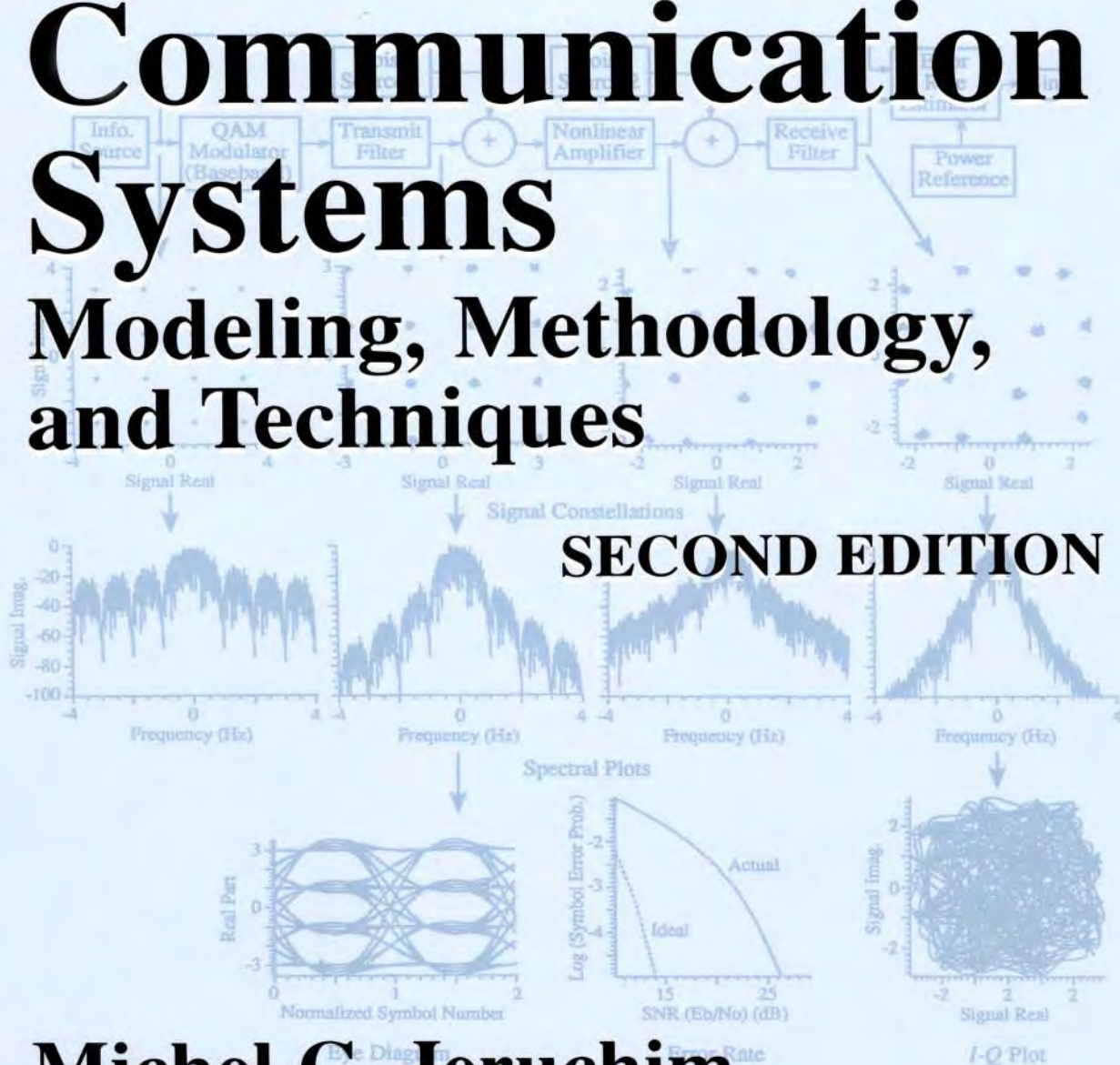


Information Technology: Transmission, Processing, and Storage
Series Editor: Jack Keil Wolf

Simulation of Communication Systems

Modeling, Methodology,
and Techniques

SECOND EDITION



Michel C. Jeruchim,
Philip Balaban, and
K. Sam Shanmugan

**Simulation of
Communication Systems Second Edition**

Information Technology: Transmission, Processing, and Storage

Series Editor: Jack Keil Wolf

*University of California at San Diego
La Jolla, California*

Editorial Board: James E. Mazo

*Bell Laboratories, Lucent Technologies
Murray Hill, New Jersey*

John Proakis

*Northeastern University
Boston, Massachusetts*

William H. Tranter

*Virginia Polytechnic Institute and State University
Blacksburg, Virginia*

Multi-Carrier Digital Communications: Theory and Applications of OFDM

Ahmad R. S. Bahai and Burton R. Saltzberg

Principles of Digital Transmission: With Wireless Applications

Sergio Benedetto and Ezio Biglieri

Simulation of Communication Systems, Second Edition: Methodology,
Modeling, and Techniques

Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan

Simulation of Communication Systems Second Edition

Modeling, Methodology, and Techniques

Michel C. Jeruchim

*Lockheed Martin Management & Data Systems
Valley Forge, Pennsylvania*

Philip Balaban

*AT&T Laboratories
Holmdel, New Jersey*

K. Sam Shanmugan

*University of Kansas
Lawrence, Kansas*

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 0-306-46971-5
Print ISBN: 0-306-46267-2

©2002 Kluwer Academic Publishers
New York, Boston, Dordrecht, London, Moscow

Print ©2000 Kluwer Academic / Plenum Publishers
New York

All rights reserved

No part of this eBook may be reproduced or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without written consent from the Publisher

Created in the United States of America

Visit Kluwer Online at: <http://kluweronline.com>
and Kluwer's eBookstore at: <http://ebooks.kluweronline.com>

To

Joan, Claude, and Kenny
and to the memory of my parents, Sonia and Samuel
—MCJ

Anna, to Victor and Nona and their families
and to the memory of my parents, Shifra and Israel
—PB

Radha, Kannon, and Ravi
and to the memory of my parents
—KSS

This page intentionally left blank

Preface

Since the first edition of the book was published, the field of modeling and simulation of communication systems has grown and matured in many ways, and the use of simulation as a day-to-day tool is now even more common practice. Many new modeling and simulation approaches have been developed in the recent years, many more commercial simulation packages are available, and the evolution of powerful general mathematical applications packages has provided still more options for computer-aided design and analysis. With the current interest in digital mobile communications, a primary area of application of modeling and simulation is now to wireless systems of a different flavor than the traditional ones.

Since the objective of modeling and simulation is to study and evaluate the behavior and performance of systems of current interest, the practice of simulation has naturally evolved along with the types of systems that have emerged or are being designed for the future. Nevertheless, to the extent that simulation is an embodiment of fundamental principles of several disciplines, communication theory in particular, the practice of modeling and simulation is still very much grounded in those basics. It is these principles, along with the many tricks of the trade that accompany their application, that still form the main focus of this second edition.

This edition represents a substantial revision of the first, partly to accommodate the new applications that have arisen. The text has been extensively reorganized and expanded. It now contains 13 chapters instead of the previous 7. Some of the former chapters have been divided into more logical units, edited for greater clarity where needed, and extended in coverage for selected topics. This division was made in part to facilitate the use of this book as a teaching text. Two new chapters were added on material only lightly covered in the first edition. One new chapter, on modeling and simulation of nonlinear systems, provides a fairly extensive discussion of “black-box” modeling of nonlinear systems with memory, and a complementary section on related measurement techniques. As hinted above, perhaps the most dramatic change in the communications/telecommunications industry since the first edition has been the explosion of wireless services. In consequence, we have included a new chapter on channel modeling, the bulk of which deals with multipath and fading channels, the usual environment for wireless systems. As in the first edition, one chapter provides several case studies as a means of illustrating different ways of approaching a problem and applying specific modeling and computational techniques from the arsenal of possibilities available to the simulation practitioner. The first case study is a thoroughly reworked version of a previous

one, and three new case studies are given. A consolidated set of problems can be found following Chapter 12.

By their nature, simulation and modeling embrace the whole of the fields to which they are applied. To cover such a breadth of material, even larger now than in the first edition, we have had again to rely on the generosity of friends and colleagues to provide us with advice and material on various topics. First, we would like to reacknowledge the contributors to the first edition, whose contributions by and large still live in these pages.

For the second edition, the list has grown longer. To our good friend and colleague at Lockheed Martin M&DS, Dr. Robert J. Wolfe, mathematician and statistician *par excellence*, we extend our gratitude for innumerable pieces of advice, proofs, and inputs on coding, nonlinear differential equations, random number generation, and interpolation, among others. Dr. Wolfe also reviewed several chapters and provided the basic material for the section on large-deviations theory (Section 11.2.5.3.2). Numerous contributions were also made by other members of the Communications Analysis and Simulation Group at Lockheed Martin M&DS. Aside from Bob Wolfe's work just mentioned, Douglas Castor and Dr. Gregory Maskarinec kindly made available their previously published work on minimum-shift-keying, which was edited into Case Study III in Chapter 12. In addition, Doug generated all the figures and carefully reviewed the final manuscript for that case study. We also benefited from many discussions with Dr. Maskarinec about nonlinear modeling, based on his extensive survey of the literature; Greg also reviewed Chapter 5 and contributed the model in Section 5.3.4.2. We appreciate the efforts of Gregory Sternberg, who used his expertise in *Mathematica* to compute Table 11.1 and to generate Figures 11.23 and 11.24. We thank Paul Beauvilliers for using his experience in simulating phase-locked loops to produce the material for Example 8.12.2 and the associated figures. We also express our appreciation to Daniel McGahey, who supplied the block diagram, its details, and the timing information that form the basis for the discussion in Section 11.2.1.

The team of Dr. Christopher Silva, Christopher Clark, Dr. Andrew Moulthrop, and Michael Muha at Aerospace Corporation were most generous in lending us the benefit of their experience and knowledge in nonlinear system modeling and measurement. The team supplied Section 5.5 on measurement techniques for nonlinear components. Dr. Silva went beyond the call of duty by providing the material on generalized Volterra models and polyspectral models in Section 5.3.3, as well as the material in Section 5.2.4.3, supplying several of the related problems, and thoroughly reviewing Chapter 5. Chris Clark is also to be thanked individually for writing Section 5.3.4.2 on nonlinear parametric discrete-time models. We have also benefited from numerous discussions with Harvey Berger of TRW on his published and unpublished work in nonlinear amplifier modeling.

Several individuals presently or formerly at AT&T Laboratories, or formerly with Bell Laboratories, made contributions that we would like to acknowledge. Our appreciation is extended to Dr. William Turin, who codeveloped and coauthored Case Study IV in Chapter 12; Bill also kindly reviewed sections of the book dealing with Markov models. We also thank Dr. Don Li for his contributions as a codeveloper of the material in Case Study IV. We are most grateful to Dr. Thomas M. Willis III for contributing the material on shadow fading in Chapter 9. We also express our gratitude to Dr. Seong (Sam) Kim for providing the material and the figures on indoor channel modeling in Chapter 9. We also acknowledge many discussions with Dr. Zoran Kostic on the workings of code division multiple-access (CDMA) systems; his advice helped shape Case Study IV.

We are indebted to Prof. Irving Kalet of the Technion, Haifa, Israel, for providing the material (and its iterations) on orthogonal frequency division multiplexing (OFDM) that

appears in Section 8.7.2.2. We much appreciate the efforts of Prof. J. Keith Townsend of North Carolina State University for many discussions on importance sampling, for inputs into Section 11.2.5.4 on stochastic importance sampling, and for the whole of Section 11.2.6 on importance splitting. Keith also made other materials available that could not be accommodated for space reasons. We thank Dr. Faroukh Abrishamkar of Qualcomm for his advice on CDMA system modeling and for providing some of the reference channel models in the Appendix to Chapter 9. Professor Vasant Prabhu of the University of Texas at Arlington was most kind to provide us with several problems that he uses for his course in simulation, and likewise we are pleased to acknowledge Prof. Brian Woerner of Virginia Polytechnic Institute for providing us with a number of projects following Chapter 12.

Finally, we renew our acknowledgment to our families for bearing with us—a second time—through this long process.

Michel C. Jeruchim
Philip Balaban
K. Sam Shanmugan

This page intentionally left blank

Contents

Chapter 1. Introduction

1.1.	Methods of Performance Evaluation.....	1
1.1.1.	Introduction.....	1
1.1.2.	Hierarchical View.....	2
1.2.	Simulation Approach: Waveform-Level Simulation of Communication Systems.....	3
1.3.	The Application of Simulation to the Design of Communication Systems.....	5
1.4.	Historical Perspective.....	6
1.5.	Outline of the Book.....	8
	References.....	12

Chapter 2. Simulation and Modeling Methodology

2.1.	Some General Remarks on Methodology.....	14
2.2.	Methodology of Problem Solving for Simulation.....	16
2.3.	Basic Concepts of Modeling.....	17
2.3.1.	System Modeling.....	21
2.3.2.	Device Modeling.....	21
2.3.3.	Random Process Modeling.....	22
2.3.4.	Modeling Hypothetical Systems.....	23
2.3.5.	Simulation with Hardware in the Loop.....	25
2.4.	Performance Evaluation Techniques.....	26
2.5.	Error Sources in Simulation.....	28
2.5.1.	Errors in System Modeling.....	30
2.5.2.	Errors in Device Modeling.....	31
2.5.3.	Errors in Random Process Modeling.....	32
2.5.4.	Processing Errors.....	35
2.6.	Validation.....	36
2.6.1.	Validating Models of Devices or Subsystems.....	37
2.6.2.	Validating Random Process Models.....	38
2.6.3.	Validating the System Model.....	39
2.7.	Simulation Environment and Software Issues.....	41
2.7.1.	Features of the Software Environment.....	42
2.7.2.	Components of the Software Environment.....	43
2.7.3.	Hardware Environment.....	45
2.7.4.	Miscellaneous.....	45

2.8. The Role of Simulation in Communication System Engineering	49
2.9. Summary.....	52
References	54

Chapter 3. Representation of Signals and Systems in Simulation: Analytic Fundamentals

3.1. Introduction to Deterministic Signals and Systems	56
3.1.1. Continuous Signals	56
3.1.2. Discrete-Time Signals	57
3.1.3. Systems.....	59
3.1.3.1. Properties of Systems.....	59
3.1.3.2. Block Diagram Representation of Systems	60
3.2. Linear Time-Invariant Systems	61
3.2.1. Continuous Linear Time-Invariant Systems.....	62
3.2.1.1. The Impulse Response	62
3.2.1.2. The Convolution Integral, ...	62
3.2.2. Discrete Linear Time-Invariant Systems.....	62
3.2.2.1. The Impulse Response	62
3.2.2.2. Convolution Sum (Discrete Convolution).....	63
3.3. Frequency-Domain Representation.....	63
3.3.1. The Fourier Transform	63
3.3.1.1. The Impulse Response	62
3.3.1.2. The Convolution Integral, ...	62
3.3.2. Frequency-Domain Representation of Periodic Continuous Signals.	65
3.3.2.1. The Fourier Series	65
3.3.2.2. Parseval's Theorem for Periodic Signals.....	66
3.3.3. The Fourier Transform	66
3.3.3.1. Convergence	67
3.3.3.2. Properties of the Fourier Transform	67
3.3.4. The Frequency Response	70
3.3.4.1. Interconnection of Systems in the Frequency Domain	70
3.3.4.2. Parseval's Theorem for Continuous Signals.....	70
3.3.5. The Gibbs Phenomenon	71
3.3.6. Relationship between the Fourier Transform and the Fourier Series	72
3.3.6.1. Introduction, ...	72
3.3.6.2. Fourier Series Coefficients.....	72
3.3.7. The Fourier Transform of a Periodic Signal	72
3.3.7.1. Periodic Convolution	73
3.3.7.2. The Poisson Sum Formula.....	74
3.4. Lowpass-Equivalent Signals and Systems	74
3.4.1. The Hilbert Transform	75
3.4.2. Properties of the Hilbert Transform	77
3.4.3. Lowpass-Equivalent Modulated Signals.....	78
3.4.4. Hilbert Transform in System Analysis.....	79
3.4.4.1. Introduction, ...	79
3.4.4.2. Lowpass Equivalent of a Bandpass Filter	79
3.4.5. Practical Considerations in Modeling of Lowpass Equivalents for Simulation. . .	82
3.4.5.1. Signals	82
3.4.5.2. Filters	83
3.5. Sampling and Interpolation	83

3.5.1.	Impulse Sampling.	83
3.5.2.	Sampling Theorem	86
3.5.3.	Multirate Sampling and Sampling Conversion	87
3.5.4.	Interpolation	89
3.5.4.1.	Introduction.	90
3.5.4.2.	Interpolator Structures for Integer Upconversion.	93
3.5.4.3.	Bandlimited and Windowed Bandlimited Interpolation	96
3.5.4.4.	Linear Interpolation.	98
3.5.4.5.	Spline Interpolation.	100
3.6.	Characterization of Linear Time-Invariant Systems Using the Laplace Transform.	106
3.6.1.	The Laplace Transform	106
3.6.1.1.	Introduction.	106
3.6.1.2.	Convergence and Stability.	106
3.6.2.	Inverse Laplace Transform.	107
3.6.3.	Properties of the Laplace Transform.	107
3.6.4.	Transfer or System Function.	108
3.6.5.	Interconnections of LTI Systems (Block Diagrams)	108
3.6.6.	Systems Characterized by Linear Constant-Coefficient Differential Equations.	110
3.6.6.1.	Properties of the Transfer Function for Linear Constant-Coefficient Differential Equations.	111
3.6.6.2.	Realizations of Rational Transfer Functions Using Biquadratic Expansion.	112
3.6.7.	Frequency Response	114
3.7.	Representation of Continuous Systems by Discrete Transfer Functions	115
3.7.1.	The z -Transform.	115
3.7.1.1.	Convergence and Stability	116
3.7.1.2.	Table of Simple z -Transforms	117
3.7.1.3.	Properties of the z -Transform	117
3.7.1.4.	Discrete Transfer or System Function	117
3.8.	Fourier Analysis for Discrete-Time Systems	118
3.8.1.	Introduction.	118
3.8.2.	The Discrete Fourier Transform.	119
3.8.3.	The Fast Fourier Transform	120
3.8.4.	Properties of the Discrete Fourier Transform.	121
3.8.4.1.	Periodic or Circular Properties	121
3.8.4.2.	The Periodic Time-Shift Property.	122
3.8.4.3.	The Periodic or Circular Convolution	123
3.8.4.4.	The Discrete Periodic Convolution Theorem	124
3.8.4.5.	The Discrete Frequency Response	124
3.8.4.6.	Relationship between the Bandwidth and the Duration of the Impulse Response.	124
3.8.4.7.	Relationship between the Discrete Fourier Transform and the z -Transform.	125
3.8.4.8.	Increasing the Frequency Resolution of the Discrete Fourier Transform.	125
3.9.	Summary.	126
3.10.	Appendix: A Brief Summary of Some Transforms and Theorems Useful in Simulation.	127
	References.	131

Chapter 4. Modeling and Simulation of Linear Time-Invariant and Time-Varying Systems

4.1. Modeling and Simulation of Linear Time-Invariant Systems	133
4.1.1. LTI Filters: Description, Specification, and Approximation	134
4.1.1.1. Filter Descriptions.....	135
4.1.1.2. Continuous Classical Filters	136
4.1.1.3. Frequency Transformations	141
4.1.1.4. Lowpass Equivalents of Bandpass Filters Represented by Rational Functions	142
4.1.1.5. Filter Specifications.....	145
4.1.1.6. Approximating Continuous Structures in Discrete Time for Simulation	145
4.1.2. Simulation of Filtering with Finite Impulse Response Filters	149
4.1.2.1. Simulation of FIR Filtering in the Time Domain	149
4.1.2.1.1. Introduction	149
4.1.2.1.2. Windowing.....	150
4.1.2.2. Simulation of FIR Filtering in the Frequency Domain	152
4.1.2.2.1. Difference between Periodic and Linear Convolution....	153
4.1.2.2.2. Linear Convolution for a Signal of Arbitrary Duration via the FFT.....	154
4.1.2.2.3. The Overlap-and-Add (OA) Method.....	155
4.1.2.2.4. The Overlap-and-Save (OS) Method.....	156
4.1.2.2.5. Efficiency of the Linear Convolution via the FFT.....	158
4.1.2.2.6. Implications of Frequency-Domain FIR Filtering	158
4.1.2.3. Mapping of Continuous Filters into Discrete FIR Filters	159
4.1.2.3.1. FIR Filters Defined in the Time Domain	159
4.1.2.3.2. FIR Filters Defined in the Frequency Domain	159
4.1.2.4. Comparison of Time-Domain (Impulse Response) and Frequency-Domain (FFT) Implementations for FIR Filtering	162
4.1.3. Simulation of Filtering with IIR Filters	165
4.1.3.1. Systems Characterized by Linear Constant-Coefficient Difference Equations	165
4.1.3.2. Structures of Recursive Discrete Filters Implemented in Simulation Models	166
4.1.3.2.1. Direct-Form (Canonic) Realization.....	166
4.1.3.2.2. The Cascade Interconnections of Biquadratic Canonic Sections.	168
4.1.3.2.3. The Parallel Realization	169
4.1.3.3. Transformations between Continuous-Time and Discrete-Time Systems Represented by Rational Functions.....	169
4.1.3.3.1. Impulse-Invariant Transformation.	170
4.1.3.3.2. The Bilinear Transformation.	173
4.1.3.3.3. Effect of Mapping on Lowpass-Equivalent Filters Represented by Rational Functions.	178
4.1.3.3.4. Guide for Mapping Recursive Filters Specified in Frequency Domain	178
4.1.4. Effects of Finite Word Length in Simulation of Digital Filters	181
4.1.4.1. Roundoff Noise in Simulations of IIR Filters.	181
4.1.4.2. Roundoff Noise in Simulations of FIR Filters	182
4.1.4.3. Effects of Quantization in Computation of the Fast Fourier Transform.	182

4.1.5.	Summary of the Process of Mapping Continuous Signals and Systems into Discrete Signals and Systems for Simulation	182
4.1.5.1.	Introduction	182
4.1.5.2.	A Guide to the Selection of the Proper Method of Filter Simulation.	183
4.2.	Time-Varying Linear Systems.	184
4.2.1.	Examples of Time-Varying Systems	185
4.2.2.	Time-Domain Description for Linear Time-Varying Systems	186
4.2.2.1.	The Impulse Response	186
4.2.2.2.	The Superposition Integral.	187
4.2.3.	Frequency-Domain Representations of Time-Varying Systems	188
4.2.3.1.	Two-Dimensional Frequency Response	189
4.2.3.2.	Bandwidth Relations in Time-Varying Systems	189
4.2.3.3.	Sampling Rate	190
4.2.4.	Properties of Linear Time-Varying Systems.	190
4.2.4.1.	Introduction.	190
4.2.4.2.	Interconnections of Linear Time-Varying Systems.	190
4.2.5.	Models for LTV Systems	192
4.2.5.1.	Linear Differential Equation with Time-Varying Coefficients	192
4.2.5.2.	Separable Models	193
4.2.5.3.	Tapped Delay-Line Channel Models	195
4.3.	Summary.	196
4.4.	Appendix: Biquadratic Factors for Classical Filters	198
	References	201

Chapter 5. Modeling and Simulation of Nonlinear Systems

5.1.	Modeling Considerations for Nonlinear Systems	204
5.2.	Memoryless Nonlinearities.	206
5.2.1.	Memoryless Baseband Nonlinearities	206
5.2.2.	Estimating the Sampling Rate for Nonlinear Systems.	207
5.2.3.	Memoryless Bandpass Nonlinearities: Analytically Based Models	209
5.2.3.1.	The Limiter Family	212
5.2.3.2.	Power Series Model.	214
5.2.4.	Memoryless Bandpass Amplifiers: Empirically Based Models	215
5.2.4.1.	Description and Interpretation of AM/AM and AM/PM Characteristics for Simulation.	218
5.2.4.2.	Lowpass Equivalent of a Bandpass Amplifier	219
5.2.4.3.	Alternative Approaches to Defining AM/AM and AM/PM Characteristics	220
5.2.4.4.	Multiple Carriers and Intel-modulation Products	221
5.2.4.5.	Setting the Operating Point of a Memoryless Nonlinearity.	223
5.3.	Nonlinearities with Memory (NLWM).	224
5.3.1.	NLWM Modeling I: Fitting Swept-Tone AM/AM and AM/PM Measurements	227
5.3.1.1.	The Poza–Sarokozy–Berger (PSB) Model.	227
5.3.1.1.1.	AM/AM Characteristics	227
5.3.1.1.2.	AM/PM Characteristics	229
5.3.1.1.3.	Combined Model	229
5.3.1.2.	The Saleh Model	229
5.3.1.3.	The Abuelma’atti Model.	232
5.3.2.	NLWM Modeling II: Fitting Preset Structures	234

5.3.2.1.	One Filter–One Nonlinearity (Two-Box) Models.	234
5.3.2.1.1.	Filter–Nonlinearity with Least-Squares Fit	234
5.3.2.1.2.	Filter–Nonlinearity ARMA Model.	235
5.3.2.1.3.	Filter–Nonlinearity with Small-Signal Transfer Function.	235
5.3.2.1.4.	Nonlinearity–Filter with Least-Squares Fit	236
5.3.2.2.	Filter–Nonlinearity–Filter (Three-Box) Models.	236
5.3.2.2.1.	Three-Box Model with Least-Squares Fit	236
5.3.2.2.2.	Three-Box Model with Specified Characteristics.	237
5.3.3.	NLWM Modeling III: Analytical Models	237
5.3.3.1.	Volterra Series Modeling.	237
5.3.3.2.	Polyspectral Models.	245
5.3.3.2.1.	Nonlinearity–Filter Polyspectral Model	246
5.3.3.2.2.	Filter–Nonlinearity Polyspectral Model	249
5.3.4.	NLWM Modeling IV: Miscellaneous Models.	252
5.3.4.1.	Power-Dependent Transfer Function Model.	252
5.3.4.2.	Nonlinear Parametric Discrete-Time Models	253
5.3.4.3.	Instantaneous Frequency Model.	255
5.3.5.	Setting the Operating Point for a Nonlinearity with Memory	256
5.4.	Nonlinear Differential Equations	257
5.4.1.	Outline of Numerical Methods	257
5.4.2.	Families of Numerical Methods.	261
5.4.2.1.	Solution Using Explicit Methods	263
5.4.2.2.	Solution Using Implicit Methods	263
5.4.2.2.1.	Iterated Predictor–Corrector Method.	263
5.4.2.2.2.	Root Finding Using Newton–Raphson Method	264
5.4.3.	Properties of Numerical Methods: Accuracy and Stability	266
5.4.3.1.	Order of a Method: Computation of Local or Truncation Error.	268
5.4.3.2.	Absolute Stability	269
5.4.4.	Computational Considerations: Methods of Quality Control	270
5.4.5.	Application of Numerical Methods.	271
5.4.5.1.	Introduction.	271
5.4.5.2.	Stand-Alone Model for a Traveling-Wave Semiconductor Amplifier.	272
5.5.	Measurement Technique for Nonlinear Components	275
5.5.1.	The Vector Network Analyzer Single-Tone Measurement	275
5.5.2.	Dynamic AM/AM and AM/PM Measurement Techniques Using a Periodically Modulated Signal.	277
5.5.3.	Time-Domain Measurement Techniques.	280
5.6.	Summary	284
	References	285

Chapter 6. Fundamentals of Random Variables and Random Processes for Simulation

6.1.	Introduction	289
6.2.	Random Variables.	291
6.2.1.	Basic Concepts, Definitions, and Notations	291
6.2.1.1.	Introduction.	291
6.2.1.2.	Statistical Averages or Expected Values	293
6.2.2.	Multidimensional Random Variables (Random Vectors)	294
6.2.3.	Complex Random Variables	297
6.3.	Univariate Models	297

6.3.1.	Univariate Models—Discrete	298
6.3.1.1.	Uniform.	298
6.3.1.2.	Binomial.	298
6.3.1.3.	Negative Binomial.	299
6.3.1.4.	Poisson.	299
6.3.2.	Univariate Models—Continuous.	300
6.3.2.1.	Uniform.	300
6.3.2.2.	Gaussian (Normal).	301
6.3.2.3.	Exponential	301
6.3.2.4.	Gamma.	302
6.3.2.5.	Rayleigh.	302
6.3.2.6.	Chi-Square.	303
6.3.2.7.	Student's t .	303
6.3.2.8.	F Distribution.	304
6.3.2.9.	Generalized Exponential.	304
6.4.	Multivariate Models.	304
6.4.1.	Multinomial.	304
6.4.2.	Multivariate Gaussian.	305
6.4.2.1.	Properties of the Multivariate Gaussian Distribution	305
6.4.2.2.	Moments of Multivariate Gaussian pdf.	308
6.5.	Transformations (Functions) of Random Variables.	308
6.5.1.	Scalar-Valued Function of One Random Variable	310
6.5.1.1.	Discrete Case.	310
6.5.1.2.	Continuous Case.	310
6.5.2.	Functions of Several Random Variables.	313
6.5.2.1.	Special Case—Linear Transformation.	313
6.5.2.2.	Sum of Random Variables.	314
6.5.2.3.	Order Statistics.	315
6.5.3.	Nonlinear Transformations.	316
6.5.3.1.	Moment-Based Techniques.	316
6.5.3.2.	Monte Carlo Simulation Techniques	316
6.6.	Bounds and Approximations.	317
6.6.1.	Chebyshev's Inequality.	317
6.6.2.	Chernoff Bound.	318
6.6.3.	Union Bound.	318
6.6.4.	Central Limit Theorem.	320
6.6.5.	Approximate Computation of Expected Values.	321
6.6.5.1.	Series Expansion Technique.	321
6.6.5.2.	Moments of Finite Sums of Random Variables.	322
6.6.5.3.	Quadrature Approximations	323
6.7.	Random Processes.	326
6.7.1.	Basic Definitions and Notations.	326
6.7.2.	Methods of Description.	328
6.7.2.1.	Joint Distribution.	328
6.7.2.2.	Analytical Description Using Random Variables	328
6.7.2.3.	Average Values.	329
6.7.2.4.	Two or More Random Processes.	330
6.7.3.	Stationarity, Time Averaging, and Ergodicity.	331
6.7.3.1.	Time Averages.	332
6.7.3.2.	Ergodicity.	333
6.7.4.	Correlation and Power Spectral Density Function of Stationary Random Processes.	334

6.7.4.1.	Autocorrelation Function and Its Properties.	335
6.7.4.2.	Cross-Correlation Function and Its Properties	335
6.7.4.3.	Power Spectral Density.	336
6.7.4.4.	Lowpass and Bandpass Processes.	337
6.7.4.5.	Power and Bandwidth Calculations.	338
6.7.5.	Cross-Power Spectral Density Function and Its Properties	338
6.7.6.	Power Spectral Density Functions of Random Sequences	339
6.8.	Random Process Models.	340
6.8.1.	Random Sequences	340
6.8.1.1.	Independent Sequences.	340
6.8.1.2.	Markov Sequences (First Order)	340
6.8.1.3.	Autoregressive and Moving Average (ARMA) Sequences	342
6.8.2.	<i>M</i> -ary Digital Waveforms	344
6.8.2.1.	Introduction.	344
6.8.2.2.	Random Binary Waveform.	345
6.8.3.	Poisson Process	346
6.8.4.	Shot Noise and Impulsive Noise	346
6.8.4.1.	Shot Noise	346
6.8.4.2.	Impulsive Noise	348
6.8.5.	Gaussian Process	350
6.8.5.1.	Definition of a Gaussian Process	351
6.8.5.2.	Models of White and Bandlimited White Noise	352
6.8.5.3.	Quadrature Representation of Bandpass (Gaussian) Signals	354
6.9.	Transformation of Random Processes	357
6.9.1.	Response of Linear Time-Invariant Causal (LTIVC) System.	357
6.9.1.1.	Stationarity	357
6.9.1.2.	Probability Distribution.	357
6.9.1.3.	Mean, Autocorrelation, and Power Spectral Density Functions	357
6.9.2.	Filtering.	358
6.9.3.	Integration	360
6.9.4.	Response of Nonlinear and Time-Varying Systems	361
6.9.4.1.	Nonlinear Systems.	361
6.9.4.2.	Time-Varying Systems	362
6.10.	Sampling of Stationary Random Processes	362
6.10.1.	Sampling	362
6.10.1.1.	Sampling of Lowpass Random Processes	362
6.10.1.2.	Aliasing Effect.	363
6.10.1.3.	Sampling Rate for Simulations	365
6.10.1.4.	Sampling of Bandpass Random Process	365
6.10.2.	Quantization	366
6.10.2.1.	Uniform Quantization.	367
6.10.2.2.	Nonuniform Quantizer	368
6.11.	Summary	369
	References	369

Chapter 7. Monte Carlo Simulation and Generation of Random Numbers

7.1.	Principle of Monte Carlo Simulation	371
7.1.1.	Definition of Monte Carlo Simulation	371
7.1.2.	Variations of Monte Carlo Simulation—Quasianalytical Monte Carlo Simulation	373

7.2. Random Number Generation	373
7.2.1. Generation of Uniform Random Numbers	374
7.2.1.1. Wichman–Hill Algorithm	376
7.2.1.2. Marsaglia–Zaman Algorithm	376
7.2.2. Methods of Generating Random Numbers from an Arbitrary pdf	377
7.2.2.1. Transform Method (Analytical)	377
7.2.2.2. Transform Method (Empirical)	379
7.2.2.3. Transform Method for Discrete Random Variables	380
7.2.2.4. Acceptance/Rejection Method of Generating Random Numbers	381
7.2.3. Generating Gaussian Random Variables	383
7.2.3.1. Sum-of-12 Method	383
7.2.3.2. Box Müller Method	383
7.3. Generating Independent Random Sequences	384
7.3.1. White Gaussian Noise	384
7.3.2. Random Binary Sequences and Random Binary Waveforms	385
7.3.3. Pseudorandom Binary Sequences	386
7.3.4. <i>M</i> -ary Pseudo noise Sequences	389
7.4. Generation of Correlated Random Sequences	392
7.4.1. Correlated Gaussian Sequences: Scalar Case	393
7.4.1.1. Autoregressive Moving Average (ARMA) Models	393
7.4.1.2. Spectral Factorization Method	395
7.4.2. Correlated Gaussian Vector Sequences	397
7.4.2.1. Special Case	397
7.4.2.2. General Case	398
7.4.3. Correlated Non-Gaussian Sequences	399
7.5. Testing of Random Number Generators	400
7.5.1. Stationarity and Uncorrelatedness	400
7.5.1.1. Introduction	400
7.5.1.2. Durbin Watson Test for Correlation	401
7.5.2. Goodness-of-Fit Tests	402
7.6. Summary	405
References	406

Chapter 8. Modeling of Communication Systems: Transmitter and Receiver Subsystems

8.1. Introduction	407
8.2. Information Sources	411
8.2.1. Analog Sources	411
8.2.1.1. Single Test Tone	412
8.2.1.2. Multiple Test Tones	412
8.2.1.3. Filtered Random Processes	413
8.2.1.4. Stored Experimental Data	413
8.2.2. Digital Sources	413
8.3. Formatting/Source Coding	414
8.3.1. Analog-to-Digital (A/D) Conversion	414
8.3.2. On Simulating the FSC Subsystem	416
8.4. Digital Waveforms: Baseband Modulation (I)	417
8.5. Line Coding: Baseband Modulation (II)	420
8.5.1. Logical-to-Logical Mapping I: Binary Differential Encoding	420

8.5.2.	Logical-to-Logical Mapping II: Correlative Coding	421
8.5.3.	Logical-to-Logical Mapping III: Miller Code.....	421
8.5.4.	Logical-to-Real Mapping I: Non-Return to Zero (NRZ) Binary Signaling	422
8.5.5.	Logical-to-Real Mapping II: NRZ <i>M</i> -ary Signaling (PAM)	423
8.5.6.	Logical-to-Real Mapping III: Return-to-Zero (RZ) Binary Signaling.....	423
8.5.7.	Logical-to-Real Mapping IV: Biphase Signaling or Manchester Code	423
8.5.8.	Logical-to-Real Mapping V: Miller Code or Delay Modulation.....	423
8.5.9.	Logical-to-Real Mapping VI: Partial Response Signaling	425
8.6.	Channel Coding.....	425
8.6.1.	Computational Load for Block Coding/Decoding.....	428
8.6.2.	Computational Load for Convolutional Coding/Decoding	431
8.7.	Radiofrequency and Optical Modulation	433
8.7.1.	Analog Modulation	434
8.7.2.	Digital Quadrature Modulation	435
8.7.2.1.	$\pi/4$ QPSK: Differential Quaternary Phase-Shift-Keying (DQSK). . . .	438
8.7.2.2.	Multitone Modulation/OFDM.	439
8.7.3.	Continuous Phase Modulation CPFSK, MSK, GMSK	443
8.7.3.1.	Continuous Phase Modulation.	443
8.7.3.2.	Continuous-Phase Frequency-Shift-Keying	445
8.7.3.3.	Minimum-Shift-Keying.	446
8.7.3.4.	Gaussian Minimum-Shift-Keying.....	447
8.7.4.	Coded Modulation.	449
8.7.5.	Modeling Considerations.....	451
8.8.	Demodulation and Detection	455
8.8.1.	Coherent Demodulation	457
8.8.2.	Noncoherent Demodulation	460
8.8.2.1.	Amplitude Demodulation.	460
8.8.2.2.	Discriminator Detection of PM/FM Signals	461
8.8.2.3.	PLL Demodulation of PM/FM Signals	465
8.9.	Filtering	467
8.9.1.	Filters for Spectral Shaping	467
8.9.2.	Filters for Pulse Shaping.....	468
8.9.3.	Linear Minimum MSE Filters.....	470
8.9.4.	Filters for Minimizing Noise and Distortion	471
8.9.5.	Matched Filters	472
8.9.6.	Adaptive Filtering (Equalization).....	474
8.9.6.1.	Tap-Gain Adaptation for Minimizing MSE	476
8.9.6.2.	Covariance Matrix Inversion Method.	479
8.9.6.3.	Simulation Considerations	480
8.9.7.	Filters Specified by Simple Functions in the Frequency Domain	481
8.9.8.	Tabular Filter for Masks and Measurements	483
8.10.	Multiplexing/Multiple Access	484
8.10.1.	Issues in the Simulation of Multiple-Access Methods.....	484
8.10.1.1.	SDMA and PDMA.	484
8.10.1.2.	FDMA	486
8.10.1.3.	TDMA	487
8.10.1.4.	CDMA (Spread Spectrum Techniques)	489
8.11.	Radiofrequency and Optical Carrier Sources.....	491
8.11.1.	Radiofrequency Sources	491
8.11.2.	Optical Sources	492
8.12.	Synchronization	495
8.12.1.	Approaches to Including Synchronization in Simulation	498

8.12.2. Hardwired Synchronization: Phase and Timing Bias	500
8.12.3. Synchronization Using an Equivalent Random Process Model	502
8.12.4. Carrier Recovery—BPSK	504
8.12.5. Timing Recovery—BPSK.	506
8.12.6. Carrier Recovery—QPSK.	510
8.12.7. Timing Recovery—QPSK.	513
8.12.8. Simulation of Feedback Loops: Application to the Phase-Locked Loop, Phase-Locked Demodulator, and Costas Loop	514
8.12.8.1. Modeling Considerations for the PLL	514
8.12.8.2. Stand-Alone PLL Model.	515
8.12.8.3. Assembled PLL Model.	522
8.12.8.4. The Phase-Locked Loop as a Phase Tracker	528
8.12.8.5. The Phase-Locked Loop as an FM Demodulator	529
8.12.8.6. Effect of Delay on the Performance of the Assembled PLL Model. . .	531
8.13. Calibration of Simulations.	534
8.13.1. Introduction.	534
8.13.2. Calibration of Signal-to-Noise Ratio or E_b/N_0 for Digital Signaling.	535
8.13.2.1. Signal Power Level	535
8.13.2.2. Noise Power Level	538
8.13.2.3. Calibrating Signal-to-Noise Ratio and E_b/N_0	538
8.14. Summary.	539
References	540

Chapter 9. Communication Channels and Models

9.1. Fading and Multipath Channels.	546
9.1.1. Introduction.	546
9.1.2. Shadow Fading.	547
9.1.3. Multipath Fading	549
9.1.3.1. Lowpass-Equivalent Characterization of Multipath Channels	550
9.1.3.2. Statistical Characterization of Multipath Channels.	551
9.1.3.3. Statistical Characterization of the Time-Variant Behavior.	551
9.1.3.4. Statistical Characterization: The WSSUS Model.	553
9.1.3.4.1. The Delay Power Profile	554
9.1.3.4.2. The Spaced-Frequency Correlation Function.	557
9.1.3.4.3. The Time-Varying Channel	558
9.1.3.5. Structural Models for Multipath Fading Channels	561
9.1.3.5.1. Diffuse Multipath Channel Model	561
9.1.3.5.2. Statistical Tap-Gain Models.	572
9.1.3.5.3. Generation of Tap-Gain Processes	575
9.1.3.6. Indoor Wireless Channels	576
9.1.3.6.1. Factory and Open-Plan-Building Model.	577
9.1.3.6.2. Office Building Model.	578
9.1.3.6.3. Ray-Tracing Prediction Model.	582
9.1.3.7. Radio-Relay Line-of-Sight (LOS) Discrete Multipath Fading Channel Model.	583
9.2. The Almost Free-Space Channel.	586
9.2.1. Clear-Air Atmospheric (Tropospheric) Channel	587
9.2.2. The Rainy-Atmospheric Channel.	587
9.2.3. The Ionospheric Phase Channel.	589

9.3.	Conducting and Guided Wave Media	591
9.3.1.	Rectangular Waveguide Medium	591
9.3.2.	The Fiber Optic Channel.....	593
9.4.	Finite-State Channel Models	596
9.4.1.	Finite-State Memoryless Models	597
9.4.2.	Finite-State Models with Memory: Hidden Markov Models (HMM).....	599
9.4.2.1.	<i>N</i> -State Markov Model.....	600
9.4.2.2.	First-Order Markov Process	601
9.4.2.3.	Stationarity	601
9.4.3.	Types of Hidden Markov Models: Gilbert and Fritchman Model.....	604
9.4.4.	Estimation of the Parameters of a Markov Model.....	606
9.5.	Methodology for Simulating Communication Systems Operating over Fading Channels.....	610
9.5.1.	Waveform-Level Simulation.....	611
9.5.2.	Symbol-Level Simulation	612
9.5.3.	Speech Coder Simulation	613
9.6.	Summary	613
9.7.	Appendix Reference Models for Mobile Channels	614
9.7.1.	Reference Channel Models for GSM Applications	614
9.7.2.	Reference Models for PCS Applications	617
9.7.3.	Reference Channel Models for UMTS-IMT-2000 Applications.....	618
9.7.3.1.	Path Loss Models	618
9.7.3.1.1.	Path Loss Model for Indoor Office Test Environment. . .	618
9.7.3.1.2.	Path Loss Model for Outdoor-to-Indoor and Pedestrian Test Environments	618
9.7.3.1.3.	Path Loss Model for Vehicular Test Environments	618
9.7.3.1.4.	Decorrelation Length of the Long-Term Fading	619
9.7.3.2.	Channel Impulse Response Model	619
	References	621

Chapter 10. Estimation of Parameters in Simulation

10.1.	Preliminaries	626
10.1.1.	Random Process Model: Stationarity and Ergodicity.....	626
10.1.2.	Basic Notation and Definitions.....	626
10.1.3.	Quality of an Estimator: Bias, Variance, Confidence Interval, and Time Reliability Product.....	628
10.1.3.1.	Bias of an Estimator	628
10.1.3.2.	Variance of an Estimator.....	629
10.1.3.3.	Confidence Interval.....	629
10.1.3.4.	Time-Reliability Product	631
10.1.3.5.	Normalized Measures	631
10.2.	Estimating the Average Level of a Waveform.	631
10.2.1.	Form of the Estimator	631
10.2.2.	Expected (Mean) Value of the Estimator.....	632
10.2.3.	Variance of the Estimator.....	632
10.2.4.	Mixture (Signal Plus Noise) Processes	635
10.2.5.	Confidence Interval Conditioned on the Signal.....	635
10.3.	Estimating the Average Power (Mean-Square Value) of a Waveform.....	636
10.3.1.	Form of the Estimator for Average Power	637

10.3.2.	Expected Value of the Estimator.....	637
10.3.3.	Variance of the Estimator.....	638
10.4.	Estimating the Probability Density or Distribution Function of the Amplitude of a Waveform.....	640
10.4.1.	The Empirical Distribution	640
10.4.2.	The Empirical Probability Density Function—Histogram	641
10.4.2.1.	Form of the Estimator	642
10.4.2.2.	Expectation of the Estimator	643
10.4.2.3.	Variance of the Estimator	644
10.5.	Estimating the Power Spectral Density (PSD) of a Process.....	645
10.5.1.	Form of the Estimator.....	646
10.5.1.1.	The Correlogram or Indirect Method	646
10.5.1.2.	The Periodogram or Direct Method	647
10.5.2.	Modified Form of the Estimator: Windowing and Averaging.....	648
10.5.3.	Expected Value of the Estimator.....	651
10.5.4.	Variance of the Estimator.....	652
10.5.5.	Some Considerations on Implementing PSD Estimators: Summary of the Simulation Procedure	653
10.5.5.1.	Welch Periodogram Procedure (Direct Method).....	653
10.5.5.2.	Windowed Correlogram Procedure (Indirect Method)	654
10.6.	Estimating Delay and Phase	655
10.6.1.	Estimating Carrier Phase and Timing Synchronization in the Noiseless Case	655
10.6.2.	Block Estimators	657
10.6.2.1.	Block Delay Estimator.....	658
10.6.2.2.	Block Phase Estimator.....	660
10.6.3.	Distribution of PLL-Based Phase and Timing Estimators	661
10.6.3.1.	Distribution of the Phase Estimator	662
10.6.3.2.	Distribution of the Timing Estimator	664
10.7.	Visual Indicators of Performance	664
10.7.1.	Eye Diagrams.....	664
10.7.2.	Scatter Diagrams.....	666
10.8.	Summary	667
	References	667

Chapter 11. Estimation of Performance Measures from Simulation

11.1.	Estimation of Signal-to-Noise Ratio	670
11.1.1.	Derivation of the Estimator.....	670
11.1.2.	Form of the Estimator.....	673
11.1.3.	Statistical Properties of the Estimator.....	673
11.1.4.	Implementing the Estimator	675
11.2.	Estimating Performance Measures for Digital Systems	678
11.2.1.	Performance Characterization for Digital Systems and Run-Time Implications.....	679
11.2.2.	A Conceptual Framework for Performance Estimation.....	683
11.2.3.	The Monte Carlo Method.....	686
11.2.3.1.	Confidence Interval: Binomial Distribution	688
11.2.3.2.	Confidence Interval: Poisson Approximation.....	691
11.2.3.3.	Confidence Interval: Normal Approximation.....	691

11.2.3.4.	Mean and Variance of the Monte Carlo Estimator	694
11.2.3.5.	Effect of Dependent Errors	696
11.2.3.6.	Sequential Estimation	697
11.2.3.7.	Estimation of Interval Measures	697
11.2.3.7.1.	Using a Generative Model.....	698
11.2.3.7.2.	Using a Descriptive Model	700
11.2.3.7.3.	Interval Simulation	701
11.2.4.	Tail Extrapolation.	703
11.2.4.1.	Form of the Estimator	706
11.2.4.2.	Asymptotic Bias of the Estimator	707
11.2.4.3.	Variance of the Estimator	707
11.2.4.4.	Summary of the Simulation Procedure for Implementing Tail Extrapolation	709
11.2.5.	Importance Sampling	710
11.2.5.1.	Formulating IS for Simulation Implementation	713
11.2.5.2.	Properties of the Importance Sampling Estimator.....	717
11.2.5.3.	Choosing Biasing Densities.....	719
11.2.5.3.1.	A Heuristic Approach	719
11.2.5.3.2.	A Formal Approach.	724
11.2.5.4.	Stochastic Importance Sampling	732
11.2.6.	Efficient Simulation Using Importance Splitting	734
11.2.6.1.	Introduction	734
11.2.6.2.	Application of DPR-Based Splitting Simulation.....	736
11.2.7.	Quasianalytical (Semianalytic) Estimation	737
11.2.7.1	General Scheme for the QA Method.....	739
11.2.7.2.	QA Method for Binary Systems	740
11.2.7.3.	QA Method for Single-Dimensional Multiamplitude Modulation. . .	743
11.2.7.4.	QA Method for QAM Modulation.	744
11.2.7.5.	QA Method for PSK Modulation	745
11.2.7.6.	QA Techniques for Coded Systems with Hard-Decision Decoding.	748
11.2.7.6.1.	Independent-Error Channel	748
11.2.7.6.2.	Dependent-Error Channel	751
11.2.7.7.	QA Method for Convolutionally Coded Systems with Soft-Decision Decoding	753
11.2.7.8.	Incorporating Jitter in the QA Technique	753
11.2.7.9.	Mixed QA Technique.....	754
11.3.	Summary	757
	References	758

Chapter 12. Four Case Studies

12.1.	Case Study I: 64-QAM Equalized Line-of-Sight Digital Radio Link in a Fading Environment.	763
12.1.1.	Introduction	763
12.1.2.	The System Model.....	765
12.1.2.1.	The Source.....	766
12.1.2.2.	Modulator	767
12.1.2.3.	Filters	767
12.1.2.4.	The Transmitted Signal	769
12.1.2.5.	The Channel.....	769

12.1.2.6.	Receiver Filtering	770
12.1.2.7.	Demodulator	771
12.1.2.8.	Receiver Noise	773
12.1.2.9.	Equalization	774
12.1.2.10.	The Detector	777
12.1.3.	The Selected Channel Snapshot Simulation	777
12.1.3.1.	Simulation Procedure	777
12.1.3.2.	Calibration Procedure	778
12.1.3.3.	Estimation of Error Probability	779
12.1.3.4.	Selected Simulation Results	781
12.1.4.	The Stochastic Channel Sequence Simulation	782
12.1.4.1.	Stochastic Channel Sequence Generation	785
12.1.4.2.	Evaluation of Error Probability: Fast Quasianalytical Method 1 (FQA-1)	786
12.1.4.3.	Evaluation of Error Probability: Fast Quasianalytical Method 2 (FQA-2)	787
12.1.4.4.	Evaluation of Error Probability: The Moment Method (Gaussian Quadrature)	787
12.1.4.5.	Simulation Procedure	790
12.1.4.6.	Evaluation of the Outage Probability	791
12.1.4.7.	Simulation Results	792
12.1.5.	Conclusions	793
12.2.	Case Study II: Phase Noise and Its Effect on Block-Coded Systems	793
12.2.1.	Introduction	793
12.2.2.	Analytical Formulation: Demodulated Signal	796
12.2.3.	Quadrupling Loop Operation	798
12.2.4.	Residual Phase Noise Model	799
12.2.4.1.	Introduction	799
12.2.4.2.	Calibrating the Model	802
12.2.5.	Residual Phase Noise Random Number Generator	803
12.2.6.	A Quasianalytical Generator for the Error Sequence	806
12.2.7.	Postprocessing	807
12.2.8.	Conclusions	809
12.3.	Case Study III: Exploring the Effects of Linear and Nonlinear Distortions and Their Interactions on MSK-Modulated Signals: A Visual Approach	809
12.3.1.	Introduction	809
12.3.2.	Linear Filter Distortions	812
12.3.2.1.	Preliminaries	812
12.3.2.2.	Bandlimiting	814
12.3.2.3.	Linear Amplitude	814
12.3.2.4.	Parabolic Amplitude	816
12.3.2.5.	Parabolic Phase	816
12.3.2.6.	Cubic Phase	817
12.3.2.7.	Residual Amplitude and Phase	818
12.3.2.8.	Combined Effects of Linear Filter Distortions	818
12.3.3.	Memoryless Nonlinear AM/AM and AM/PM Distortions	818
12.3.4.	Nonlinear Filter Distortions	820
12.3.5.	Conclusions	822
12.4.	Case Study IV: Performance Evaluation of a CDMA Cellular Radio System	822
12.4.1.	Introduction	822
12.4.2.	Brief Description of a CDMA Cellular System	825
12.4.3.	Reverse Radio Link Simulation	826

12.4.3.1.	The Simulation Model of the Reverse Radio Link	826
12.4.3.1.1.	The Transmitter	826
12.4.3.1.2.	The Channel.....	828
12.4.3.1.3.	The Receiver.....	829
12.4.3.2.	Simulation Run-Length Requirement	831
12.4.3.3.	Simulation Runs.....	831
12.4.4.	The Forward Radio Link	832
12.4.4.1.	The Simulation Model of the Forward Radio Link.....	832
12.4.4.1.1.	The Transmitter	832
12.4.4.1.2.	The Receiver.....	833
12.4.4.2.	QA Performance Evaluation of the Forward Link in a Bursty Environment	835
12.4.4.3.	Simulation Runs	836
12.4.5.	Finite-State Channel Characterization.....	836
12.4.5.1.	HMM Parameter Estimation.....	837
12.4.5.2.	Discrete Channel Modeling	838
12.4.5.2.1.	The Reverse Link.....	838
12.4.5.2.2.	The Forward Link.....	839
12.4.5.3.	The Number of States	839
12.4.5.4.	Probability Distribution of Error-Free Intervals	840
12.4.5.5.	Probability Distribution of the Number of Errors in a Block	840
12.4.6.	Conclusion	841
12.5.	Appendix: Simulation of the Tap-Gain Functions for a Rayleigh Fading Channel.....	845
12A.1.	Introduction	845
12A.2.	Estimation of the Sampling Rates and Expansion Rates	845
12A.3.	The Channel Shaping Filter.....	846
12A.4.	FIR Implementation	846
12A.5.	IIR Implementation.....	847
12A.6.	Comparison of IIR and FIR Filter Implementation	847
12A.7.	Sampling Rate Expansion and Interpolation.....	848
References		848

Problems and Projects

Chapter 3		851
Chapter 4		854
Chapter 5		856
Chapter 6		863
Chapter 7		865
Chapter 8		868
Chapter 9		871
Chapter 10		873
Chapter 11		874

Appendixes

A.	A Collection of Useful Results for the Error Probability of Digital Systems.....	879
B.	Gaussian Tail Probabilities $Q(x)$ and an Approximation $\hat{Q}(x)$	891
C.	Coefficients of the Hermite Polynomials	893
D.	Some Abscissas and Weights for Gaussian Quadrature Integration.....	895

Contents	xxvii
E. Chi-Square Probabilities.	897
Index	899

This page intentionally left blank

Introduction

The complexity of communication and signal processing systems has grown considerably during the past decades. During the same time, the emergence of a variety of new technologies such as fast and inexpensive hardware for digital signal processing, fiber optics, integrated optical devices, and monolithic microwave integrated circuits has had significant impact on the implementation of communication systems. While the growth in complexity of communication systems increases the time and effort required for analysis and design, the need to insert new technologies into commercial products quickly requires that the design be done in a timely, cost-effective, and effort-free manner. These demands can be met only through the use of powerful computer-aided analysis and design tools.

A large body of computer-aided techniques has been developed in recent years to assist in the process of modeling, analyzing, and designing communication systems⁽¹⁻⁷⁾. These computer-aided techniques fall into two categories: formula-based approaches, where the computer is used to *evaluate* complex formulas, and simulation-based approaches, where the computer is used to *simulate* the waveforms or signals that flow through the system. The second approach, which involves “waveform”-level simulation (and often incorporates analytical techniques), is the subject of this book.

Since performance evaluation and trade off studies are the central issues in the analysis and design of communication systems, we will focus on the *use of simulation for evaluating the performance* of analog and digital communication systems with the emphasis on digital communication systems.

1.1. Methods of Performance Evaluation

1.1.1. Introduction

The performance of communication systems can be evaluated using formula-based calculations, waveform-level simulation, or through hardware prototyping and measurements. (This classification is not meant to imply that the three methods are mutually exclusive; indeed, the best approach is often one that combines all three.)

Formula-based techniques, which are based on simplified models, provide considerable insight into the relationship between design parameters and system performance, and they are useful in the early stages of the design for broadly exploring the design space. However, except for some idealized and oversimplified cases, it is extremely difficult to evaluate the

performance of complex communication systems using analytical techniques alone with the degree of accuracy needed for finer exploration of the design space.

Performance evaluation based on measurements obtained from hardware prototypes of designs is of course an accurate and credible method, and is useful during the later stages of the design when the design choices are limited to a small subset. This approach is in general very costly and time-consuming and not very flexible. It is clearly not feasible to use this approach during the earlier stage of the design cycle when the number of design alternatives may be large.

With simulation-based approaches to performance evaluation, systems can be modeled with almost any level of detail desired (subject, of course, to certain limitations) and the design space can be explored more finely than is possible with formula-based approaches or measurements. With a simulation-based approach, one can combine mathematical and empirical models easily, and incorporate measured characteristics of devices and actual signals into analysis and design. Simulated waveforms can also be used as test signals for verifying the functionality of hardware.

Indeed, a simulation-based approach can be used to create a rapid prototyping environment for analysis and design of communication and signal-processing systems, an environment in which software models can be combined with hardware data and real signals to produce designs that are timely, cost-effective, and error-free.

The primary disadvantage of the simulation approach is the computational burden, which can be reduced by a careful choice of modeling and simulation techniques. A substantial part of this book is devoted to the topics of simulation models and simulation techniques.

1.1.2. Hierarchical View

In a broad sense, the term “communication system” might refer to a global communication network, a geosynchronous communication satellite, a terrestrial microwave transmission system, or a built-in modem in a personal computer. A hierarchical view that is often used to describe communication systems is shown in Figure 1.1. The top level in this representation is a communication network, which is made up of communication nodes (processors) interconnected via communication links or transmission systems as represented in the layer below. A communication link is made up of elements like modulators, encoders, filters, amplifiers, decoders, and demodulators and other components which perform signal processing operations. These elements can be analog circuits, digital circuits, or an algorithm implemented on a programmable digital signal processor (DSP). Details of these elements are represented in the bottom layer of the hierarchy shown in Figure 1.1.

A variety of simulation techniques are used to evaluate the performance of the various layers in Figure 1.1. At the network level, the flow of packets and messages over the network is simulated using an event-driven simulator, and performance measures such as network throughput, response time, and resource utilization are estimated as a function of network parameters like processor speeds, buffer sizes at nodes, and link capacities. Network simulations are used to establish specifications for the processors, protocols, and the communication links.

Communication links deal with the transmission of information-bearing waveforms over different types of communication channels (free space, cables, wires, optical fibers, etc.). For digital transmission systems, the performance of communication links is measured in terms of bit error characteristics, and the bit error rate performance is estimated by simulating the flow

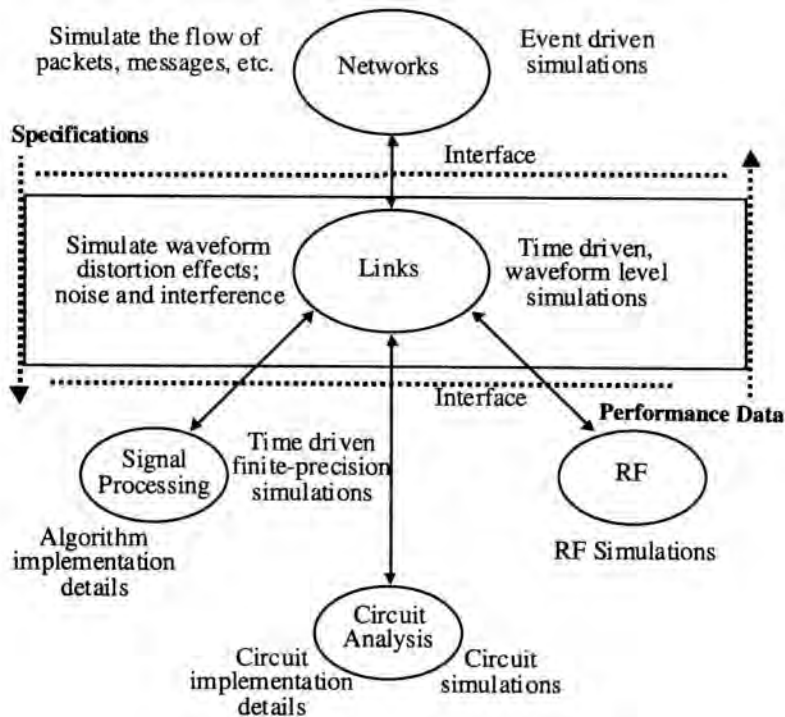


Figure 1.1. Hierarchical view of communication systems.

of waveforms using models for functional blocks such as modulators, encoders, filters, amplifiers, and channels. Whereas network simulations are used to establish specifications for communication links, link-level simulations are used to verify that the link design meets these specifications. The performance parameters obtained at the link-level simulation are exported up to the network-level simulator to verify the performance of the network.

The bottom layer in Figure 1.1 deals with implementation of components such as filters and equalizers using either analog or digital technologies. Circuit simulators like Spice or digital simulators like HDL (Hardware Description Language) are used to simulate, verify functionality, and characterize the behavior of the components. Link-level simulations establish the specifications for implementation, and simulation at the implementation level is used to provide behavioral models which are exported back to the link level. An example of a behavioral model is the transfer function for a filter.

The focus of this book is on waveform-level simulation of communication links (middle layer in Figure 1.1).

1.2. Simulation Approach: Waveform-Level Simulation of Communication Systems

To illustrate the approach used in waveform-level simulations, let us consider the simplified model of a “generic” digital communication system shown in Figure 1.2. This model shows only a subset of the functional blocks in a typical digital communication system,

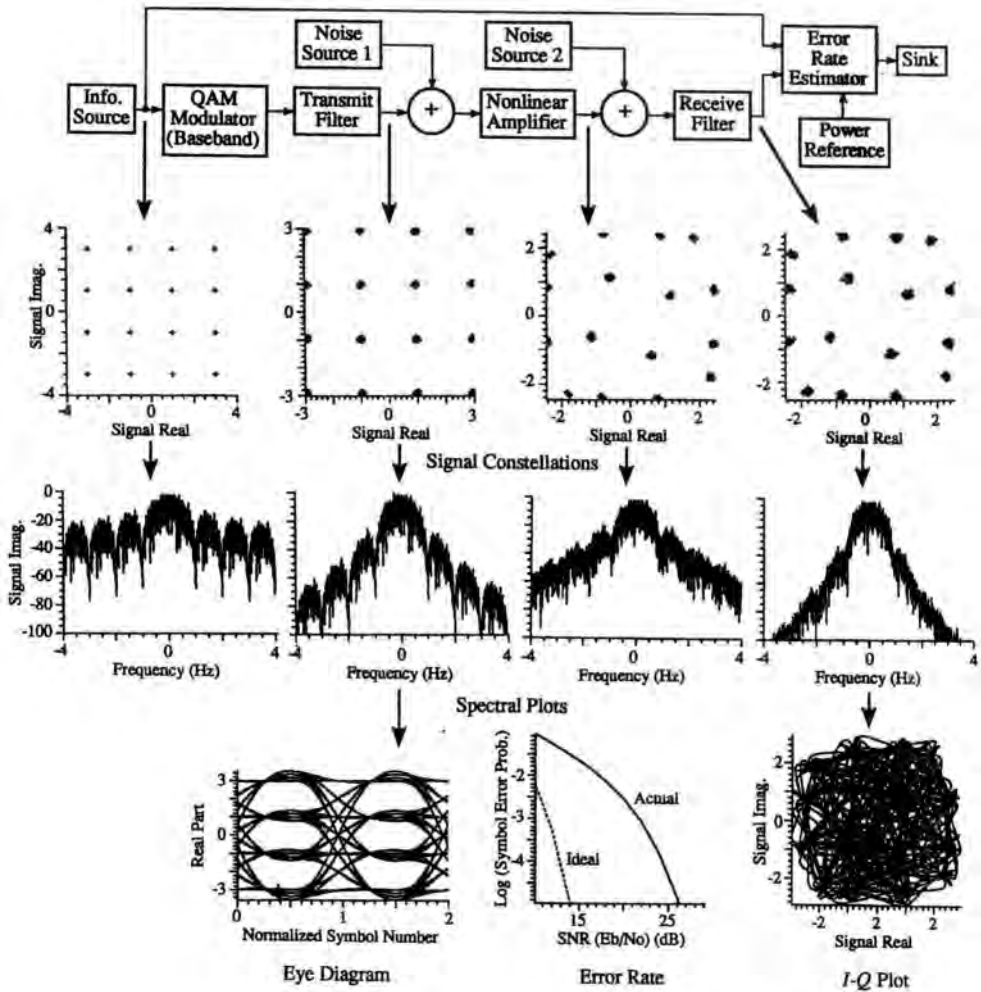


Figure 1.2. Simulation example.

and for discussion purposes let us assume that we are interested in evaluating the error-rate performance of this system as a function of the parameters of the filters (orders and bandwidths), the nonlinear amplifier (saturation level or peak power and operating point), and the signal-to-noise ratios for the two Gaussian noise sources. We are assuming that the performance of the system is determined by the signal distortions introduced by the filters and the nonlinear amplifier, and by the two noise sources.

Analytical evaluation of the performance of this system is difficult because of the presence of the nonlinearity and the filters. Bandlimiting filters introduce intersymbol interference, and the presence of noise before the nonlinearity leads to non-Gaussian and nonadditive effects, which are very difficult to characterize and analyze. Some approximations can be made by neglecting the effects of the filter preceding the nonlinearity and by combining the first noise source with the second noise source and treating the overall effect of the two noise sources as additive and Gaussian. These and other simplifications, while useful

for obtaining a “first-order” estimate of system performance, are often not accurate enough for performing detailed performance tradeoff analysis.

Estimation of the error rate via simulation involves the following steps:

1. Generate sampled values of the input processes (waveforms) (the source output and the two noise processes).
2. Process these samples through models for the filter and the nonlinearity, and generate sampled values of the output of the system.
3. Estimate the error rate by comparing the simulated values of the input sequence and the output waveform.

Examples of simulated waveforms are shown in Figure 1.2 along with sensitivity curves, which show the relationship between the performance measure (error rate) and design parameters such as the operating point of the amplifier and the receive-filter bandwidth. Smaller values of the receive-filter bandwidth reduce the effects of the noise while increasing signal distortion, whereas a larger bandwidth leads to larger noise power and smaller amounts of signal distortion. The “optimum” value of the bandwidth is chosen using the performance-sensitivity curves shown in Figure 1.2.

Note that simulated waveforms can closely mimic the waveforms that might exist in the real system. Hence, it is possible to use simulated waveforms as test signals in the real system as well as real signals to “drive” portions of the simulations. This close correspondence at the waveform level also makes it easy to incorporate in a simulation the measured values of device characteristics, such as the frequency response of filters or the transfer characteristics of amplifiers.

1.3. The Application of Simulation to the Design of Communication Systems

Simulation can play an important role during all phases of the design and engineering of communication systems, from the early stages of conceptual design through the various stages of implementation, testing, and fielding of the system.

The design process typically is initiated by the “concept definition” phase, where one imposes the top-level specifications such as information rate and performance objectives. The performance of any communication system is governed by two important factors: the signal-to-noise ratio (SNR) and the accumulated signal distortions. Generally, these are interactive and some tradeoffs are necessary. For example, with respect to the system shown in Figure 1.2, filter bandwidths affect both SNR and distortion. In most communication systems, a spread-sheet-like table called a link budget is used to keep track of factors that affect overall SNR.

The system designer starts with a candidate system and a list of design parameters. During the early phase of the design, estimates of signal-to-noise ratios and signal degradations are obtained using simpler models and educated guesses. For example, to calculate SNR, a filter may be modeled as an ideal lowpass filter with a certain bandwidth, and the distortion introduced by the actual filter may be assigned an equivalent “degradation” of, say, 2.0 dB in SNR. If the initial design produces candidate systems that meet performance objectives, then the design proceeds to the next phase. Otherwise, the topology of the candidate designs might have to be changed (say, by adding a filter or changing the encoder/decoder) and the distortion parameters must be modified.

The next phase in the design is the development of detailed specification for subsystems and components, and verification of signal distortions. Simulation plays an important role here. For example, if a filter is specified as a third-order Butterworth filter with a bandwidth-symbol time product of 0.7 (as opposed to an ideal lowpass filter with an allocation of 2.0 dB for signal distortion for link budget calculation), then waveform-level simulation can be used to verify the extent of degradation introduced by the filter. If the degradation obtained via simulation is less than 2.0 dB, then the saving can be used to relax the specification on some other component. Otherwise, additional savings will have to be sought from other components. Simulation is flexible and efficient and is often the only method available for performing these tradeoff studies and establishing detailed specifications for hardware development.

The initial step in hardware development involves the building and testing of critical components/subsystems that involve risky or new technologies. The measured characteristics of these prototype hardware components are then used in the simulations to verify the end-to-end performance of the system. Note that, at this step, simulation involves *models* for components yet to be built, and *actual characteristics* of components already built and tested. If simulations produce satisfactory values for performance objectives, then the remaining hardware components are built and a prototype hardware for the entire system is “wired together” and tested. Otherwise, specifications are modified and parts of the design are redone.

When the hardware prototype of the system is completed, it is tested and the test results are compared with simulation results. The degree of closeness between the hardware and simulation results is the basis for declaring whether or not the simulation is “valid.” The validated simulation model can be used to predict the end-of-life (EOL) performance of the system using postulated characteristics of key components due to aging. The validated simulation model can also be used during operational stages for trouble shooting and for providing answers to “what if” scenarios.

Thus, simulation can play an important role at any point in the life cycle of a communication system: at the conceptual definition stage, where top-level specifications are derived; as an ongoing part of design and development, in concert with hardware development to finalize specifications and check the influence of an as-built subsystem on the system performance as a whole; out to the operational scenario, where simulation can be used as a trouble-shooting tool; and to predict EOL performance of the system.

1.4. Historical Perspective

Waveform-level simulation started with the invention of analog computers in the 1940s, which were first used to simulate the behavior of control systems used in aircraft and weapons systems.⁽⁸⁾ An analog computer is a simulator of continuous systems composed of modular components interconnected via a patchboard into a block diagram configuration representing the system. The linear elements, such as integrators and adders, are realized using feedback DC operational amplifiers. Nonlinear modules such as multipliers and trigonometric functions were first realized by electromechanical servo systems and later by piecewise linear approximations. Any system whose behavior is described by a linear or nonlinear differential equation with constant coefficients or with time-varying coefficients can be reduced to a block diagram made up of components of the analog computer. By wiring the components

according to the block diagram and exciting the model with appropriate signals, one can simulate the dynamic behavior of a broad range of linear and nonlinear systems using the analog computer.

The development of high-speed digital computers and the availability of large-capacity memory enabled their usage in simulation applications. This development opened the field of modeling to new disciplines such as numerical analysis and programming. The large dynamic range of floating point number representation freed the user from the drudgery of signal scaling. General frameworks for digital simulation originated with block-oriented languages such as MIDAS, SCADS, and CSMP,^(9,10) which were developed in the early 1960s. These simulation languages emulated the behavior of analog computers on a component-by-component basis. For example, a summer is replaced by the code for addition, and an integrator is replaced by an integration subroutine. The interconnections between the components are specified by a block-oriented language just as the analog computer patchboard electrically links analog computing components. Block-oriented simulation languages draw their motivation from the analog block diagram as a simple and convenient way of describing continuous systems.

Applications of digital computers to circuit analysis and simulation with programs such as ECAP and SPICE⁽¹¹⁾ in the mid 1960s led to advances in numerical integration techniques and topological simplification of signal flowgraphs.

Advances in discrete-time systems and digital signal processing have led to new approaches for digital simulation of systems. Software packages for simulations based upon transform domain techniques (Fast Fourier transform for frequency domain techniques and bilinear-Z transform for time domain techniques) began to emerge in the late 1960s and the early 1970s. SYSTID,^(12,13) CSMP, CHAMP,^(14,15) LINK,¹⁶ and others^(17,18) were developed during this period for aiding in the analysis and design of satellite communication links.

While the initial versions of SYSTID and similar packages were language-oriented and designed to operate in a batch mode, later versions of SYSTID, and other packages such as ICSSM and ICS,^(19,20) were interactive and menu driven, at least in part. With these packages, simulations were performed on a mainframe or a super-minicomputer, and graphics terminals were used to provide a limited amount of interactive preprocessing as well as postprocessing.

Computer hardware and software technologies have since undergone significant changes. Powerful workstations and personal computers offer very friendly computing environments with highly visual and graphical interfaces. The Boss⁽²¹⁾ software package was the first to take advantage of the advances in workstation technologies to create a graphical user-friendly framework for simulation-based analysis and design of communication systems. The current generation of simulation software packages (SPW,⁽²²⁾ COSSAP,⁽²³⁾ MATLAB/SIMULINK⁽²⁴⁾ and others) offer interactive, graphical, and user-friendly frameworks for developing simulation models in a hierarchical fashion using graphical block diagram representations, and permit the user to configure and execute waveform-level simulations, review the results of simulations, and perform design iterations. These tools also provide database management, on-line help, on-line documentation, and other services and features. These features minimize the amount of effort the communication system engineer has to put into the creation and debugging of simulation programs, and other mundane details associated with the mechanics of simulating communication systems. With the availability of the current generation of simulation frameworks, attention is now focused on important issues such as modeling and simulation techniques, estimation of performance measures, and computational efficiency, which are the subject matter of this book. We assume that the reader's primary

interest is in understanding modeling and simulation techniques, problem formulation, and using simulations for analysis of proposed designs, and not in building simulation frameworks or tools as such. This book is intended to provide the conceptual underpinnings for such tools.

Many of the commercial packages like SPW, COSSAP, and Matlab provide interfaces to network simulators and also links to circuit design and other implementation tools. While we will briefly describe some of these interfaces; detailed discussion of network simulation and links to implementation tools is beyond the scope of this book.

1.5. Outline of the Book

Simulation of communication systems involves generating sampled values of signals and noise, processing these sampled values through discrete-time models of functional blocks in those systems, and estimating certain properties of the signal at various points in the system, with particular emphasis on performance measures at the output. The validity and accuracy of simulation results will depend on the correctness of the modeling and estimation techniques and generally, but not always, on the length of a simulation. This book covers all major aspects of the modeling and simulation of communication systems (that is, the middle layer as defined in Figure 1.1) with emphasis on providing a practical introduction rather than a theoretical discourse. By *practical* we mean those countless hints, tricks of the trade, and how-to information that will permit an individual with little previous background actually to put together a useful simulation, or to thoroughly appreciate the underpinnings of commercial software packages. On the other hand, “practical” does not mean that we have no interest in theory. Indeed, the discipline is grounded on theoretical concepts and we provide coverage of theoretical considerations that are especially relevant to simulation. Nevertheless, since such considerations are not our principal focus, this coverage is given in a condensed, review fashion with adequate references for the reader who might want to explore the theory in greater detail. In this section we provide a broad-brush overview of the following chapters. A detailed topical description is contained in the Contents.

We begin, in the next chapter, with a discussion of the methodology of modeling and simulation. The term *methodology* implies ways of dealing with and thinking about a subject, and in that sense Chapter 2 underlies the remainder of the book. Methodology includes both qualitative and quantitative aspects of the discipline, which we shall label, respectively, the “art” and the “science” of simulation. Important aspects of methodology include general concepts of modeling and modeling hierarchy, and the central notion of validation. This chapter should be useful in acquiring the right mindset for those with relatively little experience in simulation, and it may be profitable to revisit or browse through from time to time when reading other chapters.

Chapter 3 reviews and collects some basic topics in linear system theory, with emphasis on the representation of signals and linear time-invariant (LTI) systems in the simulation context. Although many of the topics covered are traditional, we strive to emphasize the connection between theoretical constructs and their practical application to the actual process of simulation. Simulation using the digital computer requires us to deal with discrete-time or sampled versions of continuous-time signals and systems. Thus, the core topic dealt with in this chapter is the link between continuous-time signals and systems and their discrete-time counterparts. This link is provided by the well-known sampling theorem. We enlarge the

typical discussion of sampling to include the important practical situation where multirate sampling is appropriate, and also discuss the related topic of interpolation. We introduce standard techniques for processing discrete-time signals, such as the z -transform and the discrete Fourier transform, and in the process expose and contrast various time- and frequency-domain relationships for discrete-time and continuous-time signals and systems. Another important practical topic addressed here is the complex lowpass representation, which is critical to the efficient simulation of bandpass signals and systems.

Probably the most ubiquitous linear operation is *filtering*; indeed, every linear operation can be thought of as a form of filtering. In the first part of Chapter 4, we examine in some detail how to perform time-invariant filtering within simulation. Although there are many ways of describing filters, a very important characterization from the point of view of simulation techniques is whether the impulse response has a finite duration (FIR) or an infinite duration (IIR). This dichotomy leads to fairly distinct sets of approaches. FIR filtering itself is divided into two approaches that are often referred to as time-domain and frequency-domain processing. The latter is most often used, and is based on the use of the discrete Fourier transform in its “fast” version, the FFT. IIR filtering is based on certain s -domain to z -domain mappings which result in recursive difference equations. We present a class of continuous filters, called the classical filters, that frequently serve as the s -domain structures that are converted to digital equivalents. In this portion of the chapter, we also provide step-by-step summaries of the process of filtering according to the several methods mentioned.

The second part of Chapter 4 deals with linear but *time-varying* (LTV) systems. A brief review is provided of various relationships in the time and frequency domains between the input and output of such systems. The time variations preclude the relatively simple input-output relationships that apply in the LTI case. The main result of this chapter portion is the derivation of a tapped delay-line model for LTV systems that is practical to implement in simulation. For the type of applications in which we are interested, the LTV “system” is typically a communication channel, such as the mobile channel. However, such a channel is a *randomly* time-varying system. The tapped delay-line model just mentioned is deterministic, but will be reinterpreted to apply in the random case when we get to Chapter 9.

Chapter 5 deals with the modeling and simulation of nonlinear systems, generally the most intractable to treat analytically. Generally, nonlinear systems are not especially difficult to simulate, given the model. Obtaining a high-fidelity model is where the main difficulty lies. Toward this goal, we define categories of models that depend largely on the ratio of signal bandwidth to system (or device) bandwidth. When that ratio is small, models called “memoryless” are often adequate and relatively easy to arrive at. Nonlinear memoryless devices are specified by input-power/output-power and input-power/output-phase characteristics that are conventionally measured. If the ratio of signal bandwidth to system bandwidth is relatively large, models “with memory” will typically have to be used. It is generally a difficult task to develop a model with memory that has validity for a wide range of signals. We outline various approaches for developing models with memory, all of them dependent to a greater or lesser degree on some type of measurement. Since measurements are central to the development of nonlinear models, we also provide in this section a primer on related measurement techniques. Such measurements, incidentally, can be used to infer whether the appropriate modeling construct is one with or without memory. A nonlinear system is one whose behavior is governed by a nonlinear differential equation. In communications, the most common forms of such systems are synchronization and tracking loops. We provide an overview of numerical methods that can be used to simulate such equations directly.

Information-bearing signals, noise, and interference in communication systems are random in nature. In addition, the characteristics of some elements of a communication system can change randomly over time. Such elements could be manufactured equipment, whose behavior is influenced by age and environment, or they could be the environment itself, for example, a fading channel. These phenomena, as well as signals, noise, and interference, are modeled using random variables and random processes. Consequently, a review of the main elements of the theory of random variables and processes is provided in Chapter 6, along with a short catalog of distributions that are commonly encountered.

In simulation, random processes such as noise must be generated in some fashion, since by their nature their values at sampling times are not given. Generating sequences of numbers that appear to behave as if they were samples of a typical path of a random process relies mainly on techniques of random number generation, which forms a discipline distinct from the theory of random processes itself. Consequently, we study such techniques in Chapter 7. An algorithm for generating a sequence that imitates a particular random process is called a random number generator (RNG). Each type of random process must have a corresponding RNG, but for any particular process there can be more than one distinct algorithm. Algorithms can differ in their properties in different ways. The most desirable ones, of course, imitate most closely the process they are intended to reproduce and are computationally efficient. In this chapter we look at several ways in which RNGs can be constructed and supply a number of algorithms for generating sequences of numbers having different types of distributions. A brief look at methods for testing the quality of RNGs is also given. A subclass of algorithms for generating “pseudorandom” sequences of digital symbols, often called shift-register sequences, is also presented, both for binary as well as M -ary symbols.

One of the central activities related to simulation is to develop models for the various building blocks of a communication system as well as for the signal and noise processes that are the stimuli to any system. A *model* is any description of the functioning or behavior of an element of a system that forms the basis for the representation of that element in a simulation. With regard to signal and noise processes, the model consists simply in deciding what the nature of the process is. The *construction* of the model in this case would be the synthesis of a proper RNG. A similar approach holds for random channels, in which the modeling per se consists in choosing a suitable set of probabilistic properties, which then have to be implemented with some RNG. In some cases, we may decide to use deterministic *test* signals or patterns as inputs to a system, which may have no resemblance to the actual signals. This, of course, is often done in practice, and the generation of such test signals is straightforward. Primarily, however, Chapter 8 deals with models of transmitter and receiver *subsystems*. In terms of the modeling hierarchy of Chapter 2, subsystems are the modeling entities closest to the entity of interest, namely, the system. Modeling at that level will generally result in the most computationally economical program. In formulating subsystem models, we advocate a “functional” approach that allows, wherever possible, the incorporation of “knobs” in the model to allow for real-life departures from ideal behavior. We discuss in varying degrees of detail a number of important subsystems in digital communications links: source formatting, modulators, demodulators, filters, equalizers, encoders/decoders, and synchronization subsystems.

All signals in communication links travel through some sort of medium, which can be “wired” or “wireless.” Such media can be the performance-limiting element of a link, and their effect must of course be included in a simulation. Most media can be considered linear, but some are time-invariant, or can be considered to be so for an application, and others are time-varying. We look at two examples of LTI wired media, namely waveguides and optical

fibers, and provide equations for their transfer functions. These can essentially be treated as filters in the same manner as those discussed earlier. We also look at some examples of wireless media that can be considered as LTI or slowly varying, namely the ionosphere and the troposphere. Most of Chapter 9, however, is devoted to a discussion of the *multipath fading channel* which generally dominates the design considerations for mobile, cellular, and personal communication systems (PCS) both indoors and outdoors. We present a general model for such channels, only the parameters of which have to be tailored for specific applications. We look at three specific multipath channels: outdoor mobile, indoor, and radio-relay. For convenience, we include certain reference channel models proposed by standards bodies for system studies. We also describe in this chapter finite-state (*discrete*) channel models. We describe the burst errors encountered in fading channels. The characterization of channels in the form of finite-state Markov models is extremely useful for network layer simulations.

The next two chapters are devoted to statistical aspects of simulation, to be distinguished from the material covered in Chapters 6 and 7.

In particular, we mean by this the estimation of quantities that one might normally wish to measure in an actual system, but are subject to statistical variations for various reasons. We divide the quantities to be estimated into two categories: parameters and performance measures. In Chapter 10 we concentrate on estimation of parameters; these may be descriptors of a waveform, such as the average power; they may be quantities inherently estimated in a real system, such as phase and timing references; or they may be rough indicators of a system's health, like eye diagrams.

Most system-level simulations are run to obtain an estimate of a performance measure. A performance measure is an indicator of the quality of signal transmission. Chapter 11 is devoted to this topic. For analog signals, the quality indicator that is almost universally accepted is the signal-to-noise ratio (SNR), and in general the SNR is a useful indicator of the noisiness of a signal. We derive the form for an SNR estimator for a general system and describe its implementation in simulation in one of two forms, a time-domain version and a frequency-domain version. The bulk of Chapter 11, however, is concerned with the performance estimation problem for digital transmissions. Performance measures for digital signals are invariably framed in terms of some description of the errors to be found in the recovered symbol (bit) stream. The traditional indicator has been the average number of errors per transmitted symbol, otherwise known as the bit-error-rate, or BER, but other measures may be even more relevant, depending on context. For example, one may be more interested in the probability of an error in a block or frame than in the BER per se.

Even though in practice it is often not thought of in this fashion, it is necessary to cast the performance estimation problem as a statistical one. It is, of course, a statistical problem: any estimate of the indicator will vary from one measurement to another because the waveforms involved are random. In many (perhaps even most) practical situations, this is not a problem because the real time involved to reduce the statistical fluctuations to negligible levels is very short. In simulation this is typically not the case: computational time per processed symbol is not real time. Therefore, it is important to bear in mind the statistical aspect and quantify it. The standard Monte Carlo technique is too slow for many problems of interest, and more clever alternatives have to be considered. Chapter 11 discusses a number of such alternatives and attempts to quantify the corresponding run-time requirements.

Chapter 12 contains four case studies that are intended to illustrate the application of as wide a range of simulation techniques and ideas as possible. The first case study deals with performance estimation for a terrestrial radio-relay system in which fading occurs, and

equalization is used to counter it. The second case study focuses on the development of an ultraefficient method to deal with the Monte Carlo estimation of very low error rates in a coded system. The dominant error source here is phase noise, and the study concentrates also on the synthesis of an efficient phase noise generator. The third case study is largely qualitative and illustrates the power that simulation has, to provide insight into problems when simulation is viewed as a sort of software laboratory. An array of scatter diagrams displays the nature of linear, nonlinear, and combined distortions that yield understanding into the system's behavior. The final case study deals with a code division multiple-access (CDMA) mobile system, in which the dominant performance-limiting effect is due to multipath fading. Dealing with that effect in a computationally efficient manner is the thrust of this study.

References

1. P. Balaban, K. Sam Shanmugan, and B. W. Stuck (eds.), *J. Select. Areas Commun. SAC-2* (January) (1984).
2. P. Balaban, E. Biglieri, M. C. Jeruchim, and K. Sam Shanmugan (eds.), *J. Select. Areas Commun.* **6**(January) (1988).
3. H. T. Moufrah *et al.* (eds.), *J. Select. Areas Commun.* **9** (January) (1991).
4. J. K. Townsend *et al.* (eds.), *J. Select. Areas Commun.* **11** (April) (1993).
5. K. Ben Letaief *et al.* (eds.), *J. Select. Areas Commun.* **15** (May) (1997).
6. K. Sam Shanmugan (ed.), *IEEE Commun. Mag.* **32** (July) (1994).
7. F. M. Gardner and J. D. Baker, *Simulation Techniques*, Wiley, New York (1997).
8. C. L. Johnson, *Analog Computer Techniques*, 2nd ed., McGraw-Hill, New York (1963).
9. R. J. Harnett *et al.*, MIDAS: An analog approach to digital computation, *Simulation* (September 1964).
10. R. D. Brennan, Continuous system modeling program, in *Proc. IFIP Conference on Simulation Programming Languages* (1968), pp. 371–396.
11. J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, New York (1963).
12. SYSTID Time Domain Simulation Program, NASA JSC contract No. NAS9-10832, Final Report (February 1971).
13. M. Fashano and A. L. Strodbeck, Communication systems simulation and analysis with SYSTID, *IEEE J. Select. Areas. Commun. SAC-2*, 8–29 (1984).
14. L. C. Palmer, Channel modeling for data transmissions: A digital simulation approach, in *Proc. IEEE Information Theory Group Workshop*, Washington, D.C. (June 1972).
15. L. C. Palmer, Computer modeling and simulation of communication satellite channels, *IEEE J. Select. Areas Commun. SAC-2* 89–102 (1984).
16. H. B. Poza, A. B. Sarkozy, and H. L. Berger, A wideband data link computer simulation model, in *Proc. NAECON '75* (June 1975), pp. 135–142.
17. D. Hedderly and L. Lundquist, Computer simulation of a digital satellite communication link, in *Proc. ICC '72* (April 1973), pp. 321–325.
18. M. C. Jeruchim, Digital computer simulation of satellite quadrature data communication systems, in *Proc. NTC '75*, New Orleans (1975), pp. 33–16–33–21.
19. W. D. Wade *et al.* Interactive communication system simulation model-ICSSM, *IEEE J. Select. Areas Commun. SAC-2*, 102–129 (1984).
20. J. W. Modestino and K. R. Matis, Interactive simulation of digital communication systems, *IEEE J. Select Areas Commun. SAC-2*, 51–77 (1984).
21. K. S. Shanmugan *et al.*, Block-oriented systems simulator (BOSS), in *Proc. MILCOM '86* (October 1986) paper 36.1.
22. *Signal Processing Work System, User's Manual*, Cadence Design Systems Inc., San Jose, California.
23. *COSSAP User's Manual*, Synopsys Inc., San Jose, California.
24. *Simulink User's Guide*, Prentice Hall, Englewood Cliffs, New Jersey (1997).

Simulation and Modeling Methodology

Building simulation models and running (executing) simulations are activities that call upon a wide variety of skills and considerations which, for present purposes, we might divide into two broad categories: the “art” and the “science” of simulation. In the latter category we include the more analytically based and quantitative aspects which form the bulk of the succeeding chapters. On the other hand, there is a set of considerations only partially or perhaps not at all related to theoretical or quantifiable matters, or difficult to describe in such terms, that are nevertheless fundamental in building simulations and in obtaining useful results. This set we might regard as making up the “art” of simulation. These latter considerations and ways of dealing with them essentially form the *methodology* of simulation. The dividing line between “art” and “science” is somewhat subjective, but is not critical in any case. In this chapter we outline a number of issues that we shall classify as methodological. These issues themselves are in general not clearly separable, but for discussion purposes certain classes of ideas have been identified, as described later.

We will begin first with an overview of various methodological ideas, how they intertwine, and how they relate to traditional methods of analysis. Probably the most fundamental methodological issue is how to map a real problem into one solvable (to an extent that is possible) by simulation. Ideally, one would like a simulation to be a perfect replica of the real system. For many, if not most, cases of interest, this objective, even if possible, would be too costly in terms of complexity or run time (run time is measured in terms of CPU time). In order to reduce the problem to a manageable one within acceptable approximation, a variety of approaches or techniques are available. These approaches or techniques can be logically divided into two levels. One is the “modeling” level, where we are interested in representing the system, or specific functions or processes within the system, in the simplest manner possible, consistent with an acceptable degree of approximation. We discuss this level in Section 2.2. The second level is the “performance evaluation” level, where we are interested in estimating the appropriate performance measure for that system. Hence, in Section 2.3 we look into explicit embodiments of performance evaluation techniques. By and large this implies some combination of simulation and analysis along the lines discussed in Chapter 11. Here, our discussion will be largely qualitative, leaving the details for Chapter 11. The subsequent two sections deal with a question of fundamental importance from a practical standpoint, namely, What is the relationship between the results of a simulation and “reality”? This question has two sides. The first is, In what ways can errors manifest themselves in a simulation? The second is, Can we quantify them? The first side is discussed in Section 2.4. The other side is more or less synonymous with the idea of “validation,” the term commonly

used in this context. This facet is examined in Section 2.5. Both sides of this question are, of course, important when dealing with an actual problem. Generally, actual problems relate to the design of proposed systems, for which a distinct methodology exists in the larger context of the systems engineering of communication systems. In Section 2.6 we look at the role of simulation in that larger context. By and large, the issues discussed in Sections 2.1–2.6 are independent of the specific software and hardware environment used to construct and run a simulation. However, since a simulation must ultimately come to life, as it were, in some environment, it is worthwhile outlining the major related considerations. Thus, in Section 2.7, the final one of this chapter, we address fairly briefly a few issues related to the software and hardware environment.

We remark that, by its nature, methodology is difficult to systematize. In any given problem, judgment and experience will dictate what methodological tricks may be useful or usable. This chapter, therefore, is intended only as a point of departure and general guidance. The material in this chapter, however, does not stand in any particular relationship with that of other chapters. Ideally, it should be read first because it sets the context for later chapters. On the other hand, because many of the terms and ideas discussed here will not be fully explored until later chapters, a full appreciation of methodology will probably not emerge until such later reading. Hence this chapter can be read before or in conjunction with other chapters, and it would probably be fruitful to revisit it from time to time.

2.1. Some General Remarks on Methodology

If it were simple to produce perfect models, and if we had unlimited computing resources, there would be little to the art of simulation. Of course, such is not the case, and under these limitations the art of simulation can be said to be to produce answers to problems within reasonable time and within an acceptable degree of approximation. In this sense, simulation is no different from traditional methods of analysis, in which one approximation or another is almost always made in order to render a problem analytically tractable. While analysis and simulation both involve approximations, they are of course different. In the former, one typically *computes* a number that represents the quantity of interest. In simulation, waveforms unfold in time, in what is hoped to be a good imitation of the system of interest. Thus, a central distinction between the two is that simulation possesses a dynamic quality absent from analysis. This aspect is most useful from an engineering standpoint, as it allows monitoring of the system at different points, thus providing insight not otherwise easily available. Ultimately, the approximations amount to simplifications of one sort or another in the representation of the system itself or of the functioning of one or more of the subsystems. Here, another distinction arises between analysis and simulation in that in the latter it is often no more difficult to use a realistic model of some operation than an idealized one, something that is generally not true in analysis. Another useful aspect of simulation is the flexibility with which the description of any one element of the system can be changed without affecting any other part of the simulation; this property is reflected in the ability to build software that is modular. In analysis, on the other hand, changing any part of the model will typically require a whole new analysis. The flexible aspect of simulation enables the system designer to track the evolution of a system from inception to finalization. As the properties of subsystems become better defined, either through refinement of specifications or measurements made on prototype hardware, the models of these subsystems can be correspondingly updated. Of course,

the ultimately preferable solution is an analytic expression, if it is sufficiently “readable” to provide insight, if it does not require numerical evaluation, and if it is sufficiently accurate—conditions often not met for real problems. The point here is not to draw a relative ranking between simulation and analysis: each has its advantages and disadvantages. In fact, as will be seen, in many cases an appropriate methodology for solving a particular problem will consist of a combination of analysis and simulation.

We have so far used the word “simulation” as if it were a uniquely identifiable technique. This is not the case. There is a range of techniques under that heading, the common element among them being the dynamic aspect mentioned earlier. What might be called the classical conception of simulation is referred to as *Monte Carlo* simulation (we shall often use the abbreviation MC for Monte Carlo). As we use this term here, MC simulation is a particular application of a general method (see, e.g., Refs. 1 and 2). The method may be applied to deterministic or probabilistic problems. In the latter class, we may define the Monte Carlo method to be a computational procedure in which random quantities are produced by drawing samples randomly from populations of values having the intended statistical properties. This is done by constructing appropriate *random number generators* (RNGs), which by and large will deliver only approximations to the intended properties. In this book, the random quantities in question will be signals, noise of different types, interference, and certain channels. Ideally, the generated “random” sequences are a completely faithful software counterpart of the actual processes: sometimes, the words *imitation* or *emulation* are used to express this notion. To the extent that the RNGs imitate the actual processes, Monte Carlo simulation is usually considered to be the most accurate simulation technique. However, accuracy involves two distinct aspects: one concerns the generation of system stimuli and channels, as mentioned, and the other concerns the models of the hardware (or software) operating upon these waveforms. Thus, it is important to separate the notion of MC simulation, as just defined, from that of accuracy itself. The sense in which Monte Carlo simulation is most often used is to imply the time evolution of waveforms (again, the dynamic implication). There are also variations of the MC method, discussed in Section 2.4, in which some, but not all, processes are emulated.

To recapitulate, accuracy implies both the faithful reproduction of the random processes in the system and high-fidelity hardware models. It will be useful for later discussion to define the term *as-is* to mean the description of a system as it really is, detail for detail. Thus, a Monte Carlo simulation does not necessarily represent a system *as-is*. The set of ideas surrounding the representation of a system, its elements, and the input processes is precisely the subject matter of *modeling*, which is discussed in the next section.

An important consequence of the Monte Carlo technique is that an MC simulation is equivalent to a random experiment. That is, the output must be observed for some period of time, and statistics collected, whose reliability depends on the length of observation. When the purpose of the simulation is to estimate some measure of error production for digital systems, the required length of observation is often such as to render the computation time prohibitive. Here, the length of observation is measured in terms of processed symbols, while the computational effort is determined by the processing time per symbol. In Chapter 11 we look at a particular system example in order to give a rough calibration of computational requirements in terms of actual run time. It is enough to say for present purposes that this time can range from hours to literally months, depending on the specific conditions. Primarily for this reason, there have evolved a number of alternative techniques to Monte Carlo simulation, designed to shorten the run time. Because the purpose of a simulation is to evaluate some property of a system, or its “performance” in some sense, we also refer to the various

simulation methodologies as *performance evaluation techniques*. We will discuss the various factors affecting the choice of technique in Section 2.3. In Chapter 11 we will discuss these techniques in detail.

Implicit in the term simulation is a software and computing environment. Essentially, such an environment is the physical enablement of the abstractions that are used to model the system. Our focus in this book is on these abstractions. But when we speak of run time, we are implicitly invoking some sort of environment. Thus, even though at one level one can separate the notions of simulation, as such, and that of the computational engine, it is worthwhile having some appreciation of the major aspects of the latter. For this reason we shall conclude the chapter with a fairly brief discussion of software considerations, including a description of the organization of typical programs. We shall also discuss recent or emerging trends such as the integration of implementation tools into simulation.

2.2. Methodology of Problem Solving for Simulation

As previously implied, a real communication system generally is far too complex to attempt to represent and simulate it in its entirety. Just as we seek analytical tractability by simplifying some aspects of a problem, so do we strive for computational tractability by similar measures. Generally, this objective will be met by reducing the complexity of a problem in one way or another. In addition to the possible use of simplifying assumptions, the methodology for achieving complexity reduction is to try to partition the problem into smaller subproblems whose solutions are then meaningful to combine in some fashion. Reducing the larger problem to simpler form can be fruitfully viewed as conducting a *conditional* experiment (or several such). To expose this perspective, consider the output waveform V_t of a system at discrete time t in the symbolic form

$$V_t = g(\Omega)$$

where g is the system transfer characteristic and $\Omega = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_K)$ is a collection of (discrete-time) input processes. The function of a simulation, generally, would be to produce a sequence of values $\{V_t\}$ for $t = kT_s, k = \pm 1, \pm 2, \dots$, with T_s the simulation sampling interval. This sequence would be processed in some fashion to extract a performance measure or other relevant information.

A conditional experiment would produce

$$V_t = g(\Omega')$$

where $\Omega' = (\mathbf{z}_1, \dots, \mathbf{z}_k, \mathbf{z}_{k+1} = \xi_{k+1}, \dots, \mathbf{z}_K = \xi_K)$. That is, the first k processes are simulated while the remainder are held at fixed (vector) values. These values can be set to zero, which is equivalent to neglecting them, of course. Conditioning in this sense produces a simpler experiment, or one which is faster, or one whose results are easier to understand. In general, the experiment would have to be repeated for a set of the conditions. The unconditioning can be done by a subsequent simulation, or by analytical means, if enough is known about the statistical properties of the conditioning variables. This latter procedure holds the potential for significant run-time reduction. Of course, we can simply omit the unconditioning if the conditional experiment provides enough information of interest, which might be the case, for example, in a sensitivity study.

A different form of conditioning is to “condition” the experiment on a simplification of the system itself.[†] The simplification can consist of a reduced complexity description of an operation, or the complete omission of one or more operations. We can combine conditioning on the system and on the input processes so that, denoting the simplified system by g' , the simulation experiment is represented by

$$V_i = g'(\Omega')$$

The system g' achieves complexity reduction. It may also be that one form of conditioning implies the other. In the case of system simplification, the unconditioning could possibly be done by further processing, or, again, may not have to be done, depending on the simulation's objective. The conditioning viewpoint just described offers a useful way of visualizing simulation methodology.

2.3. Basic Concepts of Modeling

The notion of modeling is synonymous with the formal representation, in some fashion, of a waveform, or of a system and its various signal processing functions. In this sense, modeling is not unique to simulation. In fact, most communications texts are in effect about modeling. Conceptually, there is no difference between modeling for analysis purposes or for simulation. In practice, an important distinction tends to arise in the *complexity* of models. In analysis, simplified or idealized models are often used for tractability. In simulation, on the other hand, it is typically no more difficult, although computationally more expensive, to use a more complicated model, which presumably would, or could, be a more realistic representation of the phenomenon in question, i.e., incorporating real-world impairments.

Clearly, one's objective generally will be to use the most accurate model possible because it is self-evident that inadequate models will lead to results of little value. However, it is equally important to realize that any model, whether for analysis or simulation, entails some degree of approximation to the physical entity it is attempting to represent. The issue is not perfect versus imperfect models (although in certain instances it is possible to have nearly perfect models, particularly in the area of discrete operations), but rather one of adequate models, that is, models that are necessarily approximations, but yield results that have acceptable degrees of inaccuracy.

As implied above, there is a tradeoff between accuracy and computer run time. It is generally (but not always) in the nature of things that the more accurate the model is, the more detailed will be its description; this, in turn, implies a larger number of instructions in order to invoke the model. It should be realized that the notion of modeling accuracy is not restricted to individual “boxes” or functions, but exists at every level of description of a system. A *system*, of course, is itself not uniquely defined: it is some entity of interest at any particular time. That entity is usually depicted graphically by an interconnected set of “subsystems” and referred to as a *block diagram*. Conceptually, each block (subsystem) in that original system can itself be considered a system and illuminated in terms of its own block diagram. This process can be extended until blocks can no longer be reduced to subblocks that are meaningful in this context.

The consequence of the block diagram expansion just described is that a complete description of any system can be visualized as a tree diagram, with succeeding branches

[†]Conditioning is not a term usually applied here, but it is apt in this context.

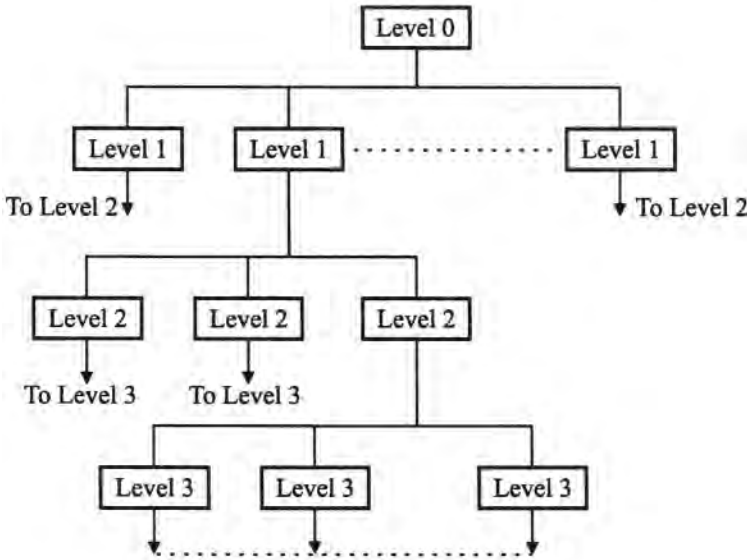


Figure 2.1. Hierarchical structure of communication systems.

representing ever-increasing levels of detail, as illustrated in Figure 2.1. This structure is also referred to as *hierarchical* in the software context, and is particularly well suited for software implementation and for managing complexity. For our purposes, we shall assume trees grow upside-down and refer to models near the root of the tree as *high-level* models, and models further down the tree as *low-level* models. These terms also have other connotations. A high-level model tends to have little or no reliance on the detailed physical modeling of its implementation. The terms *behavioral* or *phenomenological* are also sometimes used as synonyms. The classic high-level model is the transfer function of a filter, which relates the output to the input without explicit reference to the inner workings of the “box.” In contrast, a low-level model is one that to some extent has decomposed some higher level entity into constituent elements, which implies a description closer to the physical level.

The terms “high-level” and “low-level” are perhaps most meaningful in relation to each other. As a first illustration of the distinction, consider a linear filter made up of a number N of components. A low-level model, i.e., a *circuit model*, would implement Kirchhoff’s equations, representing each component by its appropriate differential equation. An input signal (voltage or current) would be transformed by this set of circuit elements, emerging at the output, bearing the effect of the structure as a whole. Of course, the filter’s effect is entirely predictable from knowledge of $H(f)$, its transfer function. Here, $H(f)$ is the high-level model, while the lower level model is the set of circuit equations.

Since each block of Figure 2.1 must have some number of instructions associated with it, it is clear that the more detailed the system description, the more computationally intensive the simulation will be. It should therefore be a standard part of good methodology to practice complexity reduction, that is, represent the system with the sparsest topology, or, in other words, the fewest blocks in Figure 2.1. If we apply this figure in terms of actual systems, the hierarchy might be interpreted as follows: Level 0 is the network; level 1 comprises the links; level 2 comprises the subsystems of each link; and so on. Our emphasis in this book is on the link level, also referred to as the physical layer. Thus, when we refer to a “system” here, we

shall usually mean a “link,” and such an entity by definition includes a transmitting portion, an intervening medium, and a receiving portion. In Chapters 8 and 9 we shall examine in detail the elements of a link and the considerations involved in modeling them.

Because a link is the entity of interest, the natural modeling elements are those one level down in the modeling tree, which we shall refer to as *subsystems*. We use this term somewhat informally because a subsystem, like a system, can have a wide variety of interpretations. Our use of it here implies only that whatever “boxes” might be used to depict the system’s block diagram constitute subsystems. Thus, a signal, whose existence would typically be captured by a box labeled “source,” would be a subsystem. A multipath channel, which would be made manifest by a box labeled “channel,” would also be a subsystem. In hardware, subsystems may contain different aggregations of functions, depending on a number of factors. Here again we shall interpret subsystem broadly. A packaged device such as a single filter may be a subsystem if it is so designated in the block diagram. Or a subsystem may be an assembly, such as a tracking loop. The common point in subsystem modeling is that the modeling paradigm consists of a “transfer function,” which we can interpret here as some rule for computing the current output value of the box, given its input values for some time span.

The preceding discussion is illustrated in Figure 2.2, which gives a more concrete example of the hierarchy of Figure 2.1. Here, we can see more clearly that complexity exists

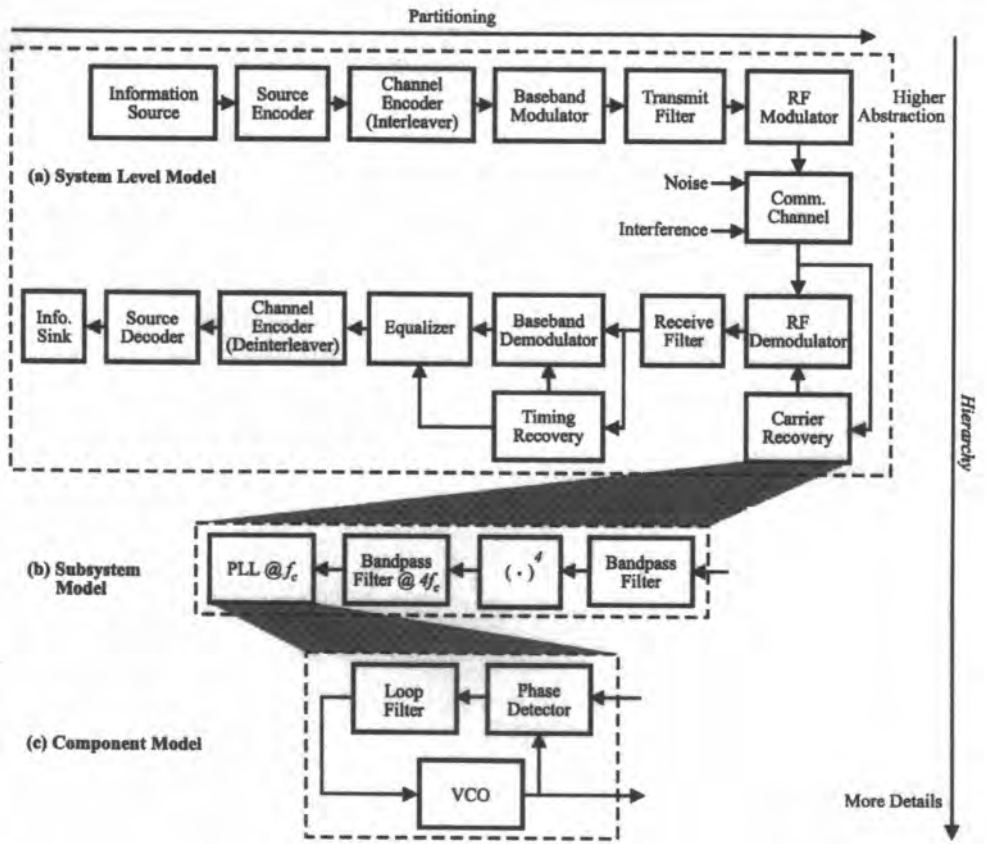


Figure 2.2. A particular illustration of hierarchical modeling.

in two “dimensions,” hierarchically (the vertical dimension), and at any level of the hierarchy (the horizontal dimension).

As implied above, for computational economy we want to model only at the subsystem level. In the case of linear elements, at least, there is no cost in fidelity in doing so. In the filter example above, both the high-level (transfer function) model and the low-level (circuit equations) model would lead to the same results for predicting the filter’s effect on a signal. For this purpose it is clearly preferable to use the high-level model. Do we ever need to use a low-level model, that is, explicitly represent a subsystem by assembling models of its next lowest level representations? There are a number of possible situations when this may be the case. In particular, when nonlinear devices are involved, the highest level model, based on an input–output relationship, may not be adequately descriptive. Here, we may indeed have to use a lower level model based on the physical workings.

Another instance of when we might want to use a low-level model is when we wish to know the sensitivity of system performance with respect to variations in one or more of a subsystem’s building blocks. Consider again the filter example above. The equivalence of the high- and low-level models is based on knowledge of the filter’s N component values. In actuality, each of these components is chosen randomly from a population having some distribution. Consequently, $H(f)$ is itself a random function, about which we may want to know, for example, the 95th percentile of the 3-dB bandwidth. Such information would be used to synthesize a transfer function for use at the subsystem level. However, the distribution of $H(f)$ would have to be obtained from the low-level model, for example by Monte Carlo simulation of the low-level model *by itself*. This illustrates a methodological point of general value, namely, that we may wish to simulate a single communication system element by itself, using a low-level model, and “float” (or “export”) the resulting information to the subsystem level, where we would use only a higher level model of that element.

In addition to simplifying the system-level simulation, there is another reason for wanting to deal with a high-level model, and this is to simplify the validation aspects and its related measurements. Eventually, it will be necessary to make such measurements to validate a simulation and to corroborate models. As can be appreciated from the filter example, a low-level model has many more associated parameters. In order to verify the accuracy of a model, it is necessary to measure the values of the parameters so they can be inserted into the simulation. A low-level model may require many such measurements, and, even more vexing, some of these may be on elements that are not directly accessible. Thus, supporting measurements for a low-level model may be costly and time-consuming.

As a general methodological guideline with respect to modeling level, we can therefore say that we should go to the lowest level of detail necessary to satisfy a desired degree of fidelity, but no lower. Equivalently, we strive to represent the system at the highest level of the hierarchy at which the input–output rule at that level is a good enough approximation to the behavior that would be observed if the input–output rules at the lowest levels of representation had been exercised.

For present purposes it is useful to subdivide modeling constructs into three categories which arise from the observation that the basic purpose of a communication system is to process waveforms (signal plus unavoidable noise or interference) such that the signal part is recovered as best as possible at a receiver. Thus, we emulate a system by generating waveforms at their point of origin and transforming them successively by the boxes (which we shall call devices here) they encounter. To the extent that the simulation block diagram is a faithful representation of the actual system (*as-is*), that the generated waveforms are close to the actual ones in their properties, and that the device models act as the actual devices, this is

the extent to which we have a good simulation. Thus, in the following we will find it convenient to speak of the system model, device models, and process models. These modeling constructs will be discussed in the next three subsections. The idea of a model has embedded into it the notion of accuracy: How *good* is the model? But it is useful to discuss this latter aspect separately. Consequently, the question of modeling errors will be taken up in Section 2.5.

2.3.1. System Modeling

As described above, a *system* here is a communications link, which, at the highest level of description, is represented by a block diagram of subsystems. The system modeling issue is a topological one in the sense that the closer the simulation block diagram is to *as-is*, the more accurate the system model is. In the preceding discussion, we emphasized the need to use the highest level model possible, for computational economy. However, at any level of the tree, it is also possible to reduce modeling complexity by using only a subset of the blocks at that level. This is the form of complexity reduction which is commonly used at the system modeling level, meaning that some of the subsystems may be completely omitted from the simulation, or perhaps represented in simplified fashion. Probably the best illustration of this idea concerns the synchronization functions. It is common to omit explicit simulation of these functions, while perhaps investigating other tradeoffs. This is similar to many analyses where synchronization is simply assumed to be present. In the simulation context, one cannot quite “assume” synchronization is magically present, but the closest thing to it (which we call “hardwired” and discuss in Chapter 8) is essentially equivalent to omitting the corresponding blocks from the simulation. Another type of subsystem which is typically not simulated is one which converts an analog signal to a digital signal (A/D converter). Instead, in such systems, it is usual simply to assume a discrete source.

Thus, generally, it is desirable to simulate the most reduced block diagram possible from the point of view of computation. For some purposes, such reduction may be quite acceptable, e.g., when doing some sensitivity studies on other parts of the system. Otherwise, some degree of approximation is inevitable. Further discussion will follow in Section 2.6.3.

2.3.2. Device Modeling

As previously implied, a *device* here is simply a block at the subsystem level which contains whatever the system designer wishes. The term further implies a piece of manufactured equipment, which can also be simply cabling, waveguide runs, or other manufactured channel media.

From the computational standpoint, the ideal device model is describable solely at the subsystem level of the modeling tree. This implies a “transfer function” model, by which we mean, as stated above, a rule for producing at each instance of the simulation clock[†] an output value based only on some set of input values. The rule in question is any appropriate mode of description, for example, a single equation, a set of equations, an algorithm, or a table lookup.

How does one arrive at a good rule? A logical starting point is a description of the subsystem function if it were ideal. But this description should then be enlarged to accommodate departures from ideal behavior in ways that are meaningful and physically realizable.

[†]In a multirate simulation, different devices may have different clocks.

Thus, a good subsystem model should have variable input parameters (sometimes called “knobs”) that can be set to reflect the actual behavior of devices. Depending upon how much is known about the device in question, the “actual” behavior may itself be only a projection or it can be based on measurements.

There are other nuances to the idea of modeling complexity besides the notion of modeling level. Even at a given level of the tree, there are sometimes more or less complicated ways to describe the functioning of a box. A case in point concerns again the synchronization functions. There are several possible models, all of which imitate the functions of synchronization at some level of abstraction, and all of which can be considered to be at the same level of the modeling hierarchy. These possibilities will be described in Chapter 8, which covers subsystem models in general.

2.3.3. Random Process Modeling

It is self-evident that the inputs and outputs of systems and subsystems are desired (information) and undesired (noise and interference) *random processes*, and that the basic goal of any simulation is to compute some measure of the goodness of transmission of the desired signal. It would then seem also fairly obvious that the fidelity of this computation is partly dependent on the degree to which the simulated processes reproduce the properties of the actual processes. The errors involved in doing this are discussed later. The modeling task, in general, is to “imitate” a random process at its point of origin because, if we have good device models, these will properly operate on the input sequence to produce a reasonably correct output sequence. The imitation in question is produced by what is called a *random number generator* (RNG). A good RNG emits a sequence of numbers that ostensibly forms a sampled version[†] of a segment of a sample path of the random process at hand. A substantial part of Chapter 7 is devoted to the theory and techniques for constructing random number generators.

Even though information sources and noise sources are both random processes in the operational setting, in the design and testing of systems the signal sources that are often used or assumed are test signals, which are usually deterministic. For example, a test signal may be a sinusoid (“test tone”) or a structured digital sequence generated by a shift register with particular connections. These latter sequences are often called *PN sequences*. In the case when such deterministic test signals would be used, there are no significant modeling issues.

There is another type of random process that we may need to model, which does not represent a noise or information process, and this is a “random” channel such as a multipath channel. Here, what is normally required to be modeled is the channel’s impulse response $h(\tau; t)$, which is typically assumed to be randomly time-varying. There are many issues related to the modeling of time-variant random channels, and these are discussed in detail in Chapter 9.

There is another modeling construct, referred to as an *equivalent random process* (erp) model, which is also not a source process. The idea is as follows. Suppose that some random process, say $x(t)$, is input to a cascade of n subsystems, the output of which is the process $y(t)$. If by some means we are able to deduce what kind of random process $y(t)$ is, then, clearly, for whatever subsequent processing takes place, all we need to do is to generate a random

[†]Although it may previously have been implied, we should now make explicit the fact that a simulation can only operate in discrete time, or, equivalently, that only discretely spaced samples of waveforms can be processed. Thus, a simulation can imitate a (continuous) random process only as a sequence. This subject will be discussed at length in the next chapter and in Chapter 6.

sequence imitating $y(t)$. If we can in fact synthesize a good RNG for $y(t)$, we will save the computation that would be required to process $x(t)$ through the cascade of subsystems leading to $y(t)$. Thus, an erp model can lead to substantial economy. We will illustrate its application to phase noise in Chapters 8 and 12.

2.3.4. Modeling Hypothetical Systems

In the preceding discussion, it was implied that the entities to be modeled were well defined. In fact, that is often not the case. Probably most simulation activity is devoted toward helping to design a system, i.e., the technical particulars of the system are not known initially, and only become better defined gradually. We refer to a system to be designed as a *hypothetical* system. Of course, it is possible for such a system to simply *assume* a set of well-defined characteristics; in that case, the methodology is not affected by the hypothetical aspect.

As just implied, one of the more useful aspects of simulation is the possibility to estimate realistically the performance of a system before it is actually built. In fact, what we want is to be able to guide the hardware development so that a specified performance does indeed materialize. The key to this process is the realization that the influence of virtually any piece of equipment can generally be dictated by just a few well-chosen descriptors or “major” parameters. Although the behavior of a device may vary arbitrarily in its fine structure, even after the aforementioned parameters are set, these fine variations should ideally induce relatively little change in performance because the major parameters should have been chosen to “anchor” the device’s essential behavior. Therefore, if we adopt these parameters as control vehicles for the hardware design and subject them to specification, a hypothetical system can be synthesized that adheres to these specifications, and we can expect the performance of the realized system to be reasonably well approximated, or bounded, by that of the software model of the hypothetical system. As pieces of the system are built, their characteristics can be measured in the laboratory and we can substitute these measurements in the simulation, which will then render a performance estimate presumably closer to “reality.” What has just been discussed is actually part of a larger context, namely, the methodology of communication systems engineering, which will be taken up in Section 2.5, from the standpoint of how simulation fits in as part of that process. Here, we shall say a few more words about modeling approaches for hypothetical systems.

Recall that as a general methodological objective, we wish to reduce complexity as much as possible in the simulation model of a system. A hypothetical digital communication system can be represented in the least complex way simply by assuming the smallest number of subsystems that would yield a reasonable system. We shall refer to the least complex system as a canonical system. Such a system would consist of a data source, a modulator, a transmit filter, a transmitter amplifier, a channel (medium), a receive filter, and a demodulator/detector. If the real system departs from the canonical form, it might still be straightforward to do an adjustment after the fact. For example, the effect of coding can be estimated from the parameters of the given code, as will be demonstrated in Chapter 11. The blocks of the hypothetical system might be described in the following way.

For the data source, with one or more bit streams, both the statistical nature and the waveform nature have to be described. Without any other information about the source properties, it is usual to assume that a binary source is either random or is structured to emulate randomness, such as a shift-register (pseudonoise, PN) sequence. Both types of sequence are easy to generate, as will be described in Chapter 7. The waveform associated

with such sequences is typically assumed to be rectangular, but can be easily adjusted to reflect specifications on real waveforms, such as finite rise time and fall time. The association of waveforms to symbols is discussed in more detail in Chapter 8.

For the modulator, a concise way of specifying it would be by setting limitations on the distortions produced in the signal space constellation, for example, by specifying a region about the ideal coordinate within which the actual phasor could fall. Alternative, or possibly complementary, specifications for a quadrature amplitude modulated (QAM) signal could be the setting of bounds on the gain imbalance (the ratio of in-phase, I, to quadrature, Q, channel amplitude) and on phase imbalance (the nonorthogonality between I and Q channels). The point here is to attempt to achieve realism by a relatively high level of abstraction, which tends to satisfy our goal of complexity reduction. Further discussion on modulator modeling will be found in Chapter 8.

Filters are specified by imposing specifications on the amplitude and phase characteristics separately. For a hypothetical system, the continuous function of frequency $H(f) = A(f) \exp[j\phi(f)]$ is assumed not to be known.

As indicated above, if certain key properties are properly specified, the unmodeled fine structure should have little additional effect. For the amplitude $A(f)$, one first needs some sense of the bandwidth, and such parameters as the -3 dB, the -10 dB, and perhaps the -20 dB bandwidths are probably sufficient. Since the transfer function is not initially known as a continuous function of frequency, its values at certain increments of frequency would be calculated by some kind of interpolation rule. In addition to bandwidth parameters, one would typically impose specifications on some of the major manifestations of imperfection. For example, one could specify the “tilt” (the mean slope in decibels) across some portion of the passband, as well as some measure of “ripple” across the same band. The “band” in question could be, for example, some fraction of the data rate. The phase characteristic would be specified by controlling the departure from ideal. There are different ways of doing this. One way would be to conceive of the phase as describable in terms of a polynomial of a certain degree plus some residual, or ripple. The synthesis of a transfer function along these lines makes use of elementary “functional” filters, as described in Chapter 8. An alternative to specifying transfer functions which is also popular is to define a “mask” for the amplitude and phase or group delay, such a mask consisting of two curves within which the characteristic in question must fall. Evidently, there exist a limitless number of characteristics that can fall within a mask. Whatever way one chooses to synthesize a particular $H(f)$, there are different options for actually simulating it, and these are fully discussed in Chapter 4.

Amplifiers are typically specified by two characteristics, the output power versus input power curve (or AM/AM characteristic) and the output phase versus input power curve (or AM/PM characteristic). Such curves can be scaled versions of measured characteristics for the type of amplifier being contemplated. What is normally specified about these characteristics is the “compression” of the AM/AM characteristic, which is the change in output power relative to the change in input power (usually in decibel units) around the saturation point; and the maximum slope of the AM/PM characteristic, in degrees per decibel. The preceding description applies to “wideband” amplifiers, namely those with no significant frequency selectivity across the signal bandwidth. If this is not the case, the model becomes much more complex, and in fact the issue is not settled with respect to what is a good high-level model; indeed, in this case a lower level model may be necessary. Amplifier modeling is discussed in Chapter 5.

The channel, or medium, must of course reflect the reality of the application. For a satellite system, for example, the channel can often be assumed ideal. For a fiber optic

channel, one should use a transfer characteristic representative of that to be encountered. For a multipath channel, the particulars of that particular channel have to be reproduced (e.g., delay profile). For such a channel, there are further options relative to the manner of simulating the channel which depend upon its time rate of change. If the fading is sufficiently slow, the channel need not be explicitly simulated as a dynamical entity. This will be further discussed in Section 2.4.

The receiver can be modeled by an approximate matched filter (which may or may not be actually specified as to type or parameter values), and by specifications on maximum phase and timing bias errors as well as root mean square jitters for each. This will satisfactorily control the performance of a hypothetical receiver.

As might now be seen, the specifications outlined above merely limit the departure of each device or subsystem from its ideal behavior. The important point is that the specifications be relatively simple in nature, which makes the associated models relatively simple to implement in software, and at the same time contain the major influences on the system performance.

2.3.5. Simulation with Hardware in the Loop

Instead of an abstract model for a piece of hardware, one can in principle conceive of the possibility of using the *actual* piece of hardware in the simulation. This arrangement, of course, stretches the definition of simulation as used so far since the model *is* the device. This situation is sometimes referred to as “hardware in the loop,” or hybrid simulation. The idea is especially attractive when a device or subsystem is particularly difficult to model. There is, of course, a potential difficulty to be dealt with, namely the simulation/hardware interface.

If the hardware is analog, such as a nonlinear amplifier, the interface may in fact be rather difficult to realize. Two complications are immediately apparent: the simulation samples will have to be fed to a digital-to-analog converter, and these samples, normally of the complex envelope, will have to be modulated (upconverted) to the carrier frequency of the device. However, the most likely problem will be a severe incompatibility between the real-time speed of the simulation and the bandwidth of the device. Hence, hardware in the loop is more likely to be achievable when the actual device does digital signal processing of one kind or another.

One example of hardware in the loop applies to the simulation of “baseband” portions of communication systems where the signal bandwidths are of the order of 1 MHz and the processing of baseband signals is performed using digital signal processors of one kind or another. This approach of including the actual signal processing hardware as part of hybrid simulations is especially attractive in the design and simulation of receivers for standards-based communication systems such as GSM or IS-95. In such systems the transmitted signal is well defined by the standards and it is easy to simulate (i.e., generate) the transmitted signal and simulate its transmission through baseband versions of the channel, the models of which are also specified in the standards documents. (Some standard or “reference” channels for performance evaluation purposes are given in Chapter 9.) It is easy to introduce a variety of transmission impairments such as multipath, fading, noise, and interference in a controlled manner and “play” the impaired signal through the receiver algorithms implemented on a digital processor.

During the early phases of design, the receiver algorithms will be executed on a workstation in floating-point arithmetic in a “simulation” mode. Many of the commercial simulation environments have the capability to convert the simulation code into (assembly

language or C) code optimized for a particular processor on which the receiver algorithms might be implemented. This code is then executed in simulation mode on a “simulation model” of the processor which is accurate at the clock cycle level. After the design is verified in the simulation mode on the simulated processor within a workstation, the code is “downloaded” to an actual processor on an interface board connected to the workstation and a hybrid simulation is carried out using simulated versions of the received signal and the actual receiver algorithms running on an actual processor.

In this hybrid approach, the boundary between simulation and implementation becomes somewhat blurred. Also, sometimes a DSP processor attached to a workstation is used for the sole purpose of speeding up simulations since special-purpose DSP processors can execute some parts of the simulation code faster than a general-purpose workstation or personal computer (PC).

2.4. Performance Evaluation Techniques

We mentioned earlier that there are two major aspects of methodology, one being modeling, and the other the subject of this section. Modeling, of course, deals with the representation of devices, subsystems, and processes for the eventual objective of obtaining an estimate of some system-level performance measure. The manner of obtaining this estimate forms the subject matter that we call *performance evaluation techniques*, which contains a number of methodological issues. Actually, as might be anticipated, there is not a complete separation between performance evaluation and modeling.

A performance evaluation technique (PET) is a set of analytical tools and assumptions applied together within a simulation software package for the purpose of efficiently estimating some performance measure. If there is any need at all to speak of PETs as a subject, it is because of the fact mentioned earlier, that the basic commodity that is consumed in simulation is computer run time, and that for many problems of interest this run time can become prohibitively long if we use the Monte Carlo method. Thus, PETs generally imply alternatives or modifications of the MC method. There are few PETs that can be applied generally. That is, they usually have to be designed on a case-by-case basis, and typically involve (1) assumptions or simplifications on the structure or properties of the system, (2) assumptions on the statistical properties of waveforms, and (3) clever statistical techniques. Item 1, in particular, implies that a performance evaluation technique may be intertwined with a modeling construct. It might be anticipated, then, that certain PETs imply or demand departure from an *as-is* model.

At the simplest level, we can consider a MC simulation of a reduced-complexity system to be a PET. It satisfies our efficiency objective to the extent that the computations associated with missing or simplified operations reduce the overall computational burden. As discussed above, reduced complexity can simply be a synonym for the simplifying assumptions often appealed to in analysis, such as the invocation of known timing. As will be seen below, the simplification or elimination of some functions are either necessary for, or facilitate, the application of other run-time-reducing techniques.

In certain instances, run-time reduction may be less connected with the degree of modeling detail than with statistical considerations. The measurand of a Monte Carlo simulation (e.g., SNR or BER) is a random variable. The longer we run the simulation, the closer the observation tends to the true value. There is thus the tradeoff mentioned earlier

between run time and accuracy of the measurement. From the run-time standpoint, the most demanding situation usually is the estimation of $\text{BER}^\dagger$ in digital systems. The reason is not hard to see. If a system is intended to deliver a BER of, say, 10^{-5} , then we expect to see one error about every 10^5 bits. In order to be convinced that the BER is indeed about 10^{-5} , it would not be sufficient to observe just a single error in the first 10^5 bits. If we observed many blocks of 10^5 bits and the number of errors were close to that number of blocks, we would have an increasing sense of reliability in the empirical estimate $\hat{p} = \text{number of errors}/\text{number of bits}$. These intuitive ideas are quantified in Chapter 11, where it is shown that a number of bits in the range of $10/p$ to $100/p$, where p is the true BER, will result in a reasonable uncertainty in the estimate. For $p = 10^{-5}$, $10/p$ corresponds to 10^6 bits. In virtually any actual system, the real time corresponding to that number of bits is small if not trivial. In simulation, it is the computational time per bit that matters. This, of course, will depend on the simulation model and on the hardware platform. But, as will be more fully discussed in Chapter 11, p in the range 10^{-5} – 10^{-4} does more or less represent the dividing line between practical or impractical run time. Thus, for small enough p , appropriate PETs are mandatory for BER estimation.

An illustration of a widely used PET in BER estimation occurs when a system is linear. The property of linearity allows us to leverage some related analytical knowledge against run time. In the linear system, Gaussian noise is known to remain Gaussian. This knowledge permits us to combine analytical expressions with *noiseless* simulation (which simply captures distortion) to yield BER estimates very efficiently, i.e., orders of magnitude more rapidly than with MC. The combination of analytical knowledge and simulation techniques will be called a *quasianalytical* (QA) technique. It turns out that in order to apply QA in the best way, it is also necessary to assume synchronization has been established; that is, synchronization is not explicitly carried out. We can think of a such a QA simulation as providing a *conditional* result, conditioned, for example, on a fixed sampling epoch. The main point of this discussion is that for an efficient QA simulation we can use a system property (linearity) and also simplify the system model itself. A number of variations of the QA approach will be developed in Chapter 11. Other classes of run-time-reducing techniques based on statistical ideas will also be discussed there.

Another example of a PET occurs when a system's performance depends upon two random processes with widely different rates of change in time (equivalently, very different bandwidths). Let us call one a “fast” process and the other a “slow” process. The fast process may be thermal noise and the slow process fading. If the latter is sufficiently slower, then it can be approximated as being in a fixed state for a significantly long interval with respect to the information process. The evolution of a received waveform, then, can be visualized as a sequence of tune segments over each of which the state of the slow process is different, but fixed. Then, a simulation can be carried out by emulating such a sequence of segments; each segment can be visualized as being a conditional experiment (conditioned on the state of the slow process) and the metric produced as being a conditional one. The final performance estimate is then the average over the conditional states. This methodology can be applied, for example, to high-data-rate systems over slowly fading channels.

Another simplification that underlies many PETs, including the QA technique just discussed, is the assumption that a system has finite memory. This assumption is also required for the effective application of importance sampling (IS), another run-time-reducing tech-

[†]We use BER here just for illustration. Any measure of error production when errors are relatively rare is computationally demanding via Monte Carlo simulation.

nique for BER estimation, as well as the use of fast Fourier transform (FFT) techniques. Since a strictly finite memory allows us to take into consideration only a finite stretch of input data to account for the system’s effect at any point in time, we can modify the simulation experiment accordingly. That is, instead of evolving the signal and noise waveforms continually (as happens in real life), a condition that we refer to as *sequential* or *stream* simulation, we can conduct a simulation by shifting into the system blocks of data whose length corresponds at least to the memory of the system. We refer to the latter as a *block* methodology.

In general, the applicable PET depends on many factors. An overview of the factors that should be taken into account is given pictorially in Figure 2.3.

2.5. Error Sources in Simulation

The usefulness of a simulation run is directly related to its accuracy. It is reasonable to interpret “accuracy” as the closeness of the simulation results to the actual behavior of the physical system being simulated. As sensible as this notion is, there are some related subtleties with respect to its application in practice. We will return to these in Sections 2.6 and 2.8. Accuracy is limited by three types of modeling error, corresponding to the modeling

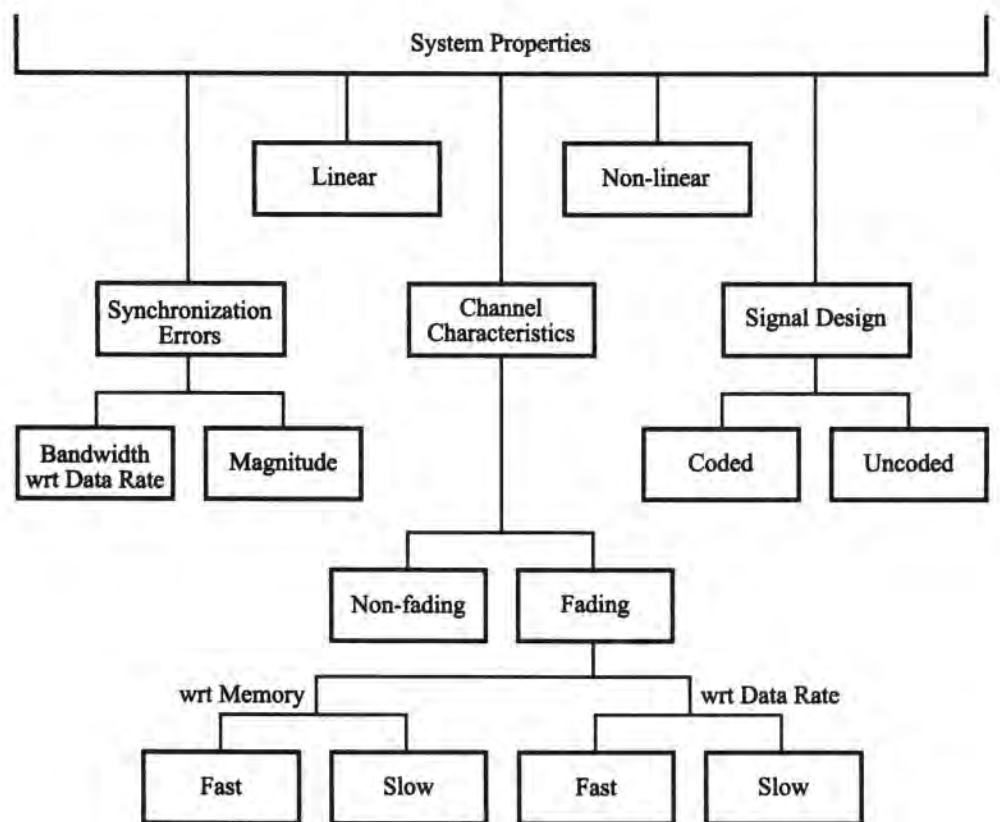


Figure 2.3. Classification of system properties for performance evaluation techniques.

categories defined earlier (system modeling, device modeling, and random process modeling) and by “processing” errors, which are due to computing limitations and the very nature of simulation. As examples of processing errors, discrete-time representation of continuous signals induces aliasing error; computer memory will not allow storage of infinite impulse responses; numerical representation will not allow unlimited accuracy; and run-time limitations will not allow zero statistical uncertainty. The errors arising from these sources, however, are controllable in the sense that in principle they can be reduced to arbitrarily small error, subject to some computational cost. Obviously, the modeling errors are of different natures and their reduction unrelated to computational cost except in the sense that a model which is known to be better than another one will often be more complicated and hence, as earlier discussed, contain more instructions. Figure 2.4 gives a pictorial summary of the error sources. In that figure, we have also added an error source called “approximations,” by which we mean the explicit use of formulas which are known to be approximate in the context of use; examples of this type of error would tend to occur in postprocessing. This idea will be clearer after reading Chapter 11.

In this section, the types of errors mentioned are reviewed and, in a few instances, numerical values in the performance measure are estimated. It is clear that such values can only be very rough since the effect of any given modeling error depends on the context. Obviously, modeling errors can be thought of as a subset of the subject of modeling in general. Our discussion of errors here is intended to be a little more focused on the way that communication systems are actually implemented than was the case in Section 2.3. The subject of errors is also clearly related to the fundamental notion of validation, and in effect, the domain of validation is to determine the impact of modeling errors on the simulation’s end product. However, we defer discussion of that topic to Section 2.6.

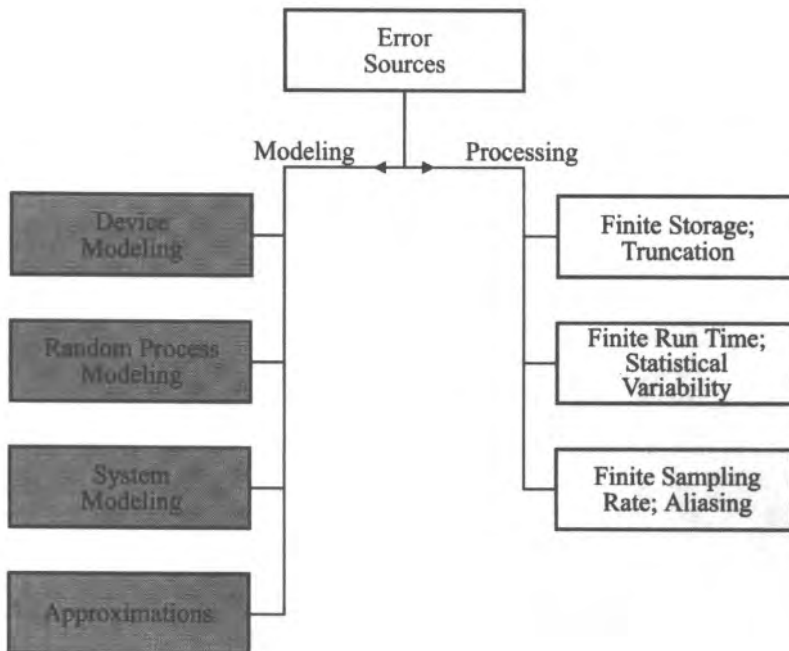


Figure 2.4. Sources of error in simulation.

2.5.1. Errors in System Modeling

If the simulation block diagram is not *as-is*, i.e., a one-to-one topological mapping of the actual system, then it is reasonable to expect that the simulation results will not correspond exactly to those of the actual system, although that is not always a necessary consequence. The reason that the simulation block diagram might not exactly mirror the actual system would be to reduce complexity. Recall from Section 2.3 that we defined a canonical or least complex digital system. It would generally be one's goal to reduce an actual block diagram to such a canonical form. In passing we may mention an obvious way to reduce the block diagram with no adverse impact on fidelity. This is the case when there is a cascade of linear elements, which clearly can be represented as a single filter. However, depending upon how we choose to simulate that filter, there may or may not be a corresponding reduction in computation.

Another situation where the topological match may not be exact without detracting significantly from accuracy is the removal from the simulation block diagram of elements that contribute little distortion. For example, one does not customarily model the antenna in a simulation because an antenna typically has very wide bandwidth relative to the signals that enter it. Similarly, in satellite applications the atmosphere is often ignored as a distorting element because its transfer characteristics are typically (but not always) transparent to the signal. There may be other elements in the actual block diagram, perhaps electromagnetic or electric couplers, low-level amplifiers, image-rejection filters, etc., which are not intended to shape the signal and which, in many cases, do have a negligible effect and can thus be simply omitted.

Assuming that we have reduced all cascades of linear elements to a set of single elements and that we have decided which elements can be omitted as essentially distortionless, the resulting block diagram would produce accurate results because it is essentially equivalent to the actual system. Any further complexity reduction will generate some degree of error.

In carrier-modulated systems, a potentially significant source of degradation is the phase noise associated with the oscillators used in up- or downconversion stages, the implementation of which, aside from the oscillator itself, consists of a mixer and a filter. If for the moment we assume the phase noise is negligible, it is safe to say that appropriately designed frequency conversion is very well approximated by ideal translation. Hence, all frequency conversion stages, as such, can be eliminated from the simulation block diagram. A somewhat less idealized version would recognize the possibility of generating spectral lines ("spurs"), and these, if significant, can be taken into account separately as additive interference, which is straightforward to simulate. Phase noise occurs wherever there is an originating carrier frequency source. That noise is modified as the carrier propagates, is distorted by various devices, and is finally "tracked" by some form of carrier-tracking loop. A form of complexity reduction which is an application of the equivalent process modeling methodology is to eliminate all explicit representation of carrier generation, frequency conversion, and tracking, and replace them all with an equivalent phase noise process at the receiver which embodies this effect. This replacement, though not perfect, appears to be well justified, as discussed in Chapter 8 and in greater detail in Case Study 2 in Chapter 12. From the point of view of performance evaluation, we estimate the error due to this simplified model to be on the order of a few tenths of a decibel, more or less, in the required E_b/N_0 (E_b = energy/bit, and N_0 = noise power spectral density) for a given BER.

Amplifiers in general should be left in place. Of course, if they are wideband and linear ("benign" as far as the signal is concerned), they can be removed. If they exhibit some nonlinear behavior, it is best to retain them as individual blocks in the simulation. Some

exceptions to this general rule may occur if we have cascaded amplifiers. This situation occurs fairly often where usually the first amplifier(s) is at a relatively low signal level and the last, a “high-power” amplifier, usually accounts for most of the nonlinear behavior. It is reasonable, therefore, to replace such a cascade of amplifiers by a single equivalent amplifier whose AM/AM and AM/PM characteristics would be measured by an “end-to-end” measurement, i.e., one spanning the cascade. Note that it is implicit here that these amplifiers have no “memory,” i.e., no frequency selectivity. If amplifiers do have memory,[†] it is not easy to say whether such a replacement would be permissible. While it is possible to make such a replacement *formally*, as described in Chapter 5, the magnitude of the error is difficult to predict.

In summary, the reduction of an actual block diagram to a simpler one for simulation purposes may lead to some error in the observed results. If this reduction is carried out according to the ideas discussed, however, the error should not exceed a few tenths of a dB, more or less, in the SNR or E_b/N_0 scale.

2.5.2. Errors in Device Modeling

It is obvious that *some* error will be produced to the extent that the model for a device is not *exactly* like the device itself. Is an exact match ever possible? There is both a philosophical as well as a practical component to this question. At the most fundamental level, we cannot expect a model to have perfect fidelity to its physical counterpart, but we can have (almost) perfect fidelity to an abstract (usually) idealized model of the physical device. In the case of discrete operations, assuming the device is working properly, it is in fact possible to have a perfect model. Ultimately, however, as in analysis, the practical goal in simulation is to develop models that are “good enough,” i.e., that produce errors, in the final results that are acceptably small for the application. It is normally possible to develop such models.

To illustrate some of these ideas, consider the modeling of a filter, say a Five-pole Chebyshev filter, hereafter referred to as a 5PC filter. The abstract mathematical model for an ideal (lossless) 5PC filter is a known rational function in $s = \sigma + j2\pi f$. Thus, if we want to simulate an ideal 5PC filter, we do have an exact model. However, while the *abstract model* is exact, the simulation implementation of it cannot be because this continuous-time structure must be approximated by a discrete-time *simulation model*. A good deal of the next chapter is devoted to this modeling issue. The error in this approximation is completely of the processing nature mentioned above. Let us assume for the remainder of this discussion that the simulation parameters have been chosen to make this processing error acceptably small.

Suppose now that we want to simulate a “real” 5PC filter, which may already be physically implemented or is yet to be built. Because no implementation can be perfect, the abstract model must be in error to some extent. The simulation model can be made closer to reality if it is known how close to ideal the design actually is. There are standard filter design parameters that indicate such closeness; in the case at hand, the “ Q factor” is such an indicator ($Q = \infty$ is ideal). Thus, if we knew the Q factor, we could calculate the corresponding response and, again, we would have an exact abstract model. Since the realized filter will again not behave exactly like the paper design, we can go still one step further toward verisimilitude by measuring the transfer function of the physical filter and using these measured characteristics as the simulation model. We would then have once more a seemingly perfect abstract model. But this “perfection” is a bit of an illusion.

[†]Of course, any amplifier must have some degree of frequency selectivity. The current discussion merely distinguishes the cases when it can be considered negligible and when it cannot.

If we observed the real and simulated filter outputs, we would generally see some difference. One reason is that the physical measurement itself is imperfect. Thus, the physically measured characteristics are not necessarily the same as the true characteristics. Another reason is that the filter characteristics are not immutable; that is, once measured, it does not mean that another measurement at another time, even if perfect, would yield the same observation. This is because components age and respond differently to different environmental conditions, such as temperature. The time-variable aspect is one reason systems are controlled by specifications (we shall return to this point later). For now, assume that we are dealing with a “snapshot,” that is, the device behavior at a point in time, and that the snapshot remains unchanged. In this case, using measured characteristics is as close as we can come to an errorless model.

From the foregoing it can be seen that, for filters at least, it is possible to have a simulation model for a device that is very close to some version of reality. If the model is not exactly like reality, in what way does it differ and what effect does this difference have on the accuracy of the simulated performance? To answer this question, it is necessary to begin with some abstract and convenient mode of description of the filter. We have previously alluded to such a mode as either a mask or a polynomial decomposition. The latter is more convenient for this discussion. For example, let us assume that the amplitude or phase is described by a degree- n least squares polynomial fit; for illustration take $n = 3$, so that, say, $H(f) = a_0 + a_1f + a_2f^2 + a_3f^3 + r(f)$ for $f \in B$, where H could be amplitude or phase, $r(f)$ is the residual (also called ripple), and B is the frequency range of the fit. Thus, the closeness between an actual filter characteristic and its model can be visualized as the closeness between the corresponding polynomial coefficients and the residuals. If the polynomial coefficients are identical, and the peak-to-peak values r_{pp} of the residuals are the same, we can say that the two descriptions differ only in the “fine” structure, this term being really appropriate only if r_{pp} is relatively small. If this is the case, then a simulation model using nearly the same $\{a_i\}$ and $r(f)$ as the actual filter will produce nearly the same effect. Errors in the polynomial coefficients of 10% or less should induce response errors of similar magnitude. It is normally possible to know these coefficients to at least this accuracy. For this degree of coefficient accuracy and for equal, but relatively small r_{pp} , we estimate the simulated performance measure to be in error by no more than plus or minus a few tenths of a decibel.

The discussion on filters illustrates a general point. It is not necessary to reproduce a device’s behavior in the finest detail in order to have a good model. As was already said in conjunction with the construction of models, it is only necessary to reproduce the major features of its distortion-producing effects. For passive devices, as we have seen, this can usually be done easily, and in fact, this is also true for most active devices. The major potential difficulty lies in the modeling of nonlinear amplifiers with memory. We have mentioned in Section 2.2.4 some possible abbreviated but essential descriptors of device behavior for the blocks of the canonical digital system. If these descriptors are given accurate values, we can again expect errors in the output of the simulation to be only on the order of a few tenths of a decibel for any one such block. It is very difficult, however, to estimate the cumulative effect of modeling errors in a sequence of blocks. That is why validation itself must be conducted at different modeling levels, as will be discussed later.

2.5.3 Errors in Random Process Modeling

It is again self-evident to say that another potential source of error lies in the degree to which models of random processes do not behave exactly like the actual processes. The processes of interest are signal sources and noise sources. As in the previous section, it is

necessary to make a distinction between the closeness of an assumed abstract model to the actual process, and our ability to imitate the adopted abstract model. The first consideration is particularly important when comparing simulation results to experimental ones. Here, however, we will assume that the abstract model is correct, and deal only with the question of implementing it as best we can.

Earlier we mentioned that the emulation of a random process is implemented with a random number generator which produces a sequence of numbers intended to behave like a sample path. In principle, it does not matter whether the process is signal or noise, but evidently, the RNG should behave in some sense like the process in question. Thus, with respect to analog signal sources, like speech, an ideal source RNG would “look” like a speech sample path. Random number generators are discussed in detail in Chapter 7, where it will be seen that the known techniques for RNGs generally have only limited ability to reproduce all the properties of random processes. In fact, as will be seen, it seems possible only to imitate two properties of a process, namely, the first-order voltage probability density and the power spectral density. These may sufficiently capture the essential properties of a process for many purposes. But, in general, the only way to ensure that the input sequence is a faithful replica of an actual input waveform is to construct an input file that is a sampled version of an experimental record. Indeed, this is a viable approach for some purposes. For example, in assessing speech processing techniques, there are certain stock words or phrases that are used in subjective testing. A waveform segment which consists of a set of test sentences or words can form a practical input source for repeated digital processing alternatives. Generally, though, such a fixed input file is of limited use.

As was mentioned earlier, if we use a deterministic test signal, there is effectively no representation error. In some analog applications, noise sources are also used as test signals. In such cases, the same issues arise as with imitating any other random process.

Source models for digital systems are discrete alphabets which may have memory. Again, we have to distinguish between the two cases where we intend to simulate the system driven by a test signal or by an actual source. In the first instance, the usual test sequence is a maximum-length (PN) sequence and this can, of course, be generated exactly in the computer. Furthermore, the performance estimate will be essentially error-free if we use a de Bruijn sequence with length q^m , where q is the number of symbols in the alphabet and m is the memory of the system, and if the symbols are independently generated. If the true source does not emit symbols independently, modeling that source with an independent symbol generator can result in some error because the intersymbol interference patterns will not be uniformly distributed. The error can be lesser or greater, depending upon which patterns are more frequent. If the source is known to have dependencies, then it should be modeled to reflect its actual properties. These dependencies are usually modeled as finite-state machines, and if that model is known, it is straightforward to incorporate it in a simulation.

As to noise sources, in the communication context we are primarily interested in three types: thermal noise, phase noise, and impulsive noise. The same ideas discussed under analog sources apply here; namely, to use files of experimental data or (the usual approach) use an RNG and rely on the principle enunciated earlier that if we can properly reproduce just a few significant properties or parameters, we would presumably induce only minor errors.

Thermal noise is typically modeled as white Gaussian noise (WGN) at its input point. It is straightforward to simulate such a noise source over the largest bandwidth defined in the simulation (which we will refer to as the *simulation bandwidth*, and is equal to the reciprocal of the sampling interval), in such a way that successive noise generator samples are independent. We will refer to such noise as bandlimited Gaussian noise (BLGN), and there exist

BLGN generators that are quite good imitators of WGN for most applications. In other words, we can consider simulation of thermal noise to be essentially error-free.

Phase noise, as a direct manifestation of oscillator instability, is a process which is incompletely understood. Hence, an explicit emulation of oscillator instability does not appear known. However, because the eventual impact of oscillator instability is also a function of tracking structures that are present, at least for coherent systems, the net, or “equivalent,” process at the output is relatively easier to deal with, and more justifiably modeled as Gaussian. If that is the case, then it suffices to know the root mean square (rms) value of that process and some descriptor of its spectral distribution in order to synthesize a good random noise generator for this purpose. Since timing jitter stems from the same kinds of causes, similar statements apply.

Impulsive noise (or shot noise) is an important class that includes noise due to man-made activity and noise in optical detectors. The impulsive noise model described in Chapter 7 is quite general, and with proper setting of the parameters can be made to fit closely many impulsive processes of interest. In any case, if one can describe the physical process accurately, it is usually possible to construct an ad hoc random number generator that is a good emulator. This is the case for optical detector noise.

How do we know how good a random number generator is? We would like the time series that unfolds from the RNG to “look” like that of the process we are emulating. In fact, there are many measures of closeness that are more or less standard statistical tests. Well-designed algorithms will (or should) pass many or most of these tests. Typically, however, these tests can only be applied to relatively short sequences without becoming computationally burdensome. Furthermore, the goodness of an RNG as determined by these tests does not provide quantitative means of evaluating the resulting error in a simulated performance measure. The test of last resort, therefore, may be the actual simulation run. As an illustration of this point, the properties of a particular RNG were empirically studied in this fashion (Ref. 3). A large number of cases was run, with different seeds; the results are shown in Figure 2.5. The figure shows five curves of BER versus number of bits, obtained by simulating a two-hop satellite system having several decibels of degradation. The two uppermost curves show the two worst BER cases observed, while the two lowermost curves show the two best BER cases observed. The middle (nearly constant) curve corresponds to a seed yielding a BER that is very close to an average of all 78 cases run. Superimposed on this figure are the upper and lower bounds on the 90% confidence interval constructed around this average curve, which we can presume is close to the true value (see Chapter 11 on how this interval is obtained). It can be seen that there is excellent agreement between the empirical bounds and the analytical bounds, the latter depending only on the assumption that the errors are independently generated. By themselves, these results do not prove that the sequences produced by this noise generator are good replicas of actual sampled paths, but the results do show the expected type of convergence with increased observation time that would occur with real processes. A more direct proof of Gaussian properties is available by simulating an idealized digital system whose theoretical BER, $\frac{1}{2} \operatorname{erfc} [(E_b/N_0)^{1/2}]$ for a binary PSK system, is directly dependent on the Gaussian assumption. It will turn out, for a good RNG algorithm, that the simulated curve is essentially indistinguishable from the theoretical curve.

Random number generators emit periodic sequences. It is necessary that the required observation time be a relatively small fraction of that period for the results to be meaningful. For example, if the period of the sequence is N , we cannot reliably estimate a BER lower than N^{-1} .

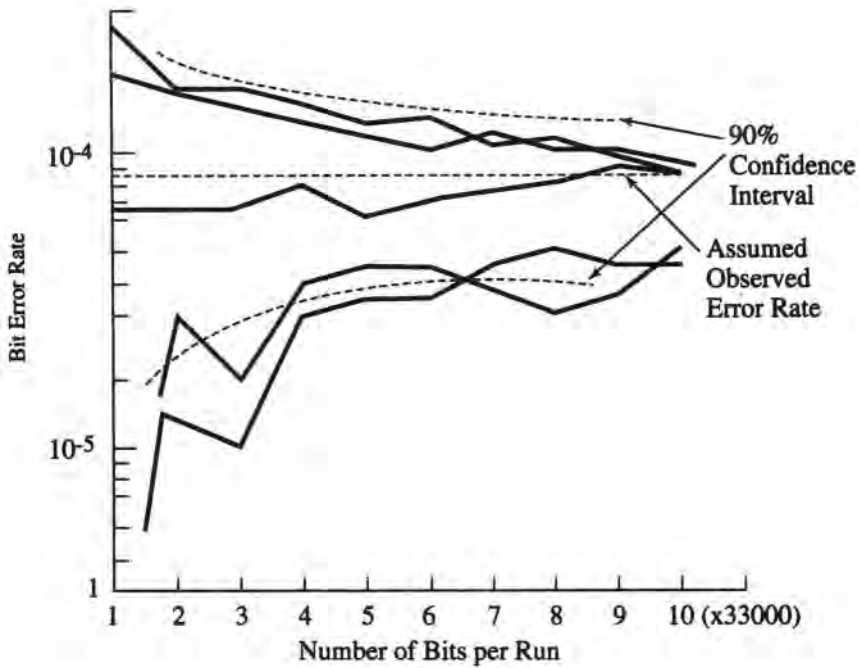


Figure 2.5. An illustration of the variability of Monte Carlo runs as a function of observation interval and random number seed.

Even though the state of the art of random number generation is not as advanced as could be wished, it does appear that relatively good RNGs exist that produce sequences that can be trusted. What we need to be aware of in Monte Carlo simulation is the statistical variability that normally occurs with relatively small samples. But this limitation is better viewed as one of the processing errors, which we consider next.

2.5.4 Processing Errors

Processing errors are due to the nature of simulation and the finiteness of computer memory, speed and precision.

The first consequence is that we must use discrete-time representations of continuous waveforms, and this leads to aliasing error, as discussed in Chapter 3. Sampling affects not only waveforms per se, but also the representation of filters. In the case of the impulse-invariant method, discrete sampling introduces an error which can also be characterized as aliasing. In the case of the bilinear z -transform technique, aliasing as such does not occur, but a distortion (frequency “warping”) does occur, which is proportional to the sampling interval. The aliasing error can be computed or bounded (see Chapter 3), so that its effect can in principle be controlled to as small a magnitude as desired. Generally, the aliasing error will present more of a problem in analog signal simulation because analog SNR requirements are typically fairly high. It is less of a problem in digital systems, at least for typical error rate requirements (say around 10^{-6}), because the equivalent noise due to aliasing is still a small fraction of the actual noise present. It is basically incumbent on the simulation user to ensure that a proper sampling rate has been chosen at any particular point in the system (as will be

seen later, it is possible to have different sampling rates at different points in the system, or for different processes at the same point). For typical digital applications, sampling rates between 8 and 16 samples per symbol are quite adequate and will result in errors on the E_b/N_0 scale of perhaps only 0.1 dB or so.

When using the impulse-invariant method for simulating filtering, it is necessary to truncate a filter's impulse response. Obviously, if the filter is IIR, this induces an error. There is a tradeoff between the size of this error, the length of the truncated impulse response, and computational load. For many cases of interest, it is possible to arrive at a viable compromise. It is recommended that the user check the truncation error as a standard part of operating procedure, for example, by computing the energy in the “ignored” portion of the impulse response; this should be a small number, perhaps 1–2%.

Another type of processing error is one that can result from the use of the complex equivalent lowpass representation of bandpass processes. Although this error is typically completely negligible so long as the carrier frequency is many times the bandwidth, it should not be overlooked as an artifact of simulation.

The speed limitations of the computer manifest themselves in our inability to run a Monte Carlo simulation long enough to eliminate statistical variability. This variability has already been raised in conjunction with Figure 2.5, and is extensively discussed in Chapter 11. This limitation may induce the most serious error (in the sense of uncertainty) that occurs in the estimation of small error probabilities.

2.6. Validation

As is implied in the preceding discussion, any quantitative notion of error must be coupled to a reference that is known or assumed to be correct. The process of certifying that the simulation results are acceptably close to the correct values is known as validation. (We note in passing that in the world of software development there is a connected notion of correctness known as *verification*, which is the process of ensuring that the software code indeed implements the desired equations or algorithms. However, we relegate this to the “mechanics” of implementation, which is not within our intended scope. We do, however, discuss some higher level aspects of software in the next section.) The criterion of closeness should be related to the intended application, but will inevitably be subjective. This process (validation) can take place at different levels of complexity and in a number of ways.

We are eventually interested in validating the simulation of some system. In doing so, we must consider the sources of error mentioned in the previous section. In practice, of course, we do not know ahead of time the detailed nature and characteristics of all the systems we may have to deal with. Hence, we can only validate a system simulation *after* the fact, that is, after the system has been given or specified. On the other hand, it is possible to prevalidate the *elements* (blocks) of a system as stand-alone models, at least up to some degree of modeling generality.

The output of a simulation, for example, a curve of BER versus E_b/N_0 , depends simultaneously on all the modeling constructs used in that simulation as well as on the performance evaluation technique invoked to generate that output. The interrelatedness of the various ingredients of a simulation vis-à-vis validation is illustrated in Figure 2.6. The “simulation environment” block simply denotes the software within which the simulation is performed. The arrows going both into and out of the other blocks are meant to indicate that

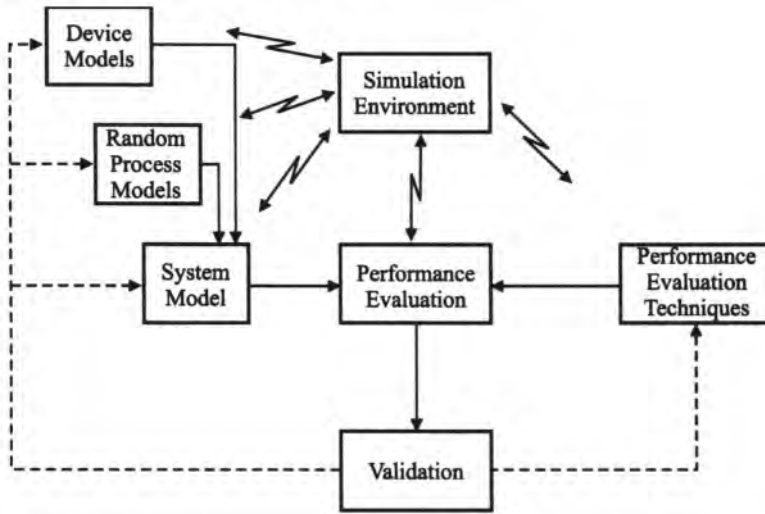


Figure 2.6. Illustration of the process of validation.

the process of validation may be iterative, i.e., if a simulation is declared not to be validated, some aspect of what produced the result must be changed, and the process repeated.

2.6.1. Validating Models of Devices or Subsystems

Any simulation package has available within it a set of models of various elements, referred to as the model library. This library contains all the blocks that may be callable in a simulation. If all the models in the library have been validated, then we certainly gain confidence that any assemblage of these elements yields “correct” results, given of course, that the assemblage is a good model of the system. Therefore, the first step in validating any simulation is to validate all the elements in the model library individually. This can imply a wide range of activities and difficulties.

What does validating mean in the context of device (or subsystem) models? This is perhaps best answered initially by revisiting the example of the five-pole Chebyshev (5PC) filter used in Section 2.4.2. Suppose we want to simulate an ideal 5PC filter. For the moment we consider such a filter to be the “correct” description of reality. Since we can describe such a filter exactly, e.g., by a ratio of polynomials $H(s)$ in the complex variable s , it is natural to model such a filter for simulation purposes by $H(s)$. Thus, $H(s)$ is the abstract simulation model which is the starting point for the simulation model itself, as indicated earlier. Generally, we shall refer to a set of equations or an algorithm that forms the basis for what is implemented in the computer as the *abstract model*, and what is eventually computed (in the simulation) as the representation of an element we shall refer to as the *simulation model*. Generally, the simulation model will not perfectly reproduce the abstract model because of processing errors or because of other conscious approximations that have been taken to simplify the model for computational reasons.

In the ideal 5PC filter case, for example, even though $H(s)$ is known precisely, the corresponding impulse response can only be computed for discretely spaced increments of time, thus incurring aliasing error. If, in addition, we truncate the impulse response, we have a representation error as well. In this instance, validation means determining the operating

conditions such that the output of the ideal 5PC filter is not in error by more than a certain amount. These conditions would include setting limits on the sampling interval and on the truncation time; the error might be measured as a peak error, mean-square error, or some other relevant criterion; and the output to which this error criterion is applied might be obtained for one specific input signal or for several classes of input signals. Of course, in order to know what the error is, it is implied that the correct value is known. In some cases this is possible *a priori*. For example, the impulse response or step response of an ideal 5PC filter is a computable expression. In other cases, the correct value is difficult to find. In those instances, a sensible approach to finding the correct answer is to remove the degree of processing error in incremental steps, e.g., by reducing successively the size of the sampling interval. When the output value appears not to change significantly after a certain number of iterations, we can safely surmise that the correct value has been reached (or the correct value can be inferred from the trend established by the iterations). Thus, even in the seemingly simple case of an ideal 5PC filter, the validation process is not trivial.

It may be noted that there is not necessarily always a difference between the simulation model and the abstract model. In particular, if we are dealing with inherently discrete or logical operations, then it may well be possible for a simulation model perfectly to reproduce the corresponding abstract model. As a simple example, a counter (which can be an element of an adaptive equalizer) can be implemented without error since integer addition can be done in the computer without approximation. Here, validation is inherent and automatic.

The question now arises, How does one validate a model of a nonideal device? Although, in detail, the answer depends somewhat on the nature of the model, the underlying general approach is the following. The model itself must have some adjustable properties or parameters that are able to reflect in some sense the types of imperfections that can arise in the device in question. Furthermore, one must be able to quantify these properties or parameters in the actual device, either by direct measurement or derived from measurements or specifications. Having done so, one can then set appropriate values in the simulation model, and this particular model is then presumably as close a replica of the real thing as it can be.

In the next step of the validation process, one would measure one or more of the properties of the *output* of both the real and simulated elements under study; the property may be a waveform, an average value, or some performance curve. This step may need to be repeated many times, for several reasons. One reason could be that if we are dealing with a class of equipment there could be a significant variation in the quality of the equipment to be encountered. Another is that, as previously indicated, the nature of the input may dictate how close the agreement is at the output. In addition, the output of the equipment should be observed under a variety of input conditions (e.g., frequency, power level, etc.) if the latter can be expected to influence the character of the former, as it would in a nonlinear device. Comparison of the simulated and measured results under some criterion of closeness would then validate (or not) the simulation model. But it should be noted that perfect agreement should not be expected because of physical measurement errors. We shall return to this point below. In any case, it is clear that the process indicated would probably lead to an unpractically large catalog if one were to try to prevalidate all useful device models.

2.6.2. Validating Random Process Models

With respect to simulating random processes, the situation is somewhat different and in a sense more complicated. A device model involves a deterministic mapping from its input to its output; the output of a random process model, on the other hand, is inherently non-

deterministic. This makes it more difficult to declare correctness by inspection. Indeed, validation can only take place in a statistical sense, and for almost every process of interest it is not possible to achieve unequivocal validation, even in well-defined cases. For random processes the abstract simulation model amounts to specifying the type of random process, e.g., Gaussian, Poisson, etc., and the associated parameters. Thus, the abstract model is usually well defined. The simulation model—the implemented version of the abstract model—is (usually) a numerical algorithm the output of which is intended to be a sample sequence of the process in question. There are a variety of statistical procedures designed to test the assumption that an RNG behaves like a given process, as discussed in Chapter 7. Generally, we cannot expect an RNG to behave exactly like the random process it is intended to simulate. Therefore, the validation criteria must be subjectively selected in a way that reflects some reasonable notion of closeness. For example, the simplest criteria that we may want to satisfy might be that only the first-order density should satisfy an accuracy requirement and also that, for independent samples, the observed autocorrelation function satisfies $\hat{R}_N(\tau) \leq \delta, \tau \neq 0$. We may also wish more stringent tests of the properties of the RNG to be satisfied. Many such tests exist, as mentioned, but they can never “guarantee” that an RNG has no shortcomings.

The criteria just mentioned are phrased in terms of the properties of the RNG itself. Since we know the RNG is imperfect, an alternative and perhaps better type of criterion is in terms of an indirect test through which properties of the RNG can be inferred. For example, the Gaussian property manifests itself in performance measures, like the BER, and since there are known expressions for many theoretical cases, one could set up such cases and declare the RNG to have passed the test if the simulated results are close enough to the theoretical results.

2.6.3. Validating the System Model

If all the elements in the model library have passed a certain validation test, then we can say that the host simulation package is validated on an individual model basis. Does this mean that if we string together some number of blocks to form a system, the simulation of this system is automatically validated? One cannot give an unequivocal “yes” to this question. Although we would feel confident that a string of validated subsystems yields “good” answers, depending on the type and number of these elements, it can be that the individual errors accumulate in such a way as to cause the simulated results to fall outside what might be considered acceptable accuracy. Therefore, in order to validate a simulation in a more encompassing sense, it is advisable to connect several systems whose behavior is known *a priori* (on theoretical grounds) and compare this behavior to the simulation outputs. If the comparisons consistently meet a specified accuracy criterion, then we can feel confident that the simulation package is system-validated to a certain level of system complexity. From a practical standpoint, it is desirable that the software environment within which one assembles strings of models be such that the mechanics of the process is as foolproof as possible. It is safe to assume that one can nearly eliminate mechanical errors in stringing all the blocks together if one uses a visual, hierarchical block diagram approach. However, if the user has to write a program, then there is room for such errors.

Suppose now that we want to validate a simulation with respect to a real, existing system. In a sense, this is the ultimate test of validity, but as was alluded to earlier, perfect corroboration should not be expected, nor should any difference be necessarily attributed to shortcomings in the simulation. We elaborate below.

Given a real system, the first step would typically be to “map” this system into a reduced complexity block diagram suitable for simulation, in accordance with the ideas discussed earlier. Presumably, the simulation block diagram will be composed of elements whose model input parameters are capable of capturing reality. Measurements will be made on the real system to determine what the values of these parameters are, and these values will be used as simulation inputs. At this point, the simulated system is as close as we can get to the real thing. If practical, both the real and simulated systems should be run under a range of conditions that mirror the expected range of operation. For each set of conditions we compare the simulated results to the physical measurements, and declare success under an appropriate criterion of closeness. We can compare waveforms, average levels, spectra, BER, or any other quantities of interest. We refer to a physical measurement made under a single set of conditions as a *snapshot* and to the procedure just described as *snapshot validation*.

Note that the described procedure validates not merely models of individual elements, but the system model as well. For example, if in the simulation we lump the effect of all phase noise sources into one equivalent process at the receiver, we validate this technique along with whatever other modeling techniques are invoked for *that* system. But validation on a system level should have a realistic criterion of acceptability. This is because measured results are themselves subject to errors, which are manifested in two ways. The first has already been alluded to: values of measured parameters that are used to “feed” the models will be in error to some degree. For example, measured filter transfer functions will have some error added to each amplitude and phase point. These errors by themselves will cause some disagreement between the simulated and measured performance metric.

A second type of measurement error occurs in the calibration of the test setup. Thus, for example, if we are interested in the BER as the parameter of interest, the measured BER curve will itself not be quite correct with respect to its horizontal placement. Typically, the measured BER is highly reliable because many errors can be observed, but it is the horizontal (or E_b/N_0) scale that cannot be perfectly calibrated. Hence, the measured BER can be thought of as some horizontal displacement of the reality. A reasonable accuracy figure for E_b/N_0 using state-of-the-art equipment should be on the order of a few tenths of a decibel.

In this example, the simulated BER curve will differ from the measured one for several reasons. As mentioned, we may initialize the simulation with errored values of parameters; the model of each subsystem will not be a perfect functional replica of the hardware; the generated signal or noise sequences will not be exactly like the real ones; and the length of observation will typically not be long enough to eliminate statistical variation. It is interesting to note that the uncertainty in the simulated results is “orthogonal” to that in the measured results in the following sense. In simulation, it is possible to calibrate the E_b/N_0 axis perfectly because we are in effect using virtual instruments that do not exhibit the shortcomings of physical equipment. On the other hand, the error sources mentioned cause an error in the BER itself. Thus, in a measurement the uncertainty is in the horizontal direction, while in a simulation the error is in the vertical direction. A picture of this idea is shown in Figure 2.7. The measured uncertainty is indicated by the swath of horizontal hashing, the simulation uncertainty by the vertical hashing, and the overlap region by the cross-hatching. Validation can be said to have occurred at some level of comfort (the word “confidence” is often loosely used instead) if there is indeed a region of overlap. Since the widths of the individual swaths should not exceed a few tenths of a decibel, this gives a quantitative idea as to the meaning of “comfort.”

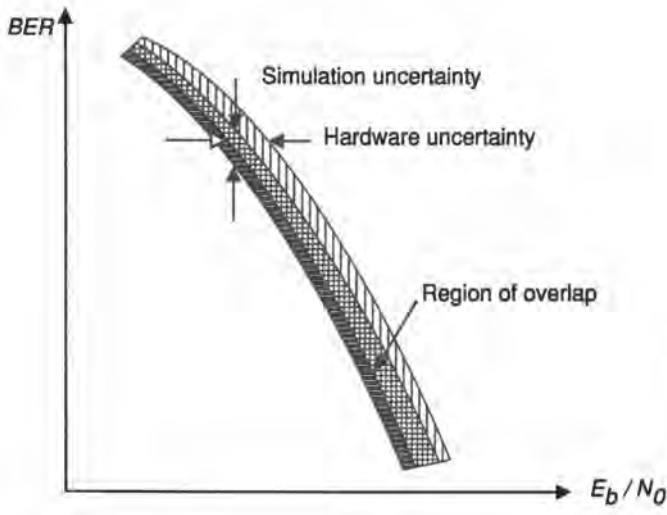


Figure 2.7. Illustrating the nature of uncertainty: simulation versus hardware.

2.7. Simulation Environment and Software Issues

Simulation activity cannot, of course, take place in the abstract. To bring such activity to life, as it were, models must be converted to programs and a software environment is necessary to link the models and execute the programs.

The methodology of simulation can be viewed as addressing two broad topic areas, one of which we might regard as dealing with the *content* of the problem and the other with the *mechanics* of the problem. The first of these subdivisions essentially encompasses the modeling and other conceptual considerations which lead to setting up a problem for solution by computational means; these considerations are the ones addressed in the other sections of this chapter and form the bulk of this book. In this section we address only briefly the “mechanics” of simulation, by which we denote the implementation of the computational process, which in principle could be independent of the specific problem itself. In fact, the degree of independence between the content and the mechanics of a problem is itself to some degree a methodological issue. Views on this issue have evolved over time and are colored by developments both in computer science and in computer technology.

For example, in the early days of simulation-based analysis and design of communication systems, the typical approach consisted in writing fairly long problem-specific simulation programs, debugging and executing them on a mainframe, and analyzing the results. The output of simulation programs often consisted primarily of endless pages of tabulated numerical outputs. Very little attention was paid to the reusability of programs and creating a simulation *environment* that could be shared by a large number of users or be reconfigured to solve other problems.

Although there is something to be said (in principle, anyway) for a problem-specific simulation mechanization, in almost all situations it is preferable to create a problem-independent *environment* for solving any number of problems within a large class. What we term here the environment is a combination of software and hardware capabilities that permits the flexibility to easily and quickly set up a problem within the class. With advances in software

and hardware technologies, particularly workstation and PC technologies, it is now possible to create an integrated and interactive environment for simulation, an environment in which software models and hardware data and signals can be combined to provide a rapid prototyping environment for the design of advanced communication systems. We describe in this section the desirable features of such an environment and also describe recent trends.

Strictly speaking, the simulation environment consists of a software environment for creating simulation programs and a hardware environment for storing, executing, and displaying the results of simulation programs. We describe below the important and desirable features of the hardware and software environments. (The current generation of workstations and PCs offers an integrated hardware/software environment, and a clear dichotomy between the two may not always be possible.)

2.7.1. Features of the Software Environment

The first step in the simulation of a communication system is the creation of a computer program (commonly referred to as the simulation “code” or program), which usually consists of a system-level “main program” that links a set of subroutines or models of various functional blocks that are preprogrammed and stored in one or more model libraries. A *model builder* or *configurator* is used to select and connect the library blocks in the desired topology as specified by the block diagram of the system being simulated. Parameters of various functional blocks are specified either when the system is being configured or prior to the execution of the simulations. In addition to the design parameters, values for simulation parameters such as the length of the simulation, sampling rates, or seeds for random numbers are also set prior to the execution of the simulation program.

During the execution (or “simulation”) stage, samples of signals are generated, processed, and stored. Signals and estimates of performance measures that are computed during the simulation are analyzed at the end of the simulation. This last phase, called *postprocessing*, might involve the computation or estimation of additional performance measures and display of the results. (It is also possible to execute the simulations in an interactive mode in which simulated signals can be analyzed “on the fly” and the results displayed during the simulations as the waveforms evolve in time). In either case, if the results of the simulation indicate that performance objectives are not met, then either the parameter values or the topology of the system are changed and several iterations are made till a suitable design is found.

To facilitate simulation-based analysis and design of communication systems, the overall software environment for simulation of communication systems should have the following features.

Integrated and Graphical Environment. Given the interactive nature of analysis and design, it is essential that the simulation environment be interactive and graphical. The current trend is toward a visual environment that permits model building in a hierarchical block diagram form and an interactive graphical environment for the analysis and display of simulation results.

Hierarchy. Modeling of very complex systems can be handled easily if the simulation environment permits the use of hierarchy for handling complexity. In a hierarchical modeling environment, complex models can be built up from very simple building blocks which are easy to code and test. For example, one could start from primitive building blocks such as adders, multipliers and integrators, and build modulators, filters, etc., and use these to build transmitters and receivers. With the hierarchical framework, each user deals with a degree of

detail that is appropriate for his or her domain of expertise. For example, a system-level designer need not deal with the details of the implementation of a filter.

A hierarchical block diagram approach is the current state-of-the-art framework for developing simulation models for communication systems. In some commercially available packages the user seldom writes simulation code. Instead, he or she relies on the hierarchical and graphical modeling framework to construct simulation models using a set of building blocks from the model libraries. Models are checked for errors and the block diagrams are automatically translated into simulation programs.

Transportability. The software environment should have a great degree of portability so that it can be implemented on a variety of hardware platforms. With the current trend toward standardizing compilers and windowing/graphical environments, it is easier now to transport simulation packages and programs from one hardware platform to another.

User Interface. Significant advances have been made in the area of user interface. Menu-driven commands with popup menus, on-line help, and error checking are standard features. Menu-driven systems sometimes also provide the option for a command-driven mode using an abbreviated set of commands. This permits the expert user a faster interface while at the same time maintaining a self-guiding, menu-driven model for the novice or occasional user.

On-line documentation is a desirable feature which provides facilities for the user to document the models while they are being developed. On-line documentation keeps the documentation dynamic rather than static, which tends to be the case with off-line documentation. On-line help and documentation encourages model reuse and sharing.

Part of the user interface should include a database manager which helps the user with the organization, storage, and retrieval of simulation models, results, and design iterations. Part of the database manager should be a consistency checker that keeps track of changes and revisions to models and assures that all the entities in the databases are consistent. It is particularly important to make sure that when a lower level model is changed, all higher level models using the revised lower-level models are flagged, checked for consistency, and brought to the attention of the owner of the higher level model.

Features such as on-line help, documentation, error checking, consistency checking, and database management take care of the mundane details associated with the mechanics of simulation, thus focusing the attention of the user on modeling, analysis, and design.

Modularity. The simulation framework should have a modular software structure. Modularity makes it easy to modify the simulation environment to take advantage of changes in software and hardware technologies. Modularity also makes it easy to modify and expand the environment to handle new applications.

2.7.2. Components of the Software Environment

A common modular structure for simulation packages has evolved (see Ref. 4 for a review). The major components of a simulation environment include model libraries, a model builder (or configurator), simulation manager (or exerciser), a postprocessor, and a database manager. These components are integrated into a seamless environment with the appropriate user interface.

Model Libraries. The model libraries contain preprogrammed models which describe the behavior of functional blocks in communication systems. These models will include both primitive models (stand-alone programs) as well as hierarchical models built using primitives

and other lower level models. The model libraries should be expandable by the user both in the form of new models built with other models already in the library and in the form of stand-alone programs (primitives). The usefulness of a model library is measured in terms of the quantity and quality of the models, model representation (hierarchical or planar), ease of expansion, on-line documentation, on-line help, and validation data.

Models should be reentrant (i.e., multiple use within a simulation) and it should be possible to initialize the models and control their conditional execution using hold or enable signals. Another convenient feature is the ability to execute models either in a “block” mode or in a point-by-point mode. In the latter mode, each invocation of a model accepts a single sample on all of its inputs and computes one value for each of its outputs. In a block mode the model accepts a sequence (a vector) of samples on each of its inputs and produces sequences of output samples. Block processing improves the computational efficiency, and coupled with an enable feature, it provides a method for implementing multirate sampling.

Model Builder (Configurator). The model builder allows the user to select a set of building blocks from the libraries, connect them together, and set or export parameter values to the next level in the hierarchy. Desirable features of a model builder are flexible (arbitrary) topology, hierarchical structure, replication of structures, block diagram input or a higher level language to describe topologies, easy handling of parameters, including capabilities to enter expressions for parameter values and exporting (floating), on-line documentation and help, and edit-time error and consistency checking. A hierarchical block diagram approach with graphical editing capabilities is a logical approach for synthesizing the model of a large variety of systems.

Simulation Manager. This component of the software translates the simulation model into an executable program, links it with appropriate model libraries, and executes and stores the simulation results. All of this is done with a minimum amount of user intervention. With the current trend toward graphical approaches to creating models in the form of block diagrams, simulation managers now contain code generators which translate graphical models into simulation programs in C or C++. The choice of programming language is not critical and it is possible to use more than one language since a program written in, say, C can link easily to a program written in C++.

The simulation manager permits the user to gather waveforms and simulation results from various levels of the hierarchical models. When executing long Monte Carlo simulations, it is desirable that the simulation manager provide some status indication (number of samples processed, for example) as well as the ability to stop the simulations, view intermediate simulation results, and restart the simulations. The simulation manager should also supervise remote and distributed execution of simulations on different host processors.

Postprocessor. The postprocessor is an important part of a simulation environment since the postprocessing routines are the ones that enable the design engineer to view the results of the simulation. The model builder and the simulation manager should be designed to allow the designer to draw direct cause and effect inferences during design iterations. Since most of the simulation results are to be presented in graphical form, the postprocessor should have built-in functions for producing a variety of plots. As a minimum, the postprocessor should be able to produce high-resolution displays of time plots, spectral plots, eye diagrams, scatter diagrams, signal constellations, $X - Y$ plots, error rate curves, histograms, and cross-correlation and autocorrelation functions. The following features also make the postprocessor more useful: automatic scaling and labeling, panning and zooming, multiple plots, overlays and inserts, display of ancillary information such as parameter values, and generation of hard copies. Graphical editing of plots (scaling, cutting and pasting, etc.) is also useful.

Database Manager. Design of large and complex communication systems involves hundreds of models and simulations, and a large number of engineers. Keeping track of models, revisions, and design histories and iterations becomes a major task. A well-designed database manager can maintain models, simulation results, and other data in one or more databases. It can minimize the amount of bookkeeping details the design engineers have to deal with and provide a variety of useful services. A consistency checker should be part of the database manager and it is also useful to have a distributed database manager.

2.7.3. Hardware Environment

The interactive and graphical nature of analysis and design requires a computer platform that offers a high-resolution graphical display with a large screen, fast I/O, adequate local processing and storage, and extensive networking capabilities to permit sharing of databases and processing over several processors if necessary. The current generation of engineering work stations (and PCs, even though the distinction is somewhat blurred now) have these capabilities and hence they seem to be the platforms of choice for implementing simulation packages. The interactive computing environment provided by the workstations combined with the graphical approach to model building and the use of popup menus and mouse-based operations provides a productive and easy-to-use interface.

The current drive toward standardizing operating systems, compilers, windowing schemes, and network protocols makes simulation programs highly transportable over a wide variety of hardware platforms.

2.7.4. Miscellaneous

While a good simulation environment minimizes the amount of mechanical details with which a communications engineer has to deal, there are a few details that a user must know to ensure that simulation models and programs function correctly. This is particularly important when creating a simulation model using “off-the-shelf” building blocks or when writing programs that someone else might use to build higher level models. Some of the important issues are discussed here.

Reentrant Programs. If a simulation model uses a building block (subroutine) more than once, then special care should be taken to ensure that the parameter values and internal states for the block are properly stored for each instance (or use) of the model. Such a situation arises, for example, when one uses exactly the same type of filter in both the transmitter and the receiver of a communication system. The filter output will depend on the coefficients and the past values of the input and output along with the current value of the input. If the coefficients and the internal states (previous values of the input and output) are not stored properly, then the second instance of the filter might incorrectly compute its output using the coefficients and internal states of the first instance. To ensure proper operation (or reentrancy), care should be exercised in allocating storage for local variables. This is the responsibility of the user who writes the model and the model builder that takes care of memory allocations for various blocks.

Order of Execution. Waveform-level simulations of communication systems are clock-driven. That is, there is a simulation clock that is advancing at the sampling rate, and each time the clock advances, one computes a new set of values for all the signals. (There are variations to this theme when block processing and/or multirate sampling is used.) The order

in which the blocks are executed depends on the topology of the system and on the presence of feedback loops.

In an open-loop system such as the one shown in Figure 2.8a, execution proceeds from source to sink. Outputs of various blocks are computed when the current values of all the inputs are available, except for sources that do not have any other inputs. Values of source outputs depend only on the time index (and possibly on previous values of their outputs), and hence their values are usually computed first.

Assuming that all signal values have been computed for the time index $n - 1$, we compute the values for time index n as follows. At time index n , we compute the output for all

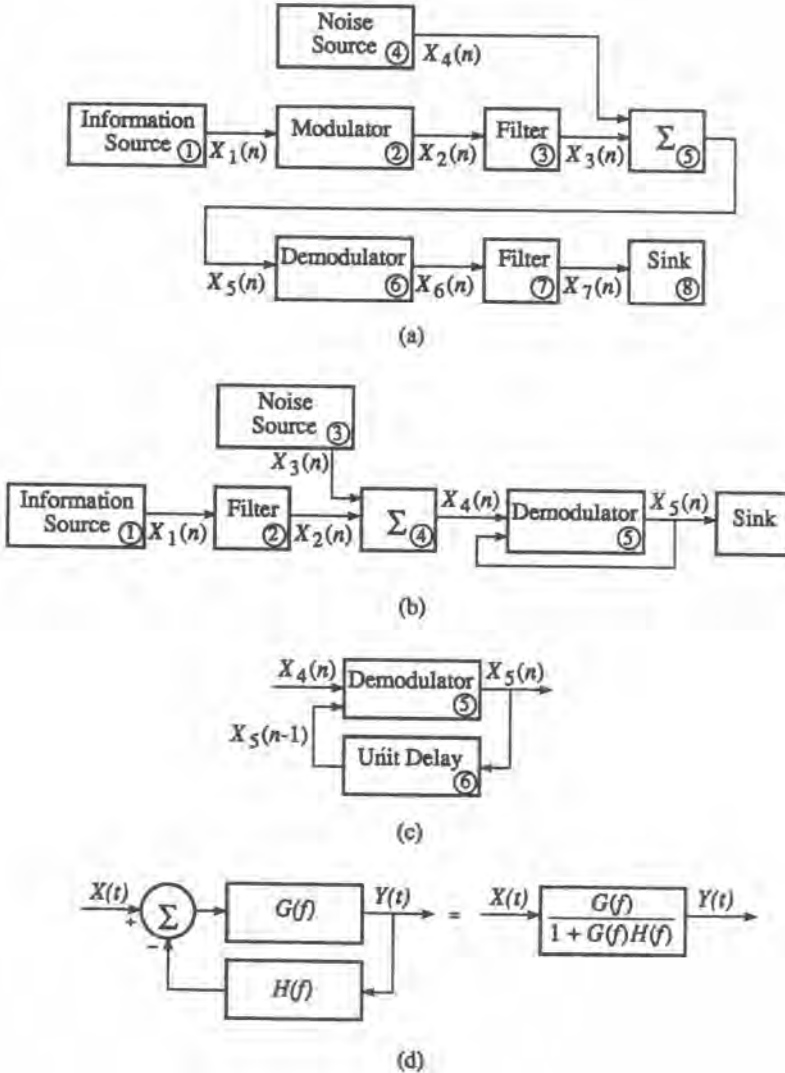


Figure 2.8. Considerations relating to computational flow. (a) Open-loop system. (b) System with feedback. (c) Breaking up a feedback loop with a delay element. (d) Analytical simplification of a feedback loop.

the sources first and then compute the output of each block when all of its inputs are available. Thus, for the example shown in Figure 2.8a, we compute $X_1(n)$ and $X_4(n)$ first and then compute the outputs of blocks 2, 3, 5, 6, and 7.

When a system has feedback loops, determining the order of execution is difficult and sometimes leads to a deadlock. For the example shown in Figure 2.8b, to compute the output $X_5(n)$, we need both $X_4(n)$ and $X_5(n)$, which has not been computed yet! To break the deadlock it is common practice to insert a unit delay in the feedback loop as shown in Figure 2.8c, although this is not necessarily the best approach. An initial value for the output of the delay, say at time index $n = 0$, has to be specified prior to the beginning of the simulation iterations. Now, to compute $X_5(n)$, we need $X_4(n)$, which is computed first, and $X_5(n - 1)$, which has already been computed during the previous iteration. With delays to break up feedback loops, the order of execution starts with all the sources, then the outputs of the delays are copied from their input values computed during the previous iteration, and the remaining blocks are executed when all the inputs for a block become available. Notice that this introduces a one-sample delay, which is an artifact of simulation. When the sampling rate is very high, this unit delay has negligible effect and this is the reason for using a very high sampling rate when simulating feedback loops such as phase-locked loops. In some cases, such as in Figure 2.8d, we can analytically simplify and replace a feedback loop with an equivalent transfer function. However, such simplifications are difficult to automate. In Chapter 8, we shall revisit in more detail some of the issues related to simulation of feedback loops.

Sample-by-Sample or Block Processing. Simulation (computation) can be carried out on a sample-by-sample basis or on blocks of samples. In the former case, each invocation of a module accepts one sample from each input port of the module and produces one sample on each output port. In block mode, blocks of N input samples from each input port are mapped to blocks of N samples on output ports. (If signals are vector-valued, then sample-by-sample processing is equivalent to block processing.)

Block mode processing coupled with FFT-type algorithms is generally (but not always) computationally more efficient than sample-by-sample processing. However, block processing introduces a time delay of N samples, where N is the block size. While this delay can be tolerated in open-loop systems, it will not be acceptable in closed-loop systems.

With appropriate buffering and scheduling, one can mix sample-by-sample and block processing. While this might present the appearance of sample-by-sample processing, the processing delay will still be present and its effects must be carefully assessed.

Stream-Driven and Event-Driven Simulations. These techniques, which are available in some commercial simulation packages, are well suited for handling the asynchronous aspects of some of the signal processing operations and protocols in communication systems. In stream (or data)-driven simulations, the availability of all necessary data items at the inputs of a block triggers the activation of the block. If the “rate” of availability of the data items can be precomputed from the fixed input/output rates of the blocks, then the sequence in which the blocks in a simulation model will be activated can be determined prior to the start of the simulations: this is called static scheduling. If the blocks do not have a constant input/output rate (in fact, this can vary from invocation to invocation), then static scheduling is not possible. The availability of data items has to be checked at run time, i.e., during simulation, and scheduling of activation of blocks has to be done dynamically. Stream-driven simulation is very efficient for simulating multirate signal processing operations.

A fundamental difference between time-driven and stream-driven simulators is that data-driven simulators typically do not have any notion of global time built into the simulation

kernel. Instead, each block can have a notion of its own local time. The time semantics is introduced by the user of the simulator instead of the simulator, itself.

In event-driven simulations, there is a notion of simulation time, and the global simulation time is advanced to the time of the next scheduled event in the event queue before execution of blocks begins. Thus, time may be advanced by an arbitrary amount rather than by a fixed increment equal to a multiple of sampling interval as is done in time-driven and stream-driven simulations. An example of this occurs in packet communication systems, where the arrival of a new packet triggers processing operations in the system and the interarrival time can be arbitrary. Each block in an event-driven simulation needs to update its internal states to be consistent with the global time. Typically, only a few blocks need to be activated to update their internal states. No processing takes place during the interarrival time between the events.

Event-driven simulations are computationally more efficient than time-driven simulations for network and logic systems because of their inherently asynchronous nature. However, event-driven simulation will be slower for waveform-level simulation of synchronous and continuous signal processing operations. In some systems, for example, in a GSM receiver, which incorporates protocols and signal processing operations, it may be necessary to mix heterogeneous simulation modes.

Time-Domain versus Frequency-Domain Processing. Simulation models can be implemented in the time domain or the frequency domain or a mixture of both. In a system like the one shown in Figure 2.8a, if we assume all the modules to be linear, then one can gather all the samples from the information and noise sources into one block, transform the samples to the frequency domain, and compute the response of all the modules in the frequency domain using block processing. At the output of the last module [labeled (7)], we can take the inverse transform and output time-domain samples. Of course, the entire system could have been simulated in the time domain using sample-by-sample processing.

The choice of frequency-domain or time-domain processing depends on the nature of the system being simulated. In general, nonlinearities and feedback loops are simulated in the time domain using sample-by-sample processing, whereas, as discussed in Chapter 4, filters can be simulated in the time domain or the frequency domain. Current simulation frameworks permit a mixture of time-domain, frequency-domain, sample-by-sample, and block processing techniques. It is the user's responsibility to select the appropriate processing technique for a given problem.

Links to Implementation. The primary uses of simulation are design optimization and performance evaluation. The usual outputs of simulations are performance plots such as BER curves and plots of performance as a function of design parameters. These plots are used for sensitivity analysis and tradeoff studies.

When simulations are used to support implementation of components such as digital filters, equalizers, decoders, etc., the waveform-level simulation environment is linked to implementation tools. Implementation of communication systems can be divided into three parts: RF/IF, baseband analog, and baseband digital portions. The "baseband" portion can be implemented in digital hardware or as code on a programmable digital signal processor.

If the implementation is in the form of digital hardware, then simulations are carried out using detailed models at the "bit" level and "clock cycle" level, and a description of the simulation model in a (high-level) hardware description language (HDL) is generated as a link to gate-level design and implementation CAD (computer-aided design) tools. If the implementation is in software, then the simulation environment produces a software description of the algorithm being simulated. The software or the code that is generated can

be optimized for efficient execution on a particular DSP processor. In most simulation packages, the block diagram model is usually translated into a simulation program by linking the code for each block in the hierarchical block diagram. There is usually very little difference between the simulation code and the implementation code for signal processing algorithms such as equalizers, encoders, and decoders. In some cases the simulation code is executed on fast DSP processors attached to the workstation or PC which is used as the primary platform for the simulation environment in order to speed up simulations.

2.8. The Role of Simulation in Communication System Engineering

In the practical world of designing and building communication systems, simulation is not an end in itself, but is a tool to aid in that process. There are aspects of that process, as it is commonly practiced, that have a fundamental impact upon the methodology of simulation. Some of these implications have been mentioned earlier. The distinguishing feature of the system engineering of real systems is that, unlike classical analysis problems, the process does not—and cannot—rely on the precise and detailed knowledge of the characteristics of every building block of the system. This is inherent to the specifications approach, which is the standard vehicle for designing systems. There are fundamental as well as practical reasons for this approach.

The basic goal of a system designer is to field a system that satisfies certain performance objectives over some intended system lifetime. Suppose we had found a detailed design which satisfied the performance objective. By a “detailed” design, we mean that each building block is completely described by some appropriate set of functions. [For example, a filter would be specified by a continuous function of frequency $H(f)$ for all f .] If this detailed design were imposed on the physical equipment, the likelihood that the manufactured equipment would *exactly* replicate this design would be vanishingly small. Real devices and components cannot be made nor measured with infinite precision, nor can parasitics and packaging effects be completely predicted. Thus, it is unrealistic to expect that we could design and build equipment that would adhere to a continuum of values. The preceding applies, of course, not just to the hypothetical detailed design we started out with, but to any possible detailed design that would also meet the performance objective.

Even if it were possible to build equipment whose characteristics reproduced exactly some given set, there are reasons why we would not want to attempt it. One is the very practical issue of test and verification. Any imposed requirement must eventually be verified as having been met, and the more detailed the requirement, generally the more costly the verification procedure, and the more likely it is to fail the test.

Suppose by sheer luck that a manufactured piece of equipment did satisfy exactly a detailed specification. This would be true only in a snapshot sense, i.e., at a point in time and under the conditions of the test, but would not continue to be so indefinitely, for the characteristics of equipment will change (generally adversely) with age, temperature, humidity, and perhaps other factors. For example, it is known that solid-state devices or tubes degrade with time. The passage of time will also lead to gradual corrosion, cracking, encrustation of deposits, etc. Radiation effects in space will damage and alter equipment behavior. The environment itself, i.e., the “channel,” also induces uncontrollable effects. Weather conditions have multiple effects such as attenuation and depolarization over the signal path, ice buildup on antennas, rain effects on radomes, and antenna mispointing due to winds.

There is thus a multiplicity of effects that are essentially unpredictable, except perhaps in a statistical way. It would be impossible to “back out” these effects to arrive at a set of detailed characteristics necessary at the beginning-of-life (BOL) in order to guarantee performance at end-of-life (EOL). However, since performance over life is in fact the designer’s goal, some such procedure is necessary. In effect, this has already been introduced in Section 2.3.4 dealing with hypothetical systems. In this approach we impose EOL *parameter* specifications on a set of parameters that are major contributors to system performance, but may not *absolutely* control that performance. The parameter set should be as small as possible in order to simplify both the synthesis and test/verification procedures, and the parameters should be susceptible to measurement at any stage of system development. These practical considerations are especially important if we are dealing with many replications of the same equipment, such as redundant chains, multiple hops, etc.

In order to use simulation at all in this process we must be able to *predict* EOL detailed characteristics. Such characteristics can be inferred, at least approximately, from the EOL parameter specifications and from a certain amount of experience with the actual type of equipment that would be used. (By “actual” type of equipment, we mean the relevant attributes such as the device technology, bandwidth, frequency, etc.) If such equipment has not been previously built, then guidance must be obtained from laboratory prototypes.

To clarify the preceding ideas, consider Figure 2.9, which shows a flow diagram of the system design process including the role of simulation. The process begins with the “concept definition” stage, where the “top-level” specifications are made, such as information rate, quality of the link, availability, connectivity, etc. Since we are confining ourselves here to the link level, only aspects relevant to link performance need be considered. The performance of any link is governed by two factors, the signal-to-noise ratio (SNR) on the link, and the accumulated distortion over the link. Generally, these are interactive and there are certain tradeoffs between them. Initially, therefore, as indicated in Figure 2.9, the system designer must make certain choices. The SNR aspect of the link is reflected in what is typically called the *link budget*, which is *not* the quantity of interest in simulation. The waveform distortion or fidelity aspect is the one that is addressed by simulation. As indicated above, we attempt to control this aspect through parameter specification: these parameters are called “distortion” or “communication” parameters. Also as mentioned, we want these parameters to constitute a moderately small set of discrete values. For example, a traveling-wave tube might be specified by a maximum AM/PM conversion factor (degrees per decibel) as opposed to the continuum of values in the entire curve of output phase versus input power. Examples of the specification approach were given in Section 2.2.4.

The system designer must determine what parameters it is sensible to specify. Based on the initial values, one can construct a simulation to determine the degradation induced by the postulated system. Together with the initial link budget, this information tells us if we have *link closure*. This term means that both the SNR and the distortion parameters are compatible, i.e., the performance measure can be met. If this is not the case, then both the initial link budget and distortion parameters must be modified and the procedure repeated.

At this point we have guidance on hardware development. The hardware designers now have targets for the quality of their equipment. The hardware developed at this point is called demonstration hardware, or engineering development models, or some such term to imply equipment whose design is not necessarily final, but is reasonably like the finished product. This hardware has three main uses. It gives an inkling of the ease or difficulty of meeting the initial specifications and provides an opportunity for adjusting those specifications to “final” form. At the same time, this hardware can be the basis for validating the simulation. The

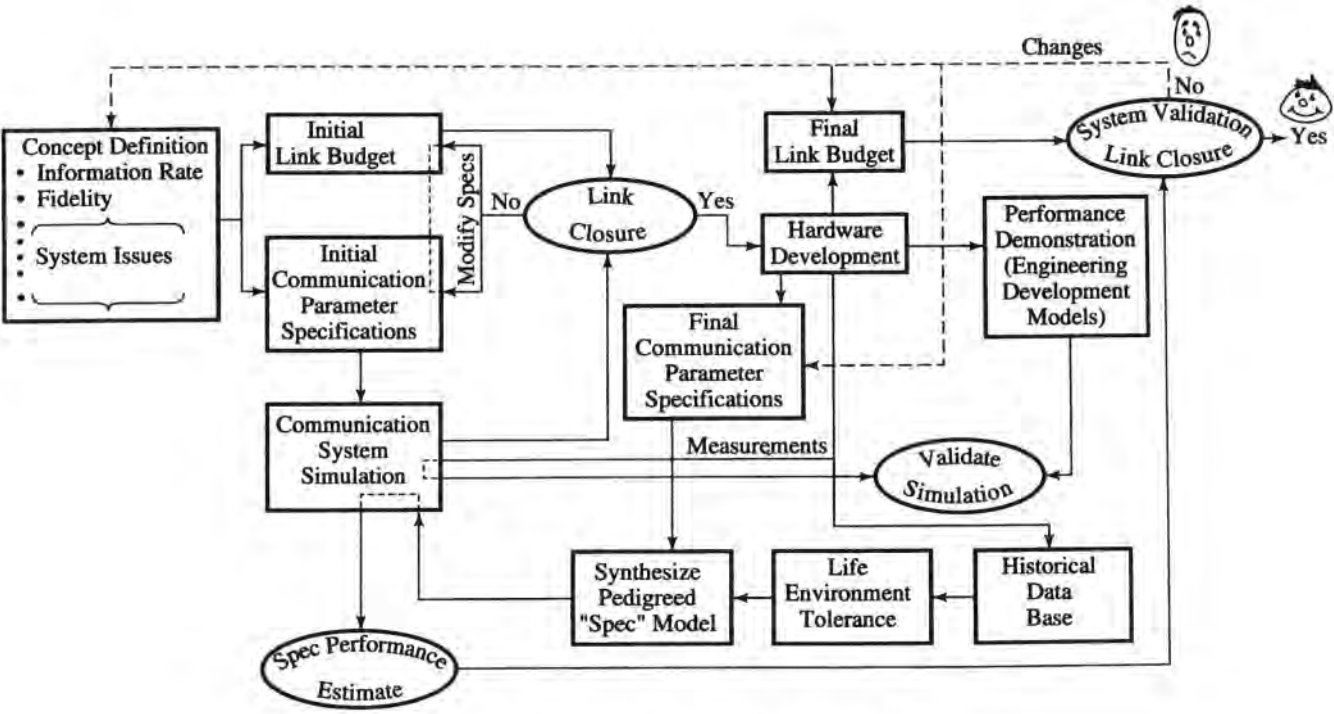


Figure 2.9. System design methodology and the role of simulation.

individual elements (devices, subsystems) can be measured for the characteristics that the simulation models require for their description, so that the simulation can be set up as a nominal mirror image of the hardware. The hardware elements are connected together in the laboratory to form the system, whose performance under these conditions can be demonstrated. The simulation system is then run, and its performance observed. The degree of closeness between the hardware and simulation results is the basis for declaring whether the simulation is “valid” with respect to this system. It is important to develop a sense of confidence in a simulation, for it will be used to predict EOL performance, which cannot be corroborated experimentally until after the fact.

The third use to which the hardware development is put is to build or augment a database for the kind of equipment in question. This database is important in painting a profile of the range of expected behavior of that equipment. This information, together with the anticipated effects of aging and environment, and allowance for measurement error, in combination with the final distortion parameters, forms the basis for synthesizing a “pedigreed” model based on specifications. That is, such a model simultaneously meets all specification limits and at the same time has detailed characteristics that “look” like those of actual equipment. This hypothetical system is the basis for projecting performance at EOL. Its characteristics are inserted into the simulation, which yields an estimate of performance. This estimate, in conjunction with the final link budget, tells us if we have closure. This validates the system design, in a certain sense, but is not validation of the simulation per se. Of course, if closure does not occur, the process must be iterated until it does.

The systems engineering methodology just described is especially appropriate for certain types of systems. It is apropos for complex systems intended for long life and for which the environmental conditions can range widely. This is the case, for example, for a satellite system or for a far-flung terrestrial system. The approach is somewhat conservative in the sense that the synthesized specification model reaches all specification limits simultaneously, something which is unlikely to occur in real life. On the other hand, there is an inherent degree of uncertainty in that the detailed characteristics taken on by the actual system may be more deleterious than those assumed for the simulation. In any case, the methodology as a whole is a useful starting point for system design and for the way in which simulation can be fruitfully put to use in that context.

2.9. Summary

In this chapter we have discussed a variety of topics that form the fabric of methodology. The discussion is not intended to be exhaustive, but rather to suggest a number of issues that should be addressed by a simulation practitioner. We defined methodology as that body of techniques and procedures from which one draws in order to “do” simulation in a way that is as efficient and accurate as possible, bearing in mind that there is usually a tradeoff between these objectives and computing as well as manpower resources.

A central aspect of methodology concerns the modeling of systems and their functional elements. As a general rule, the more detailed the representation, the more accurate it is, but the more computationally intensive it becomes. There is thus generally a tradeoff between accuracy—equivalently, complexity—and computational resources. There can be gradations of complexity reduction along with corresponding incremental reductions in accuracy. Examples of such ideas will be found in the development of various models in Chapter 8. To

the extent that a model is not a perfect representation of the real thing—which is generally to be expected—there is some error. We discussed the nature of errors for several modeling categories.

A system is represented by a string of models, about which we infer some property by running the simulation. There are in fact different ways to do a simulation. The standard is so-called Monte Carlo simulation, in which a simulation is more or less a software counterpart of the actual system. Unfortunately, Monte Carlo simulation is time-consuming. The various ways, Monte Carlo and alternative methods, to simulate a system in some sense of the word are called performance evaluation techniques. We outlined some of the underlying ideas and issues. The subject is treated in detail in Chapter 11.

The subject of errors is intertwined with the notion of validation, the process by which we determine that a simulation is valid. It is self-evident that validity exists to the degree that errors do not. This subject is so central to simulation that it is worthwhile restating here some of the main related points.

(a) The simulation of more or less arbitrary systems is performed by assembling some set of models from a model library. This set forms the simulation block diagram. Validation has two broad aspects: To what extent does the simulation block diagram mirror the actual one? and To what extent does a particular model faithfully reproduce the behavior of a device or process? The first aspect can generally be dealt with only after the fact, and certainly not before the system is conceived. The second aspect can be dealt with, to some degree, independent of any particular system.

As much as possible, of course, a simulation block diagram should be an assemblage of validated models. A model library can have hundreds of models. Many of these can in turn be assemblies of lower level models, which in turn should be validated. It is efficient both for model construction and validation to view systems in terms of this kind of hierarchical structure.

(b) As a practical matter, when a simulation block diagram is assembled, one should first verify to the extent possible that this simulated system is working correctly before committing large amounts of computer run time. Thus, it is good practice to perform short debugging runs.

(c) As an aid to the above, the simulator (the person) should perform “sanity checks” using all appropriate tools and knowledge. The latter includes theoretical formulas, bounds, approximations, etc., such as those provided in Appendix A.

(d) If, because of run-time limitations, Monte Carlo simulation cannot be used for all the tradeoff runs that might be desirable, it is necessary to resort to some variance-reduction technique. However, if at all possible, it is generally wise to perform at least one or two Monte Carlo runs of the system as close to *as-is* as possible, to calibrate other results.

(e) It is also useful to reduce the processing errors to the smallest possible, consistent with run-time efficiency. We know that the processing errors are proportional to sampling rate (e.g., samples/bit) and to run time (statistical variability). Thus, the following procedure is recommended. One representative case should be rerun with (1) double the sampling rate and (2) double the run length. If the results are sufficiently close to the original case, then the original sampling rate and sample size were chosen correctly and it can be safely assumed that the processing errors were acceptably small. Of course, the same process can be used to ensure that the original choices were not too large.

Next, we covered briefly another aspect of methodology, namely the organization of the computer program, or the software environment, that is the abstract embodiment of the simulation, as opposed to the computing hardware. Although there is not a unique way to

perform this organization, we focused on a modular, hierarchical approach as a natural software methodology for implementing the naturally hierarchical description of communication systems.

Finally, we discussed the role of simulation in the wider context of communication system engineering. Although our focus in this text is on the technical details of simulation, the ultimate application of simulation as a supporting activity in the real world should be kept in mind.

References

1. J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*, Methuen, London (1964).
2. R. Y. Rubenstein, *Simulation and the Monte Carlo Method*, Wiley, New York (1981).
3. J. H. Moore and M. C. Jeruchim, Empirical study on the variability of Monte Carlo simulation, *Proc. Int. Telemetry Conf.*, San Diego, CA, October 14–16, 1980.
4. K. Sam Shanmugan, Simulation and implementation tools for signal processing and communication systems, *IEEE Commun. Mag.* **3**, 36–43 (1994).

Representation of Signals and Systems in Simulation

Analytic Fundamentals

This chapter deals with the analysis of a system driven by deterministic (or test) signals. In a generally accepted definition of *analysis* there are three key words: the *excitation*, the *system*, and the *response*. System analysis and simulation are concerned with determining the response, given the excitation of the system. In the system design stage the problem is to synthesize the system, given the excitation and response.

For communication systems the excitation and response are given in terms of voltages and currents which are functions of time t . In general, these functions of time are called signals. This chapter is concerned only with the response of systems to deterministic signals. The response of systems to random signals is treated in Chapter 6. The real-life signals and systems that we wish to simulate are generally continuous. However, within the computer these functions are approximated by equivalent samples. These simulated functions belong to the class of discrete, or sampled, signals and systems. The main purpose of this chapter is to present in a fairly concise form the basic analytical tools that are used for modeling and analysis in the sampled domain. Simulations using these models should give results that closely approximate those obtained from the continuous systems.

Section 3.1 presents the general concepts of deterministic signals and systems. The systems are described in terms of their properties, which are used to subdivide systems into subclasses.

The principal focus of this chapter, presented in the remaining Sections 3.2–3.9, is on a subclass of systems referred to as *linear time-invariant* (LTI) systems. The properties of linearity and time invariance, described in Section 3.2, that define this class lead to a remarkable set of concepts and techniques that are of major practical importance in the simulation of communication systems. Linear time-varying systems are discussed in Chapter 4 and nonlinear systems in Chapter 5. Section 3.3 describes the representation of LTI systems in the frequency domain and develops the Fourier analysis. In Section 3.4 we introduce the concept of the lowpass equivalent of a bandpass signal and system. The lowpass representation is essential for efficient representation of continuous systems in discrete-time computer simulations. The Hilbert transform is introduced as a tool for conversion of the bandpass system to its lowpass equivalent. Section 3.5 focuses on the sampling theorem, which provides the link between continuous and discrete systems, and on multirate tech-

niques, which allow the use of different sampling rates within a simulation. Section 3.6 presents the *Laplace transform*, which is used in modeling continuous-time systems described by rational transfer functions in the *s-domain*; such systems encompass all analog filters with lumped elements. The related *z-transform*, used to describe rational functions transformed from the discrete-time domain, is then treated in Section 3.7. The *z-domain* system models include *infinite impulse response* representations of transfer functions, mapped from the *s-domain*, and *finite impulse response* representations; both are extensively used in computer simulations of communication systems. Section 3.8 presents the *discrete Fourier transform*, and its computationally efficient version, the fast Fourier transform (FFT). The FFT is extensively used in the simulation of transfer functions represented by finite impulse responses and also for determining the frequency-domain response of LTI systems in computer-based simulations.

The various tools and techniques presented in this chapter will find application in various parts of the succeeding chapters.

3.1. Introduction to Deterministic Signals and Systems

3.1.1. Continuous Signals

By the term *continuous signal* we mean a real or complex function of time $x(t)$, where the independent variable t is continuous.

Several particularly important continuous signals are introduced below. These signals serve as basic building blocks from which other signals can be constructed. Therefore, the use of these signals allows us to examine the properties of systems.

1. *The Unit Step Function and Related Functions.* The *unit step function* is defined as

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \quad (3.1.1a)$$

Other often used related functions are the *sign* or *signum function*

$$\begin{aligned} \text{sgn}(t) &= \begin{cases} -1, & t < 0 \\ 1, & t \geq 0 \end{cases} \\ &= 2u(t) - 1 \end{aligned} \quad (3.1.1b)$$

and the *rectangular pulse function*

$$\begin{aligned} p_a(t) &= \begin{cases} 1, & |t| \leq a \\ 0, & |t| > a \end{cases} \\ &= u(t + a) - u(t - a) \end{aligned} \quad (3.1.1c)$$

2. *The Unit Impulse Function or Delta Function $\delta(t)$.* This important signal belongs to a class of singularity functions that can be defined as distributions.⁽¹⁾ A distribution is a process of assigning a number to a function of a certain class.

The *delta function* $\delta(t)$ is defined as a functional assigning to a function $x(t)$, continuous at origin, the value $x(0)$:

$$\int_{-\infty}^{\infty} x(t)\delta(t) dt = x(0) \quad (3.1.2)$$

The following properties of the *delta function* can be derived from (3.1.2) under assumptions consistent with the formal properties of integrals:

$$\delta(at) = \frac{1}{|a|} \delta(t) \quad (3.1.3)$$

$$\int_{-\infty}^{\infty} x(t)\delta^{(n)}(t - t_0) dt = (-1)^n x^{(n)}(t_0) \quad (3.1.4)$$

$$\delta(t) = \frac{d[u(t)]}{dt} \quad (3.1.5)$$

3. *The sinc Function.* A function often encountered in analysis of communication systems is the *sinc function*,

$$\text{sinc}(t) = \frac{\sin \pi t}{\pi t} \quad (3.1.6)$$

4. *Complex Exponential Signals.* Another class of signals important in communication system analysis are complex exponential signals. A complex exponential is given by

$$x(t) = e^{j2\pi f_0 t} = \cos(2\pi f_0 t) + j \sin(2\pi f_0 t) \quad (3.1.7)$$

Since this function is complex, separate plots of real and imaginary parts are required to depict it. An important property of this signal is that it is periodic. It is easy to verify that

$$e^{j2\pi f_0(t+T_0)} = e^{j2\pi f_0 t} \quad (3.1.8)$$

where $T_0 = f_0^{-1}$ is the fundamental period.

Another particularly important set of functions are the harmonically related complex exponentials

$$x_n(t) = e^{j2\pi n f_0 t}, \quad |n| = 0, 1, 2, \dots \quad (3.1.9)$$

In characterizing systems we often use the signals described above as excitation signals.

3.1.2. Discrete-Time Signals

Discrete-time signals are defined only at discrete times $\{nT_s\}$, $n = 0, \pm 1, \pm 2, \dots$. In the context of this book, such discrete-time signals usually arise by sampling a continuous signal at instants separated by the sampling interval T_s . Since the independent variable takes on only discrete values, for analysis purposes these values can always be represented by the integers, without explicitly mentioning the value of T_s .

For the discrete-time case there are also a number of basic signals that play an important part in the analysis of signals and systems. These signals are counterparts of the continuous signals, and have similar properties and applications.

1. *The Unit Step Sequence.* The *unit step sequence* is defined as

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} \quad (3.1.10)$$

2. *The Unit Pulse or Delta Sequence.* For any integer k , the k -shifted *delta sequence* is defined as

$$\delta(n - k) = \begin{cases} 1, & n = k \\ 0, & n \neq k \end{cases} \quad (3.1.11)$$

from which it follows that

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n - k) \quad (3.1.12)$$

3. *Discrete Complex Exponential Signals.* Analogous to the continuous case, the discrete exponential signal is defined as

$$e^{j2\pi f_0 n} = \cos(2\pi f_0 n) + j \sin(2\pi f_0 n) \quad (3.1.13)$$

These signals differ from the continuous complex exponential in two significant properties:

- a. The exponential frequency f_0 is not unique. Indeed,

$$e^{j2\pi(f_0+1)n} = e^{j2\pi f_0 n} \quad (3.1.14)$$

Therefore, the frequency f_0 is usually defined in the principal range $0 \leq f_0 \leq 1$. For simulation of continuous signals, this property has a direct relation to the sampling frequency (see Section 3.5). Indeed, for a continuous exponential $\exp(j2\pi f t)$, a sampling frequency $f_s = 1/T_s$, and $t = nT_s$, the sampled continuous signal is

$$e^{j2\pi f t} \big|_{t=nT_s} = e^{j2\pi f T_s n} \quad (3.1.15)$$

The above relationship implies that we must have

$$f \leq \frac{1}{T_s} \quad (3.1.16)$$

that is, for the signal frequency to be in the principal range, it has to be smaller than the sampling frequency. The sampling theorem, however, actually imposes more stringent conditions on T_s (see Section 3.5).

b. The second property concerns the periodicity of the discrete complex exponential. For the signal to be periodic with period N we have

$$e^{j2\pi f_0(n+N)} = e^{j2\pi f_0 n} \quad (3.1.17)$$

which means $\exp(j2\pi f_0 N) = 1$. For (3.1.17) to hold, $f_0 N$ must be an integer,

$$f_0 = \frac{m}{N} \quad (3.1.18)$$

The implication of (3.1.18) is that sampled periodic signals are truly periodic only if the sampling rate is such that there is an integer number of samples per period f_0^{-1} ,

$$\frac{1}{f_0} = mT_s; \quad \frac{f_s}{f_0} = m \quad (3.1.19)$$

This condition is desirable in simulation of periodic signals, especially for Fourier or spectral analysis. For, if this condition is not met, the spectrum of the signal will show spectral lines that are not really present. Hence, for spectral analysis purposes one should be careful about the possibility of such spurious lines. However, for purposes of estimating the time-domain response, the condition (3.1.19) is not necessary as long as the sampling theorem (Section 3.5) is satisfied.

3.1.3. Systems

A system is a mapping (transformation) of a function $x(t)$ into a function $y(t)$. A continuous system is one that transforms continuous input signals into continuous output signals,

$$y(t) = \Gamma[x(t)] \quad (3.1.20)$$

Similarly, a discrete-time system is one that transforms discrete-time inputs into discrete-time outputs,

$$y(n) = \Gamma[x(n)] \quad (3.1.21)$$

3.1.3.1. Properties of Systems

1. *Systems with and without Memory.* A system is classified as *memoryless* or *instantaneous* if its output at any instant t depends at most on the input values at the same instant. If the output of the system at any instant depends also on some of the past (and in noncausal systems, also future) input values, the system is said to have *memory* and is sometimes referred to as a *dynamic system*. An example of a memoryless system is a resistor

$$v(t) = Ri(t)$$

Another example is

$$y(t) = ax^2(t) + bx(t)$$

An example of a system with memory is

$$y(n) = \sum_{k=-\infty}^n x(k)$$

2. *Causality.* A system is *causal* if the output at any time depends only on present and past values of the input. Such a system is sometimes referred to as being *nonanticipatory*. For example, the system $y(t) = x(t-1)$ is causal, while $y(t) = x(t+1)$ is not.

3. *Stability.* A system is defined as stable if for every absolutely bounded input $|x(t)| < K_0$, the output is also absolutely bounded, $|y(t)| < K_1$. For example,

$$y(t) = \int_{-\infty}^t x(\tau) d\tau$$

is unstable because for a bounded input $x(\tau) = u(\tau)$, we get the unbounded output

$$y(t) = t \quad \text{for } t > 0$$

On the other hand,

$$y(n) = \frac{1}{2M+1} \sum_{k=-M}^M x(n-k)$$

is stable, since for a bounded $x(n)$ summed over a finite set of values, $y(n)$ represents the average value of the set, which is also bounded.

4. *Time Invariance.* For continuous systems: A system is called *time-invariant* if, for any real t , a time shift in the input signal causes the same shift in the output signal:

$$\Gamma[x(t - t_0)] = y(t - t_0) \quad (3.1.22)$$

For discrete systems: A system is time-invariant if for any integer n ,

$$\Gamma[x(n - n_0)] = y(n - n_0) \quad (3.1.23)$$

5. *Linearity.* A linear system is one that possesses the property of superposition. For continuous systems: The system is linear if, for $y_1(t) = \Gamma[x_1(t)]$ and $y_2(t) = \Gamma[x_2(t)]$,

$$\Gamma[a_1 x_1(t) + a_2 x_2(t)] = a_1 y_1(t) + a_2 y_2(t) \quad \text{for any } a_1, a_2 \quad (3.1.24)$$

For discrete systems: If $x_1(n)$ and $x_2(n)$ are specific inputs to a linear system and $y_1(n) = \Gamma[x_1(n)]$ and $y_2(n) = \Gamma[x_2(n)]$ are their responses, then a system is linear if

$$\Gamma[a_1 x_1(n) + a_2 x_2(n)] = a_1 y_1(n) + a_2 y_2(n) \quad (3.1.25)$$

For example,

$$\begin{aligned} y(n) &= ax(n - m) && \rightarrow \text{linear, time-invariant} \\ y(t) &= t^2 x(t) && \rightarrow \text{linear, time-varying} \\ y(t) &= |x(t)| && \rightarrow \text{nonlinear} \end{aligned}$$

3.1.3.2. Block Diagram Representation of Systems

A system comprising a collection of subsystems may be conveniently represented by means of a signal flow block diagram which illustrates the interconnection of the system. Each subsystem is represented by a block.

A *cascade* interconnection is shown in Figure 3.1a. Here the output of subsystem Γ_1 is the input to subsystem Γ_2 , or

$$x_2(t) = \Gamma_1[x_1(t)] \quad \text{and} \quad x_3(t) = \Gamma_2[x_2(t)] \quad (3.1.26)$$

A *parallel* interconnection of two subsystems is shown in Figure 3.1b. Here the output of the system is the sum of the outputs of subsystem Γ_1 and subsystem Γ_2 , or

$$x_2(t) = \Gamma_1[x_1(t)] + \Gamma_2[x_1(t)] \quad (3.1.27)$$

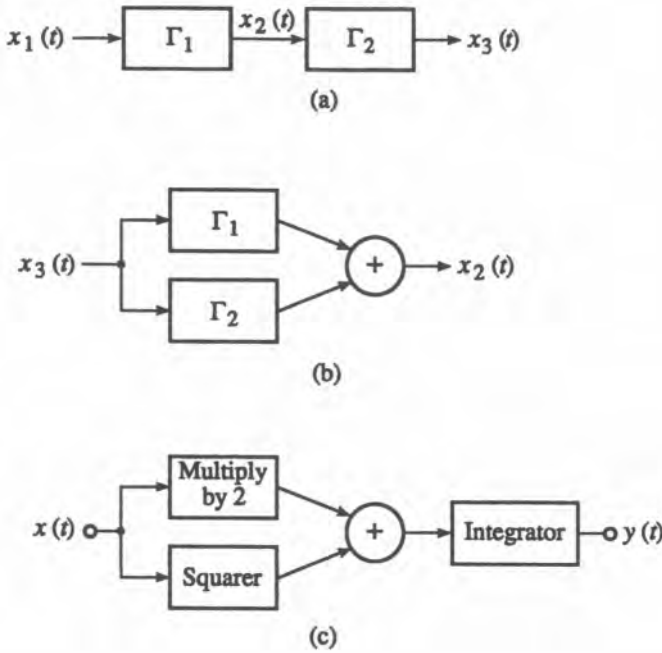


Figure 3.1. Block diagram representation of systems. (a) Cascade interconnection of systems. (b) Parallel interconnection of systems. (c) Nonlinear systems.

For example, the operation

$$y(t) = \int_0^{\infty} [2x(t) + x^2(t)] dt$$

can be represented by the block diagram in Figure 3.1c.

Block-diagram interconnections of linear time-invariant systems are described in more detail in Section 3.6.

3.2. Linear Time-Invariant Systems

In Section 3.1 we introduced a number of basic system properties. Two of these, linearity and time invariance, play a fundamental role in analysis and simulation of communication systems because of the many physical processes that can be modeled by linear time-invariant (LTI) systems. This subject is extensively covered in the literature; see, e.g., Refs. 1–6. In the simulation of communication systems this classification encompasses the whole area of spectral shaping and signal processing. The properties of continuous and discrete LTI systems are described in this section. In the following chapter we will describe the different methods used in modeling and simulation of spectral shaping and signal selection that occurs in communication systems, commonly referred to as filtering. There are several methods available with which continuous filters can be modeled for discrete simulation. Guidelines for selection of an appropriate simulation method for a given problem are described in Chapter 4.

3.2.1. Continuous Linear Time-Invariant Systems

3.2.1.1. The Impulse Response

The response to a delta function $\delta(t)$ of an arbitrary linear system defined by an operator Γ is defined as

$$h(t) = \Gamma[\delta(t)] \quad (3.2.1)$$

The function $h(t)$ is called the *impulse response*. For linear time-invariant systems

$$h(t - \tau) = \Gamma[\delta(t - \tau)] \quad (3.2.2)$$

3.2.1.2. The Convolution Integral

The response of a linear time-invariant system Γ to an arbitrary input $x(t)$ is given by

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau) d\tau \quad (3.2.3)$$

Equation (3.2.3) is referred to as the *convolution integral* or the *superposition integral*, and provides a complete characterization of a continuous-time LTI system in terms of its response to a unit impulse. Equation (3.2.3) is proved in many texts; see, e.g., Ref. 2. The convolution of the signal $x(t)$ and the impulse response $h(t)$ is denoted as

$$y(t) = x(t) * h(t) \quad (3.2.4)$$

Properties of the Convolution. The most important properties of the convolution are a consequence of the superposition properties of linear time-invariant systems.

1. The first basic property is that it is a commutative operation,

$$x(t) * h(t) = h(t) * x(t) \quad (3.2.5)$$

2. A second useful property is that it is associative,

$$x(t) * [h_1(t) * h_2(t)] = [x(t) * h_1(t)] * h_2(t) \quad (3.2.6)$$

3. The third is the distributive property,

$$y(t) = x(t) * [h_1(t) + h_2(t)] = x(t) * h_1(t) + x(t) * h_2(t) \quad (3.2.7)$$

3.2.2. Discrete Linear Time-Invariant Systems

3.2.2.1. The Impulse Response

The impulse response to an arbitrary linear time-invariant discrete system is

$$h(n) = \Gamma[\delta(n)] \quad (3.2.8)$$

3.2.2.2. Convolution Sum (Discrete Convolution)

We note from (3.2.8) and the time-invariance property that

$$\Gamma[\delta(n - k)] = h(n - k) \quad (3.2.9)$$

Hence, from the above and the linearity property, the response to $x(k)\delta(n - k)$ for any k is

$$\Gamma[x(k)\delta(n - k)] = x(k)\Gamma[\delta(n - k)] = x(k)h(n - k) \quad (3.2.10)$$

From the above and (3.1.12)

$$y(n) = \Gamma[x(n)] = \Gamma\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n - k)\right] = \sum_{k=-\infty}^{\infty} x(k)h(n - k) \quad (3.2.11)$$

The last sum is the *discrete convolution* and is denoted

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n - k) = \sum_{k=-\infty}^{\infty} x(n - k)h(k) \quad (3.2.12)$$

The procedure for evaluating the discrete convolution is illustrated in the following example.

■ **Example 3.2.1.** Consider two functions $x(k)$ and $h(k)$, as in Figure 3.2a. The convolution is evaluated as follows. $h(k)$ is “folded” and shifted by n to get $h(n - k)$, as shown in Figure 3.2b. The two functions $x(k)$ and $h(n - k)$ are multiplied and summed over k to produce $y(n)$ as shown in Figure 3.2c.

Note that the duration of the convolved signal $y(n)$ is equal to the sum of the durations of the individual signals $x(k)$ and $h(k)$ minus 1. ■

Properties of the Discrete Convolution. Similar to the continuous convolution, the discrete convolution possesses the commutative, associative, and distributive properties that are a consequence of the system being linear and time-invariant.

3.3. Frequency-Domain Representation

The signals and systems described in the previous sections can be equally well described in terms of their spectral or frequency information. The frequency-domain representation leads to a simple way of characterizing a system called the *transfer function*, also referred to as the *system function*, which is the frequently favored domain for analysis and simulation. The *Fourier transform* is used for mapping between the frequency and time domains. For linear time-invariant systems the mapping from time to frequency domain is unique, and thus the signals can be represented in either domain. This subject is widely covered in the literature; see, e.g., Refs. 7 and 8.

3.3.1. The Fourier Transform

The *Fourier transform* is a major tool in system analysis and consequently in simulation of systems. It is applied in several important areas:

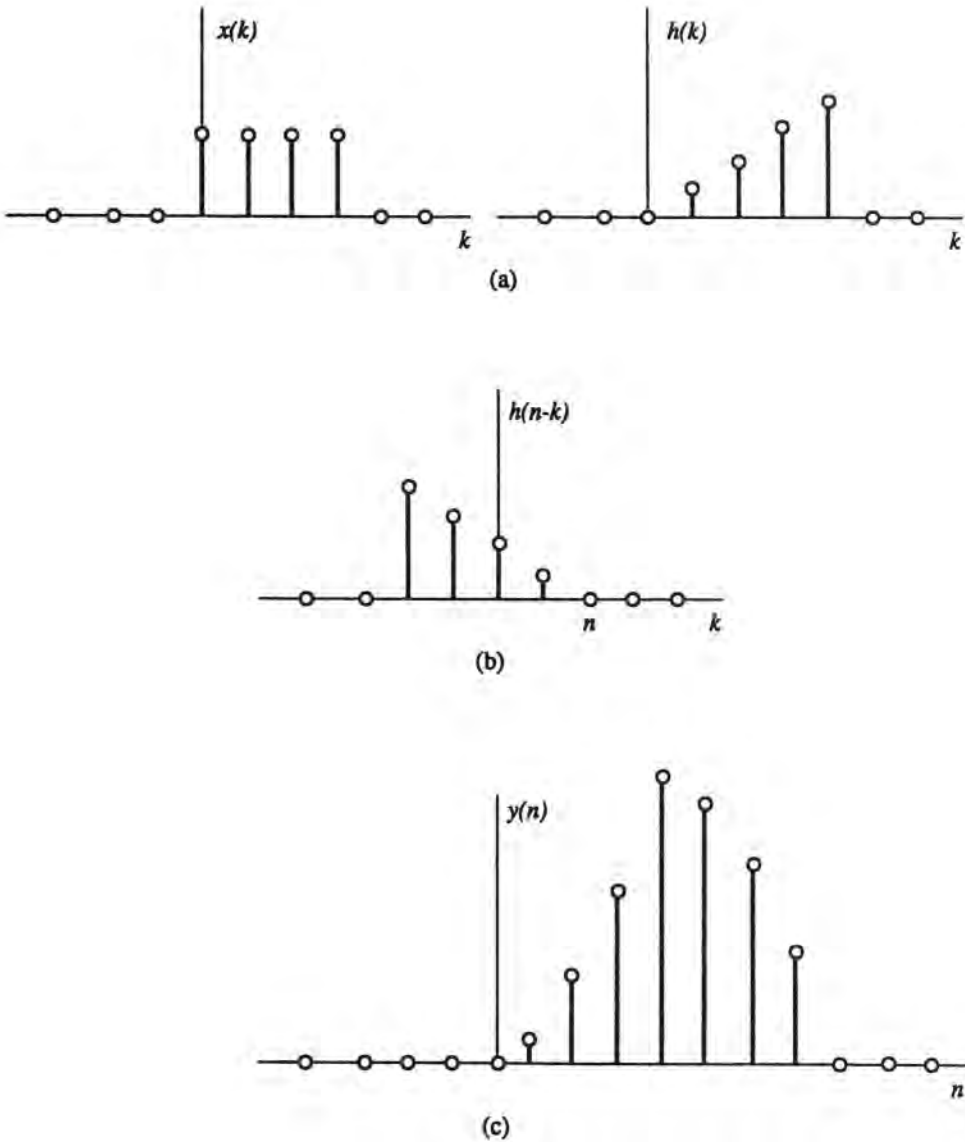


Figure 3.2. Evaluation of the convolution sum.

1. *Transformation between Frequency Domain and Time Domain.* Linear time-invariant signals and systems can be represented both in the time and frequency domains. The *Fourier integral* is the tool that enables the transformation between the two domains.
2. *Spectral Shaping/Filtering.* One of the main objectives of communication system design is spectral shaping or filtering. Simulation of filtering can be performed efficiently in the frequency domain via the *discrete Fourier transform* (DFT).
3. *Spectral Analysis.* Signals are often defined by their Fourier transform or *spectrum*. The analysis of signals in the frequency domain is one of the main applications of the

Fourier transform. The Fourier transform is also used to analyze the spectral properties of random signals; see Chapters 6 and 10.

3.3.2. Frequency-Domain Representation of Periodic Continuous Signals

3.3.2.1. The Fourier Series

A signal is *periodic* if for some positive value T_0 , $x(t) = x(t + nT_0)$ for all t . The quantity T_0 is called the period of $x(t)$.

A periodic signal can be represented by a sum of exponentials

$$x(t) = \sum_{n=-\infty}^{\infty} \alpha_n e^{j2\pi n f_0 t} \quad (3.3.1)$$

where α_n , generally a complex value, is given by

$$\alpha_n = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t) e^{-j2\pi n f_0 t} dt \quad (3.3.2)$$

The sum in (3.3.1) is referred to as the *Fourier series expansion* and α_n as the *Fourier series coefficients*.

As an example of the Fourier series representation of a time function, consider the periodic rectangular pulse train.

$$x(t) = \sum_{n=-\infty}^{\infty} p_{T_1/2}(t - nT_0) \quad (3.3.3a)$$

Fourier series coefficients for this function are

$$\alpha_n = f_0 T_1 \text{sinc}(n f_0 T_1), \quad f_0 = 1/T_0 \quad (3.3.3b)$$

Convergence of Fourier Series. The series on the right side of (3.3.1) converges to the original function $x(t)$ only under certain conditions. One such condition is square integrability, i.e.,

$$\int_0^T |x(t)|^2 dt < \infty \quad (3.3.4)$$

The above states that a periodic function $x(t)$ can be expanded into a Fourier series if the energy contained in one period is finite. For some $x(t)$ the condition (3.3.4) is more restrictive than necessary. It should be noted that at a discontinuity occurring at t_0 , the Fourier series converges to $\frac{1}{2}[x(t_0^+) + x(t_0^-)]$. A thorough discussion on convergence of Fourier series can be found, e.g., in Ref. 2.

Line Spectra. The Fourier series of a signal consists of a sum of harmonic components spaced by a frequency increment $f_0 = 1/T_0$. Let us denote

$$\alpha_n = C_x(n f_0) = A_x(n f_0) e^{j\phi_x(n f_0)} \quad (3.3.5)$$

where $A_x(n f_0)$ is the amplitude spectrum and $\phi_x(n f_0)$ is the phase spectrum of the signal $x(t)$. A periodic signal can thus be represented in the frequency domain by two plots, the amplitude spectrum and the phase spectrum.

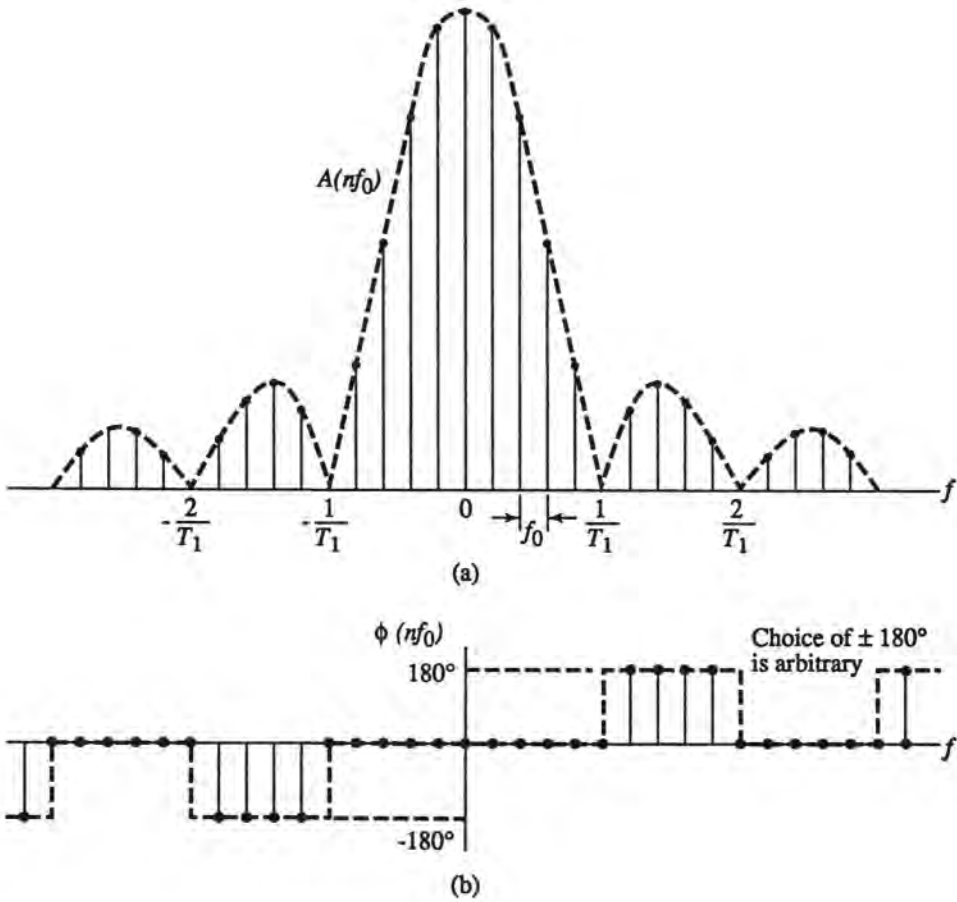


Figure 3.3. Spectrum of periodic pulse train.

As an example, the spectra of the periodic pulse train given in (3.3.3a) are shown in Figure 3.3.

3.3.2.2. Parseval's Theorem for Periodic Signals

Parseval's theorem states that the average power is the sum of the powers of the Fourier series components. That is,

$$P_{av} = \frac{1}{T_0} \int_0^{T_0} |x(t)|^2 dt = \sum_{-\infty}^{\infty} |\alpha_n|^2 = \sum_{-\infty}^{\infty} |C_x(nf_0)|^2 \quad (3.3.6)$$

A proof of this theorem can be found in, e.g., Ref. 2. The quantity $|C_x(nf_0)|^2$ is referred to as the power spectrum.

3.3.3. The Fourier Transform

In the preceding section we saw that a periodic signal can be represented by a linear combination of a discrete set of harmonically related complex exponentials. These results can

be extended to the representation of aperiodic signals via the Fourier transform. There is more than one way of arriving at the Fourier transform relationship. One popular approach⁽³⁾ is to begin with (3.3.1) and (3.3.2) and use a limiting argument as $T \rightarrow \infty$, $f_0 \rightarrow 0$, and $n \rightarrow \infty$ such that $nf_0 \rightarrow f$.

The resulting expressions are

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df \quad (3.3.7)$$

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \quad (3.3.8)$$

Equations (3.3.7) and (3.3.8) are referred to as a Fourier transform pair. The function $X(f)$ is called the *Fourier transform* or *Fourier integral* of $x(t)$, and the function $x(t)$ is called the *inverse Fourier transform* of $X(f)$. These functions are denoted as

$$X(f) = F[x(t)] \quad (3.3.9)$$

$$x(t) = F^{-1}X(f) \quad (3.3.10)$$

In analogy with the Fourier series for periodic signals, $X(f)$ is commonly called the *spectrum* of $x(t)$, since it provides us with information about the spectral composition of $x(t)$. Usually $X(f)$ is complex, and it is customary therefore to express $X(f)$ in terms of its magnitude and phase,

$$X(f) = A(f) e^{j\phi(f)} = R_x(f) + jI_x(f) \quad (3.3.11)$$

where $A(f) = |X(f)|$ is the magnitude spectrum, and $\phi(f) = \tan^{-1}[I_x(f)/R_x(f)]$ is the phase spectrum.

3.3.3.1. Convergence

For our limited exposition we can state that the Fourier transform exists if the function is square-integrable, that is,

$$\int_{-\infty}^{\infty} |x(t)|^2 dt < \infty \quad (3.3.12)$$

The above states that a Fourier transform of the function $x(t)$ exists if the total energy is finite. This condition is more stringent than necessary. Discussion of convergence can be found, for example, in Ref. 2.

Table 3.A.5 in the Appendix to this chapter shows some basic Fourier transform pairs, for many of which less restrictive conditions than (3.3.12) apply for convergence.

3.3.3.2. Properties of the Fourier Transform

Some of the simpler properties can be easily derived from the definition of the Fourier transform and will be only stated below.

1. *Linearity.* If

$$\mathbf{F}[x_1(t)] = X_1(f) \quad \text{and} \quad \mathbf{F}[x_2(t)] = X_2(f)$$

then

$$\mathbf{F}[ax_1(t) + bx_2(t)] = aX_1(f) + bX_2(f) \quad \text{for any } a, b \quad (3.3.13)$$

2. *Duality.* If $x(t)$ and $X(f)$ are a Fourier transform pair, then

$$\mathbf{F}[X(t)] = x(-f) \quad (3.3.14)$$

3. *Time Shifting:*

$$\mathbf{F}[x(t - t_0)] = X(f)e^{-j2\pi ft_0} \quad (3.3.15)$$

A shift in the time domain produces a phase shift in the frequency domain which is a linear function of f .

4. *Frequency Shifting:*

$$\mathbf{F}[e^{j2\pi f_0 t} x(t)] = X(f - f_0) \quad (3.3.16)$$

An example of the application of the frequency-shifting property is amplitude modulation,

$$\mathbf{F}[x(t) \cos(2\pi f_0 t)] = \frac{1}{2}[X(f + f_0) + X(f - f_0)] \quad (3.3.17)$$

5. *Time Scaling:*

$$\mathbf{F}[x(at)] = \frac{1}{|a|} X\left(\frac{f}{a}\right) \quad (3.3.18)$$

6. *Differentiation:*

$$\mathbf{F}\left[\frac{dx}{dt}\right] = j2\pi f X(f) \quad (3.3.19)$$

This is a particularly important property since it replaces differentiation in the time domain with multiplication by $j2\pi f$ in the frequency domain.

7. *Integration:*

$$\mathbf{F}\left[\int_{-\infty}^t x(\tau) d\tau\right] = \frac{1}{j2\pi f} X(f) + \frac{1}{2} X(0)\delta(f) \quad (3.3.20)$$

The impulse term is the average or dc value that can result from the integration.

8. *Convolution Property.* The *convolution property*, often referred to as the *convolution theorem*, is one of the most important properties of the Fourier transform because of its application in the evaluation of the system response. The convolution theorem states that

$$Y(f) = \mathbf{F}[y(t)] = \mathbf{F}[x(t) * h(t)] = X(f)H(f) \quad (3.3.21)$$

Thus the Fourier transform of the convolution of two time functions is the product of their transforms.

9. *Frequency Convolution Property.* Using the same reasoning as for the time-domain convolution property above, we can show that

$$F[x_1(t)x_2(t)] = X_1(f) * X_2(f) \quad (3.3.22)$$

■ *Example 3.3.1.* If both $x(t)$ and $h(t)$ are rectangular pulses (Figure 3.4a),

$$x(t) = h(t) = p_T(t)$$

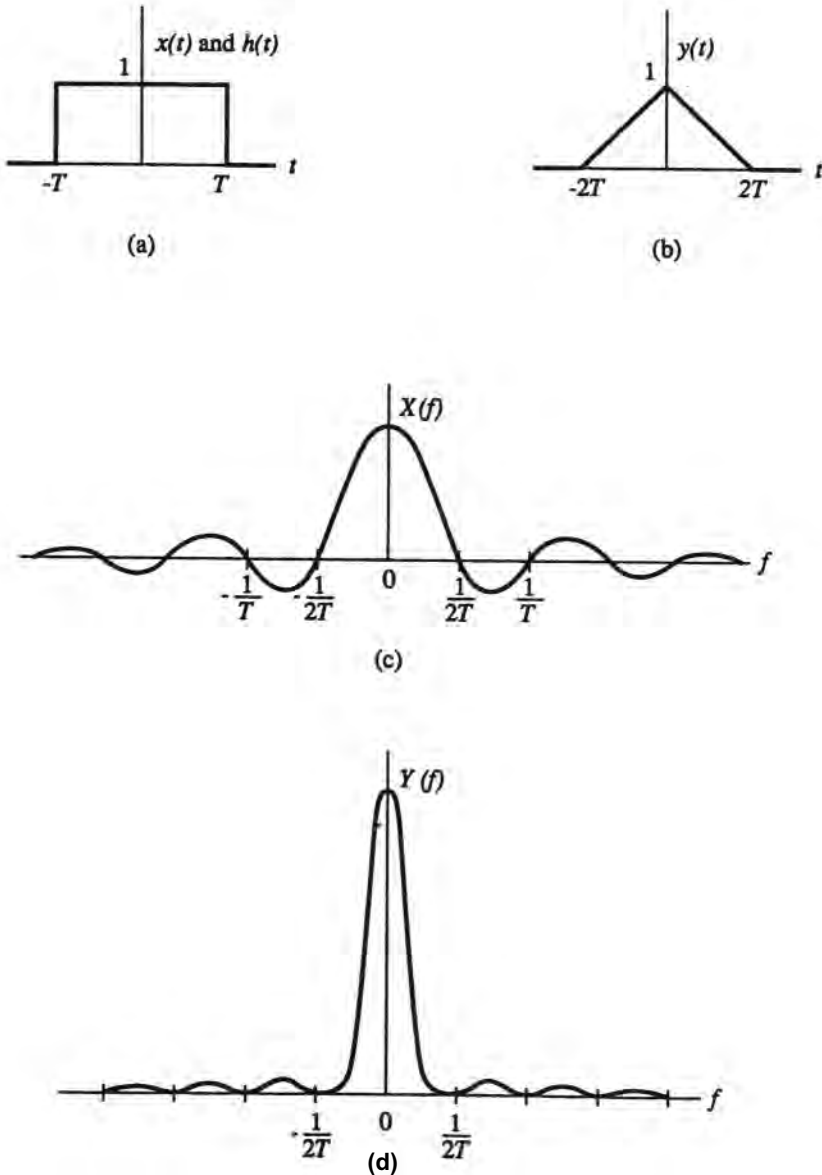


Figure 3.4. Illustration of the convolution theorem from Example 3.3.1. (a) Rectangular pulse, (b) Convolution of two rectangular pulses, (c) Spectrum of a rectangular pulse $x(t)$ and $y(t)$. (d) Spectrum of convolved pulse.

then (Figure 3.4b)

$$y(t) = x(t) * h(t) = \begin{cases} 2T - |t|, & |t| \leq 2T \\ 0, & \text{otherwise} \end{cases}$$

Since

$$\begin{aligned} X(f) &= H(f) = 2T \operatorname{sinc}(2Tf) \\ Y(f) &= 4T^2 \operatorname{sinc}^2(2Tf) \end{aligned}$$

$X(f)$ and $Y(f)$ are also shown in Figure 3.4. ■

3.3.4. The Frequency Response

When a complex exponential $e^{j2\pi ft}$ is applied to a system with an impulse response $h(t)$, the output of the system is $H(f)e^{j2\pi ft}$. Indeed,

$$\begin{aligned} y(t) &= e^{j2\pi ft} * h(t) = \int_{-\infty}^{\infty} e^{j2\pi f(t-\tau)} h(\tau) d\tau \\ &= e^{j2\pi ft} \int_{-\infty}^{\infty} h(\tau) e^{j2\pi f\tau} d\tau = e^{j2\pi ft} H(f) \end{aligned} \quad (3.3.23)$$

From (3.3.23) we see that $H(f)$ describes the change in complex amplitude of a complex exponential as a function of f . Thus $H(f)$ is referred to as the *frequency response* of the system and it plays a major role in analysis of linear time-invariant systems. In general, $H(f)$ is complex with an amplitude $A(f)$ and phase $\phi(f)$,

$$H(f) = A(f)e^{j\phi(f)}$$

Since $h(t)$ completely characterizes an LTI system, so does $H(f)$.

3.3.4.1. Interconnection of Systems in the Frequency Domain

Many of the properties of linear systems are simplified by the convolution theorem. For example, in Section 3.2 we saw that the response of a cascade connection of two systems does not depend on the order of the systems. The frequency-domain response of the cascade of two systems is the product of the frequency responses of the systems, and is therefore independent of the order of the cascade. A cascade connection of two systems can be described in the frequency domain as follows:

$$\mathcal{F}[h_1(t) * h_2(t)] = [H_1(f)] \cdot [H_2(f)] = [H_2(f)] \cdot [H_1(f)] \quad (3.3.24)$$

Further discussion on the effect of the convolution theorem on the interconnection of LTI systems is given in Section 3.6.

3.3.4.2. Parseval's Theorem for Continuous Signals

If $x(t)$ and $X(f)$ are a Fourier transform pair, then

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |X(f)|^2 df \quad (3.3.25)$$

the proof of which appears in many textbooks, eg., Ref. 3.

The integral on the left side is the total energy in the signal $x(t)$. Parseval's theorem states that the energy can be computed either by computing the energy per unit time and integrating over all time or computing the energy per unit frequency and integrating over the frequency range. The quantity

$$|X(f)|^2 = A^2(f) \quad (3.3.26)$$

is therefore often referred to as the energy-density spectrum. It should be pointed out that the total energy for a periodic signal is infinite. In this case, the Parseval's relation for average power is applicable (see Section 3.3.2.2).

3.3.5. The Gibbs Phenomenon

When we truncate the spectrum of a signal that has a discontinuity in the time domain, the resultant signal displays overshoots before and after the discontinuity. This behavior is known as the Gibbs phenomenon. The cause of the Gibbs phenomenon can be illustrated by an example. Assume that the signal $x(t) = u(t)$, the unit step function, has its spectrum truncated by an ideal lowpass filter with the transfer function $X(f) = 0$ for $|f| > B$. The impulse response of this filter (see Table 3.A.5) is $h(t) = 2B \text{sinc}(2Bt)$. The output signal is then

$$y(t) = u(t) * 2B \text{sinc}(2Bt) = 2B \int_{-\infty}^t \text{sinc}(2B\tau) d\tau \quad (3.3.27)$$

This integral cannot be evaluated in closed form, but can be expressed in terms of the sine integral function $\text{Si}(x)$, which is available in tabular form

$$\text{Si}(x) = \int_0^x \frac{\sin \eta}{\eta} d\eta \quad (3.3.28)$$

Then $y(t)$ can be expressed as

$$y(t) = \frac{1}{2} + \frac{1}{\pi} \text{Si}(2\pi Bt) \quad (3.3.29)$$

which is shown in Figure 3.5.

It is interesting to note that increasing the bandwidth B will not cause the overshoots to go away; it will only change the time scale of the oscillations without affecting their amplitude.

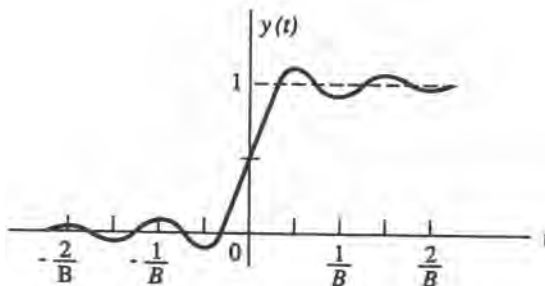


Figure 3.5. The Gibbs phenomenon.

Because of the duality property of the Fourier transform (3.3.14) the Gibbs phenomenon is also observed in the frequency domain. Indeed, truncating the impulse response of a filter whose frequency response has a discontinuity will cause Gibbs phenomenon distortion at the discontinuity of the frequency response.

To reduce the Gibbs phenomenon oscillations, we can shape the truncation into a more gradual transition. This process of shaping the frequency or impulse response to reduce the Gibbs phenomenon is called windowing. Windowing of impulse responses is described in Chapter 4 and is illustrated there by an example. Windowing is also used as a method of smoothing frequency spectra and is further described in Chapter 10.

3.3.6. Relationship between the Fourier Transform and the Fourier Series

3.3.6.1. Introduction

In the preceding section we developed the Fourier transform for aperiodic signals from the Fourier series for a periodic signal by making the period arbitrarily long. Since the Fourier series and transform are closely related, we would like to explore this relationship further. The relationships are very important since they facilitate the application of Fourier analysis to sampling and hence are particularly relevant to simulation.

3.3.6.2. Fourier Series Coefficients

We define $x(t)$ to be one arbitrary period of the periodic function $x_p(t)$

$$x(t) = \begin{cases} x_p(t), & |t| \leq T/2 \\ 0, & \text{otherwise} \end{cases} \quad (3.3.30)$$

The Fourier series coefficients are given by

$$\begin{aligned} \alpha_n &= \frac{1}{T} \int_{-T/2}^{T/2} x_p(t) e^{-j2\pi n f_0 t} dt = \frac{1}{T} \int_{-\infty}^{\infty} x(t) e^{-j2\pi n f_0 t} dt \\ &= \frac{1}{T} X(nf_0) \end{aligned} \quad (3.3.31)$$

Thus, the Fourier series coefficients of a periodic function can be computed from the Fourier transform of one arbitrary period evaluated at the discrete frequencies nf_0 and averaged over one period.

3.3.7. The Fourier Transform of a Periodic Signal

To suggest a general result, let us consider a signal with a Fourier transform that is a series of impulses

$$X(f) = \sum_{-\infty}^{\infty} \alpha_n \delta(f - nf_0) \quad (3.3.32)$$

The inverse Fourier transform is (Appendix, Table 3.A.5)

$$x(t) = \sum_{n=-\infty}^{\infty} \alpha_n e^{jn2\pi f_0 t} \quad (3.3.33)$$

Equation (3.3.33) is the Fourier series representation of a periodic signal [see (3.3.1)]. Thus, the Fourier transform of a periodic signal can be represented as a train of equidistant impulses in the frequency domain.

■ *Example 3.3.2.* A signal that is very important in the theory of sampling (see Section 3.4) is the periodic impulse train with period T , given by

$$x(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \quad (3.3.34)$$

The Fourier series coefficients for this function are

$$\alpha_n = \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) e^{-j2\pi n f_0 t} dt = \frac{1}{T}, \quad \text{where } f_0 = \frac{1}{T}$$

Inserting this into (3.3.32), we get

$$X(f) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta(f - n f_0) = F \left[\sum_{n=-\infty}^{\infty} \delta(t - nT) \right] \quad (3.3.35)$$

■

3.3.7.1. Periodic Convolution

The convolution property from (3.2.4) cannot be applied when both signals are periodic because the convolution integral would not converge. This property results from the fact that a system with a periodic impulse response is unstable. However, it is desirable to consider another form of convolution for periodic signals with equal periods. We thus define the *periodic convolution* with period T as

$$y(t) = \int_0^T x(\tau) h(t - \tau) d\tau = x(t) * h(t) \quad (3.3.36)$$

As seen in Figure 3.6, the operation is similar to the usual convolution of aperiodic signals. It is easy to show that $y(t)$ is also periodic with T .

The Periodic Convolution Property. The *periodic convolution property*, often called the *periodic convolution theorem*, can be proved by expanding $x(t)$, $h(t)$, and $y(t)$ into Fourier series whose coefficients are, respectively, $C_x(n f_0)$, $C_h(n f_0)$, and $C_y(n f_0)$. Applying (3.3.36), one can show that

$$C_y(n f_0) = T C_x(n f_0) C_h(n f_0) \quad (3.3.37)$$

The result (3.3.37) is called the *periodic convolution property*.

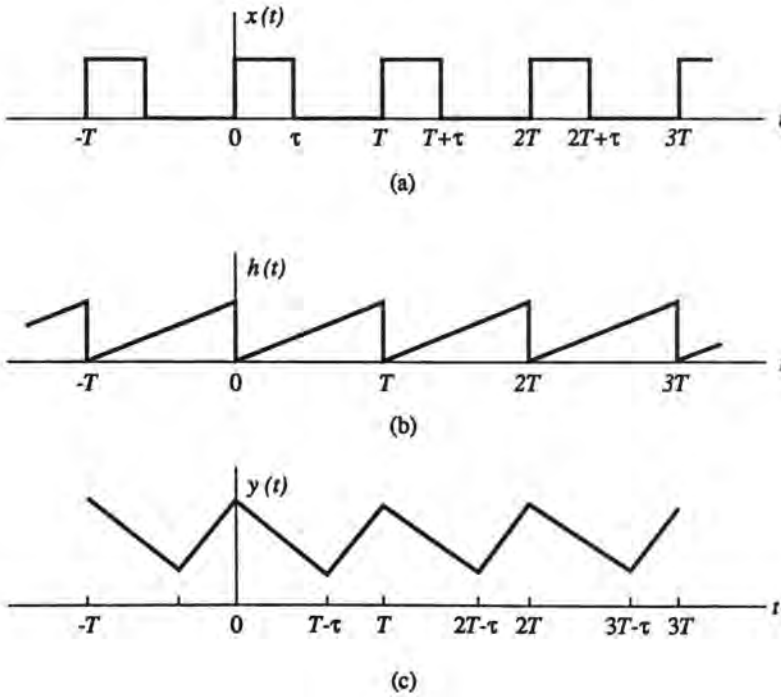


Figure 3.6. Periodic convolution.

3.3.7.2. The Poisson Sum Formula

The Poisson sum formula is an important relationship between a periodic function in the time domain and its Fourier series coefficients,⁽¹⁾

$$\sum_{n=-\infty}^{\infty} x(t + nT_0) = \frac{1}{T_0} \sum_{n=-\infty}^{\infty} e^{j2\pi n f_0 t} X(nf_0), \quad f_0 = \frac{1}{T_0} \quad (3.3.38)$$

where T_0 is an arbitrary constant.

We note that a similar formula can be derived for the dual relationship,

$$\sum_{n=-\infty}^{\infty} X(f + nf_0) = T_0 \sum_{n=-\infty}^{\infty} e^{-j2\pi n f T_0} x(nT_0), \quad f_0 = \frac{1}{T_0} \quad (3.3.39)$$

The Poisson sum formula leads very concisely to a basic result in sampling theory (Section 3.5).

3.4. Lowpass-Equivalent Signals and Systems

The time functions that we deal with in the simulation of communication systems are quite often carrier-modulated signals that are processed in bandpass systems. Since we ultimately need to sample these functions, we wish to do so in the most efficient way possible. As will be seen in the next section, a continuous signal is uniquely represented by a discrete model only if the sampling frequency is at least twice the highest frequency in the signal

spectrum. If the signal spectrum is contained in the band $f_c - B/2 \leq f \leq f_c + B/2$, a naive interpretation of the sampling theorem might indicate that the sampling frequency would have to be on the order of $2f_c + B$. Of course, as will be seen, the sampling frequency only needs to be on the order of B , which is twice the highest frequency in the *information* signal. There are “efficient” sampling schemes for bandpass signals⁽⁹⁾ that allow sampling at the rate B , but the form of the representation, using samples of the bandpass waveform, is not as simple as that for lowpass signals.

As it happens, carrier-modulated signals and systems can, with minor restrictions, be analyzed and simulated as if they were lowpass. The technique used in the implementation of this idea is the so-called *complex envelope* method, which makes use of the concepts of *lowpass-equivalent* signals and systems (see, e.g., Refs. 10 and 11).

This method is central to simulation. The idea itself is quite intuitive. Indeed, for any carrier-modulated signal $x(t)$, we can write

$$\begin{aligned} x(t) &= r(t) \cos[2\pi f_c t + \phi(t)] \\ &= \operatorname{Re}[r(t)e^{j(2\pi f_c t + \phi(t))}] \\ &= \operatorname{Re}[r(t)e^{j\phi(t)}e^{j2\pi f_c t}] \end{aligned} \quad (3.4.1)$$

where $r(t)$ is the amplitude modulation, $\phi(t)$ is the phase modulation of the signal, and f_c is the carrier frequency. The signal

$$\tilde{x}(t) = r(t)e^{j\phi(t)} \quad (3.4.2)$$

evidently contains all of the information-related variations, and is of a lowpass nature. It is often called the *complex lowpass equivalent* or the *complex envelope* of the signal. The notation in (3.4.2) will henceforth be used to denote the lowpass equivalent of $x(t)$. If the bandwidth of $x(t)$ is B , the narrowband condition $B \ll f_c$ is frequently satisfied. Under this condition, one can show that bandpass filtering of $x(t)$ can be evaluated through an equivalent lowpass filtering operation with $\tilde{v}(t)$ as an input.

The equivalent lowpass processing may be defined more formally by way of the *Hilbert transform*, which also clarifies the conditions under which equivalent lowpass processing can take place.

In the following section we give a brief introduction to the Hilbert transform. Then we will describe the mapping of modulated carrier signals into their lowpass equivalents. The modeling of lowpass equivalents of bandpass filter transfer functions, and the implementation of these functions into a lowpass-equivalent convolution to be used in simulations, will then be developed.

3.4.1. The Hilbert Transform

Given a real function $x(t)$ with a Fourier transform $X(f)$, we form the function (Figure 3.7)

$$Z_x(f) = 2X(f)u(f) \quad (3.4.3)$$

and the inverse transform

$$z_x(t) = \int_{-\infty}^{\infty} Z_x(f)e^{j2\pi ft} df = 2 \int_0^{\infty} X(f)e^{j2\pi ft} df \quad (3.4.4)$$

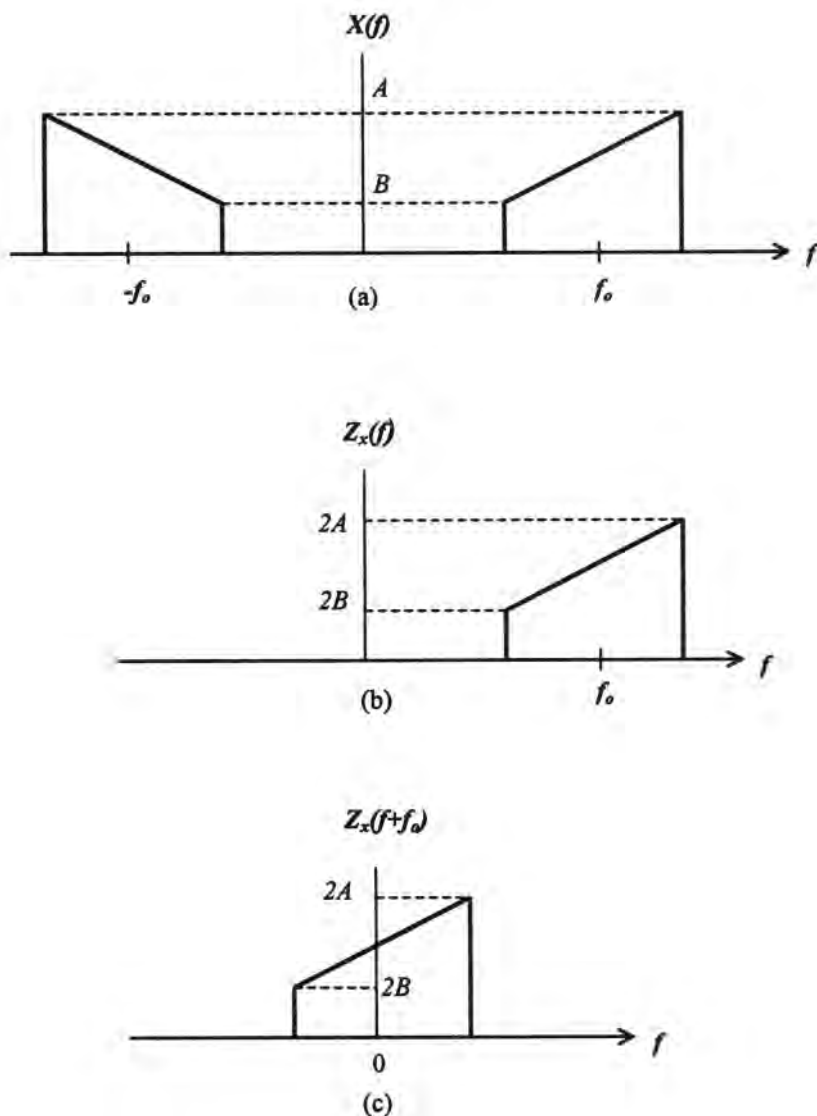


Figure 3.7. Definition of the spectrum of the analytic signal or pre-envelope of $x(t)$. (a) $X(f)$, Fourier transform of $x(t)$. (b) $Z_x(f)$, Fourier transform of the analytic signal of $x(t)$. (c) $Z_x(f+f_0)$, Fourier transform of the lowpass-equivalent signal of $x(t)$.

The function $Z_x(t)$ is complex. The real and imaginary components can be derived from the following.

Since

$$2u(f) = 1 + \text{sgn}(f) \quad (3.4.5)$$

we have

$$Z_x(f) = X(f)[1 + \text{sgn}(f)] = X(f) + X(f) \text{sgn}(f) \quad (3.4.6)$$

We have the inverse $\mathbf{F}^{-1}[X(f)] = x(t)$. We denote the inverse of $X(f) \operatorname{sgn}(f)$ as

$$\mathbf{F}^{-1}[X(f) \operatorname{sgn}(f)] = j\hat{x}(t) \quad (3.4.7)$$

where $\hat{x}(t) = \mathbf{H}[x(t)]$ is known as the *Hilbert transform* of $x(t)$

Since

$$\mathbf{F}^{-1}[\operatorname{sgn}(f)] = \frac{j}{\pi t} \quad (3.4.8)$$

the Hilbert transform can be represented as the following convolution:

$$\hat{x}(t) = -j\mathbf{F}^{-1}[X(f) \operatorname{sgn}(f)] = x(t) * \frac{1}{\pi t} \quad (3.4.9)$$

or

$$\hat{x}(t) = \mathbf{H}[x(t)] = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau \quad (3.4.10)$$

The complex function in (3.4.4),

$$z_x(t) = x(t) + j\hat{x}(t) \quad (3.4.11)$$

is referred to as the *analytic signal* of $x(t)$ or as the *preenvelope* of $x(t)$.

3.4.2. Properties of the Hilbert Transform

1. If $X(f)$ is the Fourier transform of $x(t)$, the Fourier transform of the Hilbert transform is

$$\mathbf{F}[\hat{x}(t)] = \hat{X}(f) = -jX(f) \operatorname{sgn}(f) \quad (3.4.12)$$

The above is a reiteration of (3.4.6).

A filter with a transfer function $-j \operatorname{sgn}(f)$ produces a phase shift of $\pi/2$, and is referred to as a *Hilbert transformer*.

2. The Hilbert transform of

$$x(t) = \cos(2\pi f_0 t) \quad (3.4.13a)$$

is

$$\hat{x}(t) = \sin(2\pi f_0 t) \quad (3.4.13b)$$

Because of this property, the Hilbert transformer, viewed as a filter with an impulse response $h(t) = (\pi t)^{-1}$, is also referred to as a *quadrature filter*.

3. The Hilbert transform of a Hilbert transform is the negative of the original function,

$$\mathbf{H}[\hat{x}(t)] = \hat{\hat{x}}(t) = -x(t) \quad (3.4.14)$$

From the above and (3.4.13) it follows that

$$\text{if } x(t) = \sin(2\pi f_0 t), \text{ then } \hat{x}(t) = -\cos(2\pi f_0 t) \quad (3.4.15)$$

4. If a signal $a(t)$ has a bandlimited spectrum

$$|A(f)| = 0 \quad \text{for } |f| \geq B \quad (3.4.16)$$

then the Hilbert transform of $x(t) = a(t) \cos(2\pi f_0 t)$ is given by

$$\hat{x}(t) = H[a(t) \cos(2\pi f_0 t)] = a(t) \sin(2\pi f_0 t) \quad (3.4.17)$$

and the Hilbert transform of $x(t) = a(t) \sin(2\pi f_0 t)$ by

$$\hat{x}(t) = H[a(t) \sin(2\pi f_0 t)] = -a(t) \cos(2\pi f_0 t) \quad (3.4.18)$$

provided that $f_0 \geq B$.

3.4.3. Lowpass-Equivalent Modulated Signals

An amplitude- and phase-modulated signal can always be represented as

$$x(t) = p(t) \cos(2\pi f_0 t) - q(t) \sin(2\pi f_0 t) \quad (3.4.19a)$$

with the spectrum

$$X(f) = \frac{1}{2}[P(f - f_0) + jQ(f - f_0) + P(f + f_0) - jQ(f + f_0)] \quad (3.4.19b)$$

where $P(f)$ and $Q(f)$ are the spectra of $p(t)$ and $q(t)$, respectively. If the bandwidth B of both the inphase and quadrature terms $p(t)$ and $q(t)$ is such that $B \leq f_0$,

$$\begin{aligned} P(f) &= 0, & |f| > f_0 \\ Q(f) &= 0, & |f| > f_0 \end{aligned} \quad (3.4.20)$$

then from (3.4.17) and (3.4.18) the Hilbert transform of $x(t)$ is

$$\hat{x}(t) = p(t) \sin(2\pi f_0 t) + q(t) \cos(2\pi f_0 t) \quad (3.4.21)$$

The pre-envelope is therefore

$$z_x(t) = x(t) + j\hat{x}(t) = [p(t) + jq(t)]e^{j2\pi f_0 t}$$

with

$$Z_x(f) = 2X(f)u(f) = P(f - f_0) + jQ(f - f_0)$$

The modulated signal is then

$$x(t) = \text{Re}[z_x(t)] = \text{Re}[\tilde{x}(t)e^{j2\pi f_0 t}]$$

The lowpass-equivalent signal or the complex envelope is then defined as

$$\tilde{x}(t) = z_x(t)e^{-j2\pi f_0 t} = p(t) + jq(t) \quad (3.4.22a)$$

with the spectrum

$$\tilde{X}(f) = Z_x(f + f_0) = P(f) + jQ(f) \quad (3.4.22b)$$

As shown later in this section, in the simulation of linear systems the signal $x(t)$ can be equally well represented by the complex envelope $\tilde{x}(t)$. The complex envelope is often expressed as

$$\tilde{x}(t) = a(t)e^{j\phi(t)} \quad (3.4.23)$$

where $a(t) = \sqrt{p^2(t) + q^2(t)}$ is the amplitude modulation and $\phi(t) = \tan^{-1}[q(t)/p(t)]$ is the phase modulation of the signal.

We see that for signals with bandwidth restrictions as specified in (3.4.20) the complex envelope in (3.4.22) is identical to the intuitive form of (3.4.2). The one extension beyond our intuitive notion of envelope is the fact that it is not necessary to require the usual narrowband condition $B \ll f_0$; the much weaker condition $B \leq f_0$ suffices.

3.4.4. Hilbert Transform in System Analysis

3.4.4.1. Introduction

Consider a real system $h(t)$ with a real input signal $x(t)$. The output $y(t)$ is given by the usual convolution integral

$$y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} h(t - \tau)x(\tau) d\tau \quad (3.4.24)$$

with

$$Y(f) = H(f) \cdot X(f)$$

We define the functions

$$\begin{aligned} Z_x(f) &= 2X(f)u(f) \\ Z_h(f) &= 2H(f)u(f) \\ Z_y(f) &= 2Y(f)u(f) \end{aligned} \quad (3.4.25)$$

From the above it follows that

$$Z_y(f) = \frac{1}{2} Z_h(f) Z_x(f) \quad (3.4.26)$$

If we denote the preenvelopes of $x(t)$, $y(t)$, and $h(t)$ as $z_x(t)$, $z_y(t)$, and $z_h(t)$, respectively, then from (3.4.26) we get

$$z_y(t) = \frac{1}{2} z_h(t) * z_x(t) \quad (3.4.27a)$$

and from (3.4.11)

$$y(t) = \text{Re}[z_y(t)] \quad (3.4.27b)$$

3.4.4.2. Lowpass Equivalent of a Bandpass Filter

Consider a bandlimited input signal $x(t)$ as in (3.4.20) transmitted through a bandpass system $h(t)$ with a typical transfer function $H(f)$ as shown in Figure 3.8a.

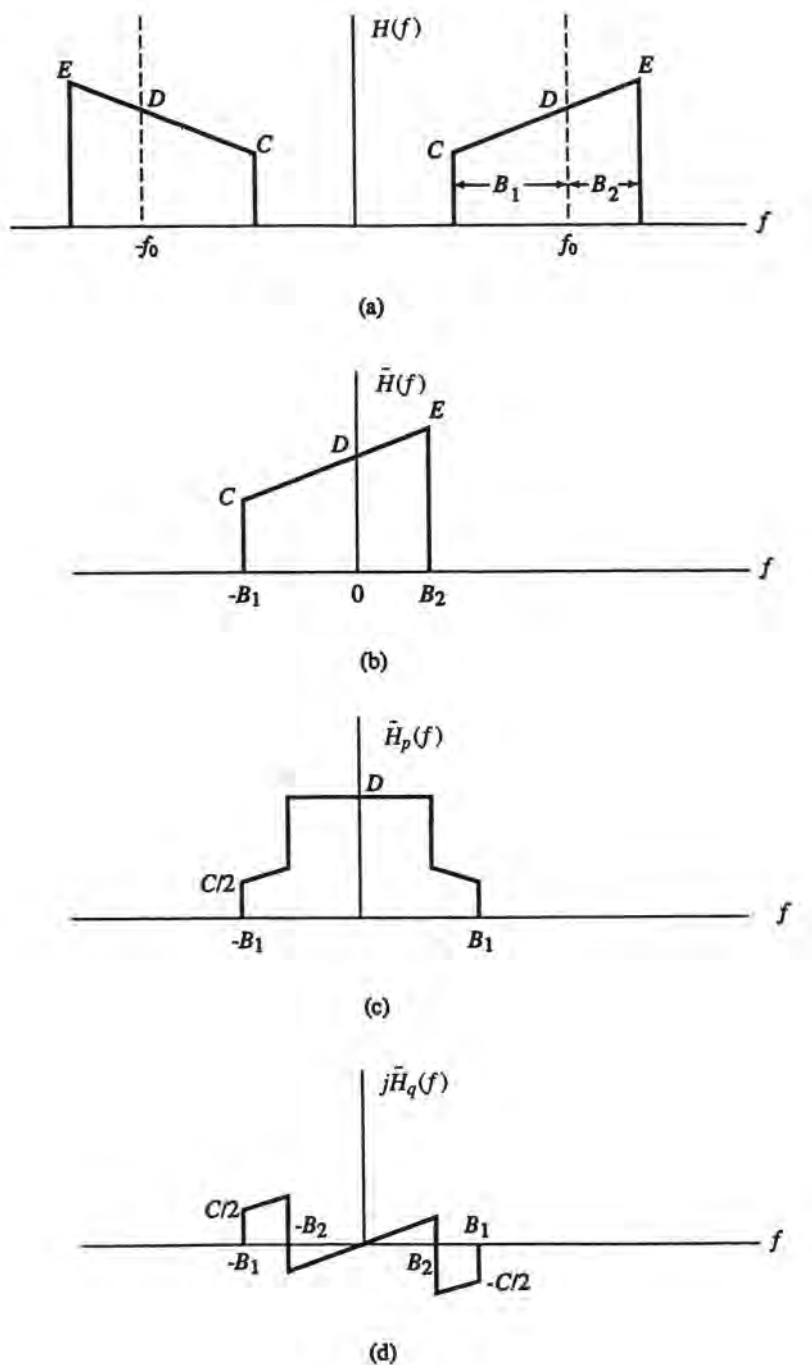


Figure 3.8. Definition of the lowpass-equivalent filter. (a) Spectrum of bandpass filter $H(f)$. (b) Spectrum of lowpass-equivalent filter $\tilde{H}(f)$. (c) Spectrum of the in-phase component of $\tilde{H}(f)$. (d) Spectrum of the quadrature component of $\tilde{H}(f)$.

The bandpass system has an output

$$y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} h(t - \tau)x(\tau) d\tau \quad (3.4.28)$$

We represent the input and output signals by their complex envelopes (lowpass equivalents) $\tilde{x}(t)$ and $\tilde{y}(t)$ as in (3.4.22). We shall show that linear filtering can be equally well represented in terms of a hypothetical filter $\tilde{h}(t)$ such that

$$\tilde{y}(t) = \int_{-\infty}^{\infty} \tilde{h}(t - \tau)\tilde{x}(\tau) d\tau \quad (3.4.29)$$

We wish now to form the lowpass-equivalent signals and filter by making explicit the frequency f_0 in (3.4.25). This will have the effect of shifting the spectra of the analytic signals and filter to the origin. The choice of f_0 is arbitrary, provided $B \leq f_0$ with B given by the larger of the two intervals B_1 and B_2 in Figure 3.8a. We have

$$\begin{aligned} \tilde{X}(f) &= Z_x(f + f_0) = 2X(f + f_0)u(f + f_0) \\ \tilde{H}(f) &= \frac{1}{2}Z_h(f + f_0) = H(f + f_0)u(f + f_0) \\ \tilde{Y}(f) &= Z_y(f + f_0) = 2Y(f + f_0)u(f + f_0) \end{aligned} \quad (3.4.30)$$

The spectrum of the lowpass-equivalent filter $\tilde{H}(f)$ is shown in Figure 3.8b. The factor 1/2 in the expression of $\tilde{H}(f)$ was introduced to simplify the forthcoming expression for the lowpass-equivalent convolution.

The complex envelopes of the input and output are $\tilde{x}(t)$ and $\tilde{y}(t)$, respectively, and $\tilde{h}(t)$ is the lowpass-equivalent impulse response:

$$\begin{aligned} \tilde{x}(t) &= z_x(t)e^{-j2\pi f_0 t} \\ \tilde{h}(t) &= \frac{1}{2}z_h(t)e^{-j2\pi f_0 t} \\ \tilde{y}(t) &= z_y(t)e^{-j2\pi f_0 t} \end{aligned} \quad (3.4.31)$$

Inserting (3.4.30) into (3.4.26), we get

$$\tilde{Y}(f - f_0) = \tilde{H}(f - f_0)\tilde{X}(f - f_0) \quad (3.4.32)$$

The pre-envelope of $y(t)$ is then

$$z_y(t) = \tilde{y}(t)e^{j2\pi f_0 t} = \int_{-\infty}^{\infty} \tilde{H}(f - f_0)\tilde{X}(f - f_0)e^{j2\pi f t} df \quad (3.4.33)$$

Substituting $f - f_0 = \zeta$, we get

$$\tilde{y}(t)e^{j2\pi f_0 t} = e^{j2\pi f_0 t} \int_{-\infty}^{\infty} \tilde{H}(\zeta)\tilde{X}(\zeta)e^{j2\pi \zeta t} d\zeta$$

The lowpass-equivalent response of the filter is then

$$\tilde{y}(t) = \int_{-\infty}^{\infty} \tilde{h}(t - \tau) \tilde{x}(\tau) d\tau = \tilde{h}(t) * \tilde{x}(t) \quad (3.4.34)$$

The output of the bandpass system can be computed from the lowpass-equivalent output by

$$y(t) = \text{Re}[\tilde{y}(t)e^{j2\pi f_0 t}] \quad (3.4.35)$$

Note that the lowpass-equivalent transfer function $\tilde{H}(f)$ can be decomposed into its in-phase and quadrature components⁽⁷⁾

$$\begin{aligned} \tilde{H}(f) &= \tilde{H}_p(f) + j\tilde{H}_q(f) \\ \tilde{H}_p(f) &= \frac{\tilde{H}(f) + \tilde{H}^*(-f)}{2} \\ \tilde{H}_q(f) &= \frac{\tilde{H}(f) - \tilde{H}^*(-f)}{2j} \end{aligned} \quad (3.4.36)$$

The amplitude of the inphase component $\tilde{H}_p(f)$ is an even function and that of the quadrature component $\tilde{H}_q(f)$ is odd (see Figures 3.8c and 3.8d). The lowpass-equivalent impulse response is then

$$\tilde{h}(t) = \tilde{h}_p(t) + j\tilde{h}_q(t) \quad (3.4.37)$$

and the impulse response is

$$h(t) = \text{Re}\left\{[\tilde{h}_p(t) + j\tilde{h}_q(t)]e^{j2\pi f_0 t}\right\}$$

The filter $\tilde{h}(t)$ is generally complex and not realizable. However, if $Z_h(f)$ is symmetric about f_0 , then $\tilde{H}(f)$ is symmetric about zero and $\tilde{h}(t) = \tilde{h}_p(t)$ is real.

3.4.5. Practical Considerations in Modeling of Lowpass Equivalents for Simulation

Earlier in this section we described the modeling of lowpass equivalents of bandpass functions using the Hilbert transform. The evaluation of the Hilbert transform in the frequency domain is usually straightforward, but it may become fairly involved in the time domain. Fortunately, the evaluation of the Hilbert transform in the time domain is seldom, if ever, necessary. The most common methods of modeling the lowpass equivalents are described below.

3.4.5.1. Signals

Carrier-modulated signals encountered in practice are generally amplitude- and phase-modulated functions that are effectively bandlimited. In order to model signals for simulation purposes, the lowpass equivalents of these signals can normally be obtained simply by mapping the inphase component of the bandpass signal into the real part of the complex envelope, and the quadrature component into the imaginary part. This is equivalent to merely expressing the modulating signal in the form of (3.4.2) or its equivalent Cartesian form, as in

(3.4.22). Hilbert transforms may need to be evaluated when the “carrier” frequency is on the order of the bandwidth, a situation that can occur in certain types of modems.

It should be noted that real modulated signals have theoretically infinite bandwidth. For practical purposes, however, these signals can be considered to have an effective finite bandwidth, often determined by empirical methods, such as *Carson’s rule*⁽¹²⁾ for FM signals.

3.4.5.2. Filters

In practical terms, the evaluation of the lowpass-equivalent transfer function $\tilde{H}(f)$ is found by truncating the negative-frequency component of the bandpass filter $H(f)$ and shifting the positive part to the zero axis by an amount equal to the carrier frequency f_c . This procedure is straightforward if the filter is defined as a finite sequence in the frequency or the time domain (see Finite impulse response filters in Chapter 4). Evaluation of the lowpass-equivalent filter when the frequency response is given in terms of a rational function of f is described in Chapter 4, which is devoted to simulation of filtering.

3.5. Sampling and Interpolation

The topic of sampling of continuous signals is of paramount importance in the simulation of communication systems since it provides the interface between the real and simulated systems. There is an extensive literature on sampling (see, e.g., Refs. 13 and 14) as well as coverage in many texts on digital communications (e.g., Refs. 12 and 15–17). Our objective is to recover, as faithfully as possible, the response of a continuous system from a discrete simulation. We will show in this section that for bandlimited signals this is possible if the sampling rate is chosen properly. As we noted above, real signals can be considered bandlimited with some suitably chosen bandwidth.

In some systems there may simultaneously exist several signals or processes with different bandwidths. In other systems the same signal may be processed so that it has a different bandwidth at different points. In these cases sampling at a single rate that is commensurate with the highest bandwidth encountered will evidently satisfy the sampling theorem, but may entail unnecessary computation, which in addition to being inefficient, may introduce roundoff errors. In principle, the most efficient processing is that for which each signal or random process is sampled at a rate just sufficient to satisfy the sampling theorem for that signal. A simulation environment in which different signals are sampled at different rates is referred to as *multirate*. Depending upon the specific nature of the system, the sampling rate of some of the signals may have to be increased or decreased at some point in the system. We shall therefore also discuss briefly multirate sampling and the associated techniques of sampling rate conversion. Interpolation plays a key role in both the reconstruction of a bandlimited signal from its samples and in sampling rate conversion. It is also important in other simulation-related contexts. Hence, we shall spend some time discussing interpolation techniques.

3.5.1. Impulse Sampling

We wish to represent a bandlimited function $x(t)$ for which $X(f) = 0$ for $|f| \geq f_M$ in terms of its values $x(nT_s)$ at a sequence of equidistant points. Let us first consider the specific

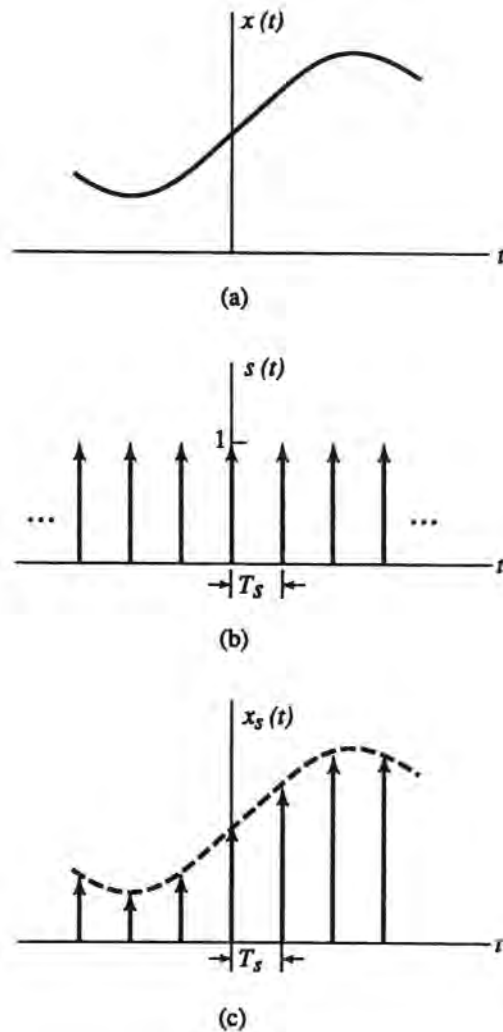


Figure 3.9. Definition of sampling. (a) Bandlimited signal $x(t)$. (b) Sampling function $s(t)$. (c) Sampled signal $x_s(t)$.

case of impulse-sequence sampling depicted in Figure 3.9. We have

$$x_s(t) = x(t)s(t) \quad (3.5.1)$$

where

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \quad (3.5.2)$$

The impulse sequence $s(t)$ is called the sampling function, T_s is the sampling period, and $f_s = T_s^{-1}$ is the sampling rate.

The function $x_s(t)$ is an impulse train with area of the impulses equal to the value of the samples of $x(t)$. That is,

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \quad (3.5.3)$$

The Fourier transform of (3.5.3) results in

$$X_s(f) = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-j2\pi nT_s f} \quad (3.5.4)$$

and from the Poisson sum formula (3.3.39) we get

$$X_s(f) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} X(f + nf_s) \quad (3.5.5)$$

$X_s(f)$ is then a periodic function of frequency consisting of a sum of scaled and shifted replicas of $X(f)$ as shown in Figure 3.10a.

In Figure 3.10b the sampling frequency is $f_s > 2f_M$ and therefore there is no overlap between the shifted replicas of the frequency spectrum. In Figure 3.10c, where $f_s < 2f_M$, there is an overlap.

In the case of no overlap ($f_s > 2f_M$; Figure 3.10b), $X(f)$ is faithfully reproduced at multiples of f_s . Therefore, $x(t)$ can be recovered exactly by means of an ideal lowpass

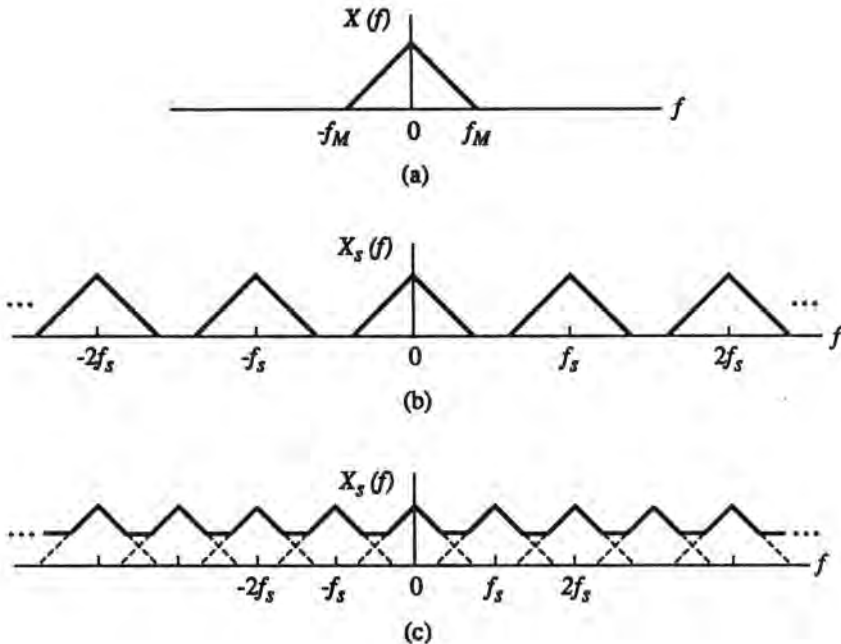


Figure 3.10. Spectrum of the sampled signal.

filter with a cutoff frequency $f_M \leq f_c \leq f_s - f_M$. Thus the recovered signal in the frequency domain is

$$X(f) = X_s(f) \cdot H_L(f) \quad (3.5.6)$$

where $H_L(f) = T_s p_{f_c}(f)$ is an ideal lowpass filter with a cutoff frequency f_c and magnitude T_s . The impulse response of such a filter is

$$\mathcal{F}^{-1}[H_L(f)] = h_L(t) = 2T_s f_c \text{sinc}(2f_c t) \quad (3.5.7)$$

The recovered signal is thus

$$x(t) = x_s(t) * h_L(t) \quad (3.5.8)$$

where $x_s(t)$ is given in (3.5.3), so that

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT_s) 2f_c T_s \text{sinc}[2f_c(t - nT_s)] \quad (3.5.9)$$

It is clearly seen from Figure 3.10 that the smallest sampling frequency for which the signal is recoverable is

$$f_s = 2f_M \quad \text{and} \quad f_c = f_M = \frac{f_s}{2} = \frac{1}{2T_s} \quad (3.5.10)$$

Equation (3.5.9) then becomes

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT_s) \text{sinc}[f_s(t - nT_s)] \quad (3.5.11)$$

Equation (3.5.11) represents an *interpolation* with the sinc function. An example of such an interpolation is illustrated in Figure 3.11.

3.5.2. Sampling Theorem

The above fundamental result is known as the sampling theorem, which can be stated as follows:

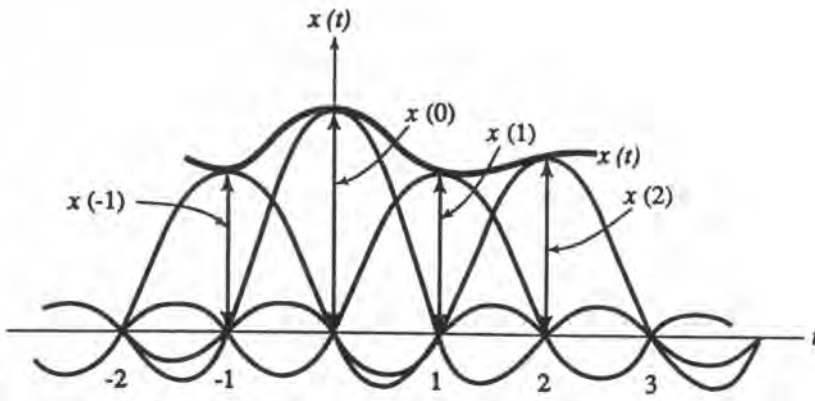


Figure 3.11. Reconstruction of a signal from its samples.

- A bandlimited signal $x(t)$ with $X(f) = 0$ for $|f| \geq f_M$ is uniquely determined by its sample values $x(nT_s)$ at a sequence of equidistant points $t = nT_s$ if $f_s > 2f_M$, where $f_s = 1/T_s$. The sampling frequency $f_s = 2f_M$ is known as the Nyquist rate.

Aliasing: The Effect of Undersampling. We saw in the preceding section that $x(t)$ could be reconstructed from $x_s(t)$ through an ideal lowpass filter having a bandwidth $f_M \leq f_c \leq f_s - f_M$. When $f_s < 2f_M$, $X(f)$ is no longer replicated in $X_s(f)$ and thus is not recoverable by filtering. It is clearly seen from Figure 3.10c that the recovered spectrum is not the same as $X(f)$.

In reality some amount of aliasing is always present because practical signals are never strictly bandlimited. Aliasing distortion of random signals is discussed in Chapter 6.

3.5.3. Multirate Sampling and Sampling Conversion

As indicated earlier, if there are two (or more) processes in a system with greatly different bandwidths, and if we use a single sampling rate, then that rate must accommodate the higher bandwidth waveform. This implies some inefficiency because there is unnecessary computation associated with the lower bandwidth signal. Generally, the most efficient situation would be to sample each process at exactly the Nyquist rate for that process. This is the motivation for multirate processing, but multirate processing is not merely sampling each process at its own necessary rate. At some stage of simulation, sampling rate conversion will be required. Sampling rate conversion simply means that the sampling rate of the waveform will be increased or decreased. Converting to a higher sampling rate is called *interpolation* and converting to a lower rate is called *decimation*⁽¹⁸⁾; these are also referred to as *upsampling* and *downsampling*, respectively⁽¹⁹⁾.

To illustrate the need for sampling rate conversion, let us consider the following examples. First, suppose we have a desired signal of bandwidth B_1 and an interfering signal of bandwidth $B_2 < B_1$, which initially might have been generated at $2B_1$ and $2B_2$ samples/s, respectively. If at some point in the system these signals must be added, we must arrange for values of each of them to be available at the same instants. The logical solution is to interpolate the lower bandwidth signal so as to produce samples of it at the same instants as those of the higher bandwidth signal. As a second example, consider a set of N contiguous frequency-multiplexed signals, each of bandwidth B . The sampling rate for the combined signal is $2NB$, while each signal might originally have been generated at $2B$ samples/s. In order to combine the N signals, we would thus have to interpolate each up to the rate $2NB$ to produce coincident samples at that rate. Subsequent to passing through the system, we want to demultiplex the group and process each signal, or a typical signal, individually. At that point, we would want to decimate any one such signal back to a rate of $2B$ samples/s. As a third example, let us consider a spread-spectrum system with information bandwidth B_1 and spread bandwidth B_2 . All of the processing that takes place before spreading can operate on a stream of $2B_1$ samples/s. But, just before applying the spreading waveform, the information signal must be interpolated to $2B_2$ samples/s so that its samples are coincident with those of the spreading waveform. After despreading, the signal can be decimated to its natural rate of $2B_1$ samples/s. An application of multirate and sampling rate conversion techniques will be given in Case Study IV, Chapter 12.

The computational saving due to using multirate sampling is a function of the specific system considered, in particular the amount of processing that takes place at either high or low bandwidth. Let N_L be the number of computations (say, equivalent multiplications or additions) associated with low-bandwidth portion of the system, *when the sampling is done at*

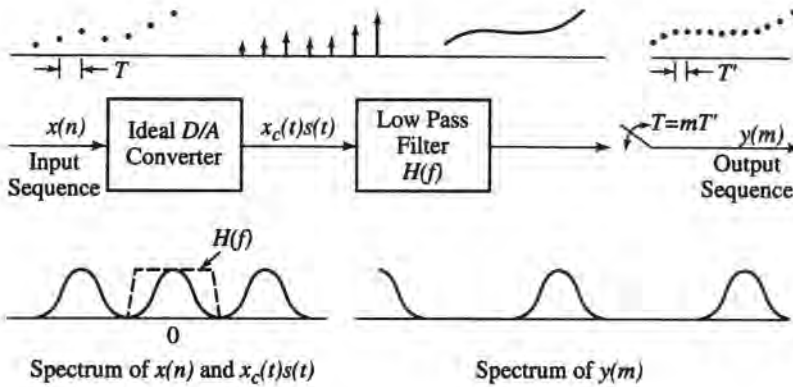


Figure 3.12. Conceptual interpretation of sampling rate conversion.

the high rate, and let N_H be the number of computations for the high-bandwidth portion of the system. Let R_H and R_L be the sampling rates corresponding to the high and low bandwidths, respectively, and assume that the number of computations is directly proportional to the number of samples. Then, using multirate sampling, the number of computations would be $N_H + N_L(R_L/R_H)$ instead of $N_H + N_L$ that would otherwise be required, the improvement factor is then $(N_H + N_L)/[N_H + N_L(R_L/R_H)]$. Its largest value is reached when $R_L/R_H \rightarrow 0$, and is $(1 + N_L/N_H)$. In actuality, the improvement due to the use of multirate processing is lessened to the degree that computational work must be done in the sampling rate conversion process. In general, it will not be worthwhile to use multirate processing if the bandwidth of the processes involved do not differ by relatively large factors.

In concept, multirate sampling is straightforward. As previously implied, it is simply an application of the sampling theorem. If a signal is represented by equispaced samples having a rate appropriate to the bandwidth, the signal can be reconstructed as a continuous signal by the process of interpolation. This much we know is possible, from (3.5.11). It can then be resampled at more closely spaced samples (but not the reverse) in preparation for higher rate processing. This process is illustrated in Figure 3.12. Samples arrive at spacing equal to T . The ideal digital-to-analog converter produces a sequence of delta functions with weights equal to the values of the samples. The lowpass filter reconstructs the analog waveform which is now sampled at m times the original rate.

Figures 3.13 and 3.14 illustrate the process of decimation. The symbols f_s and f'_s represent sampling rates. Now, if the original sampling rate f_s is appropriate for the signal, aliasing will occur if we sample at a lower rate unless we first reduce the bandwidth of the signal. This is the purpose of the filter with unit sample response $h(n)$. As can be seen in the spectral diagram, after the filter, which does not change the sampling rate, the signal is oversampled, which is evidenced by the large separation between the spectral peaks. The sampling rate compressor in effect retains only every m th sample, as illustrated in Figure 3.14. This can be viewed as multiplying the sample sequence $w(n)$ by a train of unit samples, as shown. The effect in the spectral domain is to convolve a sequence of delta functions with the spectrum of $w(n)$, which results in the final spectrum $Y(f)$. The spectra are repetitive because this is a property of sampled functions. Decimation, then, consists in retaining every m th sample and is easy to implement.

Interpolation, on the other hand, although conceptually straightforward, is computationally much more demanding. Generally, we do not have to reconstruct the analog wave-

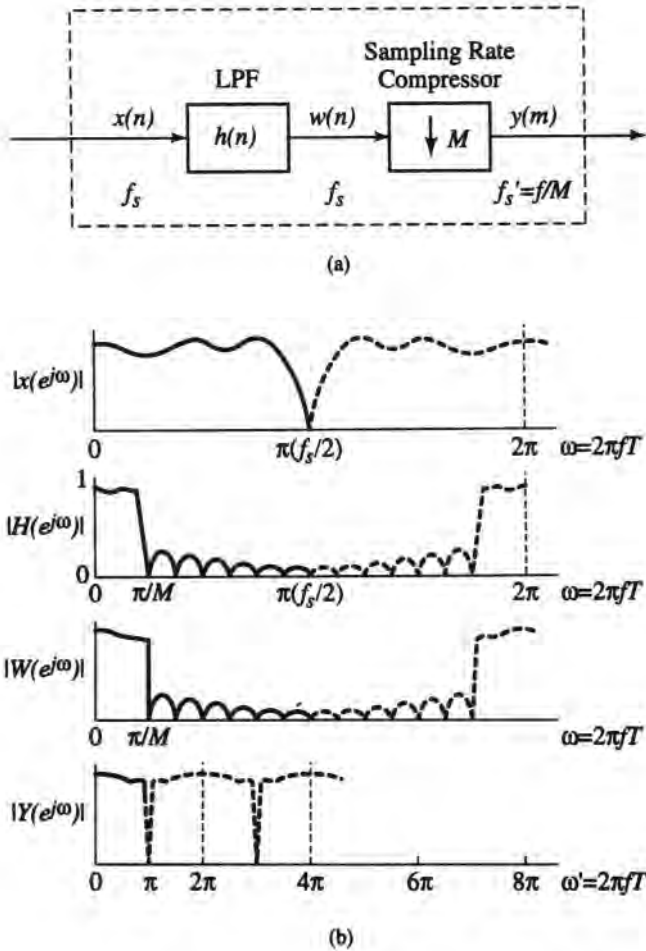


Figure 3.13. Illustration of the process of decimation by an integer factor M (from R. E. Crochiere and L. R. Rabiner, Optimum FIR digital filter implementations for decimation, interpolation and narrow-band filtering, IEEE Trans. On Acoust., Speech Signal Process., Vol. Assp.-23 No. 5, pp. 441–456, Oct 1975. © IEEE, 1975).

form (which, of course, we cannot do in simulation) in order to compute the interpolated samples. Instead, we can use the direct “digital-to-digital” approach, which is possible if f_s/f'_s is a ratio of integers. The complexity of the interpolation depends on the relationship between f_s and f'_s . One conceptualization of the process for the case when $f'_s = Lf_s$, L an integer, is shown in Figure 3.15 for $L = 3$. As indicated, the first step is to introduce $(L - 1)$ samples of value zero between the low-rate input samples, so that the new sample sequence, including the zeros, has the desired sample rate f'_s . The interpolating filter then produces the new interpolated sequence. The design of this filter is explored in the next section.

3.5.4. Interpolation

Interpolation is the key operation in the use of multirate techniques. It is thus important to make it as efficient as possible, so as not to undo the potential advantage of such techniques. Quite apart from multirate applications, interpolation is often necessary for other

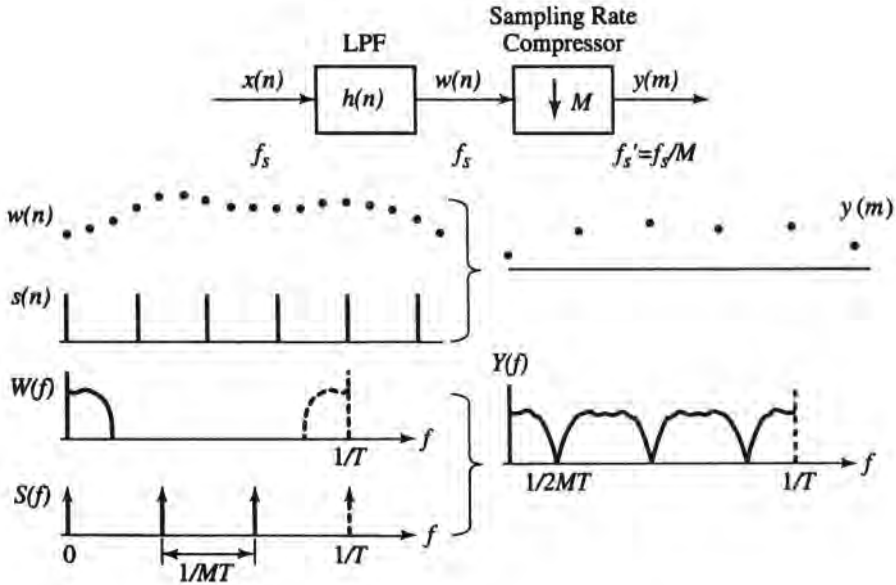


Figure 3.14. Further interpretation of decimation by an integer factor M ; $W(f)$ is the Fourier transform of $w(n)$, and $S(f)$ is the Fourier transform of $s(n)$.

purposes in simulation. In the next few subsections, therefore, we will highlight some of the relevant considerations[†].

3.5.4.1. Introduction

Let $\{x(kT_s)\}$, $k = 0, \pm 1, \pm 2, \dots$, be a sequence of samples of the waveform $x(t)$, separated by $T_s = f_s^{-1}$. An estimate of $x(t)$ for any t , call it $y(t)$, can be obtained from a general interpolation rule of the form¹

$$y(t) = \sum_{k=-\infty}^{\infty} T_s x(kT_s) h(t - kT_s) \quad (3.5.12)$$

where h is an interpolating function (or filter) with the properties

$$T_s h(0) = 1; \quad h(kT_s) = 0 \quad \text{for } k \neq 0 \quad (3.5.13)$$

The conditions (3.5.13) ensure the requirement that $y(nT_s) = x(nT_s)$. The frequency-domain version of (3.5.12) is

$$Y(f) = H(f) \sum_{l=-\infty}^{\infty} X(f + lf_s) \quad (3.5.14)$$

where X , Y , and H are the Fourier transforms of $x(t)$, $y(t)$, and $h(t)$, respectively. If $X(f) = 0$ for $|f| \geq B$ and $T_s \leq (2B)^{-1}$, it is clear that $y(t)$ perfectly reconstructs $x(f)$ if $H(f) = 1$ for $|f| \leq B$ and zero elsewhere. This is merely a reiteration of the sampling theorem. We shall return to this point later.

[†] Interpolation, not necessarily limited to the current context, is a discipline of some breadth, see, e.g., Refs. 20 and 21. Of necessity, our coverage here is limited.

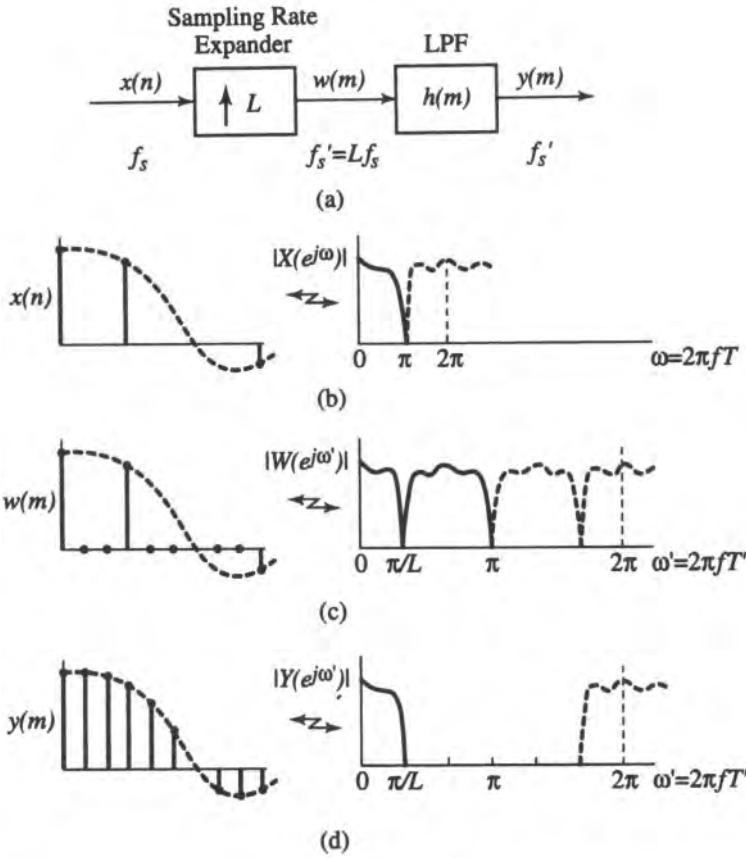


Figure 3.15. Illustration of the process of interpolation by an integer factor L (from R. E. Crochiere and L. R. Rabiner, Optimum FIR digital filter implementations for decimation, interpolation and narrow-band filtering, IEEE Trans. On Acoust., Speech Signal Process., Vol. Assp-23 No. 5, pp. 441–456, Oct. 1975. © IEEE, 1975).

For simplicity, in the following we will assume the constant T_s in (3.5.12) is absorbed in h . If we want to reconstruct a sequence of samples $\{y(mT_s')\}$, $m = 0, \pm 1, \pm 2, \dots$, at some different spacing T_s' , we can simply set $t = mT_s'$ in (3.5.12), so that[†]

$$y(mT_s') = \sum_{k=-\infty}^{\infty} x(kT_s)h(mT_s' - kT_s) \quad (3.5.15)$$

In general, of course, neither in (3.5.15) nor in (3.5.12) can we implement infinite limits in the sums for arbitrary h , although in particular cases a closed-form expression is conceivable or a recursive form may be possible. A more practical version of (3.5.15) would be to approximate h as a finite impulse response filter

$$y(mT_s') = \sum_{k=K_1(m)}^{K_2(m)} x(kT_s)h(mT_s' - kT_s) \quad (3.5.16)$$

[†] In general, T_s' may be larger or smaller than T_s . In the former case, $x(t)$ must have been sampled at faster than the Nyquist rate to avoid aliasing. Here, we are interested in $T_s' \leq T_s$, which is always permissible if the original samples satisfy the Nyquist rate.

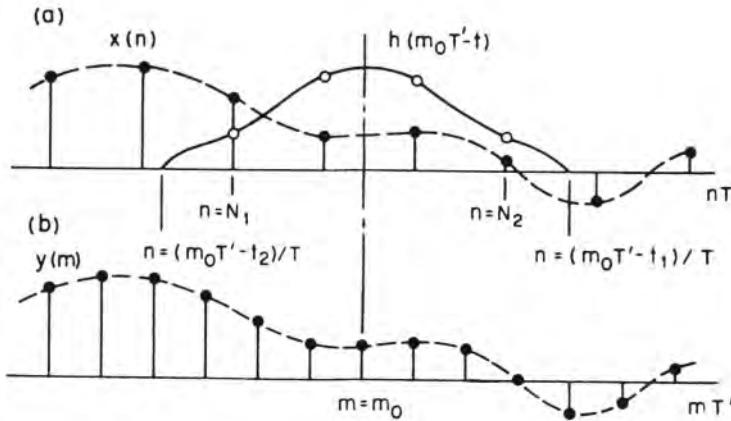


Figure 3.16. Illustration of the relationship between samples of x , h , and y , for arbitrary T_s and T'_s .

which has K terms[‡] to compute for each m , where $K = K_2(m) - K_1(m) + 1$. It is useful to write a few terms of (3.5.16) to get a sense of its nature:

$$y(mT'_s) = x(K_1 T'_s)h(mT'_s - K_1 T_s) + \cdots + x(K_2 T'_s)h(mT'_s - K_2 T_s) \quad (3.5.16a)$$

which is further illustrated in the sketch of Figure 3.16. An examination of this figure, as well as (3.5.16), indicates that the endpoints in the sum (3.5.16), i.e., the specific samples of x for each calculation of $y(mT'_s)$, are a complicated function of m , T'_s , T_s , and the filter endpoints t_1 and t_2 , which define the nonzero range of h . Even more important, for an arbitrary relationship between T'_s and T_s , for each value of m it may be possible that a *distinct* set of samples of $h(t)$ would need to be used to produce $y(mT'_s)$. Hence, a new set of K filter coefficients would (or might) have to be computed before applying the interpolation formula. Evidently, this would make the process of interpolation inefficient, especially if K were relatively large and h were not a simple function. This potential problem can be remedied to a large extent by imposing the reasonable restriction that T'_s/T_s be a ratio of integers.

Thus, we assume that

$$\frac{T'_s}{T_s} = \frac{M}{L} \quad (3.5.17)$$

where M and L are integers. Note that when $L = 1$ we have decimation by the factor M , and when $M = 1$ we have interpolation by the factor L . Under the condition (3.5.17), the seemingly simple sum (3.5.16) is susceptible to a remarkable number of nontrivial rearrangements and structural interpretations⁽¹⁸⁾ For example, by introducing the change of variable

$$n = \left\lfloor \frac{mT'_s}{T_s} \right\rfloor - k$$

[‡] In general, K may not have the same value for all m .

where $\lfloor u \rfloor$ denotes the largest integer equal to or less than u (also called the floor function), we can reformulate (3.5.16) as

$$y(mT'_s) = \sum_{n=N_1}^{N_2} g_m(n)x(\lfloor mM/L \rfloor - n) \quad (3.5.18a)$$

where

$$g_m(n) = h((n + \delta_m)T_s) \quad (3.5.18b)$$

and

$$\delta_m = \frac{1}{L}(nM \text{ modulo } L) \quad (3.5.18c)$$

The sum limits N_1 and N_2 are, respectively, $\lceil t_1/T - \delta_m \rceil$ and $\lfloor t_2/T - \delta_m \rfloor$, where $\lceil u \rceil$ denotes the smallest integer greater than or equal to u (also called the ceiling function). From (3.5.18c), it is clear that δ_m can take on only L distinct values $0, 1/L, \dots, (L-1)/L$, so that there are now only L distinct sets of filter coefficients possible. Different arrangements of these coefficient sets define different computing structures. Below we will concentrate on the case when $M = 1$, i.e., pure interpolation by the factor L .

3.5.4.2. Interpolator Structures for Integer Upconversion

As just stated, we will now assume $M = 1$, so that (3.5.18) reduces to

$$y(mT'_s) = \sum_{n=N_1}^{N_2} g_m(n)x(\lfloor m/L \rfloor - n) \quad (3.5.19a)$$

where $g_m(n)$ is notationally the same,

$$g_m(n) = h((n + \delta_m)T_s) \quad (3.5.19b)$$

but now

$$\delta_m = \frac{1}{L}(n \text{ modulo } L) \quad (3.5.19c)$$

An alternative form of (3.5.19c) is illuminating. Using the fact that $T_s = LT'_s$, we can rewrite (3.5.19b) as

$$g_m(n) = h((nL + m \bmod L)T'_s), \quad \text{all } m, n \quad (3.5.20)$$

Of course, (3.5.20) is the same as (3.5.19b), but it more clearly reveals that the digital filter $g_m(n)$ has coefficients that are sampled values of h spaced by T'_s . In other words, h can be viewed as a “high-rate” filter $h(m)$ working at the rate f'_s . In this light, we can give a meaningful interpretation to the filter in the architecture of Figure 3.15. The input sequence to $h(m)$ must arrive at the same rate. Hence, it is natural to define a modified input sequence

$$w(m) = \begin{cases} x(m/L), & m = 0, \pm L, \pm 2L, \dots \\ 0, & \text{other values of } m \end{cases} \quad (3.5.21)$$

which coincides with the input sequence at the proper instants, but with zeros filled in to make the rate Lf_s . The interpolated output is then of the form

$$y(m) = \sum h(m-k)w(k) \quad (3.5.22)$$

where h is actually given by (3.5.20). The main point of (3.5.21)–(3.5.22) is to reinforce the interpretation of interpolation as the digital processing operation shown in Figure 3.15.

■ *Example 3.5.1.* A simple illustration of an interpolation as defined in (3.5.22) for an input sequence $x(n)$ and an interpolating filter $h(n)$ for $L = 2$ and $N = 8$, is shown in Figure 3.17. [We define $N = N_2 - N_1 + 1$, where N_1, N_2 are the limits in (3.5.19a).] The interpolated sequence $y(m)$ for $m = 0, \dots, 11$ is

$$y(0) = h(0)x(0)$$

$$y(1) = h(1)x(0)$$

$$y(2) = h(0)x(1) + h(2)x(0)$$

$$y(3) = h(1)x(1) + h(3)x(0)$$

$$y(4) = h(0)x(2) + h(2)x(1) + h(4)x(0)$$

$$y(5) = h(1)x(2) + h(3)x(1) + h(5)x(0)$$

$$y(6) = h(0)x(3) + h(2)x(2) + h(4)x(1) + h(6)x(0)$$

$$y(7) = h(1)x(3) + h(3)x(2) + h(5)x(1) + h(7)x(0)$$

$$y(8) = h(0)x(4) + h(2)x(3) + h(4)x(2) + h(6)x(1) + h(8)x(0)$$

$$y(9) = h(1)x(4) + h(3)x(3) + h(5)x(2) + h(7)x(1)$$

$$y(10) = h(0)x(5) + h(2)x(4) + h(4)x(3) + h(6)x(2) + h(8)x(1)$$

$$y(11) = h(1)x(5) + h(3)x(4) + h(5)x(3) + h(7)x(2)$$

It can be seen that only even-numbered coefficients are associated with even-numbered outputs, while only odd-numbered coefficients are involved in the computation of the odd-numbered outputs. ■

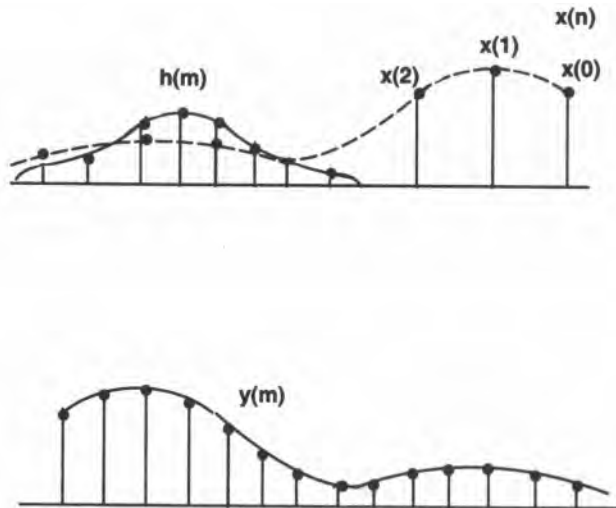


Figure 3.17. Illustration of integer upsampling for $L = 2$ and $N = 8$.

The notation $\mathbf{g}_m(n)$ is intentionally suggestive; that is, it represents a linear, periodically time-varying filter with period L in the variable m :

$$\mathbf{g}_m(n) = \mathbf{g}_{m+rL}(n), \quad r = 0, \pm 1, \pm 2, \dots$$

In it is important to reiterate that (3.5.20) implies that there are only L distinct sets of filter coefficients. That is, to generate each output sample at the high rate $y(m)$, $m = 0, 1, \dots, L-1 \pmod{L}$, a different filter coefficient set $\mathbf{g}_m(n)$ is used. In other words, $y(0), y(L), y(2L), \dots$, are generated using the same “filter” $\mathbf{g}_0(n)$; $y(1), y(L+1), y(2L+1), \dots$, are generated using the same “filter” $\mathbf{g}_1(n)$, and so on.[†] Equally important to notice is that the term $\lfloor m/L \rfloor$ in (3.5.19) increases by one for every L samples of y . Thus, the coefficients $\mathbf{g}_m(n)$ associated with the output samples $y(L), y(L+1), \dots, y(2L-1)$ multiply the samples $x(1-n)$, and in general the output samples $y(rL), y(rL+1), \dots, y(rL+L-1)$ utilize the samples $x(r-n)$. It can be seen, therefore, that $x(n)$ in (3.5.19) is updated at the low sampling rate f_s (as we would expect, of course, since that is the rate at which x is presented), while y is generated at the rate Lf_s .

Another important observation is that the coefficients within each of the subsets $\{\mathbf{g}_m(n)\}_{m=0}^{L-1}$ are separated by T_s , while each subset is offset from the adjacent one by T'_s . Thus, the partitioning of $\mathbf{g}_m(n)$ into these subsets suggests the computational structure shown in Figure 3.18. In this figure, the filter in each branch $\mathbf{p}_k(n)$ is identical to $\mathbf{g}_k(n)$:

$$\mathbf{p}_k(n) = \mathbf{g}_k(n) \quad \text{for } k = 0, 1, \dots, L-1 \quad \text{and all } n \quad (3.5.23a)$$

Furthermore, from (3.5.20), we see that

$$\mathbf{p}_k(n) = \mathbf{h}(nL + k) \quad \text{for } k = 0, 1, \dots, L-1 \quad \text{and all } n \quad (3.5.23b)$$

The coefficient sets $\mathbf{p}_k(n)$ are referred to as *polyphase filters*, and can be seen from Figure 3.18 to be interpretable as linear *time-invariant* filters operating at the low rate f_s . Notwithstanding the equality in (3.5.23a), the reason for relabeling these filters is to make explicit the subtlety that the $\mathbf{p}_k$ are time-invariant filters, while the $\mathbf{g}_k$ are subsets of a time-varying filter.¹⁸

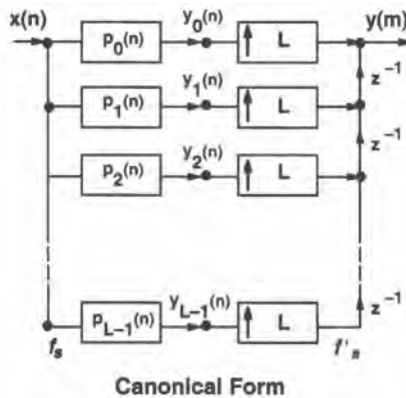


Figure 3.18. Canonical form of a polyphase network [from R. E. Crochiere and L. R. Rabiner, Interpolation and decimation of digital signals—A tutorial review, *Proc. IEEE* **69**(3), 300–331 (1981), © IEEE, 1981].

[†]Although $\mathbf{g}_0(n), \mathbf{g}_1(n), \dots$ have been called filters in quotes because they are subsets of h , they can legitimately be so interpreted, as will be seen shortly.

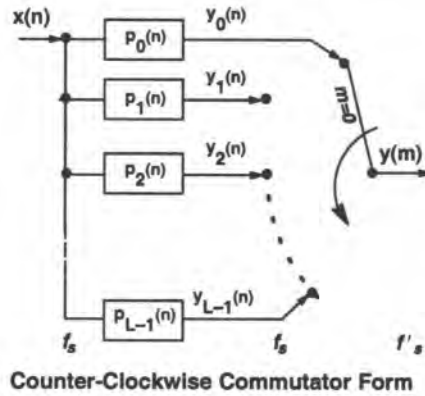


Figure 3.19. Counterclockwise form of a polyphase network [from R. E. Crochiere and L. R. Rabiner, Interpolation and decimation of digital signals—A tutorial review, Proc. IEEE **69**(3), 300–331 (1981), © IEEE 1981].

For each sample $x(n)$ there are L output samples, but it can be seen that the polyphase filters individually work at the low sample rate f_s . An alternative interpolator structure, which is mathematically equivalent, is shown in Figure 3.19. This is referred to as a *counterclockwise commutator model*. The commutator output switch rotates at the rate Lf_s and evidently produces L samples for every input sample, as required.

The polyphase filters, as implied, are decimated versions of $h(m)$. Some interesting consequences arise from this fact, as illustrated in Figure 3.20 for an FIR filter $h(m)$ with $N = 9$ taps and $L = 3$. The center of symmetry is about $m = 4$, hence the filter as a whole has a delay of four samples. Since $L = 3$, there are three polyphase filters and because the original filter has nine taps, each polyphase filter has three taps. Specifically, $p_0(n) = \{h(0), h(3), h(6)\}$, $p_1(n) = \{h(1), h(4), h(7)\}$, and $p_2(n) = \{h(2), h(5), h(8)\}$. Each of these filters is shifted in time with respect to the common reference point $n = 0$, hence have different linear phases associated with them, as indicated in the figure. This is the origin of the term *polyphase network*.⁽¹⁸⁾

The polyphase and commutator topologies are particularly useful for implementing digital signal processing hardware and arose in that context. From a simulation standpoint, there is probably little difference in computational efficiency among the various interpolator realizations. However, some of the structures may be more or less convenient to realize in software.

Up to this point the specific form of h has not been made explicit. In the sections below we briefly consider three commonly used interpolation functions.

3.5.4.3. Bandlimited and Windowed Bandlimited Interpolation

As we know from (3.5.11) and supported by the frequency-domain counterpart in (3.5.14), if a signal $x(t)$ is bandlimited to $|f| \leq B$ and if $T_s = 1/(2B)$, then the sinc function interpolates the sampled sequence perfectly everywhere. Since we can always define an effective band limitation B for any signal, it is natural to consider, as a start, the use of the sinc function for interpolation. Thus, the “ideal” interpolating function is

$$h(t) = \text{sinc}(f_s t) = \frac{\sin(\pi f_s t)}{\pi f_s t} \quad (3.5.24)$$

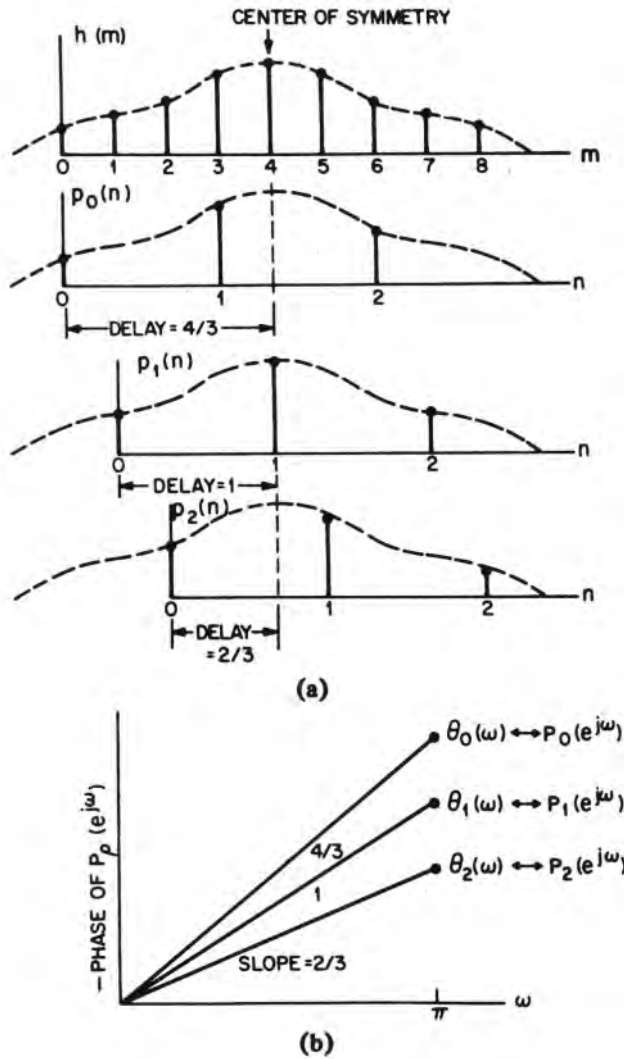


Figure 3.20. Construction of polyphase filters. (a) Original high-rate interpolating filter; (b) the polyphase filters as decimated versions of the original filters, and their phase characteristics [from R. E. Crochiere and L. R. Rabiner, Interpolation and decimation of digital signals—A tutorial review, *Proc. IEEE* **69**(3), 300–331 (1981), © IEEE 1981].

This was earlier illustrated in Figure 3.11 for continuous time. If now we consider only discrete times $t = mT_s' = m/Lf_s$, we see that

$$h(m) = \text{sinc}(m/L) = \frac{\sin(\pi m/L)}{\pi m/L} \quad (3.5.25)$$

Or, substituting directly in (3.5.19a), using (3.5.20), we have

$$y(m) = \sum_{n=-\infty}^{\infty} \text{sinc}(nL + m \bmod L) x(\lfloor m/L \rfloor - n) \quad (3.5.26)$$

A sketch of the response (3.5.25) for $L = 5$ is given in Figure 3.21a. In a polyphase interpretation, there are five polyphase filters, the coefficients of which are decimated versions (by the factor 5) of $h(m)$. Since the sinc function is not time-limited, the sum in (3.5.26) cannot be literally taken. Consequently, a practical version of the ideal sinc interpolation is to use a properly windowed sinc function, as sketched in Figure 3.21b,c. It is not obvious what window characteristics would be optimum to minimize some measure of the error in the interpolated sequence, but probably many of the commonly used windows, such as the Hamming window, can be reasonably used for this purpose.

3.5.4.4. Linear Interpolation

The simplest interpolation function possible is linear interpolation, given in continuous time by

$$h(t) = 1 - |t|/T_s, \quad |t| \leq T_s, \quad (3.5.27)$$

and zero elsewhere, as sketched in Figure 3.22b; Figure 3.22a illustrates the interpolation process itself for continuous time. The interpolated function $y(t)$ is given, as before, by

$$y(t) = \sum x(nT_s) h(t - nT_s)$$

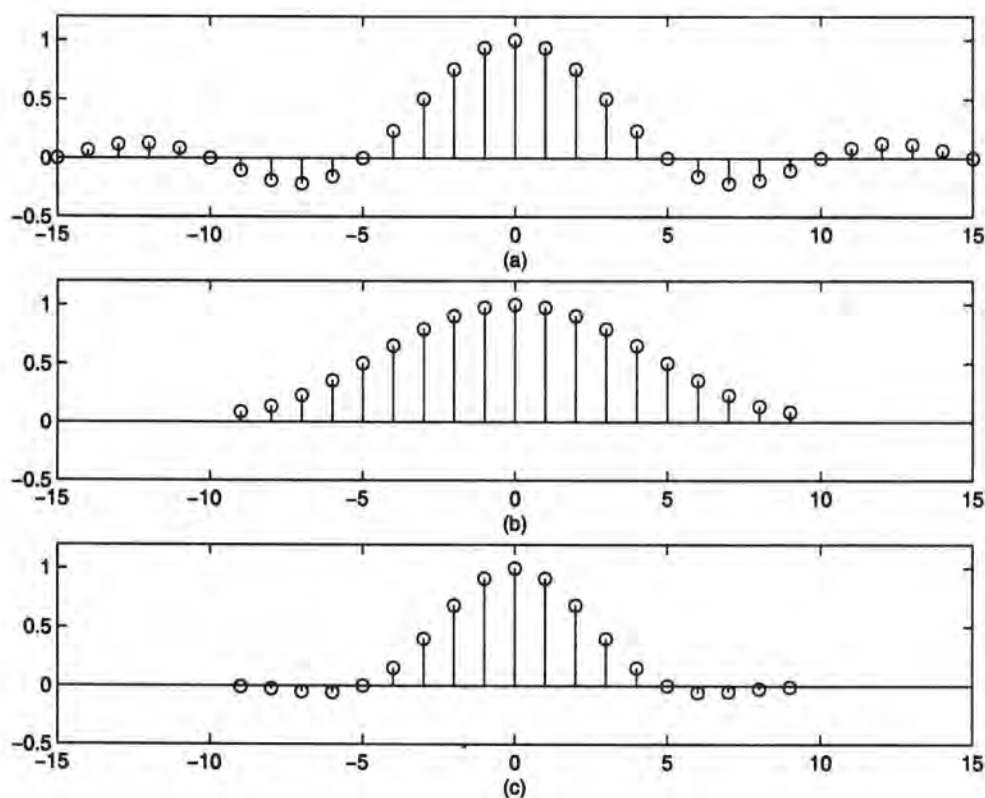


Figure 3.21. Illustration of discrete band-limited interpolation for $L = 5$. (a) The ideal response; (b) a window response $w(m)$; (c) the resulting interpolating filter $\text{sinc}(m/L) w(m)$.

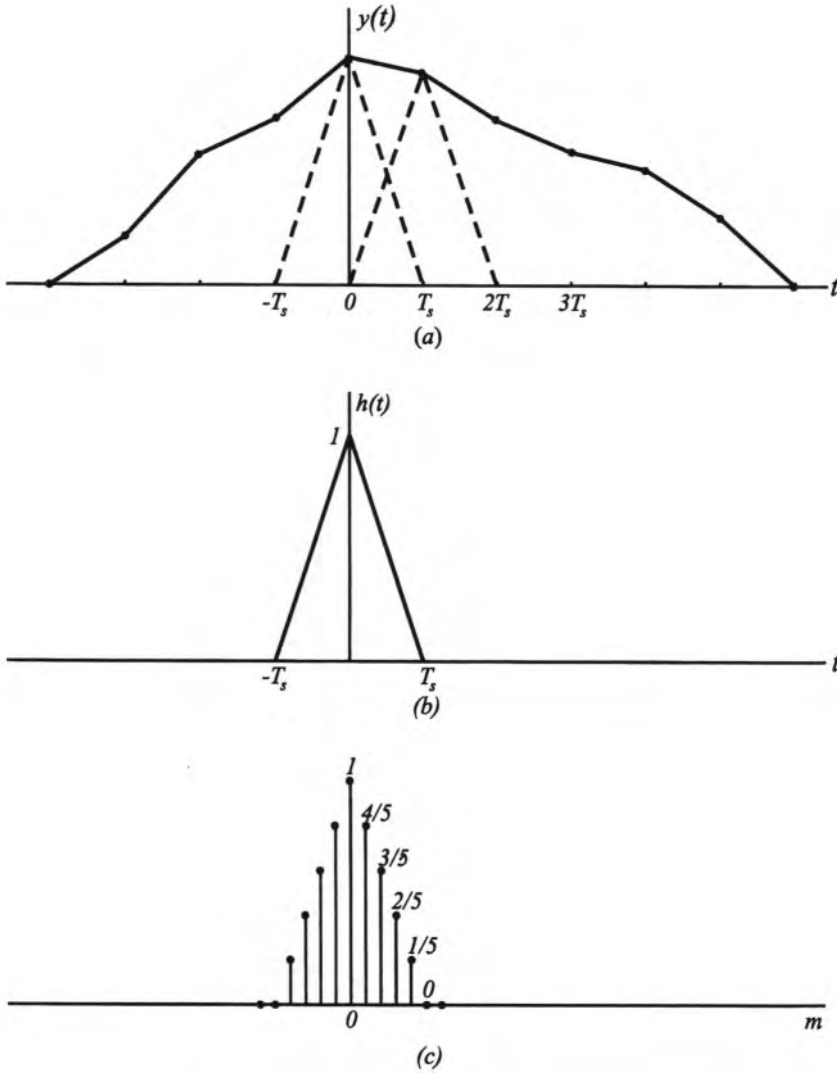


Figure 3.22. Linear interpolation, (a) Illustration of the interpolation process; (b) the continuous interpolation filter; (c) the discrete interpolation filter.

which involves at most two points of the sequence $\{x(nT_s)\}$. Hence, it is a very efficient process, which of course is why it is often used. From (3.5.14) the spectrum of the filter output is simply

$$Y(f) = T_s \operatorname{sinc}^2 f T_s \sum_{k=-\infty}^{\infty} X(f + k f_s)$$

where

$$H(f) = T_s \operatorname{sinc}^2 f T_s \quad (3.5.28)$$

is the Fourier transform of $h(t)$. Frequently, in waveform simulation, the signal is sampled at a fairly high rate, which renders linear interpolation sufficiently accurate for many purposes. A

sense of its accuracy can be gained by comparing (3.5.28), previously sketched in Figure 3.4, to the ideal lowpass filter, which is the Fourier transform of the sinc function.

The discrete-time version of (3.5.27) is simply

$$h(m) = 1 - |m|/L, \quad |m| \leq L \quad (3.5.29)$$

and zero elsewhere, as sketched in Figure 3.22c. The interpolation rule is thus simply

$$y(n + m + 1) = y(n + m) + \Delta y_{m,n} \quad (3.5.30a)$$

where

$$\Delta y_{m,n} = \frac{x(n + L) - y(n)}{L} \quad (3.5.30b)$$

Thus, discrete linear interpolation requires only one subtraction, one division, and $L - 1$ additions for every $L - 1$ output points.

3.5.4.5. Spline Interpolation

The need for interpolation arises in situations other than multirate applications. For example, one often obtains frequency response data (e.g., from a network analyzer) that are relatively widely spaced in frequency and in general may not be taken at equal intervals. For FFT-based filtering (see Chapter 4) it is often necessary to have many more points at equal intervals. This is accomplished by interpolating the given measurements. If the data are sufficiently close in frequency, the linear interpolation already discussed could be adequate, although the resulting curve will consist of straight-line segments, and will clearly not satisfy our intuitive expectation that a frequency function should be “smooth.” An interpolation scheme that is frequently used for such purposes is the cubic spline. Spline interpolation has the desirable property that the data need not be equally spaced, and it is computationally very efficient.[†] Entire texts are devoted to the subject, but it is worthwhile presenting a skeleton of the idea.

Cubic splines are a particular case of a class of functions called splines, about which there is an extensive literature, e.g., Refs. 22 and 23, and chapter sections in Refs. 24 and 25. Let Δ denote a *subdivision* on the interval $[a, b]$,

$$\Delta: \quad a = x_0 < x_1 < \cdots < x_N = b \quad (3.5.31)$$

The points $x_0, \dots, x_N$ will be referred to as nodes or knots.[‡] About the (unknown) function to be interpolated $f(x)$, we know only the function values $f(x_0), \dots, f(x_N)$. Following Ref. 24, we define *spline functions of degree m and smoothness class k* , relative to the subdivision Δ , as follows:

$$S_m^k(\Delta) = \{s: s \in C^k[a, b], s|_{[x_i, x_{i+1}]} \in \mathbb{P}_m, i = 0, 1, \dots, N - 1\} \quad (3.5.32)$$

where $C^k[a, b]$ denotes the class of k -times differentiable functions on $[a, b]$ and $\mathbb{P}_m$ denotes the class of polynomials of degree $\leq m$. In words, any $s(x) \in S_m^k(\Delta)$ is a k -times differentiable function over the *entire* interval $[a, b]$, including the knots, and is composed of contiguous

[†] In fact, bandlimited interpolation is not limited to equally spaced samples, but the nonuniformity allowed in the spacing is fairly constrained,¹⁴ unlike the case for splines. On the other hand, spline interpolation is not usually used in the sampling theorem/multirate context, although there appears to be no reason why it might not be put to good use in that context

[‡] The term *knot* is often used only for the interior points, but we shall use it freely here for all points.

segments of polynomials of degree m over each subinterval, with the further stipulation that $s(x_i) = f(x_i)$ for $i = 0, 1, \dots, N$. Hereafter, we will deal only with the cubic spline ($m = 3$) of smoothness class 2 ($k = 2$), which we will label simply $s(x)$. This cubic spline is a frequent choice because (for certain given endnode conditions described below) it possesses the optimal “smoothness” property that

$$\int_a^b |s''(x)|^2 dx \quad (3.5.33)$$

is smallest among all square-integrable functions that interpolate f at the given nodes.

The problem is to find $s(x)$, given $f(x_i)$ for $i = 0, 1, \dots, N$, and this is of course equivalent to determining the coefficients of the N cubic polynomials on each of the subintervals. We will show that the spline is uniquely determined by its moments $M_i \triangleq s''(x_i)$, and that the moments themselves can be calculated from the given data $\{f(x_i)\}$ by solving a system of linear equations.

Since $s(x)$ is twice differentiable, by assumption, differentiating twice will produce a sequence of connected straight-line segments (linear functions) between knots. Using the label M_i for $s''(x_i)$, we have the relationship

$$s''(x) = M_{i-1} \frac{x_i - x}{h_i} + M_i \frac{x - x_{i-1}}{h_i}, \quad x_{i-1} \leq x \leq x_i \quad (3.5.34)$$

where $h_i = x_i - x_{i-1}$ and $i = 1, \dots, N$. Integrating (3.5.34) twice, using the condition that $s(x_i) = f(x_i) \triangleq y_i$ to evaluate the constants of integration, we obtain, for $x_{i-1} \leq x \leq x_i$,

$$\begin{aligned} s(x) = & M_{i-1} \frac{(x_i - x)^3}{6h_i} + M_i \frac{(x - x_{i-1})^3}{6h_i} + \left[y_{i-1} - \frac{M_{i-1}h_i^2}{6} \right] \frac{x_i - x}{h_i} \\ & + \left[y_i - \frac{M_i h_i^2}{6} \right] \frac{x - x_{i-1}}{h_i} \end{aligned} \quad (3.5.35)$$

This expression shows that the spline is completely determined from its moments, and we will now show that the moments can be obtained from the data.

Differentiating (3.5.35) once, we get

$$s'(x) = -M_{i-1} \frac{(x_i - x)^2}{2h_i} + M_i \frac{(x - x_{i-1})^2}{2h_i} + \frac{y_i - y_{i-1}}{h_i} - \frac{M_i - M_{i-1}}{6} h_i \quad (3.5.36)$$

From the continuity assumption, we know that the value of $s'(x_i)$ must be the same whether approached from the left or from the right. Of course, this statement applies only to interior knots x_i , $i = 1, \dots, N-1$, because x_0 can only be approached from the right, and x_N can only be approached from the left. For these endnodes, additional conditions must be imposed, as discussed later. If we denote x_i approached from the left as x_{i-} and x_i approached from the right as x_{i+} , we get

$$s'(x_{i-}) = \frac{h_i}{6} M_{i-1} + \frac{h_i}{3} M_i + \frac{y_i - y_{i-1}}{h_i} \quad (3.5.37a)$$

and

$$s'(x_i) = \frac{h_i}{3}M_i - \frac{h_{i+1}}{6}M_{i+1} + \frac{y_{i+1} - y_i}{h_{i+1}} \quad (3.5.37b)$$

Equating (3.5.37a) to (3.5.37b) yields the $N - 1$ constraint equations

$$\frac{h_i}{6}M_{i-1} + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}, \quad i = 1, \dots, N - 1 \quad (3.5.38)$$

Introducing the notation

$$\lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad \mu_i = 1 - \lambda_i, \quad i = 1, \dots, N - 1$$

we obtain for the condition (3.5.38)

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = 6 \frac{[(y_{i+1} - y_i)/h_{i+1}] - [(y_i - y_{i-1})/h_i]}{h_i + h_{i+1}}, \quad i = 1, \dots, N - 1 \quad (3.5.39)$$

For later use we set

$$d_i = 6 \frac{[(y_{i+1} - y_i)/h_{i+1}] - [(y_i - y_{i-1})/h_i]}{h_i + h_{i+1}}, \quad i = 1, \dots, N - 1$$

so that (3.5.39) can be written more succinctly as

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i, \quad i = 1, \dots, N - 1 \quad (3.5.40)$$

which can be seen to be easily solvable recursively, once we have appropriate initial conditions. A matrix solution is also possible, as detailed below.

However, we now have $N - 1$ equations, but $N + 1$ unknowns, so additional conditions must be appended to the endpoints. A variety of such conditions have been considered:

- (i) $s''(\mathbf{a}) = s''(\mathbf{b}) = 0$; this is also called the *natural* cubic spline.
- (ii) $s^{(k)}(\mathbf{a}) = s^{(k)}(\mathbf{b})$, for $k = 0, 1, 2$; this is also referred to as the *periodic* spline.
- (iii) $s'(\mathbf{a}) = y'_0$ and $s'(\mathbf{b}) = y'_N$ for given numbers y'_0 and y'_N ; usually, we set $y'_0 = f'(\mathbf{a})$ and $y'_N = f'(\mathbf{b})$, and this is referred to as the *complete* spline.
- (iv) $s''(\mathbf{a}) = f''(\mathbf{a})$ and $s''(\mathbf{b}) = f''(\mathbf{b})$.

Which of these end conditions should be used depends on what is appropriate for the problem at hand. For example, a typical situation arises in conjunction with measured frequency response data for filters. The extreme points ($\mathbf{a}$, $\mathbf{b}$) can be assumed to be the lower and upper edges of the simulation bandwidth, respectively, and can be assumed to be well in the “skirts” region. In that region, it is typical for the filter attenuation (skirts) to roll off at some rate such as $6n$ dB/octave. A suggested procedure would be to approximate that slope either by the last two points at either edge, or by fitting the best straight line to several points in that region of the skirts, and to use that slope as the value of y'_0 and y'_N , respectively.[†] In this case, therefore, it is clear that end condition (iii) is the applicable one.

[†] This assumes that the attenuation is given in decibels, which is the usual case. The interpolation for the filter magnitude would then be carried out in the decibel domain. Of course, in implementing the actual filtering, the magnitude would have to be reexpressed in the numeric domain.

The minimality property mentioned in conjunction with (3.5.33) holds for cases (i) and (iii). It turns out (see, e.g., Ref. 25) that for cases (i), (iii), and (iv), two additional equations can be formed with a common format compatible with (3.5.40). These two equations are

$$2M_0 + \lambda_0 M_1 = d_0 \quad (3.5.41)$$

and

$$\mu_N M_{N-1} + 2M_N = d_N \quad (3.5.42)$$

where λ_0 , d_0 , μ_N , and d_N are defined as follows:

1. For case (i): $\lambda_0 = 0$, $d_0 = 0$, $\mu_N = 0$, and $d_N = 0$.
2. For case (iii):

$$\lambda_0 = 1, \quad d_0 = \frac{6}{h_1} \left(\frac{y_1 - y_0}{h_1} - y'_0 \right)$$

and

$$\mu_N = 1, \quad d_N = \frac{6}{h_1} \left(y'_N - \frac{y_1 - y_0}{h_1} \right)$$

3. For case (iv): $\lambda_0 = 0$, $d_0 = 2M_0 = 2f''(a)$, $\mu_N = 0$, $d_N = 2M_N = 2f''(b)$.

With these particular conditions, (3.5.40)–(3.5.42) can be cast in the following (tridiagonal) matrix form:

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \dots & 0 & 0 & 0 \\ \mu_1 & 2 & \lambda_1 & \dots & 0 & 0 & 0 \\ 0 & \mu_2 & 2 & \lambda_2 & 0 & \vdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \ddots & \ddots & \lambda_{N-2} & 0 \\ 0 & 0 & 0 & 0 & \mu_{N-1} & 2 & \lambda_{N-1} \\ 0 & 0 & 0 & \dots & 0 & \mu_N & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{N-2} \\ M_{N-1} \\ M_N \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{N-2} \\ d_{N-1} \\ d_N \end{bmatrix} \quad (3.5.43)$$

which is quickly and efficiently evaluated.

Case (ii) has to be treated separately. Here, it can be shown (e.g., Ref. 25) that

$$\lambda_N = \frac{h_1}{h_N + h_1}, \quad \mu_N = \frac{h_N}{h_N + h_1}$$

and

$$d_N = \frac{6}{h_N + h_1} \left\{ \frac{y_1 - y_N}{h_1} - \frac{y_N - y_{N-1}}{h_N} \right\}$$

These conditions, together with the the assumption that $M_N = M_0$, which reduces the dimension by 1, produce the system of equations

$$\begin{bmatrix} 2 & \lambda_1 & 0 & \dots & 0 & 0 & \mu_1 \\ \mu_2 & 2 & \lambda_2 & \dots & 0 & 0 & 0 \\ 0 & \mu_3 & 2 & \lambda_3 & 0 & \vdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \ddots & \ddots & \lambda_{N-2} & 0 \\ 0 & 0 & 0 & 0 & \mu_{N-1} & 2 & \lambda_{N-1} \\ 0 & 0 & 0 & \dots & 0 & \mu_N & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{N-2} \\ M_{N-1} \\ M_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-2} \\ d_{N-1} \\ d_N \end{bmatrix} \quad (3.5.44)$$

which also is easily evaluated.

Using either (3.5.43) or (3.5.44), one obtains the M_i , which then allows us to calculate directly $s(x)$ for any set of values of x , and the interpolation is complete.

■ **Example 3.5.2.** An example will help to illustrate the nature of spline interpolation. We took (arbitrarily) the function $f(x) = 5 \exp(-x/2) \sin(2\pi x)$ and sampled it *randomly* 34 times across the interval originally specified as $[0, 5]$. The sampling points were produced by using a uniform RNG, $\mathcal{U}[0, 1]$, and the last point sampled was actually about 4.67. Figure 3.23 shows $f(x)$ as the solid line and the spline interpolant $s(x)$ as the dotted line; the plus signs indicate the knots. According to the earlier presentation, the second derivative of $s(x)$ should

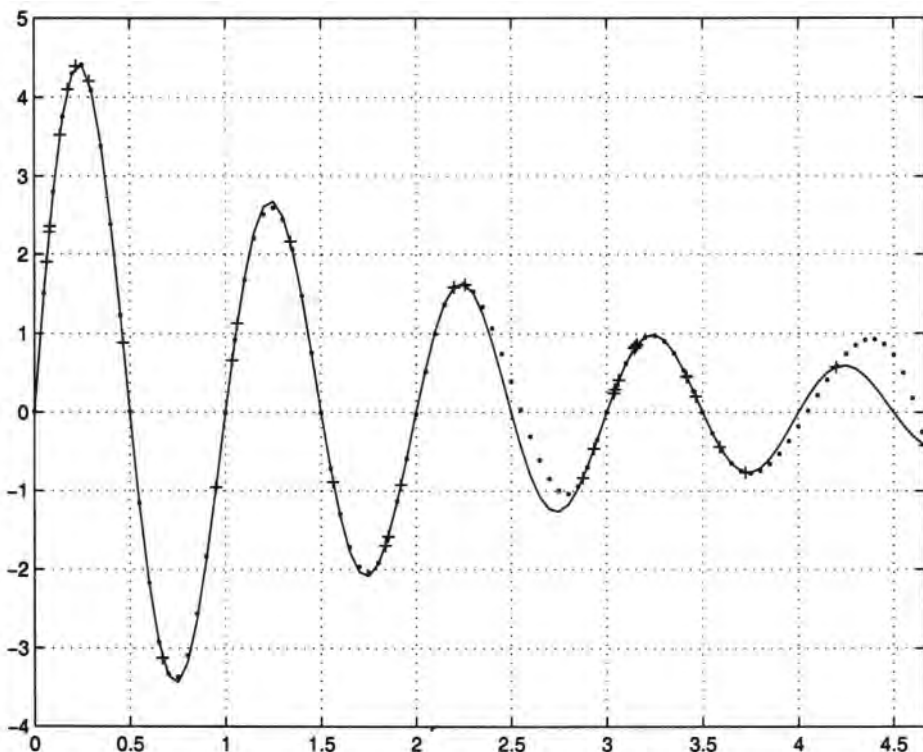


Figure 3.23. Example of a third-order spline interpolation; the solid line is the original function, the dotted line is the interpolant, and the plus signs indicate the knots.

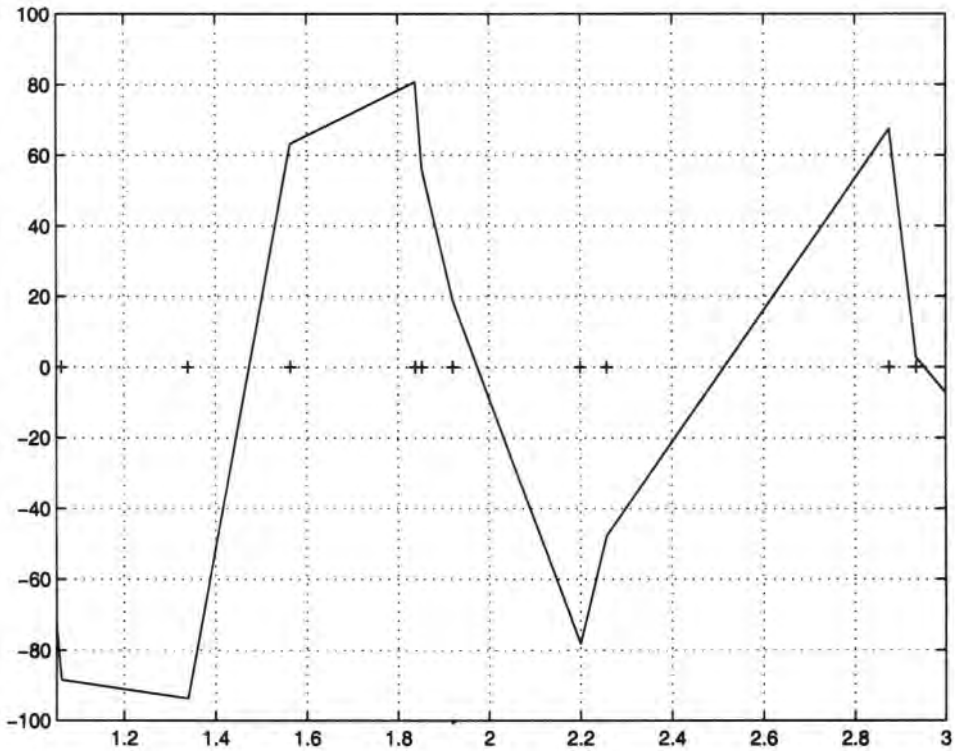


Figure 3.24. Magnified portion of the second derivative of the spline interpolant of Example 3.5.2.

consist of linear segments between the knots. This is confirmed by Figure 3.24, which shows this second derivative; for visual clarity, a smaller segment is shown. In the example shown, both $s(x)$ and its second derivative were obtained using MATLAB. ■

It can be seen from this example that the spline interpolant is quite close to the function being interpolated in most places, but with some obvious errors. The approach (convergence) of the spline to the function being interpolated is therefore a question of interest. As might be expected, it depends on the function itself and the nature of the subdivision. There are a number of theorems on the subject, with much the same character, but here we quote just one as an illustration. Let

$$|\Delta| = \max_i h_i$$

and suppose that $f \in C^4[a, b]$. Then, the maximum of the difference $|f(x) - s(x)|$ over $[a, b]$ is proportional to $|\Delta|^4$. Evidently, the interpolation error vanishes as $|\Delta| \rightarrow 0$, and does so rapidly. In practice, the subdivision will often simply be a property of a measurement, and not subject to choice, so that there is no guarantee as to how close the spline will be to the (unknown) function. Nevertheless, we can generally expect a reasonably good fit because there is typically some intuition about the behavior of the device under test, so that sampling points in practice are usually chosen to be sufficiently dense to prevent large interpolation error.

3.6. Characterization of Linear Time-Invariant Systems Using the Laplace Transform

In the following two sections we describe the methods of modeling systems characterized by linear differential equations with constant coefficients. Modeling of these equations is important because they represent a class of systems that encompass all analog filters with lumped elements. Furthermore, much of the related analytic techniques form a natural springboard for the analysis of discrete systems.

The modeling and design of these systems is done by transforming the differential equations into algebraic equations using the *Laplace transform*. This transformation, which results in a rational function representation, greatly facilitates the analysis of the system and provides us with immediate insight into the behavior of the system. The characterization of systems using the Laplace transform is described in many texts; see, e.g., Refs. 1–4, 26 and 27.

Because of the discrete form of simulation on computers, we transform the systems described by differential equations into systems described by difference equations that approximate the continuous system. Consequently, the Laplace transform, which is the natural transform domain for continuous systems, is replaced by the z -transform described in Section 3.7.

It should be noted that the transformation from the continuous into the discrete representation is necessary for simulation of continuous filters only. The design of digital filters can be done in the discrete domain without resorting to continuous filter design methods.

In the following sections we will briefly describe the analysis of filters in the s -domain and the mapping into the z -domain. The nature of the mapping as well as the sampling rate control of the accuracy of the simulation will be described.

3.6.1. The Laplace Transform

3.6.1.1. Introduction

We will limit our discussion to causal signals and systems only. The Laplace transform for a causal signal $x(t)$, often referred to as the unilateral Laplace transform, is

$$X(s) = \int_0^{\infty} x(t)e^{-st} dt = \mathcal{L}[x(t)] \quad (3.6.1)$$

where the complex variable is $s = \sigma + j\omega$. The Laplace transform is therefore a generalization of the Fourier transform for causal signals,

$$X(s)|_{s=j\omega} = \mathcal{F}[x(t)] \quad (3.6.2)$$

A table of some common Laplace transform pairs is listed in Table 3.A.6 in the Appendix to this chapter.

3.6.1.2. Convergence and Stability

In order for the Laplace transform to exist, the integral in (3.6.1) has to converge. Generally the function $X(s)$ converges only for certain regions of the s -plane. An indepth

description of convergence properties of the Laplace transform is amply covered in many texts, e.g., Refs. 1, 2, and 28.

All filters that are characterized by linear differential equations with constant coefficients, which include all classical filters, have a Laplace transform that is a rational function in the s -domain. If these functions are stable, e.g., the filters are passive, the convergence region covers the right half of the s -plane including the $j\omega$ axis. Thus, these functions have a Laplace transform and are stable if the poles are confined to the left half of the s -plane excluding the $j\omega$ axis.

3.6.2. Inverse Laplace Transform

The inverse Laplace transform can be evaluated from the contour integral

$$x(t) = \mathcal{L}^{-1}[X(s)] = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} X(s)e^{st} ds \quad (3.6.3)$$

However, this form of computation of the inverse Laplace transform is seldom used.

The system functions corresponding to constant-coefficient linear differential equations (i.e., lumped-element circuits) are rational functions. These rational functions can be expanded into simpler functions using a partial fraction expansion

$$H(s) = \frac{N(s)}{D(s)} = \sum_{i=1}^m \frac{A_i}{s + a_i} \quad (3.6.4)$$

The inverse of these simpler functions can be found easily.

3.6.3. Properties of the Laplace Transform

The properties of the Laplace transform are similar to those of the Fourier transform with the obvious substitution of $j\omega \rightarrow s$. Some of these properties are stated below:

1. *Linearity:*

$$\mathcal{L}(a_1x_1(t) + a_2x_2(t)) = a_1\mathcal{L}(X_1(t)) + a_2\mathcal{L}(X_2(t)) \quad (3.6.5)$$

2. *Time Shifting:*

$$\mathcal{L}[x(t - t_0)] = e^{-st_0}X(s) \quad (3.6.6)$$

3. *Shifting in s -Domain:*

$$\mathcal{L}[e^{s_0t}x(t)] = X(s - s_0) \quad (3.6.7)$$

4. *Time Scaling:*

$$\mathcal{L}[x(at)] = \frac{1}{|a|}X\left[\frac{s}{a}\right] \quad (3.6.8)$$

5. *Convolution Property.* The *convolution property*, often referred to as the *convolution theorem*, states that

$$\mathcal{L}[x_1(t) * x_2(t)] = X_1(s)X_2(s) \quad (3.6.9)$$

The *convolution property* is one of the most important properties of the Laplace transform because of its use in the evaluation of the system response (see *transfer function* later in this section).

6. *Differentiation in Time Domain:*

$$\mathcal{L}\left[\frac{dx(t)}{dt}\right] = sX(s) - x(0) \quad (3.6.10)$$

where $x(0)$ is the initial value of $x(t)$,

$$\mathcal{L}\left[\frac{d^2x(t)}{dt^2}\right] = s^2X(s) - sx(0) - x'(0) \quad (3.6.11)$$

7. *Integration in Time Domain:*

$$\mathcal{L}\left[\int_{-\infty}^t x(\tau) d\tau\right] = \frac{1}{s}X(s) \quad (3.6.12)$$

3.6.4. Transfer or System Function

The concept of a *transfer function*, often referred to as the *system function*, is of great importance in system analysis and therefore in system simulations. It provides a method of completely specifying the behavior of linear systems subjected to arbitrary inputs.

The transfer function is defined as the Laplace transform of the impulse response of a system

$$H(s) = \mathcal{L}[h(t)] \quad (3.6.13)$$

From (3.2.4) the response of an LTI system to an arbitrary input signal is

$$y(t) = x(t) * h(t)$$

From the convolution property (3.6.9)

$$Y(s) = \mathcal{L}[y(t)] = \mathcal{L}[x(t) * h(t)] = X(s)H(s)$$

or

$$H(s) = \frac{Y(s)}{X(s)} \quad (3.6.14)$$

Properties of the transfer function for LTI systems characterized by linear constant-coefficient differential equations will be presented later in this section. When $s = j\omega$, $H(s) = H(j\omega)$ is the frequency response and is also frequently referred to as the transfer function.

3.6.5. Interconnections of LTI Systems (Block Diagrams)

A convenient way to describe a system is as an interconnection of subsystems by means of a block diagram.

Since LTI systems possess commutative, associative, and distributive properties their block diagram interconnections can be simplified. This simplification is most effective using

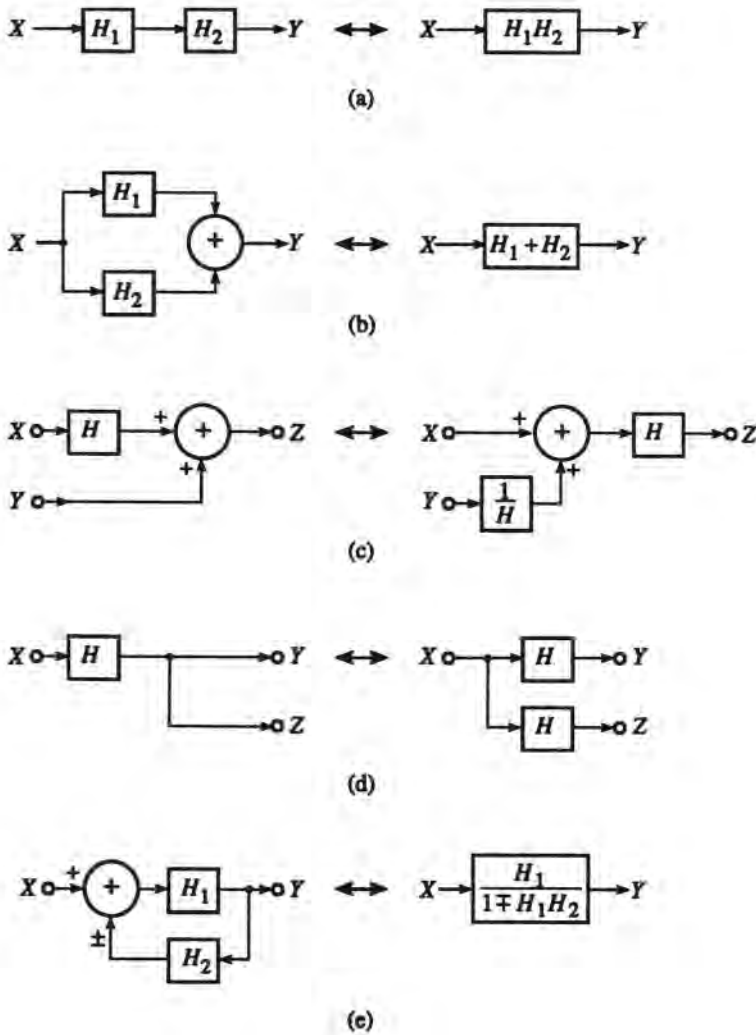


Figure 3.25. Block-diagram equivalents in the s -domain (a) $Y = H_1 H_2 X$. (b) $Y = [H_1 + H_2]X$. (c) $Z = HX + Y$. (d) $Y = Z = HX$. (e) $Y = [H_1 / (1 \pm H_1 H_2)]X$.

the Laplace transform representation. For example, the convolution property transforms the cascade connection of subsystems into one whose transfer function is the product of the subsystem transfer functions [see (3.6.9)]:

$$\mathcal{L}[h_1(t) * h_2(t) * h_3(t) \dots] = H_1(s) \cdot H_2(s) \cdot H_3(s) \dots \quad (3.6.15)$$

Using the properties of LTI systems, it is possible and convenient to simplify the system graphically using the block-diagram equivalences displayed in Figure 3.25. These equivalences and the rules to use them were developed by Mason (see Ref. 4), who called them *signal-flow graphs*.

■ **Example 3.6.1.** An example of a graphical transfer-function simplification of a complex system is shown in Figure 3.26. Using simple rules from Figure 3.25, we gradually simplify

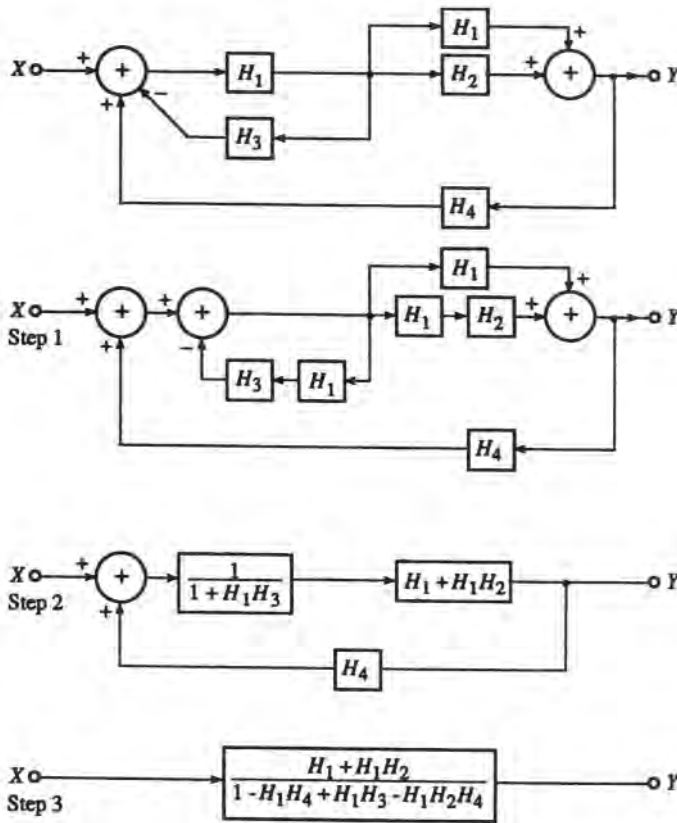


Figure 3.26. Example of a block-diagram simplification.

the system to a single block system with a transfer function

$$H = \frac{1 + H_1H_2}{1 - H_4 + H_3H_1 - H_4H_2H_1} \quad (3.6.16)$$

■

The simplification of block diagrams is an important technique in the simulation of communication systems. Because it replaces convolutions (Section 3.2) with products in the s -domain (or the frequency domain), it allows effortless combining of many filters into one, thus simplifying processing. This is an important part of the methodology of simulation, one of whose objectives is to achieve the least complex block diagram for simulation purposes (see Chapter 2). The elimination of feedback loops in LTI systems is particularly beneficial since it permits one to use frequency-domain processing, thus making the simulation more efficient.

3.6.6. Systems Characterized by Linear Constant-Coefficient Differential Equations

Consider a general differential equation of the form

$$\begin{aligned} b_m y^{(m)}(t) + b_{m-1} y^{(m-1)}(t) + \cdots + b_0 y(t) \\ = a_n x^{(n)}(t) + a_{n-1} x^{(n-1)}(t) + \cdots + a_0 x(t) \end{aligned} \quad (3.6.17)$$

where we can consider $x(t)$ as the input to the system and $y(t)$ the output. Applying the Laplace transform to both sides and using the derivative property of (3.6.10), we get

$$Y(s) \sum_{i=0}^m b_i s^i = X(s) \sum_{i=0}^n a_i s^i$$

The transfer function is then

$$H(s) = \frac{Y(s)}{X(s)} = \frac{\sum_{i=0}^m a_i s^i}{\sum_{i=0}^n b_i s^i} \quad (3.6.18)$$

For the above analysis we assumed that the system is at rest. Thus, the transfer function is defined for zero initial conditions. A large class of practical filter realizations that we refer to as “classical” filters are particular cases of (3.6.18). These include the Butterworth, Chebyshev and elliptic filters, and are presented in Chapter 4.

3.6.6.1. Properties of the Transfer Function for Linear Constant-Coefficient Differential Equations

In addition to the properties described in Section 3.6.5, the transfer function represented by linear constant-coefficient differential equations has the following additional properties listed below:

1. The transfer function is a rational function in s .
2. For a stable system the order of the numerator is equal to or smaller than that of the denominator ($m \leq n$).
3. For a causal system the coefficients of the polynomials of the rational function are real.

Thus, $H(s)$ specified by a linear differential equation with constant coefficients is always a rational function, with *zeros* specified by the roots of the numerator of (3.6.18) and *poles* by the roots of the denominator of (3.6.18). The poles and zeros are generally complex and are usually represented graphically in complex coordinates.

The impulse response of the system can be found by expanding $H(s)$ into partial fractions and using the transform pairs from Table 3.A.6 in the Appendix to this chapter.

If $H(s)$ is a proper fraction ($m < n$) and the poles are simple, then

$$H(s) = \sum_{i=1}^m \frac{A_i}{s - s_i}, \quad h(t) = \sum_{i=1}^m A_i e^{s_i t} \quad (3.6.19)$$

A system is stable if $h(t) \rightarrow 0$ as $t \rightarrow \infty$. It is easy to show that this is true if all the poles of $H(s)$ have a negative real part. A system is defined as *minimum-phase* if all its poles and zeros have negative real parts, and *nonminimum-phase* when only its poles have negative real parts. The significance of minimum versus nonminimum-phase systems arises, for example, in the equalization of such systems. An ideal equalizer has a transfer function which is the inverse of the system being equalized. If the system has zeros in the right half-plane, the inverse is unstable.

■ *Example 3.6.2.* Assume a transfer function

$$H(s) = \frac{s-3}{s^2+5s+6} = \frac{s-3}{(s+2)(s+3)} = \frac{6}{s+3} - \frac{5}{s+2}$$

$H(s)$ has a zero at $s = 3 + j0$ and poles at $s = -3 + j0$ and $s = -2 + j0$. The impulse response from Table 3.A.6 in the Appendix to this chapter is

$$h(t) = (6e^{-3t} - 5e^{-2t})u(t) \quad (3.6.20)$$

3.6.6.2. Realizations of Rational Transfer Functions Using Biquadratic Expansion

Differential equations such as (3.6.17) describing causal linear systems have real coefficients a_i, b_i . It can be shown that the poles and zeros of $H(s)$ for a differential equation with real coefficients are either real or occur in complex conjugate pairs. Therefore, the system function in (3.6.18) can also be represented in the form

$$H(s) = K \prod_{i=0}^M \frac{\alpha_{2i}s^2 + \alpha_{1i}s + \alpha_{0i}}{s^2 + \beta_{1i}s + \beta_{0i}} \quad (3.6.21)$$

The above represents a cascade connection of *biquadratic* sections (see Figure 3.27a) each with system function

$$H_{ci}(s) = \frac{\alpha_{2i}s^2 + \alpha_{1i}s + \alpha_{0i}}{s^2 + \beta_{1i}s + \beta_{0i}} \quad (3.6.22)$$

Another way of combining the complex conjugate poles into biquadratic forms is to use the partial fraction expansion of (3.6.19),

$$H(s) = \sum_{k=1}^N \frac{\alpha_{1k}s + \alpha_{0k}}{s^2 + \beta_{1k}s + \beta_{0k}} = \sum_{k=1}^N H_{pk}(s) \quad (3.6.23)$$

The above represents a *parallel* realization of a filter (shown in Figure 3.27b). Both the cascade and parallel realizations, in their discrete form, are used extensively in simulation of communication systems.

Realization of Biquadratic Sections. Biquadratic sections are used as building blocks in computer simulation of filters and in the physical implementation of analog filters using active filter technology. The realization of analog biquadratic sections for use in an active filter implementation requires the use of integrators because analog differentiators are not easily realizable. To reflect that, we rearrange (3.6.22) into

$$H(s) = \frac{Y(s)}{X(s)} = \frac{Y(s)}{W(s)} \cdot \frac{W(s)}{X(s)} = \frac{\alpha_2 + \alpha_1 s^{-1} + \alpha_0 s^{-2}}{1 + \beta_1 s^{-1} + \beta_0 s^{-2}} \quad (3.6.24)$$

where the term s^{-1} is equivalent to integration in the time domain, and where the subscript i was dropped for simplicity.

We split the transfer function $H(s)$ into two parts,

$$\frac{Y(s)}{W(s)} = \alpha_2 + \alpha_1 s^{-1} + \alpha_0 s^{-2} \quad (3.6.25a)$$

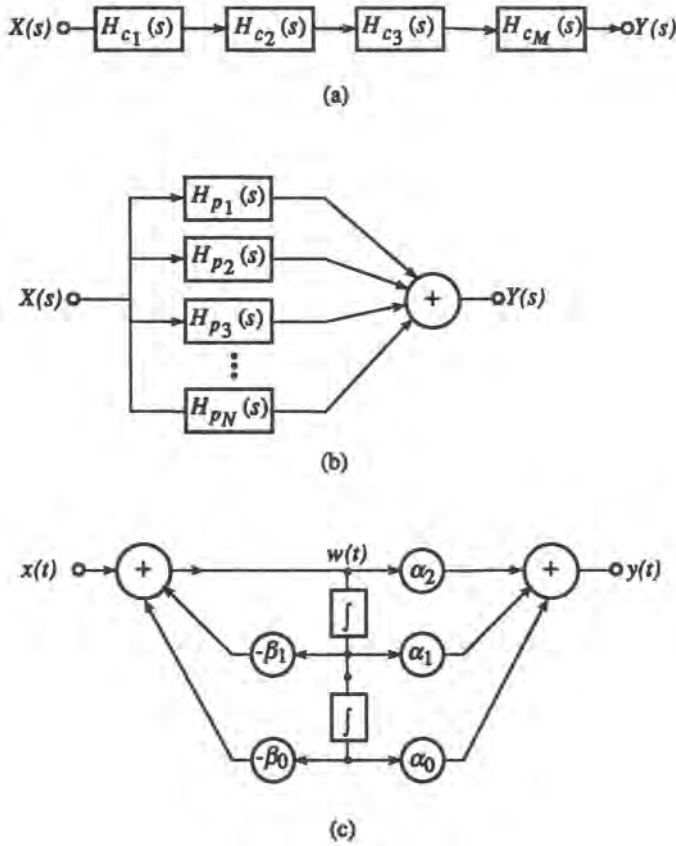


Figure 3.27. Biquadratic realization of systems. (a) Cascade interconnection of biquads, (b) parallel interconnection of biquads, (c) analog realizations of a biquad.

and

$$\frac{X(s)}{W(s)} = 1 + \beta_1 s^{-1} + \beta_0 s^{-2} \quad (3.6.25b)$$

Equations (3.6.25) are rewritten as

$$\begin{aligned} Y(s) &= \alpha_2 W(s) + \alpha_1 s^{-1} W(s) + \alpha_0 s^{-2} W(s) \\ W(s) &= X(s) - \beta_1 s^{-1} W(s) - \beta_0 s^{-2} W(s) \end{aligned} \quad (3.6.26)$$

The time-domain realization of a biquadratic section (*biquad*) from (3.6.26) is referred to as the *canonic form* and is shown in Figure 3.27c.

It should be noted that the technique used in the synthesis of the second-order section, leading to Figure 3.27c, can be extended to apply to transfer functions of arbitrary order. This extension will be used in conjunction with z -domain (discrete) filters in Chapter 4.

3.6.7. Frequency Response

If the system is stable, the region of convergence includes the $j\omega$ axis. Hence, $H(s)$ exists for $s = j\omega$ and is a rational function of ω :

$$H(j\omega) = \frac{\sum_{i=0}^N a_i(j\omega)^i}{\sum_{i=0}^M b_i(j\omega)^i} = A(\omega)e^{j\phi(\omega)}, \quad \omega = 2\pi f \quad (3.6.27)$$

$H(j\omega)$ is identical to the frequency response defined in Section 3.3.

A problem often arises in practice to determine the poles and zeros from the magnitude-squared response $A^2(\omega)$. It can be shown⁽⁷⁾ that $A^2(\omega)$ is a rational function of ω^2 ,

$$A^2(\omega) = |H(j\omega)|^2 = \frac{W(\omega^2)}{V(\omega^2)} \quad (3.6.28)$$

Furthermore, $H^*(j\omega) = H(-j\omega)$ because the system is real. Hence,

$$A^2(\omega) = H(j\omega)H^*(j\omega) = H(s)H(-s)|_{s=j\omega} = |H(s)|_{s^2=-\omega^2}^2 \quad (3.6.29)$$

Let us first find the roots of the denominator $D(j\omega)$ of $A^2(\omega)$. From (3.6.28) we have

$$|D(j\omega)|^2 = V(\omega^2) \quad (3.6.30)$$

which implies

$$D(s)D(-s) = V(-s^2)$$

Obviously, the roots of $V(-s^2)$ are symmetrically distributed on both sides of the $j\omega$ axis, and the roots on the $j\omega$ axis have even multiplicity. To form $D(s)$, we take all the roots on the left half-plane and half of the roots on the $j\omega$ axis. The rest of the roots form the roots of the polynomial $D(-s)$.

■ *Example 3.6.3.* Assume the denominator of $A^2(\omega)$ is

$$V(\omega^2) = \omega^4 + 5\omega^2 + 4$$

Then

$$V(-s^2) = s^4 - 5s^2 + 4 = (s+1)(s-1)(s+2)(s-2)$$

and the denominators of $H(s)$ and $H(-s)$ are

$$D(s) = (s+1)(s+2), \quad D(-s) = (s-1)(s-2) \quad \blacksquare$$

When the denominator of $A^2(\omega)$ is a polynomial composed of biquadratic factors of the form

$$V(\omega^2) = \omega^4 + A\omega^2 + B \quad (3.6.31)$$

the denominator of $H(-s^2)$ is

$$V(-s^2) = s^4 - As^2 + B = (s^2 + as + b)(s^2 - as + b) = s^4 + (2b - a^2)s + b^2$$

which implies $b = \sqrt{B}$ and $a = \sqrt{2b + A}$. The corresponding factors of $H(s)$ are then

$$D(s) = s^2 + as + b, \quad D(-s) = s^2 - as + b \quad (3.6.32)$$

■ *Example 3.6.4.* Let us assume the denominator of $A^2(\omega)$ is $V(\omega^2) = 1 + \omega^4$; then $V(-s^2) = 1 + s^4$, so that $A = 0$ and $B = 1$. Hence, $b = 1$, $a = \sqrt{2}$ and

$$D(s) = s^2 + \sqrt{2}s + 1, \quad D(-s) = s^2 - \sqrt{2}s + 1$$

where $D(s)$ has the roots in the left-hand side of the s -plane. ■

The above method of finding the roots of the polynomial $D(s)$ is unique only for the poles of the system function $H(s)$ since the zeros of $H(s)$ can be both on the right- and left-hand sides of the s -plane. We can apply the same procedure to the zeros as to the poles, and select the left-half-side zeros only. The system function thus selected is unique and is called *minimum-phase*. To design nonminimum-phase systems, we also need, in addition to the amplitude $A(\omega)$, information about the phase $\phi(\omega)$.

3.7. Representation of Continuous Systems by Discrete Transfer Functions

The discrete form of simulation used for communication systems requires representation of systems in the discrete-time domain. The principal focus of this section is on methods of approximating continuous systems with discrete systems that are appropriate for simulation.

The z -transform can be regarded as the discrete system analysis counterpart of the Laplace transform. By using the z -transform, the difference equations that characterize IIR discrete-time systems are transformed into algebraic equations, which are simpler to analyze and simulate. The rational functions in the s -domain that describe continuous systems can be mapped into rational functions in the z -domain directly. It should be noted that there is not a unique mapping from the s -domain to the z -domain. In principle, there can be any number of such mappings and since any given mapping results in a distinct z -domain function, the associated difference equation will also be distinct. Thus, there is more than one way of transforming a continuous system into a discrete-time system. Finite-length sequences that represent FIR systems are also conveniently modeled in the z -domain.

In this section we will give a brief introduction to the z -transform. In the next chapter we will describe the mapping of IIR filter transfer functions from the s -domain into the z -domain. The implementations of these transfer functions into models to be used in simulations will be developed there.

3.7.1. The z -Transform

To see how the z -transform naturally arises in the mathematical description of discrete-time systems, let us consider the impulse sampled causal signal

$$\hat{x}(t) = \sum_{n=0}^{\infty} x(nT_s) \delta(t - nT_s) \quad (3.7.1)$$

Since $\delta(t)$ is equal to zero except at the sampling instance $t = nT_s$, we can write

$$\hat{x}(t) = \sum_{n=0}^{\infty} x(nT_s) \delta(t - nT_s) \quad (3.7.2)$$

Taking the Laplace transform yields

$$\hat{X}(s) = \sum_{n=0}^{\infty} x(nT_s) e^{-nsT_s} \quad (3.7.3)$$

Finally, defining the complex variable $z = e^{sT_s}$, yields

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n} \quad (3.7.4)$$

$X(z)$ is referred to as the z -transform of $x(n)$, and denoted

$$X(z) = Z(x(n)) \quad (3.7.5)$$

Equation (3.7.4) represents a sequence of samples where the coefficient $x(n)$ represents the value of the sample and z^{-n} specifies the occurrence time of the sample. As an example, $5z^{-7}$ denotes a sample of value 5 which occurs seven sample periods after $t = 0$.

Note that the z -transform can be defined simply as a sequence of sample values as in (3.7.4). However, relating z to the Laplace variable s allows us to make good use of the previously developed theory for continuous systems in Section 3.6.

3.7.1.1. Convergence and Stability

Recall from Section 3.6 that the Laplace variable s was given by $s = \sigma + j\omega$. From the definition $z = \exp(sT_s)$, we have

$$z = e^{\sigma T_s} e^{j\omega T_s} \rightarrow |z| = e^{\sigma T_s} \quad (3.7.6)$$

Thus, strips of width $2\pi/T_s$ in the left-half s -plane, $\sigma < 0$, map into the interior of the unit circle in the z -plane (see Figure 3.28), and strips of width $2\pi/T_s$ in the right-hand s -plane map into the outside of the unit circle in the z -plane.

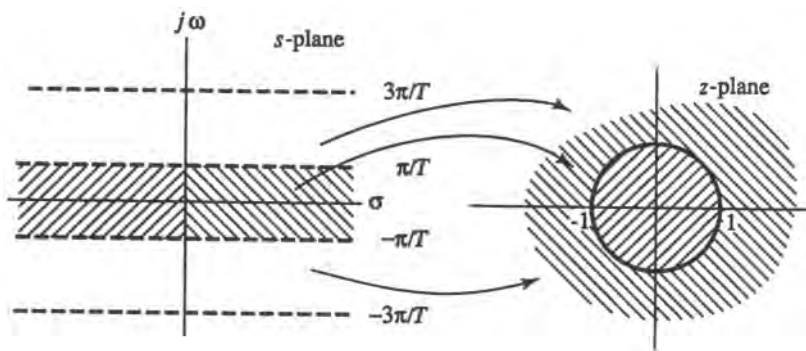


Figure 3.28. The mapping from the s -plane to the z -plane defined by $z = e^{sT_s}$.

From Section 2.6 we recall that stable rational functions of the Laplace variable s have a region of convergence that covers the right half of the s -plane including the $j\omega$ axis. Thus, these functions must not have any poles in the right-half s -plane or on the $j\omega$ axis. Equation (3.7.6) and Figure 3.28 then yield an equivalent definition for stability for the function of z . A stable rational function of z has a region of convergence that covers the outside and including the unit circle in the z -plane. Such a function has no poles outside or on the unit circle in the z -plane.

3.7.1.2. Table of Simple z -Transforms

A short table of z -transforms evaluated using the relationship (3.7.4) is given in Table 3.A.7 in the Appendix to this chapter. The region of convergence for each entry can be found, e.g., in Ref. 2.

3.7.1.3. Properties of the z -Transform

As with other transforms we developed, the z -transform possesses properties that make it an extremely valuable tool in the study of discrete-time systems. The derivation of the properties is similar to the derivation of properties of the transforms described in previous sections and can be found in the literature.⁽²⁹⁾

1. *Linearity.* The z -transform is linear,

$$\mathbf{Z}[a_1x_1(n) + a_2x_2(n)] = a_1X_1(z) + a_2X_2(z) \quad (3.7.7)$$

2. *Time Shifting:*

$$\mathbf{Z}[x(n - n_0)] = z^{-n_0}X(z) \quad (3.7.8)$$

3. *Convolution Theorem.* Similar to the convolution property of the Laplace transform,

$$\mathbf{Z}[x_1(n) * x_2(n)] = X_1(z)X_2(z) \quad (3.7.9)$$

where $x_1(n) * x_2(n) = \sum_{k=-\infty}^{\infty} x_1(k)x_2(n - k)$

3.7.1.4. Discrete Transfer or System Function

The *discrete transfer function* or *discrete system function* is of great importance in system analysis and therefore in system simulations because it completely specifies the behavior of discrete linear systems subjected to arbitrary inputs.

The discrete transfer function is defined as the z -transform of the sampled impulse response of a system

$$H(z) = \mathbf{Z}[h(n)] \quad (3.7.10)$$

From (3.2.12) the response of an LTI system to an arbitrary input signal is

$$y(n) = x(n) * h(n)$$

From the convolution property (3.7.9),

$$Y(z) = \mathbf{Z}[y(n)] = \mathbf{Z}[x(n) * h(n)] = X(z)H(z)$$

or

$$H(z) = \frac{Y(z)}{X(z)} \quad (3.7.11)$$

For z evaluated on the unit circle, the transfer function reduces to the periodic frequency response

$$H(z)|_{z=\exp(j\omega T_s)} = H_p(e^{j\omega T_s}) \quad (3.7.12a)$$

The recovered frequency response [see (3.5.6)] is then

$$H(\omega) = T_s p_f(f) H(e^{j\omega T_s}) \quad (3.7.12b)$$

Properties of the transfer function for LTI systems characterized by linear constant-coefficient difference equations will be presented in Chapter 4.

3.8. Fourier Analysis for Discrete-Time Systems

3.8.1. Introduction

The techniques for continuous-time Fourier analysis developed in Section 3.3 are important in analyzing and gaining insight into the properties of signals and systems. The representation of signals and systems in simulation is as finite sets of sampled values. Fourier analysis for sampled systems is performed by the *discrete Fourier transform* (DFT). The DFT, as part of modern numerical analysis, was developed for computer applications, which are always limited in storage capacity. The use of the DFT in simulation as well as in digital signal processing was accelerated by the development of efficient algorithms for computing DFT known as the *fast Fourier transform* (FFT).

The DFT is a discrete-sum spectral representation of a finite-length sequence. To provide continuity with the naming conventions used in the previous sections, we will refer to this latter sequence as the *time-domain sequence*. The transformed function, the *frequency-domain sequence*, is also a finite-length sequence of the same length.

The two most important applications of the DFT in simulation of communication systems are spectral analysis, which is covered in Chapter 10, and spectral shaping of signals or filtering, which is taken up in Chapter 4. The impulse response of a DFT-type filter is a finite-length sequence. Therefore, the DFT is used in applications of *finite-impulse-response* (FIR) filters (see Section 4.2). FIR filtering via the DFT is often referred to as *frequency-domain filtering*, and is the preferred method for simulations of filters specified in the frequency domain. The method, because of its use of the FFT algorithm, is also a computationally efficient method of filtering in simulation applications.

Although the properties of the discrete Fourier analysis are quite similar to the continuous case, there are differences. These differences are caused by both the discrete time and finite duration representation of the sampled signals.

3.8.2. The Discrete Fourier Transform

Although the DFT can be developed independently as a theory of finite sequences, we choose here to derive the DFT from the continuous form. This approach is important in simulation of communication systems because we deal with discrete representation of continuous signals and the accuracy of this representation has to be estimated. In addition, it allows us to better understand the similarities and the differences between the continuous and discrete transforms.

The Fourier series for a periodic signal with a period T , as developed in Section 3.3, and repeated here for convenience, is

$$x(t) = \sum_{n=-\infty}^{\infty} \alpha_n e^{j2\pi n f_0 t} \quad (3.8.1)$$

and the spectral coefficients are computed from

$$\alpha_n = \frac{1}{T} \int_0^T x(t) e^{-j2\pi n f_0 t} dt \quad (3.8.2)$$

where $T = f_0^{-1}$ is the period of $x(t)$.

To obtain a finite-sequence approximation of (3.8.1) and (3.8.2), we examine the periodic analog signal in Figure 3.29a and the sampled version in Figure 3.29b. The approximation of the integral in (3.8.2) is obtained by substituting $t \rightarrow kT_s$, $f_0 \rightarrow \Delta F = 1/T$, $\alpha_n \rightarrow X(n\Delta F)$, and $dt \rightarrow T_s = T/N$, where N is the number of samples in one period, ΔF is the frequency increment, and T_s is the sampling period. We have

$$X(n\Delta F) = \frac{1}{N} \sum_{k=0}^{N-1} x(kT_s) e^{-j2\pi n \Delta F k T_s} = \frac{1}{N} \sum_{k=0}^{N-1} x(kT_s) e^{-j2\pi n k / N} \quad (3.8.3)$$

The infinite sum in (3.8.1) becomes

$$x(kT_s) = \sum_{n=0}^{N-1} X(n\Delta F) e^{j2\pi n \Delta F k T_s} = \sum_{n=0}^{N-1} X(n\Delta F) e^{j2\pi n k / N} \quad (3.8.4)$$

The expressions in (3.8.3) and (3.8.4) are essentially the formulation of a Fourier transform for discrete signals of finite length. However, the DFT in universal usage was historically developed independently of the continuous Fourier analysis and has therefore a somewhat different form,

$$X(n) = \mathbf{F}[x(k)] = \sum_{k=0}^{N-1} x(k) W_N^{nk}, \quad n = 0, 1, \dots, N-1 \quad (3.8.5)$$

and the inverse transform (IDFT) is

$$x(k) = \mathbf{F}^{-1}[X(n)] = \frac{1}{N} \sum_{n=0}^{N-1} X(n) W_N^{-nk}, \quad k = 0, 1, \dots, N-1 \quad (3.8.6)$$

where for notational simplicity we define the weighting kernel

$$W_N = e^{-j2\pi/N}$$

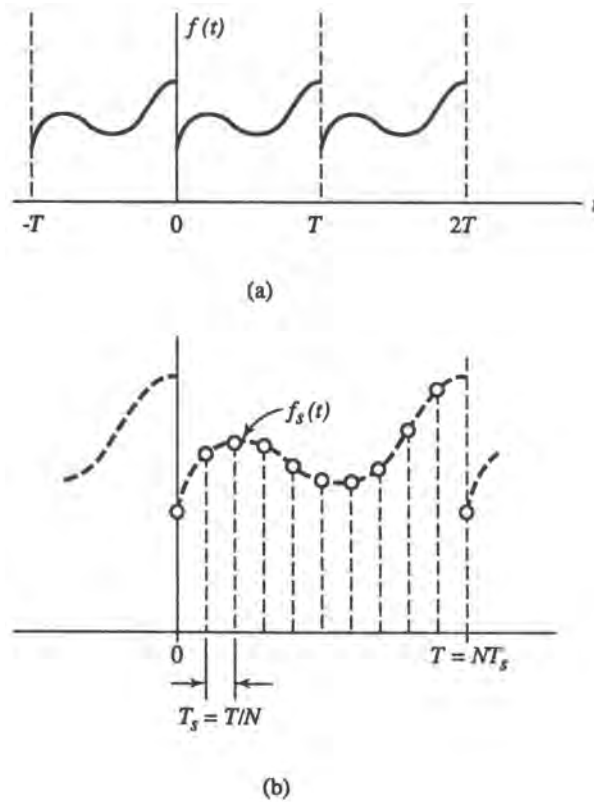


Figure 3.29. Definition of a sampled periodic signal. (a) Continuous periodic signal; (b) sampled periodic signal.

As can be seen from the comparison of (3.8.3) with (3.8.5), and (3.8.4) with (3.8.6), the factor $1/N$ was dropped in (3.8.5) and reinserted in (3.8.6) in the above definitions for historical reasons.

3.8.3. The Fast Fourier Transform

The discrete Fourier transform plays an important role both in the simulation and implementation of signal processing and filtering algorithms. One of the reasons that Fourier analysis is so important in simulation of signal processing is because of the existence of efficient algorithms for computing the DFT. The fast Fourier transform (FFT) is the name applied to various algorithms that rapidly make the computation to obtain the DFT of a sequence. Some introductory discussion of the FFT will be given below. The detailed description of the various FFT algorithms is beyond the scope of this book and can be found in many excellent references.^(3,30,31)

One method is the decimation-in-time algorithm. The method is applicable when N is a product of two or more integers. The most efficient algorithms are for $N = 2^v$. The sequence in (3.8.5) is recursively divided into two sequences till two one-point sequences are obtained. By cleverly arranging these successive sequences and using the periodicity property of

$W_N^k = W_N^{k(n+N)} = W_N^{(k+N)n}$, we can greatly reduce the number of operations. The entire FFT requires $(N/2) \log_2 N$ complex multiplications and $N \log_2 N$ complex additions. On the other hand, it can be shown that the direct computation of the N values of $X(n)$ in (3.8.5) requires approximately N^2 complex multiplications and additions. Obviously the same is true for $x(k)$ of (3.8.6) [$x(k)$ is usually complex in simulation applications]. Since addition is a much simpler operation than multiplication, we will use the number of multiplications as a measure of the efficiency of the algorithm. Thus the computational effort is decreased by $R = N^2 / (0.5N \log_2 N)$. As an example, for $N \approx 1000$ the number of complex multiplications is reduced by $R \approx 10^6 / 5000 = 200$.

Other algorithms allow N to be a composite number of the integers 2, 3, 4, 5, etc.³² These algorithms are less time-efficient than the $N = 2^v$, but are more flexible in the choice of the size of the DFT.

3.8.4. Properties of the Discrete Fourier Transform

As we will see in the following section, there are many similarities and some differences between the continuous and the discrete Fourier transforms. We will focus primarily on those properties that are unique to the DFT. When the properties of the DFT can be directly translated from the properties of the continuous transform, we will simply state this property.

1. *Linearity*:

$$F[a_1x_1(k) + a_2x_2(k)] = a_1X_1(n) + a_2X_2(n) \quad (3.8.7a)$$

2. *Symmetry*. If $x(k)$ and $X(n)$ are a DFT pair, then

$$F\left[\frac{1}{N}X(k)\right] = x(-n) \quad (3.8.7b)$$

3. *Parseval's Theorem*. For discrete functions the relationship between the power computed in the time domain and frequency domain is

$$\sum_{k=0}^{N-1} |x(k)|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |X(n)|^2 \quad (3.8.7c)$$

3.8.4.1. Periodic or Circular Properties

The DFT is useful in approximating the continuous Fourier transform. However, the DFT is defined as an operation on finite sequences $x(k)$ with N samples. That results in another finite sequence $X(n)$ also of length N . As shown in Section 3.8.2, the DFT is equivalent to a Fourier transform of a periodic sampled signal. The values produced by the IDFT of $X(n)$ outside the range $0 \leq k \leq N-1$ are thus repetitions of the values within the range. These values can be regarded as periodic extensions of $x(k)$. We thus represent the circular or *periodic operation* as the replacement of the finite sequence $x(k)$ by a periodic extended sequence

$$x_p(k + N) = x(k) \quad \text{for all } N$$

(Another way to look at the circular property is view the sequence as being arranged on a circle with N divisions, one for each sample.)

The result of the periodic operation is another periodic sequence $y_p(k)$ and the finite sequence extracted from it is

$$y(k) = \begin{cases} y_p(k), & 0 \leq k \leq N-1 \\ 0, & \text{otherwise} \end{cases}$$

A prime example of a circular property of the DFT is described below.

3.8.4.2. The Periodic Time-Shift Property

As in the continuous Fourier transform, a shift in the time domain corresponds to a phase shift in the frequency domain

$$F[x(k-m)] = X(n)W_N^{nm} \quad (3.8.9)$$

The problem is to understand what kind of time shift $x(k-m)$ represents, or in other words, what is the time function produced by an IDFT of $X(n)W_N^{nm}$? An example of a circular time shift is illustrated in Figure 3.30. The sequence $x(k)$ from $0 \leq k \leq 4$ is shown in Figure 3.30a.

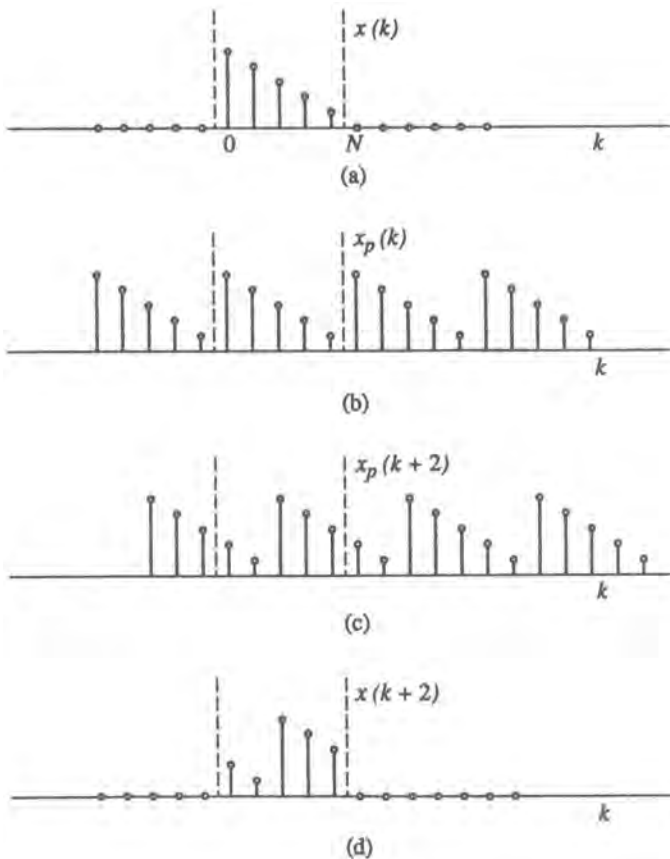


Figure 3.30. Periodic time shift of a sequence.

The $x_p(k)$ shown in Figure 3.30b is the periodic extension of $x(k)$. Figure 3.30c shows the shifted periodic extension $x_p(k + 2)$. Thus the IDFT of the DFT of the sequence $x(k + 2)$ is the sequence shown in Figure 3.30d.

3.8.4.3. The Periodic or Circular Convolution

As in the time-shifting theorem, the convolution theorem for finite sequences is also a circular operation. One way to perform the circular convolution is by way of the periodic extension method used in the time-shifting property above. [Another way to represent the circular convolution is by placing the N samples of $x(k)$ on the circumference of one circle and the N samples of $h(k)$ in reverse on another circle, then rotating one circle against the other and multiplying and adding the corresponding samples.] The circular or periodic convolution is the discrete form of the periodic convolution of continuous waveforms in Section 3.3.6,

$$y(n) = x_p(n) \otimes h_p(n) = \sum_{m=0}^{N-1} x_p(m) h_p(n - m) \quad (3.8.10)$$

An example of a circular convolution of two finite sequences $x(m)$ and $h(m)$ of length N is illustrated in Figure 3.31. Shown in the figure are the periodically extended sequences $x_p(m)$

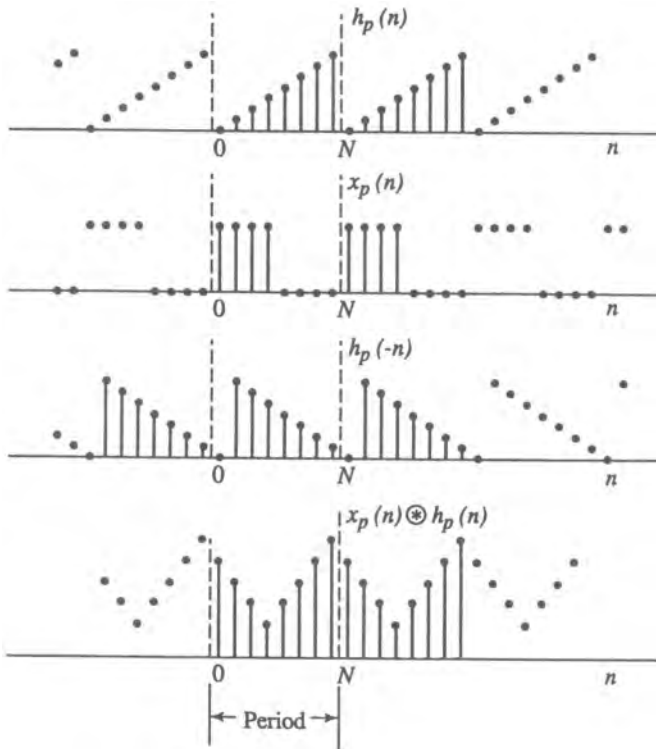


Figure 3.31. Circular convolution.

and $h_p(\mathbf{m})$ and the periodically extended convolution $y_p(\mathbf{k})$. The circular convolution $y(k)$ is one period of the periodically extended convolution $y_p(k)$.

3.8.4.4. The Discrete Periodic Convolution Theorem

The discrete periodic convolution theorem states that if $X_n = \mathbf{F}(x_k)$ and $H_n = \mathbf{F}(h_k)$, then the DFT of the convolution

$$y(k) = x_p(k) * h_p(k)$$

is the product

$$Y(n) = \mathbf{F}[y(k)] = X(n)H(n) \quad (3.8.11)$$

3.8.4.5. The Discrete Frequency Response

Analogous to the continuous system definition in Section 3.3, the *discrete frequency response* is defined as the response of a discrete system to a discrete complex exponential input. A discrete complex exponential input

$$x(n) = W_N^{-mn}$$

where n indexes the sampling instant and m/N is the relative frequency, is applied to a system with an impulse response $h(k)$. The output of the system is

$$\begin{aligned} y(n) &= \sum_{k=0}^{N-1} h(k)x(n-k) = \sum_{k=0}^{N-1} h(k)W_N^{-m(n-k)} \\ &= W_N^{-mn} \sum_{k=0}^{N-1} h(k)W_N^{mk} \end{aligned}$$

Hence

$$y(n) = W_N^{-mn}H(m); \quad H(m) = \sum_{k=0}^{N-1} h(k)W_N^{mk} \quad (3.8.12a)$$

From (3.8.12a) we see that $H(m)$ describes the change in complex amplitude of the discrete complex exponential as a function of the discrete frequency m . Thus $H(m)$ is referred to as the *discrete frequency response* of the system and it plays a major role in analysis of linear time-invariant systems. In general, $H(m)$ is complex with an amplitude $A(m)$ and phase $\phi(m)$,

$$H(m) = A(m)e^{j\phi(m)} \quad (3.8.12b)$$

Since $h(k)$ completely characterizes a discrete FIR system, so does $H(m)$.

3.8.4.6. Relationship between the Bandwidth and the Duration of the Impulse Response

In the FFT we use the same number of samples for both the frequency domain and time domain. In the frequency domain the frequency span is $\mathbf{F} = N \Delta\mathbf{F}$, where N is the number of frequency increments $\Delta\mathbf{F}$. In the time domain the duration of the impulse response is

$T = NT_s$, where T_s is the sampling interval. Because of the periodicity of the DFT in both the frequency and time domains, we have the relationships

$$\Delta F = 1/T \quad \text{and} \quad T_s = 1/F \quad (3.8.13)$$

This relationship is obvious if the DFT is derived via the Fourier series (see Section 3.8.2). It is clear that because $f_s = 1/T_s$, then $f_s = F$.

3.8.4.7. Relationship between the Discrete Fourier Transform and the z -Transform

We want to find a relationship between the z -transform of a finite-duration sequence and the DFT coefficients of the same sequence. Given a sequence $x(n)$, $0 \leq n \leq N-1$, it can be shown³³ that

$$\hat{X}(z) = \sum_{n=0}^{N-1} x(n)z^{-n} = \sum_{k=0}^{N-1} \frac{X(k)}{N} \frac{1 - z^{-N}}{1 - z^{-1} \exp[j(2\pi/N)k]} \quad (3.8.14)$$

where $\hat{X}(z)$ is the z -transform and $X(k)$ is the DFT of the sequence $x(n)$. If (3.8.14) is evaluated on the unit circle, we get the frequency response [see (3.7.12)]

$$\hat{X}[e^{j2\pi f T_s}] = \sum_{k=0}^{N-1} \frac{X(k)}{N} \frac{\exp[-j2\pi f T_s(N-1)/2] \sin(2\pi f T_s N/2)}{\exp(j2\pi f T_s/N) \sin(2\pi f T_s/2 - \pi k/N)} \quad (3.8.15)$$

The function $\sin(2\pi f T_s N/2) / \sin(2\pi f T_s/2 - \pi k/N)$ acts as an interpolation for the samples of the DFT $X(k)$ and is a periodic equivalent of the sinc function interpolation for nonperiodic bandlimited functions (see Section 3.5).

3.8.4.8. Increasing the Frequency Resolution of the Discrete Fourier Transform

For a finite-duration N -point sequence $x(n)$ the above provides the means for evaluating the frequency response for L uniformly spaced frequencies on the unit circle where L can be much larger than N , thus yielding an increased resolution. From the left-hand side of (3.8.14) we get

$$\hat{X}[e^{j2\pi f T_s}] = \sum_{n=0}^{N-1} x(n)e^{-j2\pi f T_s n} \quad (3.8.16)$$

If $\hat{X}[e^{j2\pi f T_s}]$ is evaluated at L frequencies $f_l = l \Delta F = l/(LT_s)$ for $l = 0, 1, \dots, L-1$, we find

$$\hat{X}[e^{j2\pi l/L}] = \sum_{n=0}^{N-1} x(n)e^{-j2\pi l n/L} \quad (3.8.17)$$

We want to express (3.8.17) as an L -point DFT. Since the DFT assumes that the time- and frequency-domain sequences have the same number of samples, we have to find a finite sequence $x_a(n)$ with L samples that produces the same values as (3.8.17). To do this, let us define an L point sequence $x_a(n)$ with $L > N$ as

$$x_a(n) = \begin{cases} x(n), & 0 \leq n \leq N-1 \\ 0, & N \leq n \leq L-1 \end{cases} \quad (3.8.18)$$

The sequence $x_a(n)$ is often referred to as a *zero-padded* sequence. By taking the L -point DFT we get

$$\begin{aligned} X_a(k) &= \sum_{n=0}^{L-1} x_a(n) e^{-j2\pi kn/L} \\ &= \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/L} = \hat{X}[e^{-j2\pi k/L}] \end{aligned} \quad (3.8.19)$$

The reader can verify that the same results would arise from (3.8.15) by setting $f = k/T$ (see Problem P4.7)

Thus we have shown that the simple technique of augmenting a finite-duration sequence with zero-valued samples allows one to compute the frequency response with arbitrary resolution. This simple technique is one of the most useful techniques in spectrum analysis of finite-duration sequences.

3.9. Summary

The main thrust of this book is to develop the theoretical tools and associated computing techniques for simulating communication systems whose waveforms are continuous both in the amplitude and the time domains. In this chapter we concentrated on the theoretical tools that are the basis for simulating linear time-invariant (LTI) systems driven by deterministic signals. The tools for describing LTI systems apply, of course, whether the inputs are deterministic or random. For random signals, however, different modes of description are necessary, and are covered in Chapters 6 and 7.

Strictly speaking, simulation of time-continuous waveforms in the digital computer is not possible. Signals and the responses of systems can only be simulated in discrete time. We therefore focused initially on the transition between the continuous-time and the discrete-time representation of signals and systems. This, of course, is carried out through the sampling theorem. However, in order to understand this transformation, as well as any errors that might be associated with it, it was first necessary to develop the proper language of description, namely the Fourier series and Fourier transform. We then extended the coverage to the Laplace transform and reviewed the associated discrete-time description of signals via the z -transform.

Since one of our main objectives is the simulation of carrier-modulated, or bandpass, systems, we developed an efficient representation of such signals for simulation purposes, namely the complex envelope. The complex envelope applies, of course, to both continuous-time and discrete-time representations.

The most ubiquitous linear time-invariant (LTI) operation is filtering. Indeed, all LTI operations can be regarded as filtering. Filtering in discrete time can be regarded as a form of digital signal processing. There are two basic forms of discrete-time processing. One is based on the z -transform, which itself is the discrete-time counterpart of the Laplace transform. A brief review of these topics was given. The net embodiment of a discrete-time filter based on these ideas is a linear recursive difference equation. This aspect is discussed in detail in Chapter 4.

The second approach to discrete-time filtering is based on the discrete Fourier transform (DFT), which is normally implemented using the fast Fourier transform (FFT) technique. We

spent some time on the basis for the FFT as well as its computational requirements. The actual application of the FFT to filtering, along with specific guidance and practical hints on how to handle specific types of filtering situations, is also covered in the next chapter.

3.10. Appendix: A Brief Summary of Some Transforms and Theorems Useful in Simulation

In analysis and simulation, it is common to process signals in the time domain or in the frequency domain, and switch from one domain to the other as convenience dictates. Certain operations are repeatedly used, such as filtering and correlation. Furthermore, although we process in discrete time in simulation, this processing is in many cases an approximation to a continuous-time operation.

In light of the preceding, it is useful to collect in this Appendix a number of relationships and transforms that are commonly used. Tables 3.A.1–3.A.4 provide various versions of basic theorems (relationships) in the continuous, discrete, or mixed mode domains. The subscript p in some of the expressions stands for “periodic” and indicates that the function so labeled is periodic. The basic analytical description of such periodic functions has the form given for $Z_p(f)$ in Table 3.A.1. The symbol $\oplus$ appearing in certain places stands for periodic convolution.

Table 3.A.5–3.A.7 list some basic Fourier, Laplace, and z -transform pairs that arise frequently in practice.

Table 3.A.1. The Convolution Theorem in Various Domains

Continuous-time, continuous-frequency

$$z(t) = \int_{-\infty}^{\infty} x(\tau)y(t-\tau) d\tau = \int_{-\infty}^{\infty} x(t-\tau)y(\tau) d\tau$$

$$Z(f) = \int_{-\infty}^{\infty} z(t)e^{-j2\pi ft} dt = X(f)Y(f)$$

Discrete-time, continuous-frequency

$$z(kT_s) = \sum_{n=-\infty}^{\infty} T_s x[(k-n)T_s] y(nT_s)$$

$$Z_p(f) = T_s \sum_{n=-\infty}^{\infty} z(kT_s) e^{-j2\pi n T_s f} = \sum_{k=-\infty}^{\infty} Z(f + k/T_s) = X_p(f)Y_p(f)$$

Discrete-time, discrete-frequency

$$z_k = \sum_{n=0}^{N-1} x_n y_{k-n} = \sum_{m=0}^{N-1} x_{k-m} y_m \equiv x_k \oplus y_k$$

$$Z_r = \sum_{k=0}^{N-1} z_k \exp(-j2\pi rk/N) = X_r Y_r$$

where indices are taken modulo N

Table 3.A.2. Parseval's Theorem in Various Domains

Continuous-time, continuous-frequency

$$\int_{-\infty}^{\infty} x(t)y^*(t) dt = \int_{-\infty}^{\infty} X(f)Y^*(f) df$$

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |X(f)|^2 df$$

Discrete-time, continuous-frequency

$$\sum_{n=-\infty}^{\infty} T_s x(nT_s)y^*(nT_s) = \int_0^{f_s} X_p(f)Y_p^*(f) df; \quad f_s = 1/T_s$$

$$\sum_{n=-\infty}^{\infty} T_s |x(nT_s)|^2 = \int_0^{f_s} |X_p(f)|^2 df$$

Discrete-time, discrete-frequency

$$\sum_{k=0}^{N-1} x_k y_k^* = \frac{1}{N} \sum_{r=0}^{N-1} X_r Y_r^*; \quad k, r = 0, 1, \dots, N-1$$

$$\sum_{k=0}^{N-1} |x_k|^2 = \frac{1}{N} \sum_{r=0}^{N-1} |X_r|^2$$

Table 3.A.3. The Frequency Convolution Theorem in Various Domains

Continuous-time, continuous-frequency

$$z(t) = x(t)y(t)$$

where $x(t)$ and $y(t)$ may be complex

$$Z(f) = \int_{-\infty}^{\infty} X(f')Y(f-f') df' = \int_{-\infty}^{\infty} X(f-f')Y(f') df'$$

Discrete-time, continuous-frequency

$$z(kT_s) = x(kT_s)y(kT_s)$$

$$Z_p(f) = \int_0^{f_s} X_p(f')Y_p(f-f') df' = \int_0^{f_s} X_p(f-f')Y_p(f') df' \equiv X_p(f) \oplus Y_p(f)$$

Discrete-time, discrete-frequency

$$z_k = x_k y_k; \quad k = 0, 1, \dots, N-1$$

$$Z_r = \frac{1}{N} \sum_{n=0}^{N-1} X_n Y_{r-n}; \quad r = 0, 1, \dots, N-1 \equiv \frac{1}{N} X_r \otimes Y_r$$

Table 3.A.4. The Correlation Theorem in Various Domains

Continuous-time, continuous-frequency

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t + \tau)y^*(t) dt = \int_{-\infty}^{\infty} X(f)Y^*(f)e^{j2\pi f\tau} df$$

Discrete-time, continuous-frequency

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} T_x x[(n+k)T_s]y^*(nT_s) = \int_0^{f_s} X_p(f)Y_p^*(f)e^{j2\pi kT_s f} df$$

Discrete-time, discrete-frequency

$$R_{xy}(k) = \frac{1}{N} \sum_{n=0}^{N-1} x_{n+k}y_n^*, \quad k = 0, 1, \dots, N-1 = \frac{1}{N^2} \sum_{r=0}^{N-1} X_r Y_r^* e^{j2\pi rk/N}$$

Table 3.A.5. Basic Fourier Transform Pairs^a

Signal $x(t)$	Fourier transform $X(f)$
$\delta(t)$	1
$u(t)$	$\frac{1}{j2\pi f} + \frac{1}{2}\delta(f)$
$\delta(t - t_0)$	$e^{-j2\pi f t_0}$
$e^{-\alpha t}u(t), \alpha > 0$	$\frac{1}{\alpha + j2\pi f}$
$e^{-\alpha t }, \alpha > 0$	$\frac{2\alpha}{\alpha^2 + (2\pi f)^2}$
$e^{-\pi(t/T)^2}$	$T e^{-(\pi f T)^2}$
$t^n e^{-\alpha t}u(t), \alpha > 0$	$\frac{n!}{(\alpha + j2\pi f)^{n+1}}$
$e^{j2\pi f_0 t}$	$\delta(f - f_0)$
$\cos(2\pi f_0 t)$	$\frac{1}{2}[\delta(f - f_0) + \delta(f + f_0)]$
$\sin(2\pi f_0 t)$	$\frac{1}{2j}[\delta(f - f_0) - \delta(f + f_0)]$
$\cos(2\pi f_0 t)[p_{(t/T)}]$	$T \text{sinc}(f - f_0)T$
1	$\delta(f)$
$\sum_{n=-\infty}^{+\infty} \delta(t - nT)$	$\frac{1}{T} \sum_{k=-\infty}^{+\infty} \delta\left[f - \frac{k}{T}\right]$
$p_T(t)$	$2T \text{sinc}(2fT)$
$2W \text{sinc}(2Wt)$	$p_w(f)$
$\begin{cases} 1 - t /T & \text{for } t \leq T \\ 0 & \text{for } t > T \end{cases}$	$T \text{sinc}^2 fT$
$J_0(2\pi at)$	$\frac{1}{\pi\sqrt{a^2 - f^2}}$

^aNote the rectangular function: $p_a(t) = 1$ for $|t| \leq a$, and $p(t) = 0$ for $|t| > a = 0$ for $|t| > a$; see (3.1.1c).

Table 3.A.6 Laplace Transforms of Elementary Functions

Signal $f(t)$	Transform $F(s)$
$\delta(t)$	1
$u(t)$	$\frac{1}{s}$
$t^n u(t)$	$\frac{n!}{s^{n+1}}$
$e^{-\alpha t} u(t)$	$\frac{1}{s + \alpha}$
$t^n e^{-\alpha t} u(t)$	$\frac{n!}{(s + \alpha)^{n+1}}$
$\delta(t - T)$	e^{-sT}
$[e^{-\alpha t} \cos(\omega_0 t)]u(t)$	$\frac{s + \alpha}{(s + \alpha)^2 + \omega_0^2}$
$[e^{-\alpha t} \sin(\omega_0 t)]u(t)$	$\frac{\omega_0}{(s + \alpha)^2 + \omega_0^2}$
$\sin at$	$\frac{a}{s^2 + a^2}$
$\cos at$	$\frac{s}{s^2 + a^2}$
$\sinh at$	$\frac{a}{s^2 - a^2}$
$\cosh at$	$\frac{s}{s^2 - a^2}$
$\frac{1}{\sqrt{\pi t}}$	$\frac{1}{\sqrt{s}}$

Table 3.A.7. Some Common z-Transform Pairs

Signal $f(n)$	Transform $F(z)$
$\delta[n]$	1
$u[n]$	$\frac{1}{1 - z^{-1}}$
$\delta[n - m]$	z^{-m}
$\alpha^n u[n]$	$\frac{1}{1 - \alpha z^{-1}}$
$n \alpha^n u[n]$	$\frac{\alpha z^{-1}}{(1 - \alpha z^{-1})^2}$
$e^{-\alpha n} u[n]$	$\frac{1}{1 - e^{-\alpha} z^{-1}}$
n	$\frac{z}{(z - 1)^2}$
n^2	$\frac{z(z + 1)}{(z - 1)^3}$
$\frac{1}{n}, n > 0$	$\ln \frac{z}{z - 1}$
$[r^n \cos(\omega_0 n)]u[n]$	$\frac{1 - [r \cos \omega_0]z^{-1}}{1 - [2r \cos \omega_0]z^{-1} + r^2 z^{-2}}$
$[r^n \sin(\omega_0 n)]u[n]$	$\frac{[r \sin \omega_0]z^{-1}}{1 - [2r \cos \omega_0]z^{-1} + r^2 z^{-2}}$
$[r^n \sinh(\omega_0 n + \psi)]u[n]$	$\frac{z^2 \sinh \psi - zr \sinh(\omega_0 - \psi)}{z^2 - 2zr \cosh \omega_0 + r^2}$

References

1. A. Papoulis, *Signal Analysis*, McGraw-Hill, New York (1977).
2. A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*, Prentice-Hall, Englewood Cliffs, New Jersey (1983).
3. R. E. Ziemer, W. H. Tranter, and D. R. Fannina, *Signals and Systems Continuous and Discrete*, Macmillan, New York (1983).
4. R. J. Schwarz and B. Friedland, *Linear Systems*, McGraw-Hill, New York (1965).
5. C. D. McGillem and G. R. Cooper, *Continuous and Discrete Signal and System Analysis*, Holt, Rinehart and Winston, New York (1974).
6. R. A. Gabel and R. A. Roberts, *Signals and Linear Systems*, 2nd ed., Wiley New York (1980).
7. A. Papoulis, *The Fourier Integral and Its Applications*, McGraw-Hill, New York (1962).
8. R. N. Bracewell, *The Fourier Transform and Its Applications*, 2nd ed., McGraw-Hill, New York (1978).
9. H. Urkowitz, *Signal theory and Random Processes*, Artech House, Dedham, Massachusetts (1983).
10. A. D. Whalen, *Detection of Signals in Noise*, Academic Press, New York (1971).
11. M. Schwartz, W. R. Bennett, and S. Stein, *Communication Systems and Techniques*, McGraw-Hill, New York (1966).
12. K. S. Shanmugan, *Digital and Analog Communication Systems*, Wiley, New York (1979).
13. A. J. Jerri, The Shannon sampling theorem—Its various extensions and applications; A tutorial review, *Proc. IEEE* **65**(11), 1565–1596 (1997).
14. A. I. Zayed, *Advances in Shannon's Sampling Theory*, CRC Press, Boca Raton, Florida (1993).
15. J. G. Proakis, *Digital Communications*, 2nd ed., McGraw-Hill, New York (1989).
16. S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
17. R. E. Ziemer and W. H. Tranter, *Principles of Communications: Systems, Modulation, and Noise*, Houghton Mifflin, Boston (1976).
18. R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1983).
19. A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1989).
20. P. J. Davis, *Interpolation and Approximation*, Dover, New York (1975).
21. T. J. Rivlin, *An Introduction to the Approximation of Functions*, Dover, New York (1981).
22. J. E. Ahlberg, E. N. Nilson, and J. L. Walsh, *The Theory of Splines and Their Applications*, Academic Press, New York (1967).
23. C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York (1978).
24. W. Gautschi, *Numerical Analysis*, Birkhäuser, Boston (1997).
25. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York (1980).
26. A. Papoulis, *Circuits and Systems*, Holt, Rinehart and Winston, New York (1980).
27. L. Weinberg, *Network Analysis and Synthesis*, McGraw-Hill, New York (1962).
28. G. Doetsch, *Introduction to the Theory and Application of the Laplace Transformation with a Table of Laplace Transformations*, Springer-Verlag, Berlin (1974).
29. E. I. Jury, *Theory and Application of the z-Transform Method*, Wiley, New York (1964).
30. A. V. Oppenheim and R. W. Schaffer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1977).
31. E. O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, New Jersey (1974).
32. R. C. Singleton, An algorithm for computing the mixed radix fast Fourier transform, *IEEE Trans. Audio Electroacoust.* **AU-17**(2), 93–103 (1969).
33. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1975).

This page intentionally left blank

Modeling and Simulation of Linear Time-Invariant and Time-Varying Systems

In this chapter we begin our discussion of modeling and simulation per se, confining ourselves exclusively to linear systems. Probably the most ubiquitous devices, components, or subsystems are *linear* and *time-invariant* (LTI), or at least can be reasonably so modeled.[†] Certainly, the class of linear time-invariant filters forms a large set of LTI elements, and for analytical purposes there is no distinction between such filters and the seemingly more general term LTI system. For brevity, therefore, we shall often refer simply to filters, LTI being implied unless otherwise stated. The first part of this chapter will be devoted to the modeling and simulation of LTI systems. The second main topic of this chapter is the important subclass of linear *time-varying* systems. In Chapter 9 we shall study specifically randomly time-varying (multipath) channels, which we encounter in mobile applications. In this chapter, we shall give a somewhat more general introduction to the subject, though without the random variation: the models of Chapter 9 will be seen to be structurally the same as those introduced here, with statistical properties superimposed.

The discrete form of simulation used for communication systems requires representation of continuous systems in the discrete-time domain. The principal focus of this chapter, therefore, is on methods of approximating linear continuous systems with discrete systems that are adapted for simulation.

4.1. Modeling and Simulation of Linear Time-Invariant Systems

There are many possibilities for simulating LTI systems, depending to a large extent on the manner in which the transfer characteristics of the filters are described or specified. For example, continuous systems that are described by LTI differential equations are typically transformed into LTI difference equations.^(1, 2) Thus obtained discrete systems are represented by recursive equations that generally result in filters having *infinite impulse responses* (IIR) in discrete time. Discrete systems can also be modeled with impulse responses that are repre-

[†] Since a cascade of linear elements is another linear element, we need not make a great distinction among systems, subsystems, components, etc., so long as they are linear.

sented by finite-length sequences. These sequences can be obtained by truncating infinite impulse responses, or from frequency response specifications or measurements taken over a finite frequency interval. Thus, the approximating discrete system could have a *finite impulse response* (FIR).

The *z-transform* can be regarded as the discrete system analysis counterpart of the Laplace transform. By using the *z-transform*, we transform the difference equations that characterize the IIR discrete-time systems into algebraic equations, which are simpler to analyze. The rational functions in the *s*-domain that describe many continuous systems can be mapped into rational functions in the *z*-domain directly. It should be noted that there is not a unique mapping from the *s*-domain to the *z*-domain. In principle, there can be any number of such mappings and since each mapping results in a distinct *z*-domain function, the associated difference equation will also be distinct. Thus, there is more than one way of transforming a continuous system into a discrete-time system.

Finite-length sequences that represent FIR systems can also be conveniently modeled in the *z*-domain. However, precisely because of the finiteness, advantage can be taken of the finite (or discrete) Fourier transform and its efficient implementation via the FFT. In fact, the techniques for simulating filtering are sufficiently different for IIR and FIR systems that it is convenient to separate the discussion of filtering according to the type of impulse response. In addition, it is useful to distinguish between a filter per se and the act of filtering. In this fashion, we can address the representation and approximation of filter structures independent of the filtering process.

Consequently, this section will begin with a discussion of various aspects of describing, specifying, and approximating filters. Along the way, we will present the analytical structure of several classes of widely used continuous filters. We will then present methods of filtering according to whether the approximating filter structure is FIR or IIR. A section at the end will summarize the various steps that must be undertaken to set up a simulation of an LTI system.

It should be noted that both IIR and FIR filters are often used in communication systems design in the form of digital signal processors and transversal equalizers. These filters are formulated directly in discrete form and do not require conversion from their continuous counterparts.⁽¹⁻⁴⁾

4.1.1. LTI Filters: Description, Specification, and Approximation

There are a number of considerations leading to the ultimate representation of filters for purposes of simulation. These considerations are naturally intertwined, but for expository purposes it is useful to identify individual key features and discuss them separately.

To begin, we review some basics about the representation of (continuous) filters. One of these representations leads to a well-known class of filters that we refer to as the “classical” filters, whose structures we will present. Although filters can be described precisely in various kinds of abstractions, in practice it is common to impose requirements on filters (or, more generally, on a string of equipment) in the form of *specifications*. As was discussed in Chapter 2, specifications inherently imply an *imprecise* description. We will briefly discuss two common approaches to specifications, and possible ways of extracting specific filters from them. Lastly, we will address our final objective in this section, namely, how to approximate a continuous-time structure by a discrete-time equivalent suitable for simulation: in effect, this is the modeling problem.

4.1.1.1. Filter Descriptions

Filters may be described in one of several ways. Generally, the descriptions given below can apply to lowpass, bandpass, highpass, or bandstop structures.

1. *Time-Domain Description.* A filter is described in the time domain by its impulse response $h(t)$. For simulation purposes, the impulse response is usually given or expressed in complex lowpass-equivalent form. The impulse response forms the link between the input $x(t)$ and the output $y(t)$ through the well-known convolution theorem^(5,6)

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t - \tau) d\tau \quad (4.1.1)$$

to which we will return in Section 4.1.1.6. Although the impulse response is often used in an approximate implementation of (4.1.1), its form is usually derived from the filter's frequency-domain description rather than given directly in the time domain.

If $h(t) \neq 0$ only for a finite stretch of time, it is called a finite impulse response (FIR) filter; otherwise, it is an infinite impulse response (IIR) filter. How we obtain one or the other representation will unfold below.

2. *Frequency-Domain Description.* One frequency-domain representation can be a *prescribed* function of frequency $H(f)$ for the amplitude and phase $A(f)$ and $\theta(f)$, respectively:

$$H(f) = A(f) \exp[j\theta(f)] \quad (4.1.2)$$

which is used in the frequency-domain counterpart of (4.1.1),

$$y(t) = \int_{-\infty}^{\infty} H(f)X(f)e^{j2\pi ft} df \quad (4.1.3)$$

where H and X are the Fourier transforms of h and x , respectively.

A second frequency-domain description is a *measured* function of frequency $\hat{H}(f)$ with corresponding amplitude and phase functions $\hat{A}(f)$ and $\hat{\theta}(f)$, respectively. The distinction between this and the previous mode of description is that measurements can only be made at a finite number of frequencies $f_i, i = 1, \dots, N$. Hence, to arrive at a complete description will generally require interpolation and extrapolation, implying some degree of approximation. A side point about using measured characteristics is the fact that such measurements can never be perfect, and thus create additional error in the simulation results.

3. *Complex-Plane (s -Plane) Description.* A complex-plane description consists of a transfer function $H(s)$, which is usually a ratio of polynomials in the complex variable s . In that case, the filter can be described alternatively as a set of prescribed pole and zero locations, by the coefficients of the polynomials, or by the coefficients of the associated biquadratic sections (see Section 3.6.6). The resulting function $H(s)$ is $H(f)$ when we set $s = j2\pi f$, and is the key to the version of (4.1.1) or (4.1.3) based on the Laplace transform. While more or less arbitrary pole/zero sets can be legitimate filters (subject to realizability conditions), certain classes of filters defined by their pole/zero locations have come into general use. We refer to these as the *classical* filters, and include the well-known Butterworth, Chebyshev, Bessel, and elliptic types. Because of their importance, we now give a short description of these filter classes.

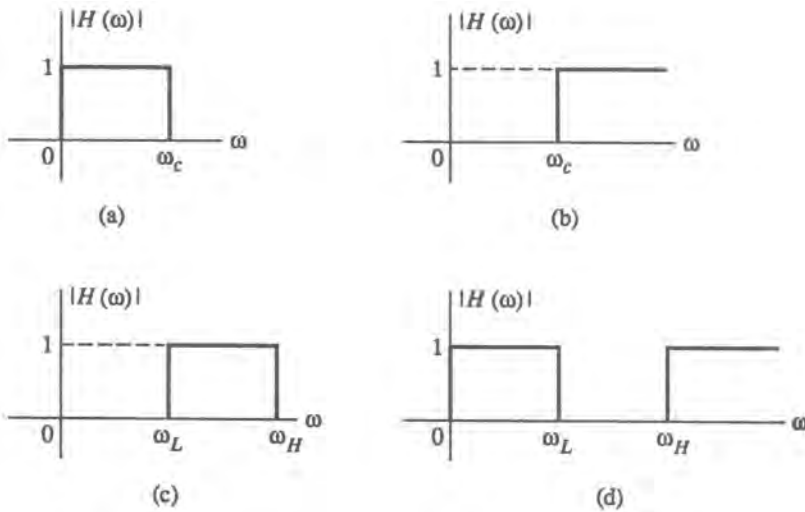


Figure 4.1. Ideal filters in the frequency domain. (a) Lowpass, (b) highpass, (c) bandpass, (d) bandstop.

4.1.1.2. Continuous Classical Filters

A class of filters, often called the *classical* filters, have been developed to approximate the amplitude characteristics of an *ideal* filter. In the current context an ideal frequency-selective filter is one that exactly passes complex exponentials at one set of frequencies and completely rejects them at others. The frequency response $H(\omega)$ of such a filter is shown in Figure 4.1. A filter with frequency response as in Figure 4.1a is called lowpass, as in Figure 4.1b, highpass, as in Figure 4.1c, bandpass, and as in Figure 4.1d, bandstop. The classical filters are particular cases of filters defined by linear constant-coefficient differential equations (see Section 3.6.6). There are several members in this class of filters, each representing a different type of approximation to the ideal filtering characteristics.

Classical filters are specified using only a few parameters (Figure 4.2) such as the passband, stopband, and the tolerances allowed within these bands. These filters are analytically described by the type of polynomial used in $H(s)$, such as Chebyshev, elliptical, etc.

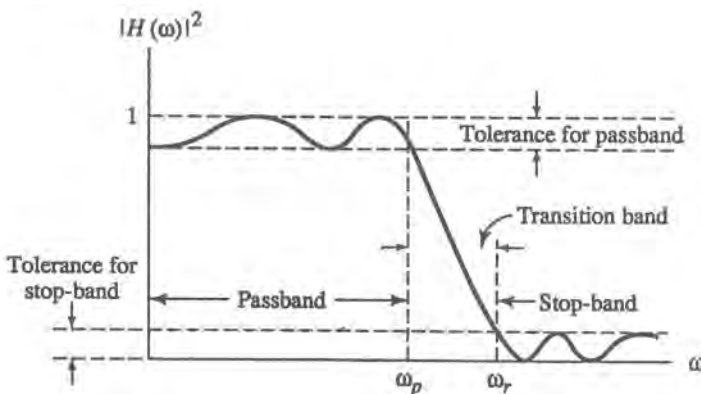


Figure 4.2. Specifications for selective filters.

The properties of these filters are described below. From the filter type and its parameters, the transfer function $H(s)$ in its various forms (poles and zeros, rational function of polynomials in s , etc.; see Section 3.6.6) is computed from given algorithms. Moreover, the transfer function needs to be computed for the normalized lowpass filter only. The highpass or bandpass filters are computed from the normalized lowpass filter through a transformation of variables (see the next section), while different passband edge frequencies amount only to a simple scaling. The transfer functions for classical lowpass filters are tabulated and available in the literature.⁽⁷⁾ For simulation purposes the biquadratic form of $H(s)$, (given in 3.6.21) and repeated here for convenience,

$$H(s) = K \prod_{i=0}^M \frac{\alpha_{2i}s^2 + \alpha_{1i}s + \alpha_{0i}}{s^2 + \beta_{1i}s + \beta_{0i}}$$

is the most desirable. Tables of coefficients for the biquadratic realization of classical filters are given in the Appendix to this chapter.

Butterworth Filters. Butterworth filters form a class of all-pole filters with a maximally flat frequency response (see Figure 4.3a):

$$|H(\omega)|^2 = \frac{1}{1 + (\omega/\omega_b)^{2n}} \quad (4.1.4)$$

The parameter n is referred to as the order of the filter and ω_b is the 3-dB bandwidth. As $n \rightarrow \infty$, $|H(\omega)|^2$ approaches the ideal lowpass filter characteristics of Figure 4.1a.

It can be shown that the system function is of the form

$$H(s) = \frac{\omega_b^n}{\prod_{k=1}^n (s - s_k)} \quad (4.1.5)$$

where the $s_k = \omega_b \exp\{j\pi[1/2 + (2k-1)/2n]\}$ are the poles, located on a semicircle as in Figure 4.3b.

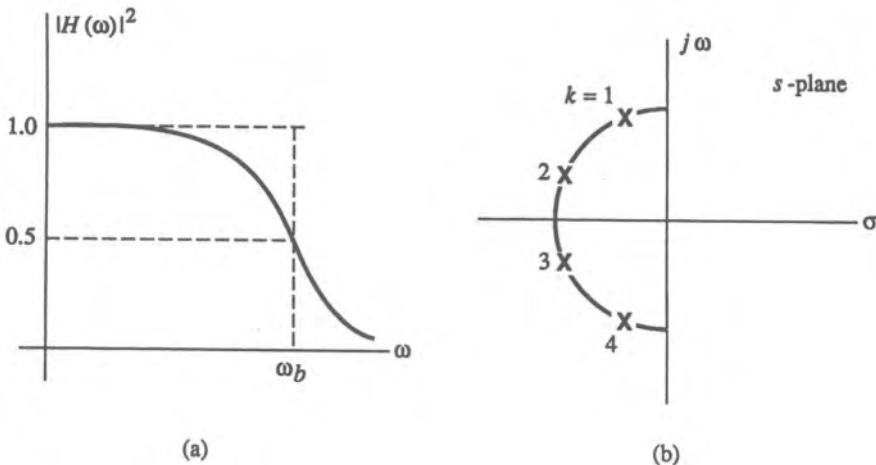


Figure 4.3. Properties of Butterworth filter. (a) Typical frequency response. (b) Pole constellation. Radius of the circle in the s plane is ω_b .

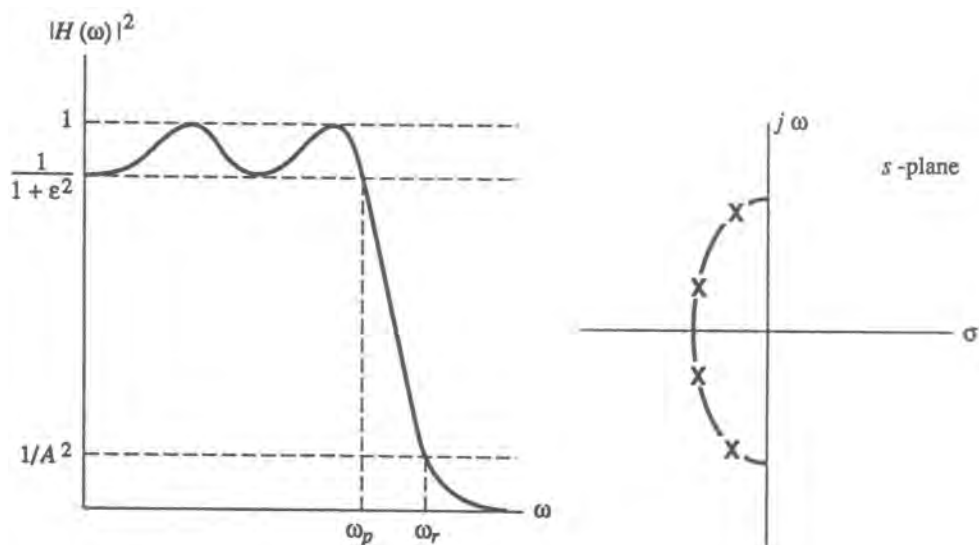


Figure 4.4. Properties of Chebyshev filter. (a) Typical frequency response of Chebyshev filter, (b) Chebyshev filter pole constellation.

The filter, normalized to the bandwidth ω_b , is completely specified by the order n . A short table of biquadratic sections for the normalized Butterworth filter for various values of n is given in Table 4.A.1 in the Appendix. Extensive tables can be found in the literature (e.g., Ref. 7).

Chebyshev Filters. The Chebyshev filter is characterized by the property that the magnitude error is equiripple in the passband (see Figure 4.4a).

The frequency response of the Chebyshev filter is

$$|H(\omega)|^2 = \frac{1}{1 + \epsilon^2 C_n^2(\omega/\omega_p)} \quad (4.1.6)$$

where $C_n(\omega)$ is the n th-order *Chebyshev polynomial*.

The Chebyshev filter is an optimal all-pole filter in the sense that it has the best performance in both the passband and stopband. The poles of the Chebyshev filter are located on an ellipse around the origin (Figure 4.4b).

The coefficients of the biquadratic sections for Chebyshev filters, normalized to the passband edge frequency ω_p , are tabulated for the filter order n and the passband ripple parameter $\alpha_p = 10 \log(1 + \epsilon^2)$ [dB]. The necessary filter order n to satisfy the passband and stopband specification can be obtained from⁽⁸⁾

$$n = \frac{\cosh^{-1} \{[(10^{\alpha_r/10} - 1)/(10^{\alpha_p/10} - 1)]^{1/2}\}}{\cosh^{-1}(\omega_r/\omega_p)} \quad (4.1.7)$$

where the stopband attenuation is given as $\alpha_r = 20 \log A$ [dB]. A short table of coefficients for the biquadratic sections for normalized Chebyshev filters is given in Table 4.A.3 in the Appendix. Extensive tables can be found in the literature (e.g., Ref. 7).

The *inverse Chebyshev* filter is characterized by the property that the magnitude error is equiripple in the stopband. This filter is used in cases where a flat response in the passband is desired. The frequency response of the inverse Chebyshev filter is

$$|H(\omega)|^2 = \frac{1}{1 + [\epsilon^2 C_n^2(\omega_p/\omega)]^{-1}}$$

The filter order n and the passband ripple parameter for the normalized inverse Chebyshev filter are determined from specifications similar to the ones for the standard Chebyshev filter; see, e.g., Ref. 8. The coefficients of the biquadratic sections are given in Table 4.A.3 in the Appendix.

Elliptic Filters. The elliptic filters are characterized by an equiripple both in passband and stopband (Figure 4.5a). It can be shown that elliptic filters are optimal in the sense that for a given ripple and order they provide the best selectivity (smallest transition frequency ω_r/ω_p).

The frequency response of a lowpass elliptic filter is given by

$$|H(\omega/\omega_p)|^2 = \frac{1}{1 + \epsilon^2 R_{n,\omega_p}^2(\omega/\omega_p)} \quad (4.1.8)$$

where $R_{n,\omega_p}(\omega/\omega_p)$ is a Jacobian elliptic function. To obtain equiripple performance in both the passband and stopband, elliptic filters must have both poles and zeros. The poles are located on an ellipse around the origin within the passband, and the zeros are distributed along the $j\omega$ axis in the stopband; see Figure 4.5b.

The coefficients of the biquadratic sections for elliptic filters, normalized to the passband edge frequency ω_p , are tabulated for the filter order n , the ripple parameter $\alpha_p = 10 \log(1 + \epsilon^2)$ [dB], and the ratio of the stopband edge to the passband edge frequencies

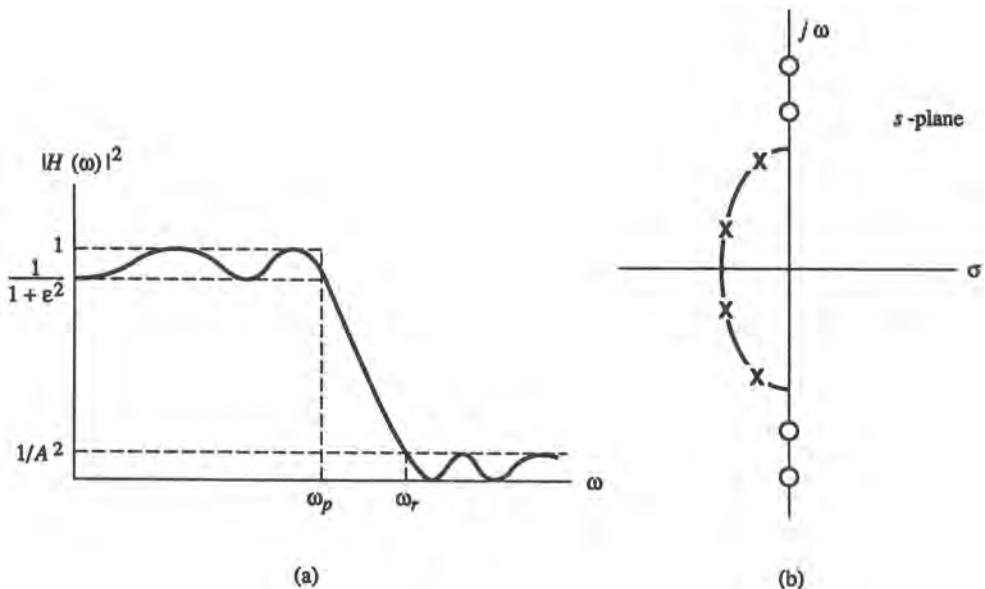


Figure 4.5. Properties of elliptic filter. (a) Typical frequency response of elliptic filter. (b) pole and zero constellation of elliptic filter.

ω_r/ω_p . The edge frequencies are defined as the frequencies where the prescribed passband or stopband loss is attained. A short table of elliptic polynomials in biquadratic form is given in Table 4.A.4 in the Appendix. Extensive tables can be found in the literature (e.g., Ref. 7).

Bessel Filters. Bessel lowpass filters are characterized by the property that the group delay is maximally flat. The step response of Bessel filters exhibits very low overshoot and the impulse response has a Gaussian shape for high-order filters. The impulse responses of the Bessel, Butterworth, and Chebyshev filters are shown in Figure 4.6.

Bessel filters have a transfer function of the form

$$H(s) = \frac{d_0}{B_n(s)} \quad (4.1.9)$$

where $d_0 = (2n)!/(2^n n!)$ is a normalizing constant and $B_n(s)$ is an n th-order Bessel polynomial of the form

$$B_n(s) = \sum_{k=0}^n d_k s^k \quad (4.1.10)$$

where

$$d_k = \frac{(2n-k)!}{2^{n-k} k! (n-k)!} \quad (4.1.11)$$

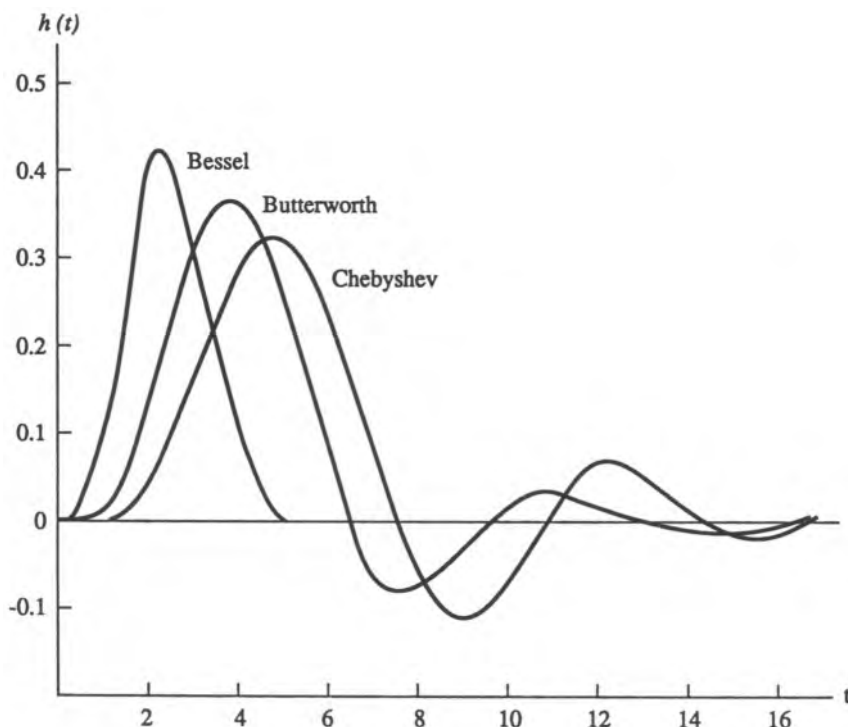


Figure 4.6. Impulse responses for fifth-order filters.

The coefficients d_k are normalized to the zero-frequency delay $\tau(\omega) = -d\phi(\omega)/d\omega|_{\omega=0} = 1$. The 3-dB bandwidth ω_b varies with the order of the filter and can be approximated by $\omega_b \approx \sqrt{(2n-1) \ln 2}$ for $n \geq 3$ ⁽⁹⁾. To normalize the 3-dB bandwidth of the filter to 1, the roots of the filter are divided by $\sqrt{(2n-1) \ln 2}$.

Normalized Bessel lowpass filters are specified by the order n of the filter. The coefficients of the biquadratic sections for Bessel filters, normalized to the zero-frequency delay $\tau(0)$, are tabulated for the filter order n . A short table of Bessel polynomials in biquadratic form is given in Table 4.A.2 in the Appendix. Extensive tables can be found in the literature; e.g., Ref. 7.

4.1.1.3. Frequency Transformations

Filter designs are usually given for normalized lowpass configurations. Other types of filters can be derived from the lowpass design and simple frequency transformations. These transformations are based on the following substitutions (see Figure 4.1):

$$\omega_g = \sqrt{\omega_H \omega_L} \quad \text{is the geometric center frequency}$$

and

$$\omega_b = \omega_H - \omega_L \quad \text{is the bandwidth of the filter} \quad (4.1.12)$$

where ω_H is the upper critical frequency and ω_L is the lower critical frequency. It should be noted that for lowpass and highpass filters ω_b represents the bandwidth between zero and the cutoff frequency ω_c , i.e., $\omega_b = \omega_c$, while for bandpass and band-reject filters ω_b represents the bandwidth between $\pm\omega_c$, i.e., $\omega_b = 2\omega_c$.

The frequency transformations normally used are as follows:

1. *Lowpass to lowpass.* A normalized lowpass characteristic is transformed into lowpass by

$$s \rightarrow \frac{s}{\omega_b} \quad (4.1.13)$$

■ *Example 4.1.1.* A third-order normalized Butterworth filter with $H_N(s) = 1/(s^3 + 2s^2 + 2s + 1)$ is transformed into

$$H_L(s) = \frac{\omega_b^3}{s^3 + 2s^2\omega_b + 2s\omega_b^2 + \omega_b^3} \quad \blacksquare$$

2. *Lowpass to highpass.* A lowpass characteristic is transformed into highpass by

$$s \rightarrow \frac{\omega_b}{s} \quad (4.1.14)$$

■ *Example 4.1.2.* The same Butterworth filter from Example 4.1.1 is transformed into

$$H_H(s) = \frac{s^3}{\omega_b^3 + 2\omega_b^2s + 2\omega_b s^2 + s^3} \quad \blacksquare$$

3. *Lowpass to bandpass.* The lowpass-to-bandpass transformation is given by

$$s \rightarrow \frac{s^2 + \omega_g^2}{\omega_b s} \quad (4.1.15)$$

■ *Example 4.1.3.* A first-order normalized Butterworth filter

$$H_N(s) = \frac{1}{1+s}$$

is transformed into an

$$H_B(s) = \frac{s\omega_b}{s^2 + s\omega_b + \omega_g^2}$$

bandpass filter. ■

4. *Lowpass to band-reject.* The lowpass-to-band-reject transformation is given by

$$s \rightarrow \frac{s\omega_b}{s^2 + \omega_g^2} \quad (4.1.16)$$

■ *Example 4.1.4.* Using the filter from Example 4.1.3, we get

$$H_R(s) = \frac{s^2 + \omega_g^2}{s^2 + s\omega_b + \omega_g^2} \quad \blacksquare$$

4.1.1.4. Lowpass Equivalents of Bandpass Filters Represented by Rational Functions

As we previously noted, in simulating the filtering of bandpass signals it is usually desirable for computational simplicity to deal with the *complex envelopes* of signals and *lowpass equivalents* of bandpass filters.

In Section 3.4 we used the Hilbert transform to derive lowpass equivalents for bandpass filters. The technique consisted in truncating the negative-frequency component of the transfer function and shifting the positive-frequency component to the zero-frequency axis. Given $H(s)$ as a frequency response $\mathbf{H}(\omega) = \mathbf{A}(\omega)e^{\Phi(\omega)}$, one can always implement this procedure. However, for a “narrowband” bandpass filter described by a rational transfer function it is possible to determine the lowpass equivalent in an analytical form directly from the given transfer function. This alternative procedure is described below.

The procedure is based on the reasoning that for narrowband bandpass filters ($B \ll f_0$) the positive-frequency response is dominated by the poles and zeros located in the positive-frequency half of the s -plane. The rational transfer function $H(s)$ is expanded into partial fractions as in (3.6.19). Since $H(s)$ has real coefficients the poles and zeros are either real or complex conjugate pairs. Therefore, the frequency response can be expressed as

$$H(j\omega) = H_p(j\omega) + H_n(j\omega) = \sum_{i=1}^N \frac{K_n}{j\omega - s_i} - \sum_{i=1}^N \frac{K_n^*}{j\omega - s_i^*} \quad (4.1.17)$$

where $H_p(s)$ is the component of $H(s)$ that contains the poles and zeros on the positive-frequency half of the s -plane, with poles at $s_i = \alpha_i + j\omega_i$, and $H_n(s)$ contains the poles and zeros on the negative-frequency half of the s -plane, with poles at $s_i^* = \alpha_i - j\omega_i$.

For narrowband bandpass filters with $B \ll f_0$ the poles and zeros are concentrated around the center frequencies $\pm\omega_0$ as in Figure 4.7a. To obtain the lowpass equivalent, we substitute $\omega \rightarrow \omega + \omega_0$ (see Figure 4.7b). The contribution of the negative-frequency poles and zeros, located now around $-2\omega_0$, to the lowpass frequency response is negligible; however, they may introduce a high-frequency ripple at $\omega = 2\omega_0$ if the input signal has such components. Additionally, the negative-frequency poles and zeros could introduce consid-

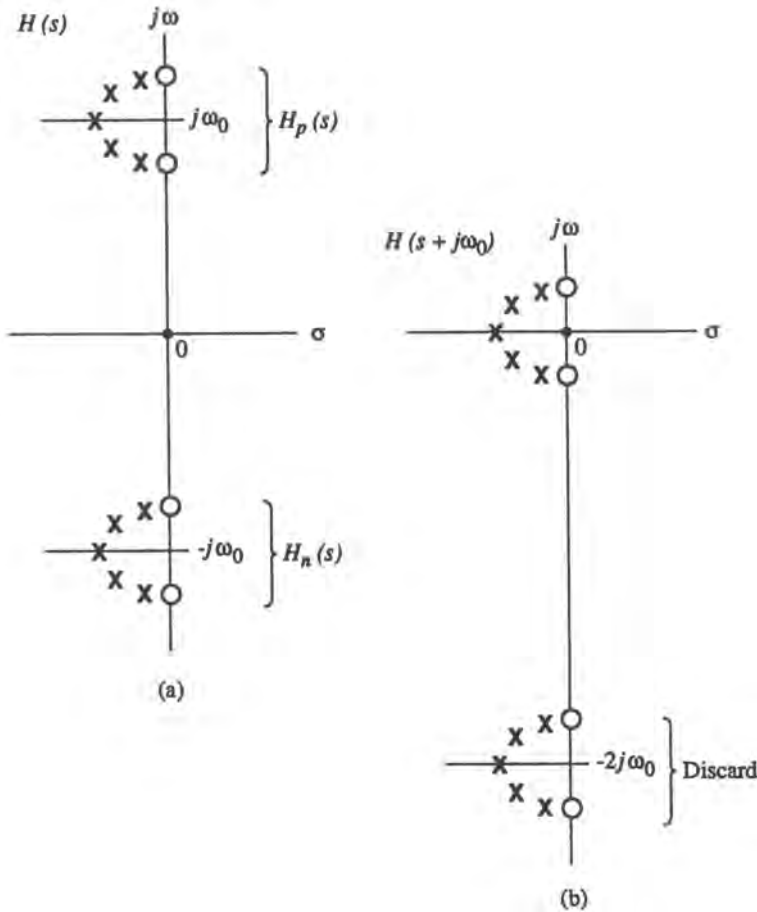


Figure 4.7. Poles and zeros of narrowband bandpass filter. (a) Bandpass filter; (b) lowpass-equivalent filter.

erable distortion into the lowpass band, through aliasing caused by sampling (see Section 3.7). It is therefore advisable to eliminate the contribution of $H_n(j\omega)$.

The lowpass-equivalent frequency response from (3.4.30) can be approximated by

$$H_L(j\omega) = H[j(\omega + \omega_0)]u(\omega + \omega_0) \approx H_p[j(\omega + \omega_0)] \quad (4.1.18)$$

Thus, the creation of a lowpass-equivalent filter $H_L(s)$ consists in extracting $H_p(s)$ from $H(s)$ by identifying and separating the conjugated poles and then shifting $H_p(s)$ to the zero axis. The lowpass-equivalent filter is then

$$H_L(s) = H_p(s + j\omega_0) \quad (4.1.19)$$

The procedure of mapping the bandpass transfer function into its lowpass equivalent can be summarized in the following steps:

- Decompose the transfer function $H(s)$ using the partial fraction expansion.
- Discard the poles that are located on the negative-frequency half-plane.
- Shift the poles located on the positive-frequency half-plane to the zero axis by substituting $s \rightarrow s + j\omega_0$.

■ *Example 4.1.5.* Given a bandpass filter with a transfer function $H(s)$ and a center frequency $\omega_0 = 2\pi f_0 = 50$, find the lowpass frequency equivalent $H_L(s)$:

$$H(s) = \frac{2(s^2 + 5s - 2494)}{(s^2 + 4s + 2504)(s^2 + 6s + 2509)}$$

Expanding the above into partial fractions, we get

$$H(s) = \frac{1}{s + 2 + j50} + \frac{1}{s + 2 - j50} - \frac{1}{s + 3 + j50} - \frac{1}{s + 3 - j50}$$

The positive-frequency component is

$$H_p(s) = \frac{1}{s + 2 - j50} - \frac{1}{s + 3 - j50}$$

The lowpass-equivalent transfer function is

$$H_L(s) = \frac{1}{s + 2} - \frac{1}{s + 3} = \frac{1}{s^2 + 5s + 6}$$

■

Lowpass-Equivalent Classical Filters. In simulating bandpass or band-reject classical filters with $\omega_g \gg \omega_b$, which is often the case, it is also desirable for computational efficiency to use lowpass-equivalent models of these filters. Since classical filters are a subset of filters having a rational transfer function, the approach outlined just above applies here also. However, in the case of classical filters, a simpler approach is possible because, as was seen earlier, any classical filter is derivable from its lowpass prototype.

In the following discussion, for simplicity we analyze bandpass filters only, but the discussion applies to band-reject filters also. Defining $\omega_0 = (\omega_H + \omega_L)/2$ as the center frequency of the passband, we get

$$\omega_L = \omega_0 - \frac{\omega_b}{2} \quad \text{and} \quad \omega_H = \omega_0 + \frac{\omega_b}{2} \quad (4.1.20)$$

The geometric center frequency from (3.6.44) is then

$$\omega_g = \sqrt{\omega_L \omega_H} = \sqrt{\omega_0^2 - (\omega_b/2)^2} \quad (4.1.21)$$

Thus, $\omega_0 = \omega_g$ for $\omega_g \gg \omega_b$.

It can be shown that in this case the filter becomes symmetric about the center frequency ω_0 . The frequency response of this filter is thus a shifted replica of the two-sided lowpass filter response. Indeed, performing the lowpass-to-bandpass transformation as in (4.1.15) and then deriving the lowpass equivalent, as in the section above, will result in the original lowpass filter with the bandwidth $\omega_b/2$. Thus, the lowpass-equivalent filter of a bandpass filter with $\omega_g \gg \omega_b$ is the prototype lowpass filter with the bandwidth $\omega_b/2$.

Under the condition that $\omega_g \gg \omega_b$, lowpass-equivalent filters that are centered about $\omega = 0$ are real, otherwise they are complex. For example, in simulation of adjacent channel interference in carrier-modulated systems, we have to find the lowpass equivalent of three channels with carrier frequencies ω_c , $(\omega_c - \omega_1)$, and $(\omega_c + \omega_1)$. The logical choice would be to shift the transfer functions by ω_c , resulting in lowpass-equivalent filters around 0, $-\omega_1$, and ω_1 . All three filters are identical to the lowpass prototype filters. The bandpass filters are real, so is the filter centered around zero, but the lowpass-equivalent filters centered around $-\omega_1$ and ω_1 are complex.

In the case that ω_c is not much larger than ω_b , as in voice-band modems, for example, the use of lowpass equivalent filters is not needed and not recommended. Thus, the procedure

of truncating the negative-frequency component of the bandpass transfer function to derive the lowpass-equivalent transfer function as described in the above section is not needed here.

4.1.1.5. Filter Specifications

In practical systems, filtering characteristics are typically required to satisfy some set of specifications, rather than given as a prescribed function of frequency. Specifications inherently imply some degree of flexibility or “fuzziness,” so as to allow the designer some freedom of design and to recognize the impossibility of building *exactly* some given transfer function. Specifications are typically given in the frequency domain. There are two prevalent methods for imposing specifications.

Frequency-Domain Mask. One widely used technique is to require the amplitude and phase (or delay) characteristics to lie within a *mask* (a separate one for amplitude and delay). A mask is merely a tolerance band or region within which the response in question can lie. For example, as was shown in Figure 4.2, a passband can be defined, and for that passband, the amplitude response (illustrated in Figure 4.2) can lie anywhere within the defined tolerance. A mask can be defined both for the amplitude and phase, but it may not be simple simultaneously to meet both specifications. The classical filters defined above (specifically the Chebyshev and elliptic types) are designed to satisfy an amplitude mask, but once a particular filter has been selected, the phase is automatically determined. Given an amplitude mask, one could select a member of the classical filters to implement. Such a filter would naturally have an infinite impulse response, and one could choose to implement it as such, or truncate it into an approximating FIR filter. Alternatively, one could choose a more or less arbitrary function satisfying the specification. In that case, no explicit structure need be invoked. One could simply construct a *tabular* filter (see Chapter 8), which is defined from a finite set (table) of amplitude and phase points satisfying the specification. In this case the filter is more naturally implemented (in simulation) as an FIR filter, but one could implement it as an IIR filter obtained from a rational function approximation to the chosen points. We should remark that the approximations we have been speaking of are different in nature from those we shall discuss in the next section, which deal with approximating continuous structures by discrete ones. We should also point out that filters (or systems in general) that are defined by specifications will usually require multiple simulations to sufficiently sample the specification space.

Filters Specified by Simple Functions in the Frequency Domain. A second method of specification is to impose limitations on the terms of certain decompositions of the amplitude and phase characteristics. One common method is to represent the amplitude or phase as low-order polynomials plus a residual (or “ripple”) term. The coefficients of the polynomials are specified not to exceed certain values, and the peak-to-peak ripple is similarly controlled. The polynomials, of course, are well-defined functions, but the residual may have many realizations satisfying the specification. Further discussion of this approach is given in Chapter 8.

4.1.1.6. Approximating Continuous Structures in Discrete Time for Simulation

Now we discuss possibilities for approximating continuous structures in discrete time, in effect, how to create simulation models. Discrete-time approximations can be derived corresponding to the three modes of description in Section 4.1.1.1. In this subsection we give

only an overview of these ideas. In Section 4.1.2.3, we will return to this topic with an eye toward implementation details.

1. *Approximation in the Time Domain.* The basis for approximation in the time domain is to discretize the convolution integral (4.1.1) into the sum

$$y(kT_s) \cong \sum_{n=-\infty}^{\infty} T_s h(nT_s) x[(k-n)T_s] \quad (4.1.22)$$

which is formed by sampling $h(t)$ as well as $x(t)$ at discretely spaced intervals T_s . Equation (4.1.22) is referred to as a convolution sum, and may be formally justified as a discrete approximation to the integral. The goodness of the approximation and the ability to reconstruct $y(t)$ from discretely spaced samples depends on the ratio of $1/T_s$ to the bandwidth of $x(t)$ and $h(t)$. In (4.1.22), T_s is often normalized to unity (or made implicit), so that the following alternative notations are also often seen:

$$y(k) = \sum_{n=-\infty}^{\infty} h(n)x(k-n) \quad (4.1.23a)$$

or

$$y_k = \sum_{n=-\infty}^{\infty} h_n x_{k-n} \quad (4.1.23b)$$

We note that while there is no restriction on the analytical form of $h(t)$, the convolution sum will generally not be appropriate for computing the output of highpass systems because the sampled form requires the involved functions to be effectively bandlimited for a good approximation. As with any sampled function, the sampled version of $h(t)$ is an aliased version of the original. With $H(f)$ the Fourier transform of $h(t)$, and $H_p(f)$ the aliased version, we have

$$H_p(f) = \sum_{k=-\infty}^{\infty} H(f + kf_s) \quad (4.1.24)$$

where $f_s = T_s^{-1}$ as in Figure 4.8. Thus, the impulse response must be sampled sufficiently rapidly to control aliasing error (see Section 3.5).

In general, neither $x(t)$ nor $h(t)$ is bandlimited: what is the consequent effect on the spectral properties of the output sequence $y(k)$?

$$y(k) \leftrightarrow Y_p(f) = H_p(f)X_p(f) \quad (4.1.25)$$

Therefore, if $h(t)$ is bandlimited with bandwidth B_1 and $x(t)$ is bandlimited with bandwidth B_2 , then $y(k)$ is computed without error if $f_s \geq 2B$, where $B = \max(B_1, B_2)$. The latter is a sufficient condition, but it is not necessary. A necessary condition is $f_s \geq \frac{1}{2}(B_1 + B_2)$; see Figure 4.9. If $B_2 \gg B_1$, it may be advantageous to downsample (decimate) after filtering.

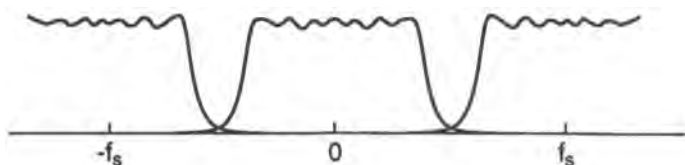


Figure 4.8. Illustration of a possible aliasing of the transfer function of a filter.

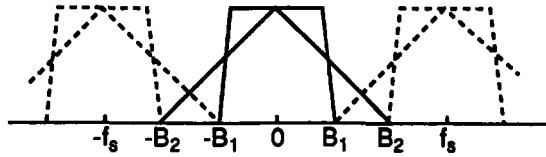


Figure 4.9. Illustration of non aliasing conditions when a filter and a signal are both bandlimited with different bandwidths.

Evaluating the convolution sum literally by numerical means is impossible if the sequence $h(k)$ is infinite [assuming, of course, that $x(t)$ is defined for all time]. Practical means for evaluating (4.1.22) or (4.1.23) requires one of the following conditions:

- The sequence $h(k)$ is finite, i.e., $h(t)$ is an FIR filter.
- The infinite sequence $h(k)$ is truncated to create an approximate FIR filter; note that this truncation represents an additional source of error (besides aliasing).
- The convolution sum is expressible in recursive form, i.e., $h(t)$ can be implemented as an IIR filter.[†]

For simulation purposes, the first two conditions imply the same computational implementation, while the third possibility represents a different approach. Both cases will be developed below. Note that, as developed so far, only t is discretized, not f . A practical approach is to discretize both so that advantage can be taken of the FFT. This will be taken up in Section 4.1.2.

2. Approximation in the Frequency Domain. As might be inferred, approximation in the frequency domain is the discretization of the frequency-domain counterpart of the convolution integral, as expressed in (4.1.3). Thus,

$$y(t) = \sum_{m=-\infty}^{\infty} H(m\Delta f)X(m\Delta f)e^{2\pi\Delta f t} \quad (4.1.26)$$

From the time–frequency duality, sampling in the frequency domain implies an aliased function in the time domain. That is, the corresponding impulse response would be periodic and, within each period, an approximation to the actual impulse response. Assuming that $X(f)$ could be meaningfully defined, (4.1.26) also could not be reasonably computed unless one or the other of the transforms in the summand could be truncated. This is of course generally reasonable if H or X is (almost) bandlimited.

As written, however, (4.1.26) cannot be used as the basis for simulation since t is continuous, although f is discretized. Thus, to produce a useful sequence $y(k)$ also requires sampling t at T_s intervals in this formulation. In that case, both H and h are finite sequences, and we can go back and forth from the “time” to the “frequency” domain using FFT techniques, as will be seen.[‡] Thus, as developed to this point, approximation in the frequency domain implies an FIR simulation technique.

An alternative leading to an IIR implementation is to synthesize an IIR structure from the given points $H(m\Delta f)$, or more generally from *some* set of samples of $H(f)$, not necessarily equally spaced. Generally, this involves obtaining an approximating rational function in

[†]Of course, $h(k)$ is already assumed to have an infinite impulse response. The term IIR typically implies a recursive form.

[‡]The quotation marks on time and frequency are placed here to remind us that these sequences have the same mathematical properties; time and frequency are attributes no longer inherently distinguishable.

s or in z whose corresponding frequency function passes through the given points. Many mathematical and software applications packages will automatically produce these rational transfer functions, given the points.

3. *Approximation in the Transform Domain.* The basis for what we shall call approximation in the transform domain begins with the standard relation

$$Y(s) = H(s)X(s) \quad (4.1.27)$$

where $Y(s)$, $H(s)$ and $X(s)$ are the Laplace transforms of the output, impulse response of the system, and the input, respectively.

We assume that $H(s)$ is a ratio of polynomials in s , which implies the system is characterized by a linear time-invariant differential equation:

$$H(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \cdots + a_1 s + a_0}{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0} \quad (4.1.28)$$

with $m \leq n$. Dividing the numerator and denominator by s^n leads to

$$\begin{aligned} H(s) &= \frac{a_m (1/s)^{n-m} + a_{m-1} (1/s)^{n-m-1} + \cdots + a_0 (1/s)^n}{b_n + b_{n-1} (1/s) + b_{n-2} (1/s)^2 + \cdots + b_0 (1/s)^n} \\ &= A(1/s)/B(1/s) \end{aligned} \quad (4.1.29)$$

Therefore, the input-output relationship in (4.1.27) can be rephrased as

$$Y(s)B(1/s) = X(s)A(1/s) \quad (4.1.30)$$

Since $(1/s)^k$ represents k -fold integration, the relationship above, translated back into the time domain, leads to a schematic interpretation as shown for a biquadratic transfer function in Section 3.6.6.1 (with $b_n = 1$). This representation suggests that one approach to a discrete-time approximation of the system is to approximate the integration operation $(1/s)$ by some discrete form, or “integration formula.”

Approximation in the transform domain means formally to employ the substitution

$$\frac{1}{s} \rightarrow f(z^{-1}) \quad (4.1.31)$$

where z^{-1} is the delay operator. In principle, the *integration formula* $f(\cdot)$ could be arbitrary, subject only to certain desirable properties. Let us consider the case when $f(z^{-1})$ is itself a ratio of polynomials in z^{-1} , viz.,

$$f(z^{-1}) = \frac{\alpha_r z^{-r} + \alpha_{r-1} z^{-(r-1)} + \cdots + \alpha_0}{\beta_l z^{-l} + \beta_{l-1} z^{-(l-1)} + \cdots + \beta_0} = \frac{N(z^{-1})}{D(z^{-1})} \quad (4.1.32)$$

which corresponds to the *linear multistep methods* of integration (see Chapter 5). With $X(z)$ and $Y(z)$ the z -transforms of $x(n)$ and $y(n)$, respectively,

$$\begin{aligned} X(z) &= \sum x(n)z^{-n} \\ Y(z) &= \sum y(n)z^{-n} \end{aligned} \quad (4.1.33)$$

we obtain

$$B[f(z^{-1})]Y(z) = A[f(z^{-1})]X(z)$$

The preceding equation leads to the following explicit form:

$$\left\{ \sum_{i=0}^p b'_i (z^{-1})^i \right\} Y(z) = \left\{ \sum_{i=0}^q a'_i (z^{-1})^i \right\} X(z) \quad (4.1.34)$$

The coefficients a'_i, b'_i are combinations of the old coefficients $a_i, b_i, \alpha_i, \beta_i$, and $p = l \times n, q \leq p$. Thus, under our assumptions, the “equivalent” discrete system function

$$H_d(z) = \frac{\sum a'_i (z^{-1})^i}{\sum b'_i (z^{-1})^i} \quad (4.1.35)$$

is *always* a ratio of polynomials in z^{-1} (or z).

The above discrete approximation also implies the following difference equation:

$$y(n) = \sum_{i=1}^p b'_i y(n-i) + \sum_{i=0}^q a'_i x(n-i)$$

If $b'_i = 0$, for $i > 0$, then we have an FIR filter. If $b'_i \neq 0$ for some $i > 0$, we have a *recursive* equation, which implies an *infinite impulse response* (IIR) filter. In Section 4.1.3, in conjunction with IIR filters, we shall examine in more detail two particular cases of the mapping (4.1.31). One is a specific instance of a ratio of polynomials in z^{-1} or z called the *bilinear transformation*; the other is the so-called *impulse-invariant transformation*, which is analytically of a different form.

4.1.2. Simulation of Filtering with Finite Impulse Response Filters

In this subsection we begin our discussion of techniques for filtering signals with filters having finite impulse responses. There are two principal approaches, commonly referred to as “time-domain” and “frequency-domain” filtering. We will take these up in turn, following which we will summarize the procedures for mapping continuous filters into discrete ones. Finally, we will compare the two approaches in Section 4.1.2.4.

4.1.2.1. Simulation of FIR Filtering in the Time Domain

4.1.2.1.1. Introduction. By definition, an FIR filter is one for which $h(n) = 0$ for $n < 0$ and $n \geq M$, where M is chosen at the discretion of the user. Then, the convolution sum (4.1.23) simplifies to

$$y(k) = \sum_{n=0}^{M-1} h(n) x(k-n) \quad (4.1.36)$$

which is straightforward to implement as shown, in simulation applications. This representation is equivalent to a tapped delay line, as in Figure 4.10. If the filter is a lowpass equivalent of a bandpass filter, the tap coefficients will typically be complex, as will also be the input signal sequence.

The computational load for implementing the above equation is M multiplications and $M - 1$ additions for each output point. Therefore, time-domain processing is usually used for low or moderate response duration, say $M \leq 128$. If M is large, the frequency-domain approach is more efficient, as will be seen.

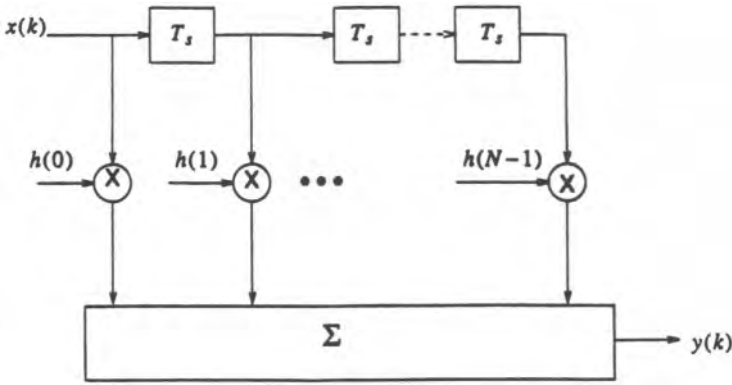


Figure 4.10. Representation of an FIR filter as a tapped delay-line.

The frequency response of a tapped delay line is periodic whether the input is continuous or discrete. For discrete time and continuous frequency, we have

$$H(f) = \sum_{n=0}^{M-1} h(n)e^{-j2\pi nT_s f} \quad (4.1.37)$$

and for discrete time and discrete frequency

$$H_d(k\Delta f) = \sum_{n=0}^{M-1} h(n)e^{-j2\pi nk/N} \quad (4.1.38)$$

where $MT_s = (\Delta f)^{-1}$.

Continuous structures generally have infinite impulse responses. The usual procedure for obtaining a finite discrete impulse response is to sample and truncate the original response. The latter might have been obtained via the so-called impulse invariant method presented later. Direct truncation of the impulse response leads to distortion which manifests itself in the frequency domain as a ripple, known as the Gibbs phenomenon (see Section 3.3.5 and Example 4.1.6 below). A more successful way to obtain an FIR filter is to use a finite weighting sequence $w(n)$, called a window, to modify the response $h(n)$ in such a way as to minimize or at least reduce distortion.⁽²⁾ Below, we give a very brief introduction to windowing in this context.

4.1.2.1.2. Windowing. The windowed impulse response is formed by creating the sequence

$$h_w(n) = h(n) w(n) \quad (4.1.39)$$

where $w(n) = 0$ for $n < 0$ and $n \geq N$. Hence, $h_w(n)$ is M long, and the finite convolution sums remain as above with h replaced by h_w . The windowed frequency response is the periodic convolution (see Table 3.A.3)

$$H_w(f) = H(f) \circledast W(f) \quad (4.1.40)$$

where H_w , H , and W are the discrete Fourier transforms of h_w , h , and w , respectively. Traditionally, the window function is designed to reduce the Gibbs phenomenon distortion. For such an end, it is easy to show that desirable window characteristics are as follows:

1. Narrow main lobe of the frequency response, containing the maximum energy possible.
2. Small and rapidly decaying spectral sidelobes.

These are conflicting requirements, as is well known from Fourier theory.⁽¹⁰⁾

The Gibbs phenomenon can be moderated through the use of a less abrupt truncation of the impulse response. By tapering the window smoothly at each end of the N -point interval, the height of the sidelobes in $W(f)$ can be diminished; however, this is achieved at the expense of a wider main lobe and thus a smoother change at sharp transitions in the frequency response $H_w(f)$. In the following two examples of windows we assume the windows to be symmetric sequences with N samples centered about the $n = 0$ axis. To apply the window to causal impulse responses, they have to be shifted by $n \rightarrow n + (N - 1)/2$.

Reducing the Gibbs phenomenon addresses only one aspect of distortion. An alternative objective for a window is to reduce the waveform distortion due to the truncation. Here, an appropriate criterion might be to lessen the mean-square error between the true waveform and that produced by the truncated filter. Optimal windows for this purpose may not be the same as those designed to deal with the Gibbs' phenomenon. Further discussion of this point can be found in Ref. 11.

Rectangular Window. The rectangular window corresponds to direct truncation of the impulse response. This window does not reduce distortion, and is given here for reference only:

$$w_r(n) = \begin{cases} 1.0, & |n| \leq (N - 1)/2 \\ 0.0, & \text{elsewhere} \end{cases} \quad (4.1.41a)$$

The frequency response of the rectangular window is

$$W_r(e^{j2\pi f}) = \sum_{n=-(N-1)/2}^{(N-1)/2} e^{-j2\pi f n} = \frac{\sin(2\pi f N/2)}{\sin(2\pi f/2)} \quad (4.1.41b)$$

Hamming Window. An often used window is the Hamming window. The Hamming window sequence is given by

$$w_h(n) = 0.54 + 0.46 \cos\left[\frac{2\pi n}{N}\right], \quad \frac{N-1}{2} \leq n \leq \frac{N-1}{2} \quad (4.1.42a)$$

The frequency response of the Hamming window is

$$W_h(e^{j2\pi f}) = 0.54W_r(e^{j2\pi f}) + 0.23W_r(e^{j(2\pi f - 2\pi/N)}) + 0.23W_r(e^{j(2\pi f + 2\pi/N)}) \quad (4.1.42b)$$

where $W_r(e^{j2\pi f})$ is the frequency response of the rectangular window. Illustrations of W_r and W_h are given in Figure 4.11.

An example of the use of the Hamming window in the simulation of an FIR filter is given later in Example 4.1.6.

In the preceding discussion, the purpose of the windows is to improve, in some sense, the behavior of the FIR filter. For similar reasons (i.e., distortion due to truncation) windows are also useful in a different but related context, namely the estimation of power spectra. This subject is taken up in Chapter 10, where additional examples of some commonly used windows and their frequency responses are given.

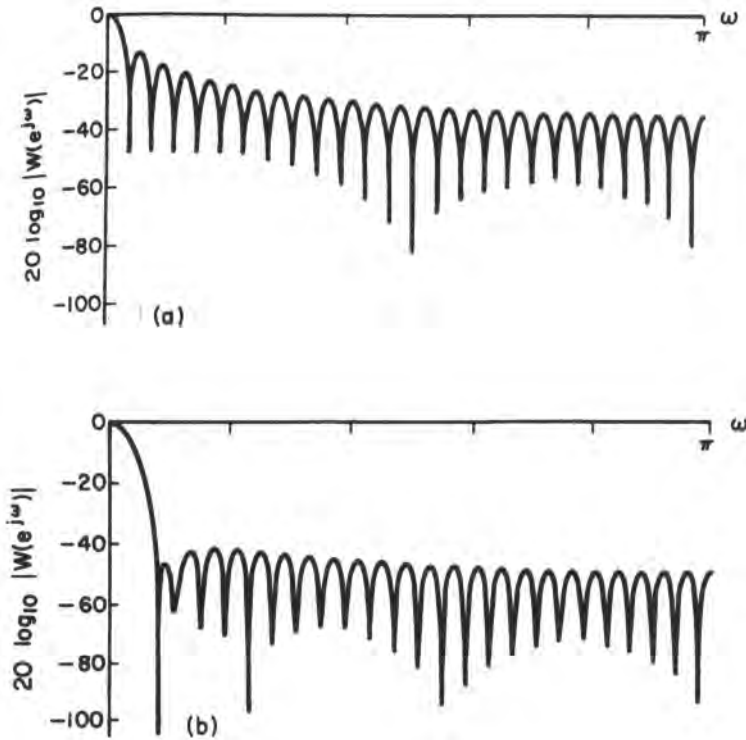


Figure 4.11. (a) Frequency response of the rectangular window. (b) Frequency response of the Hamming window.

4.1.2.2. Simulation of FIR Filtering in the Frequency Domain

Simulation of FIR filtering in the frequency domain means essentially the implementation of the filtering operation through a truncated version of (4.1.26) rather than a direct application of the convolution sum (4.1.36). Under most conditions, the frequency-domain approach will be much more computationally efficient, primarily because of the efficiency of obtaining the finite Fourier transform through the fast Fourier transform (FFT). Recalling the frequency-domain version of the convolution theorem [Equation (3.8.11)], we can compute the system response as

$$y(n) = F^{-1}\{F[x(n)] \cdot F[h(n)]\} \quad (4.1.43)$$

where the time-domain sequences $x(n)$ and $h(n)$ are transformed by the DFT into frequency-domain sequences, multiplied, and then transformed into the time domain using the IDFT. In this case both $x(n)$ and $h(n)$ must be of the same length. This can be always achieved by padding the shorter sequence with zeros. This type of signal processing is often referred to as *frequency-domain FIR filtering* or simply *frequency-domain filtering*. In many cases, the filter description is initially given in the frequency domain, rather than as a sequence in the time domain. In such cases, the operation $F[h(n)]$ evidently need not be performed.

The saving in computational effort by using the FFT algorithm can be considerable. The whole operation will require two FFTs, one multiplication of the sequences, and one IFFT. For an impulse response and signal sequences of respective lengths M and N , an FFT requires

$(L/2)\log_2 L$ multiplications, where $L = \max(M, N)$. The whole operation therefore will require

$$N_1 = L \log_2 L + L \quad (4.1.44a)$$

multiplications, while the time-domain convolution sum will require

$$N_2 = (M + N) \times N \quad (4.1.44b)$$

multiplications.

The saving in computational effort is significant when both M and N are large. For example, with $M = N = 1000$ the time-domain convolution will require 2×10^6 operations, while the FFT approach requires only 11,000. This amounts to a reduction in computational effort by a factor of about 200.

In the form presented above the computation of the system response by the FFT using the periodic convolution theorem is limited to periodic signals. Such signals are often used in simulations of communication systems with quasianalytical probability error estimators (see Chapter 11). In general, however, the signals are neither periodic nor time-limited, and simulating a filter response using the circular convolution may lead to distortions. Frequency-domain FIR filtering methods using the linear convolution are described next.

4.1.2.2.1. Difference between Periodic and Linear Convolution. The difference between the periodic and linear convolution can be best illustrated with an example. Figure 4.12 shows two sequences $x_1(n)$ and $x_2(n)$ of length N . The circular convolution will produce the sequence

$$x_3(n) = \sum_{m=0}^{N-1} x_{p1}(m)x_{p2}(n-m)$$

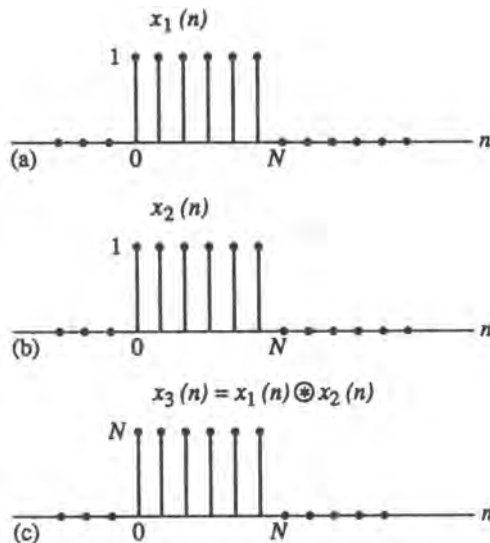


Figure 4.12. N -point periodic convolution of two rectangular sequences of duration N .

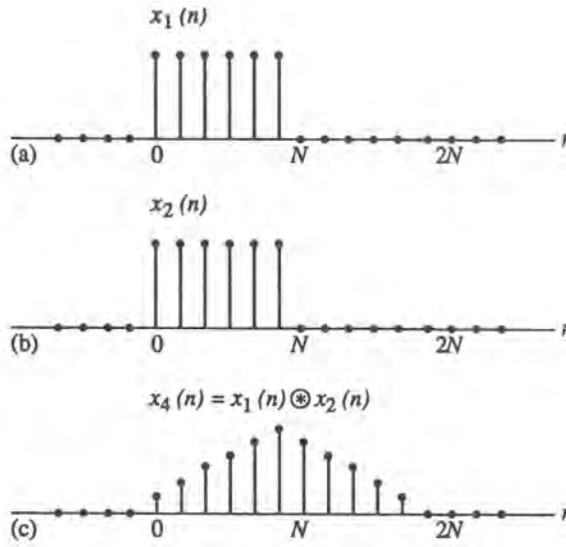


Figure 4.13. $2N$ -point periodic convolution of two rectangular sequences of duration N .

also shown in Figure 4.12. (The subscript p indicates periodic extension, as introduced in Chapter 3.) Computing the convolution by using the DFT

$$x_3(n) = F^{-1}\{F[x_1(n)] \cdot F[x_2(n)]\}$$

will clearly produce the same result.

Figure 4.13 shows the two sequences $x_1(n)$ and $x_2(n)$ with augmented zeros at the end so that each sequence has length $2N$. The convolution of these sequences, also shown in Figure 4.13, is

$$x_4(n) = \sum_{m=0}^{2N-2} x_{p1}(m)x_{p2}(m-n)$$

This result is the linear convolution of the sequences and has the length $2N$. Clearly $x_3(n)$ is the distorted result of the linear convolution. Normally, x_1 and x_2 are sequences which represent the system (impulse) response and a data input. The correct result is then the linear convolution. Thus, to compute the linear convolution through the DFT, we have to use the circular convolution of length $2N$. To perform this convolution by way of the Fourier transform, the DFTs and IDFT also have to be of length $2N$.

4.1.2.2.2. Linear Convolution for a Signal of Arbitrary Duration via the FFT. The procedure described above permits linear convolution of two sequences of equal length. However, quite often we are faced with a problem of computing the system response to a fairly long input function. The impulse response is usually short compared to the input. Such long input sequences are generally too large for the DFT to compute. To perform a convolution while still using the DFT, the long input sequence is segmented into sections of length L . These sections are then convolved with the impulse response using the DFT. The filtered segments are combined in an appropriate way, so that the resultant output represents a linear convolution of the original sequences. Two such algorithms for a segmented DFT are described below.

4.1.2.2.3. *The Overlap-and-Add (OA) Method.* To illustrate the procedure of partitioning and combining the filtered segments together, consider the input sequence $x(n)$ and the impulse response $h(n)$ as in Figure 4.14. The input sequence $x(n)$ is divided into segments of length L , with the k th section $\mathbf{x}_k(n)$ defined by

$$\mathbf{x}_k(n) = \begin{cases} x(n), & kL \leq n \leq (k+1)L - 1 \\ 0, & \text{otherwise} \end{cases} \quad (4.1.45a)$$

Then

$$x(n) = \sum_{k=-\infty}^{\infty} \mathbf{x}_k(n) \quad (4.1.45b)$$

The individual segments are shown in Figure 4.14a. Since convolution is a linear operation, the convolution of $x(n)$ with $h(n)$ is equal to the sum of $\mathbf{x}_k(n)$ convolved with $h(n)$:

$$x(n) * h(n) = \sum_{k=-\infty}^{\infty} \mathbf{x}_k(n) * h(n) \quad (4.1.46)$$

Since each segment $\mathbf{x}_k(n)$ has L nonzero points and $h(n)$ has M points, to obtain a linear convolution, the DFT must have at least $L + M - 1$ points. Thus the output segments will overlap each other by M points. The overlapping sequences are added up to produce the entire convolution. Figure 4.14b shows the individual convolved segments and the resultant convolution, which is the sum of the convolved segments.

4.1.2.2.4. *The Overlap-and-Save (OS) Method.* An alternative procedure to compute a linear convolution is to perform a circular convolution, identify and save the linear part, and discard the aliased part. By performing a circular N -point convolution for an N -point segment with an M -point impulse response, the first $M - 1$ points are “aliased,” while the remaining $N - M + 1$ points are identical to those obtained with a linear convolution. We therefore section $x(n)$ into segments of length N so that each section overlaps the preceding section by $M - 1$ points (see Figure 4.15a)

$$\mathbf{x}_k(n) = x(n + k(N - M + 1)), \quad 0 \leq n \leq N - 1 \quad (4.1.47a)$$

The circular convolution segments $\mathbf{y}_k(n)$ are shown in Figure 4.15b. The first $M - 1$ points of each output section are discarded. The remaining points are combined to produce the linear convolution

$$y(n) = \sum_{k=0}^{\infty} \mathbf{y}_k[n - k(N - M + 1)] \quad (4.1.47b)$$

4.1.2.2.5. *Efficiency of the Linear Convolution via the FFT.* To provide a linear convolution for one L -point segment of the input function $\mathbf{x}_k(n)$ using the overlap-and-add (save) method, we have to perform one $(L + M - 1)$ -point FFT and IFFT and one array multiplication; we will refer to the ratio $(L + M)/L$ as the overlap-and-add (save) ratio. The transform of the impulse response can be precomputed and stored prior to performing the convolution.

The number of multiplication operations required for K convolutions of the L -point segments with an M -point impulse function, is from (4.1.44a),

$$N_1 \approx K[(L + M) \log_2(L + M) + (L + M)] \quad (4.1.48a)$$

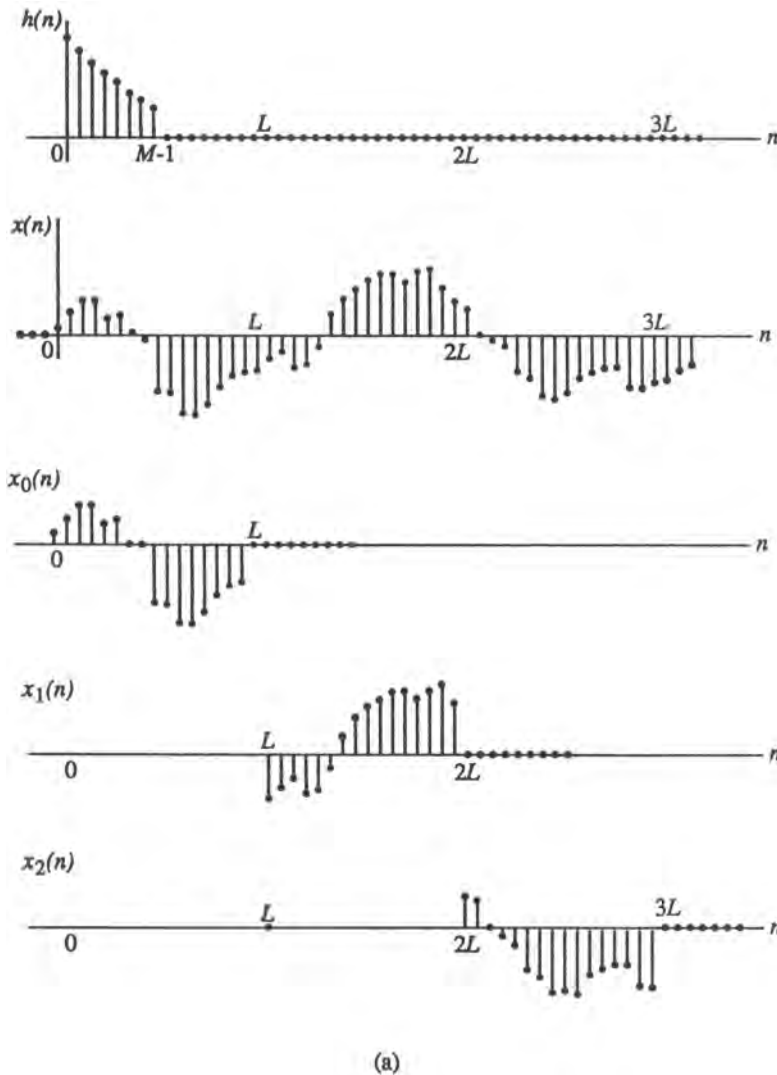


Figure 4.14. Overlap-and-add linear convolution, (a) Decomposition of $x(n)$ into nonoverlapping segments.

The number of operations for a time-domain convolution sum is, from (4.1.44b),

$$N_2 \approx (K \cdot L + M)M \quad (4.1.48b)$$

As an example, let us assume that a long input function with 10^6 samples is subdivided into $K = 1000$ segments of $L = 1000$ points. If $M = 1000$ is the number of samples in the impulse function, then

$$N_1 \approx 10^3 [2 \cdot 10^3 \log_2 2 \cdot 10^3 + 2 \cdot 10^3] \approx 2.4 \cdot 10^7$$

and

$$N_2 \approx (10^6 + 10^3)10^3 \approx 10^9$$

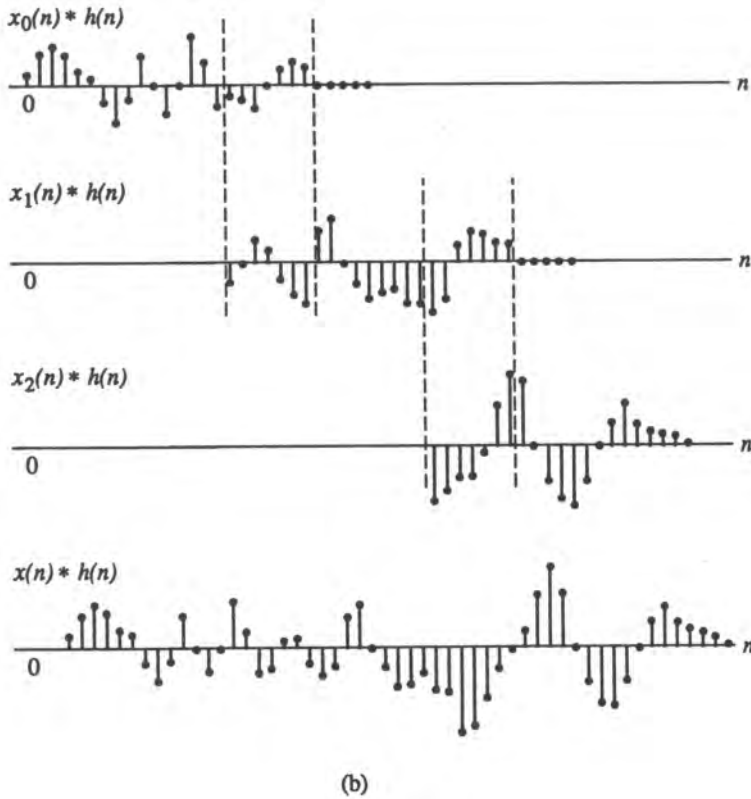


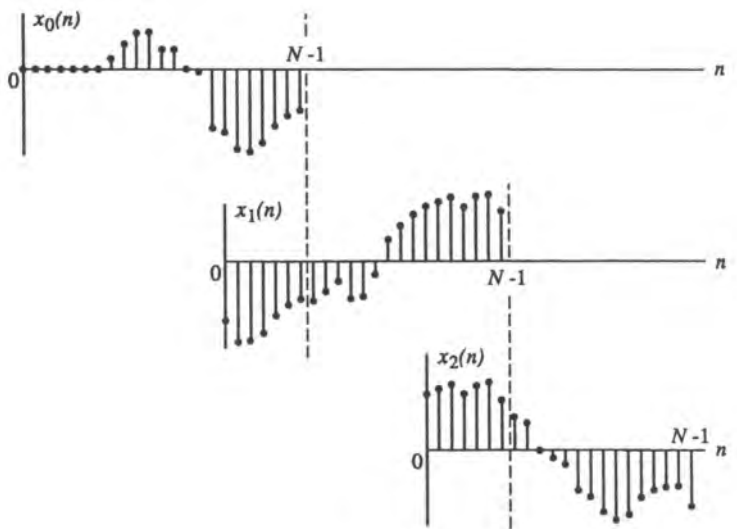
Figure 4.14. (b) Generation of partial convolutions.

The computational effort is reduced by a factor of $N_2/N_1 \approx 40$.

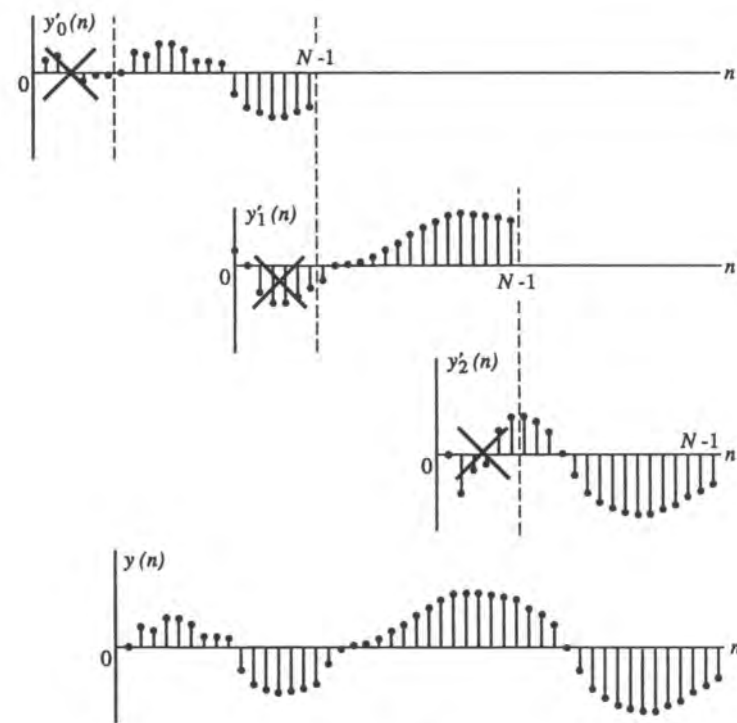
It is interesting to note that the number of operations N_1 is little dependent on the length of the segment L for $L \geq M$. As an illustration, for the example above, for $1 \leq K \leq 1000$, one can show that $1.7 \cdot 10^7 \leq N_1 \leq 2.7 \cdot 10^7$.

The linear convolution via the FFT is fairly efficient even for small impulse response durations M . It can be shown from (4.1.44) that the crossover point is achieved when $2^M - M = L$. The crossover point for $L = 1000$ is at $M = 11$.

4.1.2.2.6. Implications of Frequency-Domain FIR Filtering. One of the most important applications of frequency-domain FIR filtering in simulation is when using filters specified in the frequency domain. The frequency response $H(n)$ for these filters is usually given in tabular form for both amplitude and phase, and may come from design specifications, measured data, or approximations to desired characteristics. In addition, the discrete frequency response for FIR filters having specified characteristics can be obtained from the digital signal processing methodology that is available in many texts.⁽¹⁻³⁾ As was seen, one of the main features of using frequency-domain filtering is the fact that the input signal has to be broken up into segments (blocks). In this situation the whole simulation process is often performed one block at a time. This type of simulation is referred to as *block processing*. The main advantage of using block processing is efficient simulation of linear systems. A major disadvantage of block processing is that it creates virtual delays in the signal flow. This may complicate the calculation of such parameters as signal time delay. Because of this artificial



(a)



(b)

Figure 4.15. Overlap- and save-linear convolution, (a) Decomposition of $x(n]$ into overlapping segments; (b) generation of partial convolutions.

time delay, frequency-domain filters cannot be used in feedback loop simulations (that is, within the loop itself). Another complication with block processing arises when linear convolutions such as OA and OS are used within systems that contain nonlinear elements. This situation is described below.

Block Processing Using the OA and OS Methods. Block processing with frequency-domain FIR filtering using the OA or OS method may cause complications because the signal segments are overlapping. If linear modules are cascaded, processing of individual blocks can be performed separately, using either the OA or the OS method, with domain conversion being performed only when required. For instance, a cascade of filters can be processed without FFTs and IFFT's required in between. The use of OS and OA methods is functionally equivalent for linear systems, except that the OS method requires the input sequence to be broken up into overlapping segments, and OA allows it to be split into contiguous segments.

Since the output segments are contiguous for the OS method, block processing can be extended to memoryless nonlinear modules. An IFFT is required before the nonlinear element, but blocks can be processed separately. This is not the case for the OA method, however. The overlapping output signal segments must be recombined before the nonlinearity. On the other hand, if there are linear modules following the nonlinear element, then the OA method is more appropriate to use since the output of the nonlinear elements consists of contiguous segments.

4.1.2.3. Mapping of Continuous Filters into Discrete FIR Filters

Here we summarize the procedure, step by step, and some of the practical details that need to be taken into account in order to prepare continuous filters into discrete FIR filters ready for simulation.

4.1.2.3.1. FIR Filters Defined in the Time Domain. FIR filters may be described or specified by an impulse response in functional form, which often arises from a given function that is truncated. The procedure to follow to prepare these filters for simulation is described below and shown in Figure 4.16a.

1. Apply an appropriate technique, such as the Hilbert transform (Section 3.4) to obtain the lowpass equivalent of a bandpass filter, if necessary.
2. If the FIR filter has to be derived from an infinite impulse response, the response has to be truncated and windowed. The truncation should be done so that no more than a few percent of energy is lost in the truncated tail.
3. If needed, the impulse response is interpolated and resampled at the sampling rate $f_s = 1/T_s$.
4. The resulting impulse response is ready for filtering via time-domain convolution.

4.1.2.3.2. FIR Filters Defined in the Frequency Domain. Most of these filters are given in a tabular form for amplitude and phase or delay as a function of frequency. The data in tables are often taken from measurements which themselves may have been taken at key nonuniformly spaced frequencies. To provide a uniformly spaced table of N sampling points at ΔF frequency increments, the data in the tables are interpolated and resampled. The two interpolation methods most commonly used are the *cubic spline* and *linear* methods (Section 3.5). When initially setting up a problem, it may not be clear what are “good” numbers to use for the number of samples N and the frequency increment ΔF . Therefore there may have to be some trial and error; that is, the interpolation and resampling may have to be iterated a few times.

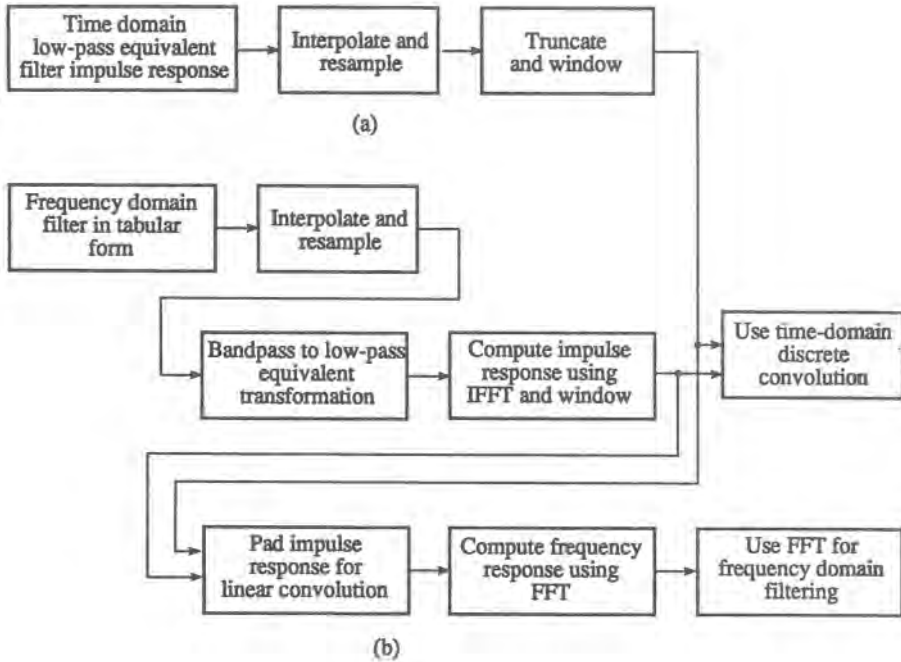


Figure 4.16. Flowchart for simulation of FIR filters, (a) Time-domain specified FIR filter design; (b) frequency-domain specified FIR filter design.

The procedure to follow for mapping filters from the continuous to the discrete domain is shown in the flowchart in Figure 4.16b and is described below.

1. The first step, if applicable, is to map the bandpass filter specified for a carrier-modulated system into a lowpass equivalent (see Section 3.4). In most cases this will simply mean translating the table to a frequency band centered about zero, i.e., the carrier frequency is set to zero. If the resultant response is not symmetric about zero (even amplitude and odd phase), the IFFT will produce a complex impulse response (see Section 3.4).
2. The second step is to determine the number of frequency increments ΔF and the frequency span F of the FFT. To do that, we have to remember a few simple relationships mentioned earlier. In the frequency domain the total frequency span in the FFT is $F = N\Delta F$, where N is the number of frequency increments. In the time domain the total time $T = NT_s$, where T_s is the sampling interval. Moreover, $\Delta F = 1/T$ and $T_s = 1/F$ (see Section 3.8.4.6). The process of selecting of F , ΔF , T , and T_s is as follows.
 - a. Determine the sampling period T_s . This is done by inspecting the frequency response and defining a bandwidth B such that the filter response outside B is sufficiently small (e.g., below -40 dB). Then, usually it will be adequate if F is on the order of 3–5 times the filter bandwidth B . Note that T_s may have been already determined independently for other models. The smallest T_s is then selected for the system.

- b. We then select a frequency increment $\Delta F = 1/T$ by estimating the total time T . ΔF has to be small enough to represent adequately the tails of the impulse response in detail. To center the impulse response, we may have to introduce a delay to the frequency response. A delay of $e^{j2\pi fT/2}$ will shift the impulse response by $T/2$. We then perform an IFFT. Since we do not know the correct value of T *a priori*, this step may have to be repeated a few times.
 - c. Truncate the impulse response to a length such that it does not affect the specified filter performance. A good rule of thumb is that the truncated tails contain no more than a few percent of the energy of the impulse response.
3. It is good practice to window the impulse response to reduce the Gibbs phenomenon distortion. This of course will smooth the frequency response. Before windowing it is necessary to “center” the impulse response so that it is lined up with the window.
 4. The resulting impulse response is ready for time-domain convolution FIR filtering.
 5. If the FIR filtering is to implement linear convolution via the FFT, the impulse response has to be augmented with zeros. If block processing is used, the impulse response is augmented with zeros to fill the block. Otherwise the choice of the segment size is not critical. One rule of thumb is to augment the impulse response with zeros so that its size doubles. We perform the FFT, and the frequency response FIR filter is ready to be used in a simulation.

The following is a simulation of a frequency-domain filter using windowing.

■ **Example 4.1.6.** The frequency response of the asymmetrical bandpass filter shown in Figure 4.17 is

$$H(f) = \begin{cases} a, & f_0 - B/2 \leq |f| \leq f_0 \\ b, & f_0 \leq |f| \leq f_0 + B/2 \\ 0, & \text{otherwise} \end{cases}$$

The lowpass-equivalent frequency response evaluated by the methods described in Section 3.4 is

$$H_L(f) = \begin{cases} a \exp(j2\pi f t_0), & -B/2 \leq |f| \leq 0 \\ b \exp(j2\pi f t_0), & 0 \leq |f| \leq B/2 \\ 0, & \text{otherwise} \end{cases}$$

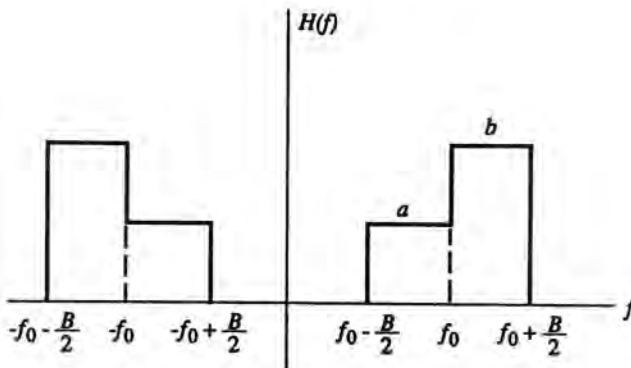


Figure 4.17. Frequency response of an asymmetric bandpass filter

The phase $\exp(j2\pi f_0 t)$ was added in order to center the impulse response in the time domain.

The real and imaginary components of the lowpass-equivalent impulse response $h(t)$ can be shown to be

$$h_1(t) = \frac{a + b \sin[\pi B(t - t_0)]}{2 \pi(t - t_0)}$$

$$h_2(t) = j \frac{a - b \cos[\pi B(t - t_0)]}{2 \pi(t - t_0)}$$

To demonstrate the simulation techniques described in previous sections, we present below the results of a simulation of the above filter for $a = 0.5$, $b = 1.0$, and the bandwidth $B = 1.0$. A delay $t_0 = T/2$ was added to center the impulse response.

To examine the frequency and impulse responses with high resolution, we used an FFT with 1024 samples and a sampling period of $T_s = 0.05$ s. This corresponds to an impulse response with duration $T = 50$ s, a frequency span $F = 20$ Hz, and a sampling rate $f_s = 20$ samples/s. For most practical purposes such an idealized frequency response is not needed and often not desired because it produces long ringing tails in the time domain (Gibbs phenomenon distortion). We therefore truncated the impulse response to about five sidelobes on each side of $h_1(t)$ or a total of $T \approx 10$ s. Since the bandwidth of the filter is $B = 1.0$ Hz, a sampling rate $f_s = 5.0$ samples/s will be adequate. We chose, therefore, $T_s = 1/f_s = 0.2$ s and $T = NT_s = 12.9$ s, since it is desirable for the number of samples $N = 64$ to be an integer power of two, $N = 2^n$.

The truncated real and imaginary impulse responses are shown in Figure 4.18a. To be useful in a linear convolution (see Section 3.8.6), we augmented the impulse responses with an additional 64 zeros and performed the FFT with $N = 128$ samples. Figure 4.18b shows the resultant frequency response. As expected, the Gibbs phenomenon distortion is severe. It is interesting to note that the Gibbs phenomenon is not apparent in the 64-sample frequency response, which is due to the fact that the samples occur at the zero crossings of the frequency response (see Section 3.8.4.7 and Problem P4.8).

Figure 4.19a shows the impulse responses smoothed by a Hamming window

$$w(n) = 0.54 + 0.46 \cos\left(\frac{2\pi n}{N}\right), \quad -\frac{N-1}{2} \leq n \leq \frac{N-1}{2}$$

Since windowing is performed before augmenting with zeros, $N = 64$ in the equation above.

As seen from the frequency response of the windowed functions in Figure 4.19b, the Gibbs phenomenon distortion is reduced by about 20 dB. The frequency response, however, has slightly gentler rolloff.

The model of an FIR filter developed in this example can be used either for frequency- or for time-domain filtering. In the frequency domain it is used in a linear convolution type FFT filtering, using the appropriate technique. ■

4.1.2.4. Comparison of Time-Domain (Impulse Response) and Frequency-Domain (FFT) Implementations for FIR Filtering

The convolution sum is simple and straightforward to implement for FIR filters. Hence, time-domain convolution is the preferred type of filtering for FIR filters if the number of samples in the impulse response is not prohibitively large. As a rule of thumb, FIR filters requiring $N = 128$ samples or less may be implemented using the time-domain convolution method, without loss of efficiency.

Filters with sharp cutoffs generally have a “longer” impulse response, i.e., a larger interval over which the response is significant. Hence, to create an FIR filter there will

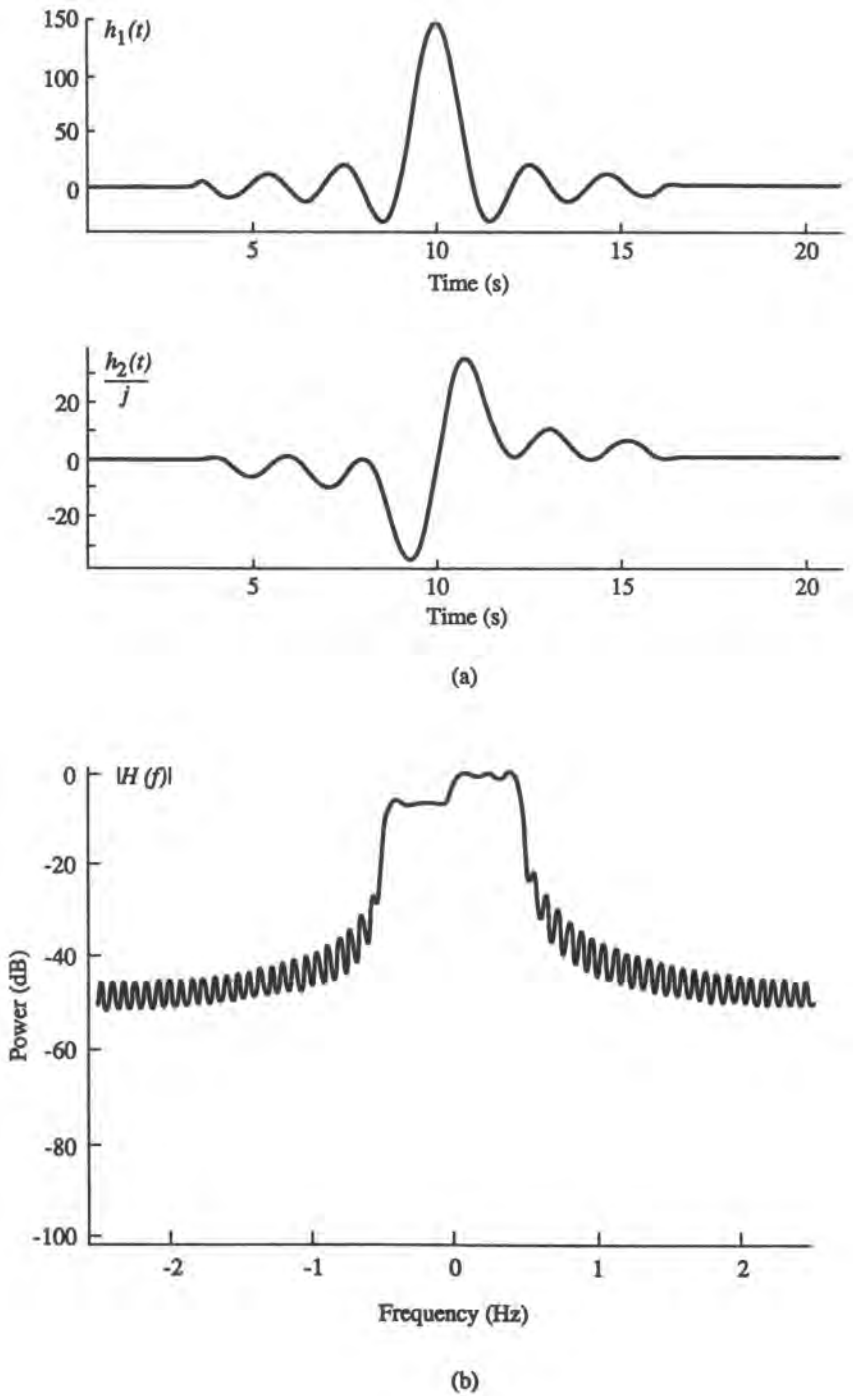


Figure 4.18. (a) Truncated real and imaginary components of the lowpass-equivalent impulse response $h(t)$; (b) lowpass-equivalent frequency response of filter with truncated impulse response.

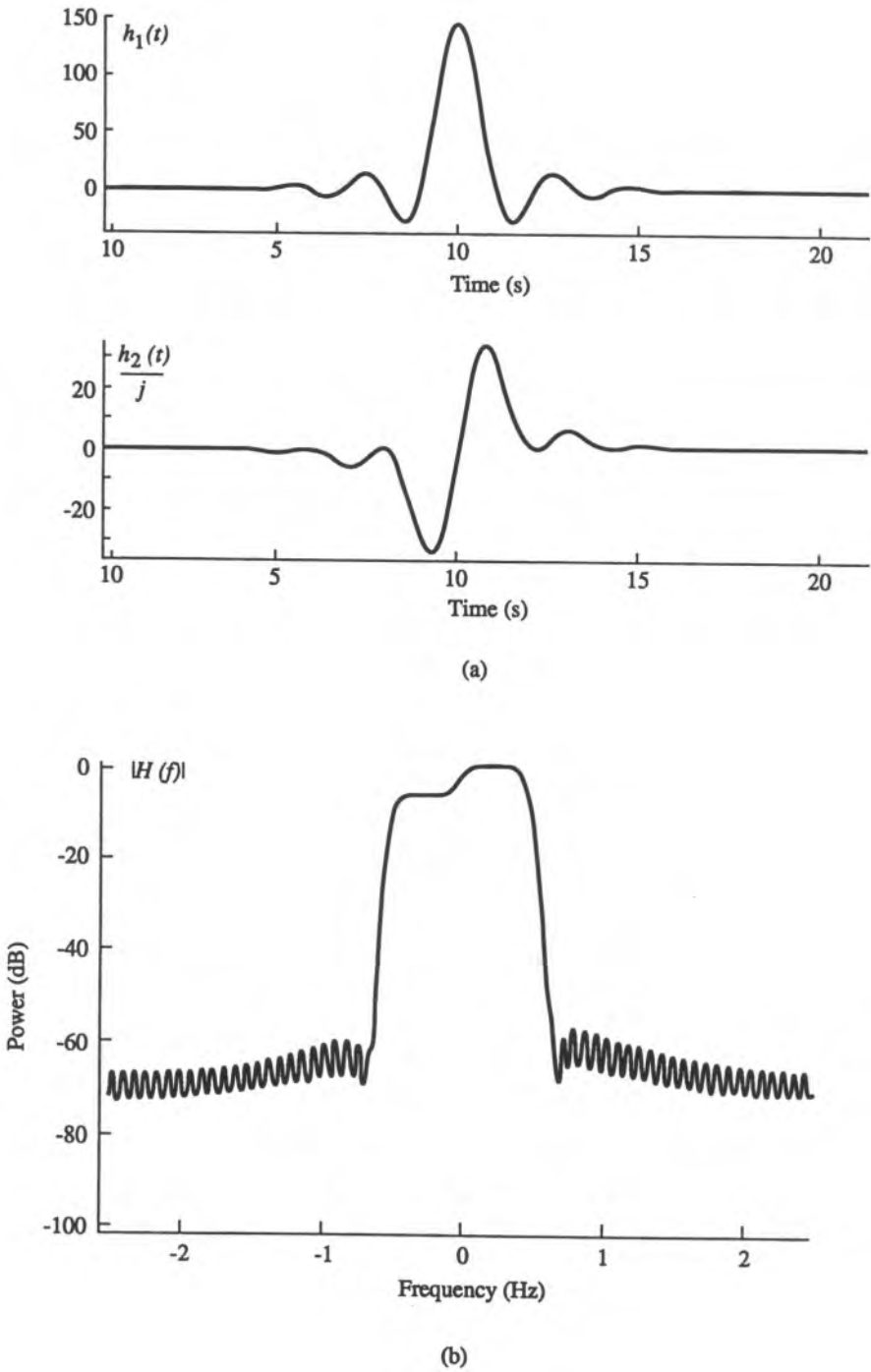


Figure 4.19. (a) Windowed real and imaginary components of the lowpass-equivalent impulse response $h(t)$; (b) lowpass-equivalent frequency response of filter with truncated impulse response.

generally be a larger number of points within the truncated response. Here, FFT-type filtering is generally preferred because of computational efficiency. The disadvantages are:

1. For nonperiodic processes we have to use the overlap-and-add or overlap-and-save methods. This adds some complexity to the simulation.
2. FFT filtering is inherently a block process. Input signal samples are accumulated in arrays before each filter process, thus creating artificial delays within the simulated system. This adds complexity when combining signals within a simulated system. It also prevents using these filters within a feedback loop.

The FIR filter techniques described above are used for simulation of analog filters in communication systems. However, as noted before, FIR filters are used extensively in digital signal processing. These frequency-selective filters are designed directly in the discrete-time domain. The techniques used in the design of these filters are amply described in the literature.^(1,2) The two most frequently used methods in frequency-selective filter design are the following:

1. *The windowing design method*, which is similar to the one described above for simulation.
2. *Equiripple approximation in the frequency domain for FIR filters*. Several techniques have been developed to obtain an optimal Chebyshev solution to the equiripple FIR filter. The most commonly used is the algorithm described by Parks and McClellan.⁽¹²⁾

4.1.3. Simulation of Filtering with IIR Filters

Infinite impulse responses in continuous systems can arise in a number of ways, but are most often associated with systems described by linear time-invariant differential equations (LTI/DE), such as the “classical” filters introduced earlier. The natural discrete counterparts to such continuous systems are LTI difference equations. We saw earlier, in fact, that one type of s -to- z mapping transforms a continuous LTI/DE system into one described by an LTI difference equation. We also pointed out in conjunction with (4.1.22) that direct computation of the convolution sum is not feasible for an IIR filter. An equivalent formulation for the output of an IIR system that *is* computable is in fact a (recursive) difference equation. Thus, filtering with IIR filters can be implemented in simulation with recursive structures.

We thus begin this section with a very brief introduction to linear difference equations and in particular their z -domain representations. We then present the standard structural interpretations through which such discrete systems may be realized. We then look at two often-used s -to- z mappings through which continuous structures may be transformed into discrete ones. We then summarize the steps for producing IIR discrete systems from continuous ones, and illustrate the process with examples.

4.1.3.1. Systems Characterized by Linear Constant-Coefficient Difference Equations

Let us consider the case of a general N th-order difference equation,

$$\sum_{k=0}^N b_k y(n-k) = \sum_{k=0}^M a_k x(n-k) \quad (4.1.49a)$$

Suppose $\{x(n)\}$ is the input sequence and $\{y(n)\}$ the output sequence. Since we can write (4.1.49a) as (with the usual normalization $b_0 = 1$)

$$y(n) = \sum_{k=0}^M a_k x(n-k) - \sum_{k=1}^N b_k y(n-k) \quad (4.1.49b)$$

we can see that the output at any time depends not only on the present and past inputs, but also on previous outputs. This latter fact is what makes the system *recursive*. In the case $b_k = 0$, for all k , (4.1.49b) reduces to the FIR filtering model discussed above.

Applying the z -transform to both sides of the equation and using the linearity and time-shifting property yields

$$\sum_{k=0}^N b_k z^{-k} Y(z) = \sum_{k=0}^M a_k z^{-k} X(z)$$

or

$$Y(z) \cdot \sum_{k=0}^N b_k z^{-k} = X(z) \sum_{k=0}^M a_k z^{-k}$$

so that

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M a_k z^{-k}}{\sum_{k=0}^N b_k z^{-k}} \quad (4.1.50)$$

Thus, the transfer function of a linear constant-coefficient difference equation is *always* rational.

The zeros of $H(z)$ are specified by the roots of the numerator and the poles by the roots of the denominator of (4.1.50).

Since the difference equation (4.1.49) has real coefficients, the poles and zeros of $H(z)$ are either real or occur in complex conjugate pairs.

The complex conjugate roots may be combined to produce biquadratic polynomials

$$(1 + cz^{-1})(1 + c^*z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2}$$

where $a_1 = c + c^*$ and $a_2 = c \cdot c^*$ are real numbers. Thus, the transfer function $H(z)$ can be represented as a product of M_1 real roots and M_2 biquadratic factors. All coefficients in the rational function $H(z)$ are real,

$$H(z) = K \frac{\prod_{i=0}^{M_1} (1 + a_i z^{-1}) \prod_{i=0}^{M_2} (1 + b_{1i} z^{-1} + b_{2i} z^{-2})}{\prod_{i=0}^{N_1} (1 + c_i z^{-1}) \prod_{i=0}^{N_2} (1 + d_{1i} z^{-1} + d_{2i} z^{-2})} \quad (4.1.51)$$

where $M = M_1 + 2M_2$ and $N = N_1 + 2N_2$.

4.1.3.2. Structures of Recursive Discrete Filters Implemented in Simulation Models

The widespread use of *digital signal-processors* (DSP) has prompted the development of many methods of realization of discrete filters.^(1-4, 13) Two of these methods are of particular interest in communication system simulation and are described below. The development of this section parallels that in the corresponding s -domain case that was examined in Section 3.6.6.1.

4.1.3.2.1. Direct-Form (Canonic) Realization. The result in (4.1.50) can, without loss of generality, be rewritten in a form where we set the order of both polynomials in the

numerator and denominator to the larger value, say M . If $M \neq N$, some of the coefficients in the lower order polynomial can simply be set to zero. Thus, the transfer function $H(z)$ can be represented as a cascade connection of $H_1(z)$ and $H_2(z)$,

$$\begin{aligned} H(z) &= H_1(z) \cdot H_2(z) = \frac{W(z)}{X(z)} \cdot \frac{Y(z)}{W(z)} \\ &= \left[\frac{1}{\sum_{k=0}^M b_k z^{-k}} \right] \left[\sum_{k=0}^M a_k z^{-k} \right] \end{aligned} \quad (4.1.52)$$

The function $W(z)$ introduced in (4.1.52) is an artifice that facilitates the development of an explicit realization for $H(z)$.

Since cascade connections are commutative [see (3.2.5)], we can arbitrarily assign the transfer functions $W(z)/X(z)$ or $Y(z)/W(z)$ to $H_1(z)$ or $H_2(z)$.

A simpler realization of the discrete transfer function results from the assignment

$$H_1(z) = \frac{W(z)}{X(z)} = \frac{1}{\sum_{k=0}^M b_k z^{-k}} \quad (4.1.53a)$$

and

$$H_2(z) = \frac{Y(z)}{W(z)} = \sum_{k=0}^M a_k z^{-k} \quad (4.1.53b)$$

The expression for $H_1(z)$ in (4.1.53a) can be rewritten in the form

$$X(z) = \sum_{k=0}^M b_k W(z) z^{-k}$$

which can be rephrased as

$$W(z) = \frac{1}{b_0} \left[X(z) - \sum_{k=1}^M b_k W(z) z^{-k} \right] \quad (4.1.54)$$

The difference equation corresponding to (4.1.54) is

$$w(n) = \frac{1}{b_0} \left[x(n) - \sum_{k=1}^M b_k w(n-k) \right]$$

The expression for $H_2(z)$ in (4.1.53b) can be rewritten in the form

$$Y(z) = \sum_{k=0}^M a_k W(z) z^{-k} \quad (4.1.55)$$

The difference equation corresponding to (4.1.55) is

$$y(n) = \sum_{k=0}^M a_k w(n-k)$$

Upon examining (4.1.54) and (4.1.55), we note that both equations contain chains of delays with identical inputs $W(z)$. Therefore, the corresponding transfer functions can be realized by a single chain of delays as shown in Figure 4.20. In this configuration, implementation of the difference equation requires only M delay elements. This realization is usually referred to as the *canonic realization* because it requires the minimum number of delay elements for implementation of (4.1.50).

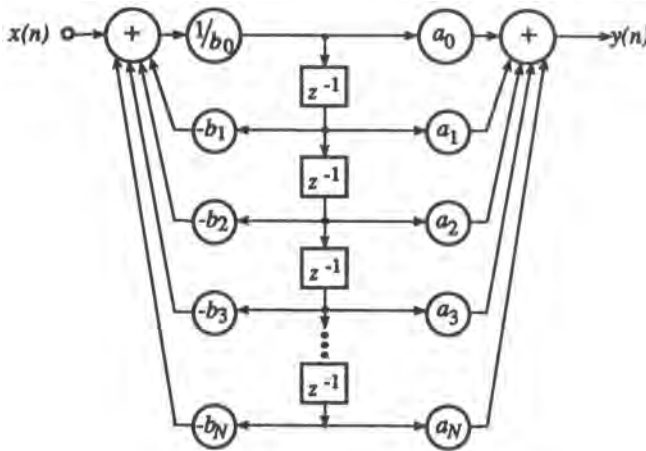


Figure 4.20. Discrete recursive filter, canonic realization.

4.1.3.2.2. *The Cascade Interconnections of Biquadratic Canonic Sections.* The canonic structure in Figure 4.20 was obtained directly from the transfer function $H(z)$ in (4.1.50). We can also realize the transfer function from the factored form of $H(z)$ in (4.1.52).

It is convenient to think of $H(z)$ in general form as a cascade connection of biquadratic sections, as was done in the s -domain case in Section 3.6.6.1,

$$H(z) = K \prod_{i=0}^L \frac{1 + \alpha_{1i}z^{-1} + \alpha_{2i}z^{-2}}{1 + \beta_{1i}z^{-1} + \beta_{2i}z^{-2}} \quad (4.1.56)$$

where L is $(M + 1)/2$. In this form we assumed that real poles or real zeros were combined in pairs. If there is an odd number of real poles or zeros, one of the coefficients α_{2i} or β_{2i} will simply be zero.

A cascade realization of a fourth-order system is shown in Figure 4.21. Each section in Figure 4.21 is the z -domain counterpart of the s -domain realization previously shown in Figure 3.27c. The cascade realization is the most commonly used model in communication systems simulation and in implementation because of its modular form. That is, since an arbitrary filter can be represented by a cascade of biquadratic sections, one needs to develop only a single, fairly simple program to simulate a biquadratic section (with coefficient values as inputs), and to simulate a more complicated filter, one simply makes multiple calls to the same basic program.

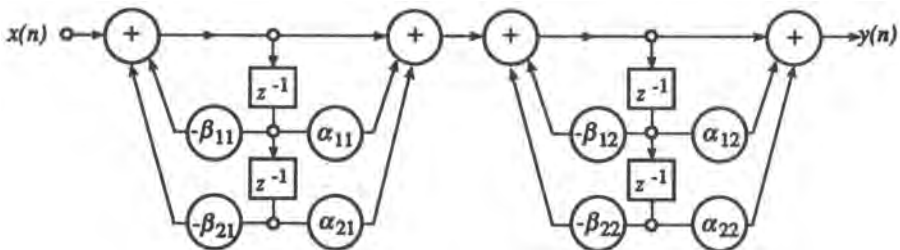


Figure 4.21. Cascade structure with a canonic realization for each second-order subsystem (biquad).

4.1.3.2.3. *The Parallel Realization.* The parallel-form discrete filter structure is obtained by performing a partial fraction expansion of the transfer function $H(z)$ in (4.1.50),

$$H(z) = \frac{b_n}{a_n} + \sum_{k=1}^N \frac{A_k}{1 + B_k z^{-1}} \quad (4.1.57)$$

Combining the complex conjugate poles into biquadratic forms yields

$$H(z) = C + \sum_{i=1}^{N/2} \frac{\alpha_{0i} + \alpha_{1i} z^{-1}}{1 + \beta_{1i} z^{-1} + \beta_{2i} z^{-2}} \quad (4.1.58)$$

Any two real poles can also be combined to yield a biquadratic section. If there is an odd number of real poles, the appropriate α_1 and β_2 are simply set to zero.

A parallel-form realization for a fourth-order filter using canonic biquadratic sections is shown in Figure 4.22.

4.1.3.3. Transformations between Continuous-Time and Discrete-Time Systems Represented by Rational Functions

In this section we consider two transformations often used for mapping continuous-time systems with rational transfer functions (hence, infinite impulse responses) to discrete-time systems with rational transfer functions. The first is a technique referred to as *impulse-invariant transformation* and is used mainly in modeling filters specified by the impulse response in the time domain. In this technique a continuous system is transformed into a discrete system by setting the impulse response of the latter to a sampled version of the impulse response of the continuous system. The discrete impulse response is obtained by sampling the continuous impulse response at a rate sufficient to avoid severe aliasing (see Section 3.5 on the sampling theorem).

The second technique we discuss is the *bilinear transformation*, which is a particular mapping from the s domain to the z domain belonging to the general class (4.1.32). This

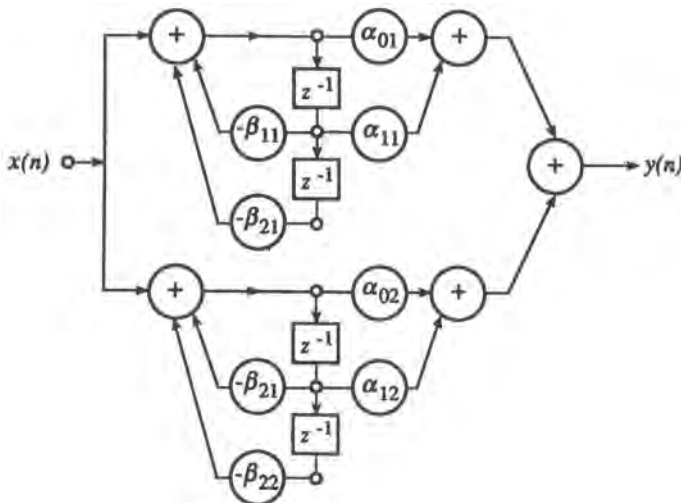


Figure 4.22. Parallel-form structure with a canonic realization for each second-order subsystem (biquad) associated with the bilinear transformation.

transform is used mainly for filters specified by masks in the frequency domain which are attempted to be met by one or another classical filter. It is therefore the preferred mapping method used in simulation of continuous systems except when specific time-domain characteristics are of paramount importance.

4.1.3.3.1. Impulse-Invariant Transformation. The characteristic property of the impulse-invariant transformation is that the impulse response of the resultant discrete filter is a sampled version of the impulse response of the continuous filter. As a consequence, the frequency response of the discrete filter is an aliased version of the continuous filter.

In (3.7.1)–(3.7.5) it was shown that, to map a continuous function $h_c(t)$ with a transfer function $H_c(s)$ into a discrete function $h_d(nT_s)$ with a transfer function $H_d(z)$, one can use the mapping

$$z = e^{sT_s} \quad (4.1.59)$$

It can be shown as a generalization of the sampling theorem in (3.5.6) that the z -transform is related to the Laplace transform of $h_c(t)$ by the equation

$$H_d(z)|_{z=\exp(sT_s)} = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} H_c \left[s + j \frac{2\pi}{T_s} k \right] \quad (4.1.60)$$

From the above relationship it is seen that strips of width $2\pi/T_s$ in the s -plane are mapped into the entire z -plane as shown in Figure 4.23.

The left half of each strip maps into the interior of the unit circle and the right half of the strip maps into the exterior of the unit circle. Each $2\pi/T_s$ segment of the imaginary axis maps linearly once around the unit circle. Therefore the frequency response of the discrete filter is evaluated for $z = e^{sT_s}|_{s=j\omega}$ and is given by

$$H_d(e^{j\omega T_s}) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} H_c \left[j\omega + j \frac{2\pi}{T_s} k \right] \quad (4.1.61)$$

From the above relationship, it is clear that the impulse-invariant transformation can introduce severe aliasing if the frequency response of the continuous filter spans more than one strip. Thus, to avoid aliasing, we have

$$H_c(f) = 0 \quad \text{for } |f| \geq f_s/2 \quad (4.1.62)$$

where $\omega = 2\pi f$ and $f_s = T_s^{-1}$. Equation (4.1.62) is merely a reiteration of the sampling theorem in Section 3.5.

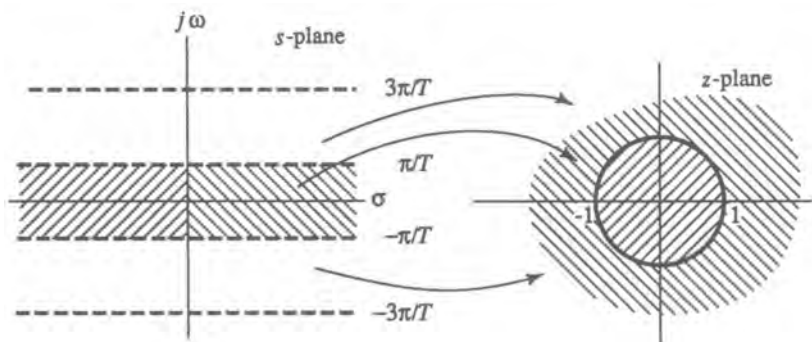


Figure 4.23. The mapping from the s plane to the z plane defined by $z = e^{sT_s}$.

Impulse-Invariant Mapping for Rational Functions. To demonstrate how one implements the impulse-invariant mapping technique, assume that the continuous filter has a transfer function such that the partial fraction expansion has the form

$$H_c(s) = \sum_{i=1}^m \frac{A_i}{s - s_i} \quad (4.1.63)$$

The impulse response of this filter is

$$h_c(t) = \sum_{i=1}^m A_i e^{s_i t} u(t) \quad (4.1.64)$$

The z -transform of the impulse response is

$$H_d(z) = \sum_{n=0}^{\infty} h_c(nT_s) z^{-n} \quad (4.1.65)$$

which, upon substitution of (4.1.64), gives

$$H_d(z) = \sum_{n=0}^{\infty} \sum_{i=1}^m A_i (e^{s_i T_s} z^{-1})^n$$

and which, after summing on n , yields

$$H(z) = \sum_{i=1}^m \frac{A_i}{1 - e^{s_i T_s} z^{-1}} \quad (4.1.66)$$

By comparing (4.1.63) to (4.1.66), it is seen that simple poles are mapped using the relation

$$\frac{1}{s - s_i} \rightarrow \frac{1}{1 - z^{-1} e^{s_i T_s}} \quad (4.1.67a)$$

When s_i is complex the poles appear in complex conjugate pairs

$$\frac{a_i}{s - s_i} + \frac{a_i^*}{s - s_i^*} = \frac{2\gamma_i s - 2(\sigma_i \gamma_i + \omega_i \delta_i)}{s^2 - 2\sigma_i s + (\sigma_i^2 + \omega_i^2)} \quad (4.1.67b)$$

where $s_i = \sigma_i + j\omega_i$ and $a_i = \gamma_i + j\delta_i$.

By substituting (4.1.67a) into each of the terms on the left side of (4.1.67b), we get

$$\begin{aligned} & \frac{a_i}{1 - z^{-1} e^{s_i T_s}} + \frac{a_i^*}{1 - z^{-1} e^{s_i^* T_s}} \\ &= \frac{2\gamma_i - z^{-1} e^{\sigma_i T_s} [2\gamma_i \cos(\omega_i T_s) + 2\delta_i \sin(\omega_i T_s)]}{1 - z^{-1} 2e^{\sigma_i T_s} \cos(\omega_i T_s) + z^{-2} e^{2\sigma_i T_s}} \end{aligned}$$

Hence a biquadratic term in the s domain is mapped into a biquadratic term in the z domain as

$$\begin{aligned} & \frac{\gamma_i s - (\sigma_i \gamma_i + \omega_i \delta_i)}{s^2 - 2\sigma_i s + (\sigma_i^2 + \omega_i^2)} \\ & \rightarrow 2 \frac{\gamma_i - z^{-1} e^{\sigma_i T_s} [\gamma_i \cos(\omega_i T_s) + \delta_i \sin(\omega_i T_s)]}{1 - z^{-1} 2e^{\sigma_i T_s} \cos(\omega_i T_s) + z^{-2} e^{2\sigma_i T_s}} \end{aligned} \quad (4.1.68)$$

From (3.6.23) and (4.1.68) we find that a biquadratic partial-fraction-expanded rational function in the s domain

$$H(s) = \sum_{i=1}^n \frac{s - \sigma_i - \omega_i(\delta_i/\gamma_i)}{s^2 - 2\sigma_i s + \sigma_i^2 + \omega_i^2} \quad (4.1.69)$$

is mapped into a partial-fraction-expanded rational function in the z domain

$$H(z) = \sum_{i=1}^n \frac{1 - z^{-1} e^{\sigma_i T_s} [\cos(\omega_i T_s) + (\delta_i/\gamma_i) \sin(\omega_i T_s)]}{1 + 2z^{-1} e^{\sigma_i T_s} \cos(\omega_i T_s) + z^{-2} e^{2\sigma_i T_s}} \quad (4.1.70)$$

where $n = m/2$. It should be noted that this modeling technique yields the parallel realization in Figure 4.22 with $\alpha_{0i} = 1$, $\alpha_{1i} = -e^{\sigma_i T_s} [\cos(\omega_i T_s) + (\sigma_i/\gamma_i) \sin(\omega_i T_s)]$, $\beta_{1i} = 2e^{\sigma_i T_s} \cos(\omega_i T_s)$, and $\beta_{2i} = e^{2\sigma_i T_s}$.

In summary, the use of impulse invariance preserves the time-domain behavior of the filter. To avoid aliasing, the frequency response of the continuous filter must be limited to the band $-f_s/2 \leq f \leq f_s/2$. In some situations, the incoming signal may have high-frequency components that are not of interest, or the actual filter bandwidth may be much larger than that of the signal itself, as in a highpass filter. In these cases, in order to avoid aliasing distortion, one can use a suitable lowpass filter (sometimes called a prefilter) to produce a bandlimited response that will not suffer from aliasing at the chosen sampling rate.

A summary of the procedure for transforming a filter defined by its poles and zeros into an impulse-invariant discrete equivalent is shown in the flowchart of Figure 4.24. As the first step, if applicable, this filter should be transformed into a lowpass equivalent. The continuous filter is next transformed into the discrete form. In the example below, we transform the lowpass equivalent of a filter into a discrete filter ready to be used in a simulation.

■ *Example 4.1.7.* Consider

$$H(s) = \frac{1}{(s+2)(s+3)} = \frac{1}{s+2} - \frac{1}{s+3}$$

The squared amplitude of the frequency response of this filter is

$$|H(j\omega)|^2 = \frac{1}{(6 - \omega^2)^2 + 25\omega^2}$$

At the frequency $f_c = \omega_c/(2\pi) = 12.5$ Hz the frequency response is down to -60 dB, which, for this example, defines the bandwidth:

$$\frac{|H(j\omega)|_{\omega_c=2\pi 12.5}^2}{|H(j\omega)|_{\omega=0}^2} = -60 \text{ dB}$$

We therefore choose the sampling rate $f_s = 2f_c = T_s^{-1} = 25$ samples/s. The discrete filter, using the mapping in (4.1.67a), is then

$$H(z) = \left(\frac{1}{1 - z^{-1} e^{-2T_s}} - \frac{1}{1 - z^{-1} e^{-3T_s}} \right)$$

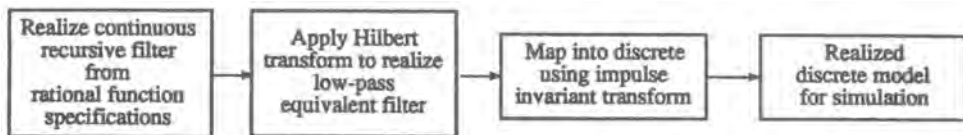


Figure 4.24. Flowchart for modeling of recursive filters from rational functions for simulation.

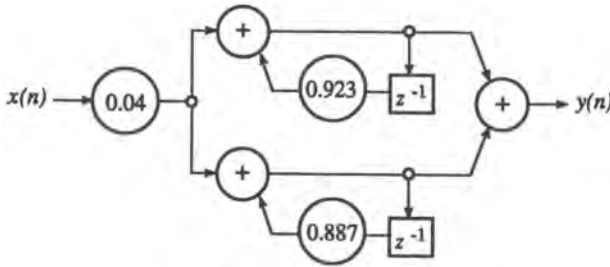


Figure 4.25. Parallel realization of filter in Example 4.1.7.

The parallel realization of the discrete filter above is shown in Figure 4.25.

The frequency response is evaluated from (3.7.12), which results in

$$H(\omega) = T_s |H(z)|_{z=\exp(j\omega T_s)} = T_s \left| \frac{e^{-j\omega T_s}(e^{-2T_s} - e^{-3T_s})}{1 - e^{-j\omega T_s}(e^{-2T_s} + e^{-3T_s}) + e^{-j2\omega T_s}e^{-5T_s}} \right|$$

The frequency responses for amplitude and phase of the continuous and discrete filters at various sampling rates are shown in Figures 4.26a and 4.26b. It is clearly seen from the plots that there can be considerable aliasing present. We should therefore use extreme caution when using the impulse-invariant transform. ■

4.1.3.3.2. The Bilinear Transformation. In Section 4.1.1.6 we introduced a general form of s -to- z mappings

$$\frac{1}{s} = f(z^{-1})$$

which we developed for the case where f is a ratio of polynomials in z^{-1} . By far the most commonly used instance of this type of mapping is the so-called bilinear transformation,

$$s = \frac{2}{T_s} \frac{1 - z^{-1}}{1 + z^{-1}} \quad (4.1.71a)$$

or

$$\frac{1}{s} = \frac{T_s}{2} \frac{1 + z^{-1}}{1 - z^{-1}} \quad (4.1.71b)$$

which has many desirable properties. To obtain the digital structure corresponding to a continuous filter $H(s)$, we use

$$H_d(z) = H\left(\frac{2}{T_s} \frac{1 - z^{-1}}{1 + z^{-1}}\right) \quad (4.1.72)$$

The order of the denominator of $H_d(z)$ is the same as that of $H(s)$; however, the order of the zeros may differ.

■ **Example 4.1.8.** Consider

$$H(s) = \frac{1}{s + a}; \quad H_d(z) = \frac{1 + z^{-1}}{2/T_s + a + z^{-1}(a - 2/T_s)}$$

The pole at $s = -a$ is mapped to a pole at $(2/T_s - a)/(2/T_s + a)$ in the z domain and the zero at ∞ in s is sent to -1 in the z domain. ■

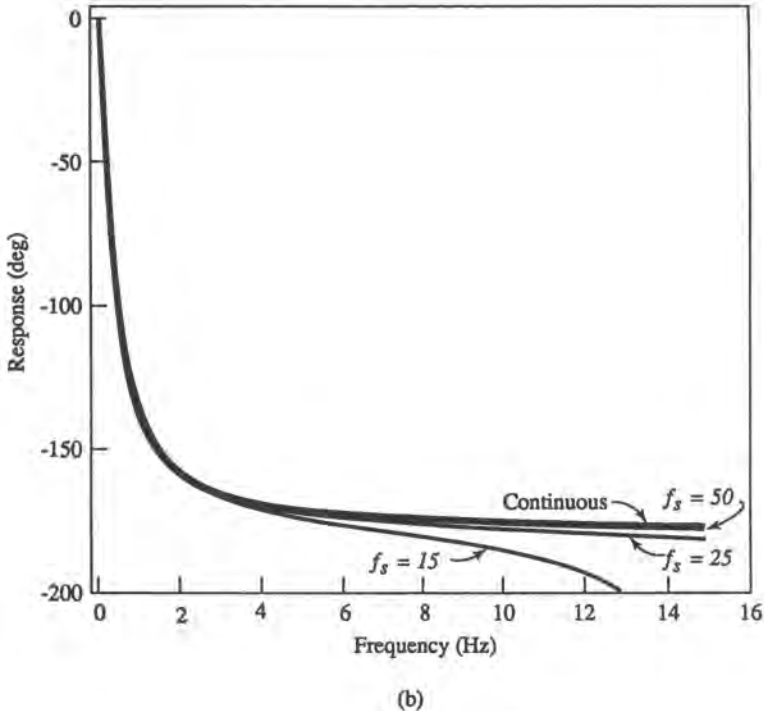
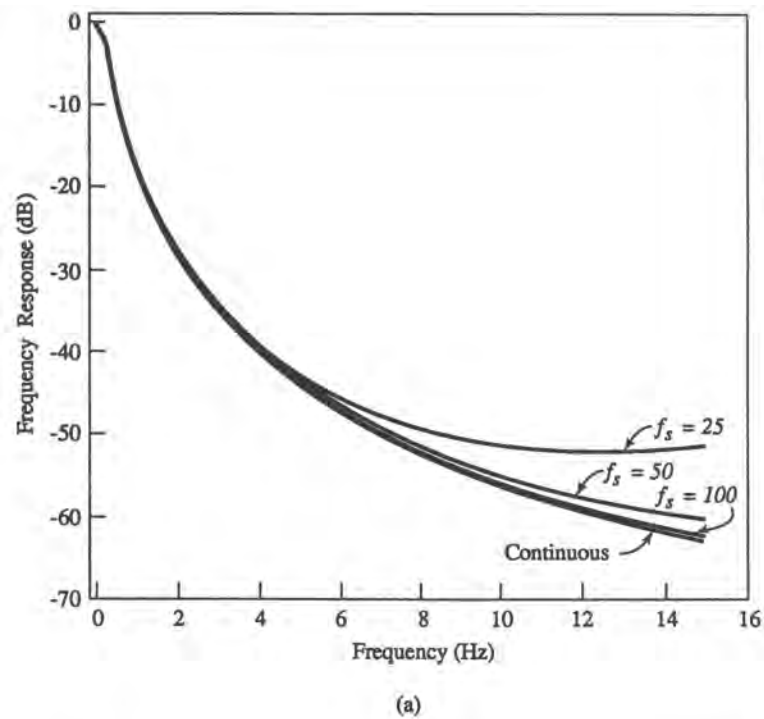


Figure 4.26. (a) Continuous and discrete frequency responses for different sampling rates. (b) Continuous and discrete phase responses for different sampling rates.

Note that the substitution

$$\frac{1}{s} \rightarrow \frac{T_s}{2} \frac{1 + z^{-1}}{1 - z^{-1}}$$

corresponds to the *trapezoidal rule* integration formula, which is a second-order implicit integration formula given by the difference equation (see Section 5.4)

$$y_n = y_{n-1} + \frac{T_s}{2}(x_n + x_{n-1})$$

By taking the z -transform of the above difference equation, we get the transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{T_s}{2} \frac{1 + z^{-1}}{1 - z^{-1}}$$

which, of course, is identical to the expression (4.1.71b).

This transformation is invertible. The inverse mapping is given by

$$z = \frac{2f_s + s}{2f_s - s} \quad (4.1.73)$$

where $f_s = T_s^{-1}$. If we let $s = \sigma + j\omega$, we get

$$z = \frac{2f_s + \sigma + j\omega}{2f_s - \sigma - j\omega}$$

For $\sigma < 0$ we find $|z| < 1$ and for $\sigma > 0$, $|z| > 1$. The entire left half-plane is mapped in the interior of the unit circle (see Figure 4.27) and the right half-plane is mapped to the outside of the unit circle.

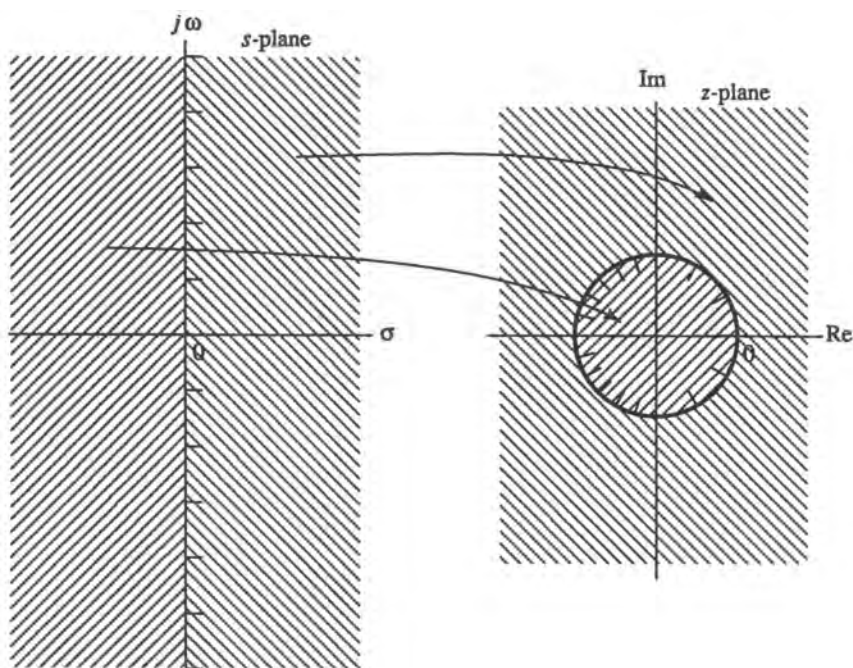


Figure 4.27. Mapping from the s plane to the z plane for the bilinear transformation.

Since the entire $j\omega$ axis of the s plane is mapped onto the unit circle in the z plane, aliasing errors are eliminated. There is, however, a nonlinear relationship between the continuous frequency ω and the discrete frequency Ω . Indeed, for $s = j\omega$ and $z = e^{j\Omega T_s}$, we get from (4.1.71a)

$$j\omega \rightarrow \frac{2}{T_s} \frac{1 - e^{-j\Omega T_s}}{1 + e^{-j\Omega T_s}}$$

or

$$\omega \rightarrow \frac{2}{T_s} \tan\left(\frac{\Omega T_s}{2}\right) \quad (4.1.74)$$

For small values of Ω , the mapping is almost linear. For most of the frequency scale it is highly nonlinear. Thus, the bilinear transform preserves the amplitude, but warps the frequency (see Figure 4.28). This implies that the bilinear transform will satisfy specifications only when the frequency response is described by a mask composed of piecewise-constant bands, and the exact shape of the response within the bands is of no consequence (see Figure 4.2). This requirement is met by a large class of filters which includes all classical filters described earlier. These filters can be modified from their original definition so that the frequency warping is compensated. This prewarping is performed at a set of critical frequencies and is best illustrated by example.

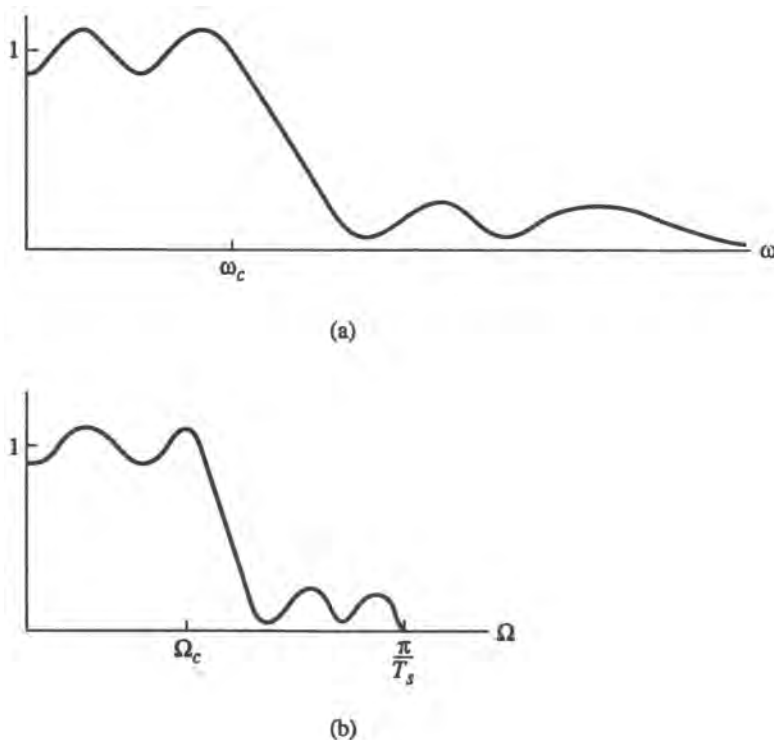


Figure 4.28. Nonlinear frequency warping caused by the bilinear transformation. (a) Transfer function in continuous domain; (b) transfer function in discrete domain.

■ **Example 4.1.9.** Figure 4.29 shows the frequency response of a bandpass filter. The set of desired cutoff frequencies of the discrete filter ($\Omega_1, \Omega_2, \Omega_3, \Omega_4$) is shown below the abscissa of Figure 4.29. Using the frequency warping relation (4.1.74), we find the new set of cutoff frequencies of the continuous filter ($\omega_1, \omega_2, \omega_3, \omega_4$). The continuous filter with the new set of cutoff frequencies is shown on the ordinate of Figure 4.29. ■

The advantages and disadvantages of the bilinear mapping are summarized below.

Advantages of the bilinear transformation:

1. Mapping the s plane to the z plane is one-to-one.
2. Aliasing is avoided.
3. The closed left half of the s plane is mapped onto the unit disk of the z plane.

Disadvantages of the bilinear transformation:

1. The frequency scale is warped.
2. Waveforms are not preserved if T_s is not small enough.

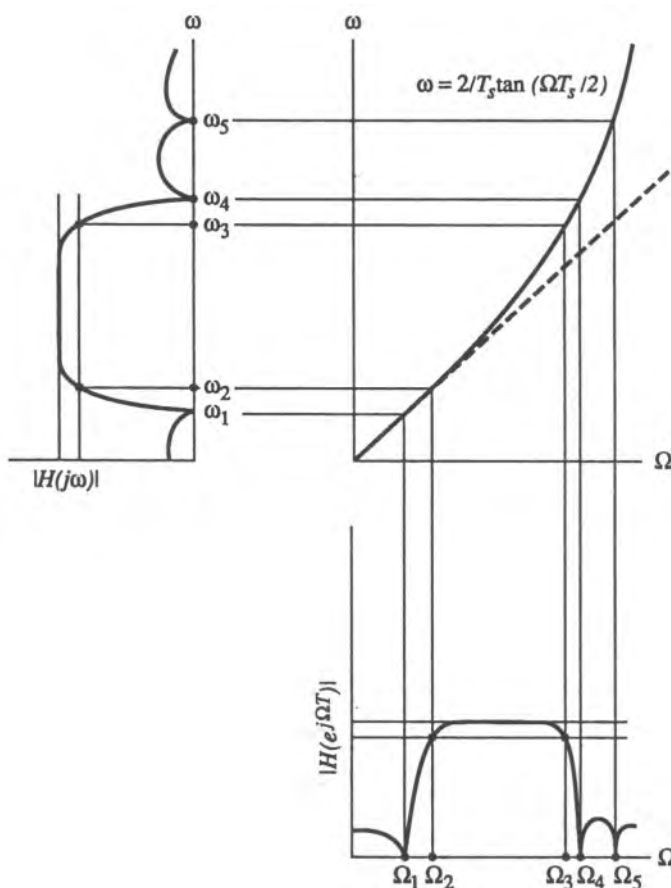


Figure 4.29. Technique for compensating the nonlinear frequency warping of the bilinear transformation.

Most classical filters can be modeled as a cascade connection of biquadratic sections. The coefficients of these sections are computed from the filter specifications including the prewarped cutoff frequencies. The biquadratic sections for the continuous filter are transformed into biquadratic sections of the discrete filter using the mapping

$$\frac{\alpha_1 s^2 + \beta_1 s + \gamma_1}{\alpha_2 s^2 + \beta_2 s + \gamma_2} \rightarrow \frac{z^{-2}(4f_s^2 \alpha_1 - 2f_s \beta_1 + \gamma_1) + z^{-1}(2\gamma_1 - 8f_s^2 \alpha_1) + (4f_s^2 \alpha_1 + 2f_s \beta_1 + \gamma_1)}{z^{-2}(4f_s^2 \alpha_2 - 2f_s \beta_2 + \gamma_2) + z^{-1}(2\gamma_2 - 8f_s^2 \alpha_2) + (4f_s^2 \alpha_2 + 2f_s \beta_2 + \gamma_2)} \quad (4.1.75)$$

This modeling technique yields a cascade realization as in Figure 4.21.

In summary, the bilinear transformation provides a simple mapping between continuous and discrete filters which is free of aliasing. It is ideal for filters in the frequency domain that have a piecewise-constant frequency response specification. One disadvantage of this technique is that neither the impulse response nor the phase response of the continuous filter is preserved.

4.1.3.3.3. Effect of Mapping on Lowpass-Equivalent Filters Represented by Rational Functions. Lowpass equivalents of narrowband bandpass filters represented by rational functions were derived in Section 4.1.1.4. The procedure was based on the reasoning that for narrowband bandpass filters ($B \ll f_0$) the positive-frequency response is dominated by the poles and zeros located in the positive-frequency half of the s plane. For narrowband bandpass filters with $B \ll f_0$ the poles and zeros are concentrated about the center frequencies $\pm f_0$ as in Figure 4.7a. To obtain the lowpass equivalent we substitute $f \rightarrow f + f_0$. The contribution of the negative-frequency poles and zeros, located now about $-2f_0$, to the lowpass frequency response is negligible for continuous filters. However, if these filters are mapped into the z domain, this contribution may be substantial and cannot be neglected.

If the continuous lowpass equivalents of bandpass filters are mapped into discrete filters via the *impulse-invariant transformation*, it is necessary to discard the negative-frequency poles located about $-2f_0$, since they may fold into the lowpass region and introduce severe aliasing. That is, we cannot merely translate the bandpass characteristics so that the response about $+f_0$ is now centered about $f = 0$, but we must also filter out anything else. This filtering is accomplished, as described in Section 4.1.1.4, by discarding the poles corresponding to the response about $-2f_0$.

The *bilinear transformation* does not introduce aliasing distortion. However, because of nonlinear frequency warping, the frequency band at $-2f_0$ is moved closer to the lowpass response, which will increase distortion.

4.1.3.3.4. Guide for Mapping Recursive Filters Specified in Frequency Domain. Filters that are specified by a mask of piecewise-constant acceptance strips as shown in Figure 4.2 are usually realized by classical filters. These filters are characterized by the *type*, for example third-order Butterworth or fifth-order elliptic, and by a small number of parameters at a few critical frequencies. The flowchart for preparation of classical filters for simulation is shown in Figure 4.30. The steps in the flowchart are briefly described below.

1. Select a lowpass-equivalent filter for bandpass or band-reject filters. If the center frequency is much larger than the bandwidth, then $\omega_0 \gg \omega_b$ (Section 4.1.1.4). In that case the lowpass-equivalent filter is identical to a lowpass prototype filter possibly shifted in frequency.

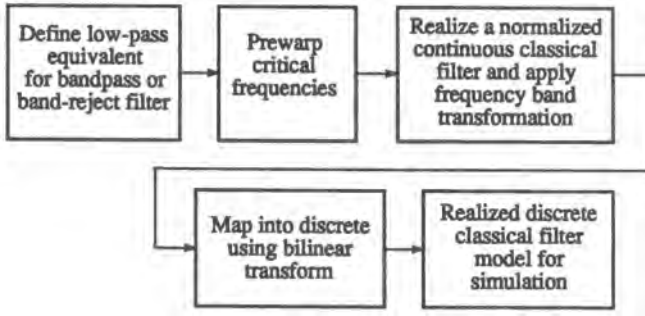


Figure 4.30. Flowchart for modeling of classical filters for simulation.

2. Since the bilinear transformation is most often used for mapping these filters, we have to prewarp the critical frequencies using the relationship in (4.1.74).
3. Next, the filter type and parameters are used to obtain, from filter design tables, the coefficients of the rational functions (usually in biquadratic form) for the normalized lowpass realization of the filter. The filter is then transformed into the appropriate continuous filter using the prewarped critical frequencies. The transformation to obtain the actual filter parameters is done using the relations in Section 4.1.1.3.
4. The biquadratic sections of the continuous filter in the s domain are transformed into the discrete biquadratic sections of the z -domain filter using the relation (4.1.75).

■ *Example 4.1.10.* To illustrate the application of the above procedure, let us simulate a lowpass filter using a Chebyshev filter type with the following specifications:

1. Passband ripple $\alpha_p = 2 \text{ dB} (\epsilon = 0.7648)$
2. Stopband loss $\alpha_r = 22 \text{ dB}$
3. Passband edge frequency $F_p = \Omega_p/(2\pi) = 100 \text{ Hz}$
4. Frequency where the prescribed stopband loss is achieved $F_r = \Omega_r/(2\pi) = 200 \text{ Hz}$
5. Sampling rate $f_s = 1/T_s = 600 \text{ samples/s}$

From the prewarping relation in (4.1.74) we compute that the prewarped frequencies of the continuous filter $f_p = 110.26 \text{ Hz}$ and $f_r = 330.79 \text{ Hz}$. The transition frequency ratio is then $f_r/f_p = 3$. The filter order that satisfies the above specifications is computed from (4.1.7) and results in $n = 2$. The transfer function of a second-order normalized lowpass Chebyshev filter with $\alpha_p = 2 \text{ dB}$ is, from Table 4.A.3 in the Appendix,

$$H(s) = \frac{1}{s^2 + 0.804s + 0.637}$$

Using the lowpass-to-lowpass transformation in (4.1.13), with $\omega_b = 2\pi f_p = 692.92$, we get the prewarped continuous filter

$$H(s) = \frac{4.77 \times 10^5}{(s^2 + 555s + 3.04 \times 10^5)}$$

By applying the bilinear transformation of (4.1.71a), we obtain the transfer function of the discrete filter

$$H(z) = \frac{(z^{-1} + 1)^2}{2.26z^{-2} - 4.75z^{-1} + 5.05} = 0.198 \frac{z^{-2} + 2z^{-1} + 1}{0.436z^{-2} - 0.94z^{-1} + 1.0} \quad (4.1.76)$$

The frequency response of the discrete filter is

$$|H(z)|_{z=\exp(j\Omega T_s)}^2 = \left| \frac{(e^{-j\Omega T_s} + 1)^2}{2.26e^{-j2\Omega T_s} - 4.75e^{-j\Omega T_s} + 5.05} \right|^2$$

The canonic realization of the discrete filter in (4.57) is shown in Figure 4.31a, and the frequency response in Figure 4.31b.

It should be noted that the above filter is a discrete model for a lowpass filter with specified passband and stopband tolerances. It is *not* a simulation of a continuous second-order Chebyshev filter. Indeed, a design of a continuous filter with the same requirements will produce a third-order Chebyshev filter. To simulate the performance of a given filter design, we will have to resort to a much higher sampling rate, to reduce frequency warping. For example, to obtain a warping effect of no more than 2% at a given frequency f , the sampling rate has to be $f_s = 13f$. ■

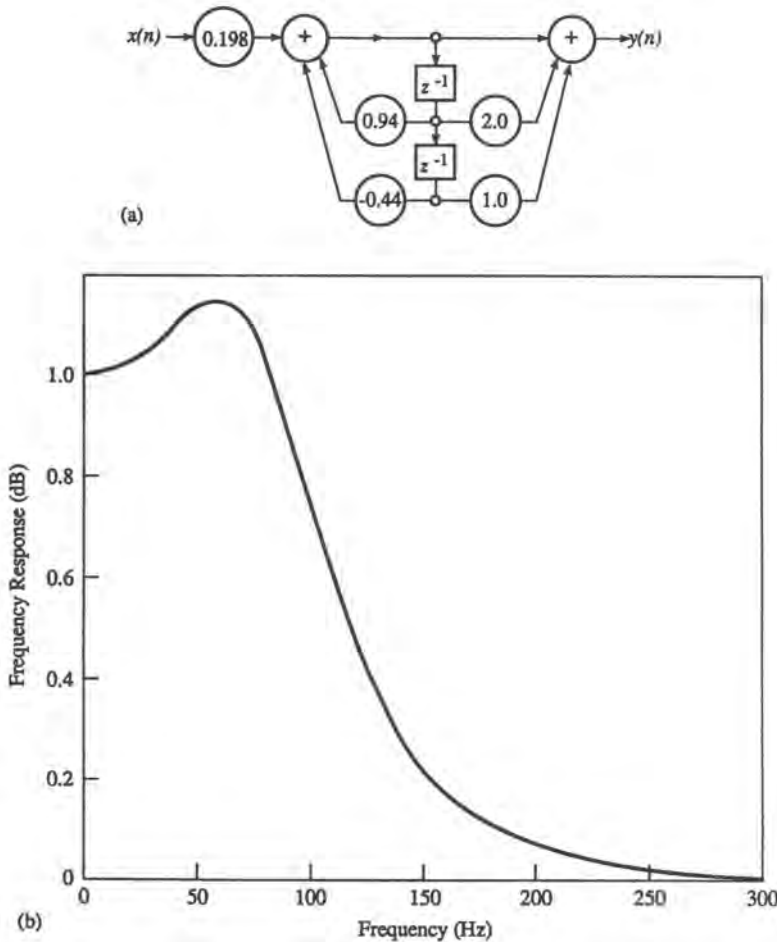


Figure 4.31. (a) Biquadratic realization of Chebyshev filter in Example 4.1.10; (b) frequency response of simulated Chebyshev filter in Example 4.1.10.

Classical filters are computationally fairly efficient since they require a minimum number of delay elements if implemented in the time domain. However, for sharp-cutoff filters as might be used in frequency multiplexing it is more practical to use the frequency-domain implementation of an ideal FIR filter for simulation purposes (see Sections 4.1.1.2 and 4.1.2.3).

4.1.4. Effects of Finite Word Length in Simulation of Digital Filters

The simulation algorithms for linear filters in IIR and FIR configurations are usually realized as programs on general-purpose computers. In both cases sequence values and coefficients are stored as finite-length floating-point numbers. This finite-word-length constraint produces quantization of the simulated values and is manifested in a variety of ways. The effect of this quantization on sampling rates of signals, structures of IIR filters, and discrete Fourier transforms will be briefly described below. Quantization effects in fixed-point arithmetic arise only in modeling of real hardware signal processors and are not covered in this section. The interested reader can find in-depth coverage of the subject in the vast literature available for digital signal processing, e.g., Refs. 1 and 2.

4.1.4.1. Roundoff Noise in Simulations of IIR Filters

The basic arithmetic operations in the simulation of an IIR filter are multiplication by a constant (the filter coefficients) and addition. It can be shown that both addition and multiplication lead to quantization noise that is proportional to the output of the operation.

The most damaging effect in the simulation of IIR filters results from the quantization of the filter coefficients. In the general approach to the analysis and synthesis of IIR filters with finite-precision coefficients, the effects of coefficient quantization can be represented by a stray transfer function in parallel with the corresponding ideal filter. The expected mean-square difference between the frequency response of the actual and ideal filter can be readily evaluated.

Knowles and Olcayto⁽¹⁴⁾ formulated the coefficient quantization problem as a statistical one. They showed convincingly that for any reasonably complex filter with steep transitions between passband and stopband, the use of the *direct form* (see Section 4.1.3.1) should be avoided because of the extreme sensitivity problem. The statistical model as derived by Knowles and Olcayto predicted that the *parallel form* had the smallest degradation of performance for a fixed number of bits, followed by the *cascade connection* and then the *direct form*. The relationships between the errors and the filter parameters for the different filter configurations are quite complex and are beyond the scope of this book. Because of the results mentioned, it is recommended that for high-order filters, the biquadratic configuration should be used. It is also recommended that the coefficients should be declared as double-precision numbers and the simulation of the filter should be performed in double-precision arithmetic. The biquadratic coefficients should be computed directly from the pole and zero values. It is obvious that first obtaining the transfer function of a filter as a ratio of polynomials and subsequently factoring to obtain biquadratic forms will degrade the accuracy of the coefficients.

It should be noted that in the design of hardware digital filters, optimization techniques have been developed that optimize the performance of filters for any given truncation.

4.1.4.2. Roundoff Noise in Simulations of FIR Filters

The output of an FIR filter can be represented as a sum

$$w(n) = y(n) + f(n)$$

where $y(n)$ is the output of the ideal filter and $f(n)$ represents the effects of rounding of floating-point products and sums. It can be shown that $f(n)$ has zero mean and the signal-to-noise ratio is

$$\frac{\sigma_f^2}{\sigma_y^2} \leq (N+1) \frac{2^{-2b}}{3}$$

where σ_f^2 is the variance of $f(n)$, σ_y^2 is the variance of $y(n)$, N is the number of taps in the filter, and b is the number of bits in mantissa of the floating-point number. This shows that increasing the number of taps N in the impulse response will decrease accuracy. If this causes a problem, it is recommended that, for large N , double-precision arithmetic be used.

4.1.4.3. Effects of Quantization in Computation of the Fast Fourier Transform

The FFT is widely used in frequency-domain filtering and spectral analysis. It is important, therefore, to understand the effects of finite register length in FFT calculations⁽²⁾.

If we assume that the input spectrum of the input signal is white, we can express the noise-to-signal ratio as

$$\frac{E[|F(k)|^2]}{E[|X(k)|^2]} = 2v\sigma_\epsilon^2$$

where $E[|F(k)|^2]$ is the expected value of the noise power caused by the mantissa roundoff, $E[|X(k)|^2]$ is the expected value of the signal power, $v = \log_2 N$, where N is the number of points in the FFT, and σ_ϵ^2 is the variance of the roundoff noise for each operation of multiplication and addition. Since σ_ϵ^2 is proportional to 2^{-2b} , where b is the number of bits in the mantissa, quadrupling v (which is equivalent to raising N to the fourth power) results in an increase in noise-to-signal ratio of one bit in the mantissa. Thus, the increase of the noise-to-signal ratio as a function of N is relatively low.

4.1.5. Summary of the Process of Mapping Continuous Signals and Systems into Discrete Signals and Systems for Simulation

4.1.5.1. Introduction

It is apparent from previous sections that there are several methods available with which continuous LTI systems can be modeled for discrete simulation. The selection of the appropriate method is partially dictated not only by the system specifications and by the type of information that we expect to obtain from the simulation, but also by the computational efficiency of the method and by the personal preference of the user. A short recommendation guide to how one selects an appropriate simulation method for a given problem is described below.

To prepare a communication system and the associated signals for simulation we have to perform two main operations: transform the system and signals into lowpass equivalents, and determine the sampling rate of the simulation.

1. *Mapping into a Lowpass-Equivalent System.* Quite often the communication systems and the associated signals are of the carrier-modulated type. It is computationally inefficient to include the carrier into the simulation process. Therefore, *all* linear and most nonlinear systems are, without any loss of generality, simulated using lowpass-equivalent signals and systems. In general these modulated signals and systems have both an in-phase and quadrature component. The lowpass-equivalent signals and system will, therefore, be complex.

The mapping from carrier-modulated systems into lowpass-equivalent systems is described in general terms in Section 3.4 and for systems described by rational functions in s in Section 4.1.1.4. In most cases the carrier modulated systems and the associated signals are narrowband. The mapping in this case is straightforward. The quadrature component maps into the imaginary part and the in-phase component maps into the real part of the complex signal or system.

2. *Determining the Sampling Frequency.* The sampling frequency of a strictly bandlimited signal has to be at least twice the bandwidth of the signal (see the sampling theorem, Section 3.5). In practice, physical signals or systems are never strictly bandlimited. Usually the bandwidth is determined as the frequency band outside of which the power or energy in the signal is negligible, e.g., below -60dB . This frequency band is normally determined from the energy spectrum for aperiodic signals and from the power spectrum for periodic signals. The determination of the power spectrum for random signals is described in Chapter 6. Generally, the sampling rate is chosen to control aliasing to tolerable levels. As a rule of thumb, 8–16 samples/Hz is adequate. Higher sampling rates may be required in particular cases, for example, in feedback loops (see Chapter 8).

Many communication system simulators use a single sampling rate for the whole system. The sampling rate is thus determined by the subsystem with the largest bandwidth. When determining the largest bandwidth, special attention should be paid to subsystems with nonlinearities, as described in Chapter 5.

4.1.5.2. A Guide to the Selection of the Proper Method for Filter Simulation

To summarize this section, we describe below and in the flowchart of Figure 4.32 the process by which an appropriate method for simulation of filters is selected. The corresponding sections in this chapter that describe these methods are also indicated in the chart.

1. *Finite Impulse Response Method.* If the filter shape is specified in the frequency domain by a finite set of values in tabular form, we have two choices, either to compute the impulse response and perform time-domain convolution, or use the equivalent and more efficient method of frequency-domain filtering via the FFT. The latter method lends itself especially well to modeling filters from frequency response data. As was also seen, FFT-based frequency-domain filtering retains its efficiency even for long-duration-impulse responses (i.e., sharp-cutoff filters) as long as that duration remains a relatively small fraction of the segment length in either the OA or OS method.

Since FIR filters do not have to be physically realized before the simulation, or even to be realizable, this method is invaluable in exploring system performance for different shapes of the amplitude and phase of the channel, e.g., a Nyquist filter with zero phase.

If the filter is specified in the time domain, we perform the same operations as just mentioned, but in reverse order. We have two choices, either perform time-domain convolution or compute the frequency response and use frequency-domain filtering via the FFT. If the filter is initially IIR, then its impulse response must first be truncated.

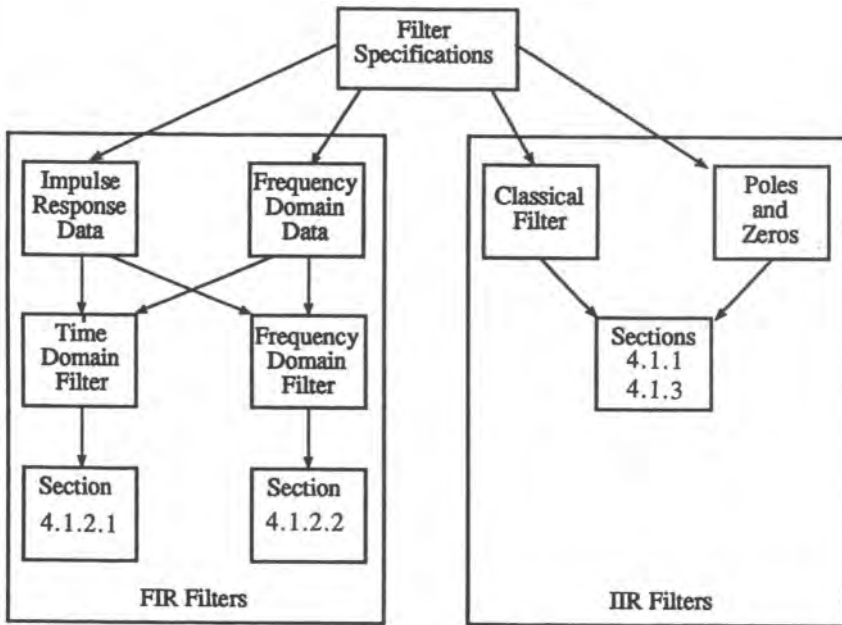


Figure 4.32. Selection of a simulation method for filters.

One advantage of FIR filters relative to IIR structures is unconditional stability. The main disadvantage of FIR filtering implemented with FFT techniques is the need to use block processing, which introduces a virtual delay in the simulation stream, thereby preventing its use in feedback loops.

2. *Infinite Impulse Response Method.* If the filter shape is specified by a classical filter, e.g., a fifth-order lowpass elliptic filter, or given as a transfer function in the form of a rational function for a designed filter, then the natural method to use is that of recursive filter simulation. Since these filters are modeled by a rational function, they are also computationally very efficient, because it takes far fewer delay elements to characterize a rational function as compared to a polynomial fit.

3. *Combination of Approaches.* The modeling choices described above are not unique and have a large area of overlap. For example, we can obtain the frequency response of a classical filter from the transfer function and use the FFT approach for simulation, either by truncating the impulse response or the frequency response, or we can fit a rational function to a specified frequency response and use a recursive filter simulation.

4.2. Time-Varying Linear Systems

Linear elements in a communication system can be time-invariant or time-varying in nature. Linear time-invariant (LTI) models and their application to the modeling and simulation of such elements as filters in communication systems were discussed in detail in the preceding section and in Chapter 3. We now turn our attention to the modeling and simulation of time-varying components.

The assumption of time invariance implies that the properties of systems being modeled do not change over (long periods of) time. If we are modeling a system by a time-invariant transfer function, it is of course implied that the transfer function remains fixed and shows no changes as a function of time. If, for example, the model is a third-order Butterworth filter with a 3-dB bandwidth of 1MHz, time invariance implies that element values and consequently the bandwidth do not change as a function of time. In order for this assumption to hold, the values of all physical components of the filter such as the resistors and active components cannot change as a function of time. While this assumption might hold over a short period of time that might be long relative to the rate of change of the information process, component values do change, albeit slightly, as a function of time due to environmental variations or aging, particularly when observed over a very long period of time. Thus, while time invariance may be a good assumption over some limited time interval, it will not hold indefinitely.

Whether to use a time-invariant or time-varying model is usually determined by the rate at which the characteristics of the communication system being modeled are changing in comparison to other parameters of the communication system such as the symbol rate. If the time constant associated with the time variations is very large compared to, say, the symbol rate in the system, then a (locally) time-invariant model may be justified. On the other hand, if the system parameters are changing at a rate approaching the symbol rate, a time-varying model is appropriate. Thus, one has *slow* or *fast* variations compared to the symbol rate or some other attribute of the system that influences the choice of a time-varying or time-invariant model. Let us further examine this point using two examples.

4.2.1. Examples of Time-Varying Systems

Consider a microwave radio communication system in which the transmitting and the receiving antennas are located on fixed microwave towers. The *channel* for these antennas is the atmosphere and the changes in the channel characteristics are due to changes in the atmospheric conditions, which typically have a time constant of several minutes to hours. Now, if the communication link is operating at a symbol rate of 100Mbit/s, then the time constant associated with the channel variations is very long compared to the symbol time. Indeed, the channel will remain in nearly the same state while billions of symbols flow over the link. Now, if the objective of simulation is BER estimation, then, during the estimation interval, the channel can be assumed to be in a *static* state and a time-invariant model can be used during the BER estimation. The long-term behavior of the channel and its impact on long-term system performance can be evaluated by analyzing system performance over a series of *snapshots* of static channel conditions, using a different time-invariant model for each snapshot. From such simulations, one can obtain performance measures such as *outage probabilities* which describe the portion of time during which the channel performance might fall below some BER threshold.

As a second example, consider a mobile communication system which consists of a fixed base station and a mobile user. The characteristics of the communication channel between the transmitter and the receiver in this system will be changing as a function of time since the parameters of the channel such as the attenuation and delay are changing because of relative motion between the base station and the user, in addition to changes in atmospheric conditions. If the mobile user is moving fast and if the symbol rate is of the order of 10 Ksymbols/s, then the rate at which channel conditions change might be comparable to the symbol rate and hence we need to consider a time-varying channel model. While a time-

varying model may not be needed for BER estimation, such a model will be necessary to study the behavior of receiver subsystems such as synchronizers and equalizers.

4.2.2. Time-Domain Description for Linear Time-Varying Systems

4.2.2.1. The Impulse Response

The commonly used form of the time-varying impulse response is modeled on the definition of the impulse response for linear time-invariant systems defined in Chapter 3,

$$h(t, \tau) = \Gamma[\delta(t - \tau)] \quad (4.2.1a)$$

where τ is the time (from the origin) when the impulse is applied, and Γ is the linear time-varying system operator.

The most important application of time-varying models in communication systems is in the modeling of time-varying (multipath) channels. To describe the behavior of such channels, Kailath⁽¹⁵⁾ introduced several alternative formulations for the impulse response, one of which is analytically more convenient than the others in different situations. The most convenient for purposes of modeling communication channels is to define $c(\hat{\tau}, t)$ as the response measured at time t to a unit impulse applied at time $t - \hat{\tau}$,

$$c(\hat{\tau}, t) = \Gamma[\delta(t - (t - \hat{\tau}))] = \Gamma[\delta(\hat{\tau})] \quad (4.2.1b)$$

We will refer to the above impulse response as the *channel impulse response*.

Note that the relationship of the above definition of the impulse response to the channel impulse response is $c(\hat{\tau}, t) = h(t, t - \hat{\tau})$ or $h(t, \tau) = c(t - \hat{\tau}, t)$. We will use one or the other definition of the above impulse responses in this section, depending on which is most appropriate to use in a given situation.

Since the system is time-varying, the impulse response will change as a function of both the time at which the impulse is applied and the time at which the output is measured. Hence, both $c(\hat{\tau}, t)$ and $h(t, \tau)$ are surfaces in three-dimensional space. Cross sections of these surfaces are illustrated in Figure 4.33. Figure 4.33a shows two hypothetical impulse responses $h(t, \tau)$ for values of $\tau = 0, 2$. Observe that causality requires $h(t, \tau) = 0$ for $t < \tau$. Because of the relationships $c(\hat{\tau}, t) = h(t, t - \hat{\tau})$ and $\hat{\tau} = t - \tau$, it is simple to establish a correspondence between the two functions. For example, the cross section $h(t, 2)$ can be mapped to a cross section $c(\hat{\tau}, t)$ with the help of the following table:

t	τ	$\hat{\tau}$
2	2	0
3	2	1
4	2	2
5	2	3
.	.	.
.	.	.
.	.	.

Thus, for example, $h(5, 2)$ is numerically equal to $c(3, 5)$. Hence, the cross sections of Figure 4.33a correspond to the cross sections in Figure 4.33b of the surface $c(\cdot, \cdot)$. In order to sharpen the distinction between h and c , we have used the symbols τ and $\hat{\tau}$ ($\hat{\tau} = t - \tau$). In Chapter 9,

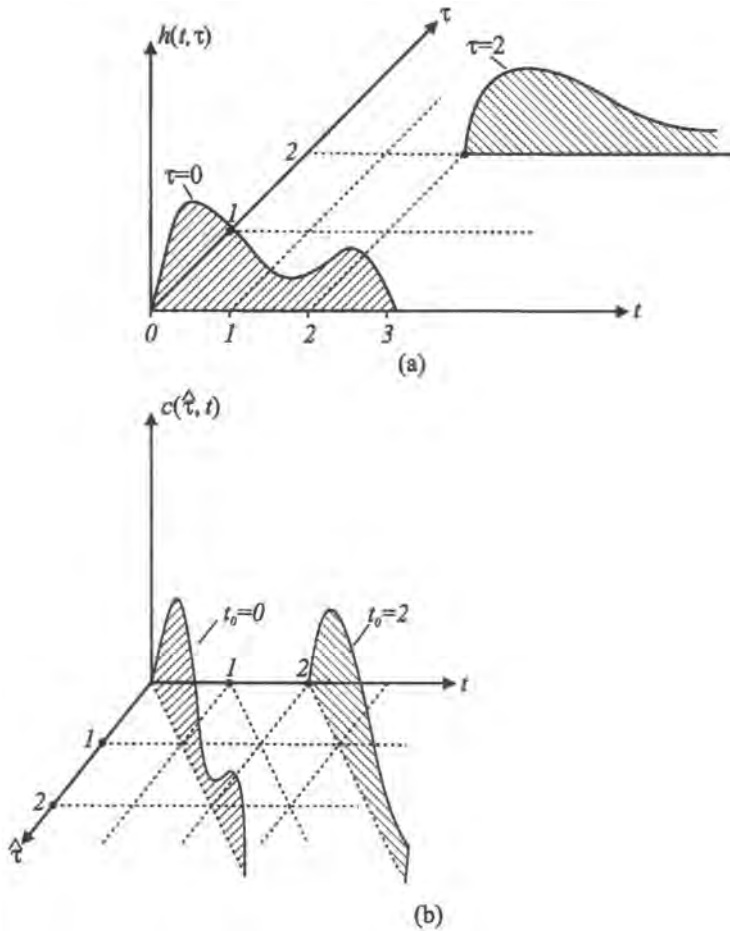


Figure 4.33. Two-dimensional representation of a time-varying impulse response. (a) $h(t, \tau)$; (b) $c(\hat{\tau}, t)$.

where we use c extensively, we will use the simpler notation τ for what we labeled here $\hat{\tau}$. For a time-invariant system the impulse response will strictly be a function of the elapsed time and

$$h(t_1, \tau) = h(t_2, \tau) = \cdots = h(t - \tau) \text{ or } c(\hat{\tau}, t_1) = c(\hat{\tau}, t_2) = \cdots = c(\hat{\tau})$$

While the impulse response of an LTI system maintains the same functional form irrespective of when the impulse was applied at the input, the impulse response of an LTV system depends on when the input was applied, as mentioned above. An illustration of this point is given in Figure 9.4 in Chapter 9.

4.2.2.2. The Superposition Integral

The response of the system to an arbitrary input is determined by a superposition integral^(15,16,17) (actually, a Fredholm integral⁽¹⁸⁾). Using the impulse response, the superposition integral is

$$y(t) = \int_{-\infty}^{\infty} h(t, \tau)x(\tau) d\tau \quad \text{or} \quad y(t) = h(t, \tau) * x(\tau) \quad (4.2.2a)$$

In terms of the channel response we have

$$y(t) = \int_{-\infty}^{\infty} c(\hat{\tau}, t)x(t - \hat{\tau}) d\hat{\tau} \quad \text{or} \quad y(t) = c(\hat{\tau}, t) * x(t - \hat{\tau}) \quad (4.2.2b)$$

When the system is time-invariant, $h(t, \tau) = h(t - \tau)$ and $c(\hat{\tau}, t) = c(\hat{\tau})$, and (4.2.2) is the convolution integral. Although the convolution integral is also sometimes referred to as the superposition integral, we will reserve the term *superposition* for the time-varying case in (4.2.2). For causal systems this integral may be written as

$$y(t) = \int_0^{\infty} h(t, \tau)x(\tau) d\tau \quad (4.2.3a)$$

because $h(t, \tau) = 0$ for $\tau < 0$, and

$$y(t) = \int_0^{\infty} c(\hat{\tau}, t)x(t - \hat{\tau}) d\hat{\tau} \quad (4.2.3b)$$

where $c(\hat{\tau}, t) = 0$ for $\hat{\tau} > t$.

4.2.3. Frequency-Domain Representations of Time-Varying Systems

For an LTV system one can develop the notion of a transfer function,⁽¹⁹⁾ albeit a time-varying transfer function $C_{\hat{\tau}}(f, t)$, by simply taking the Fourier transform of $c(\hat{\tau}, t)$ with respect to $\hat{\tau}$ as

$$C_{\hat{\tau}}(f, t) = \int_{-\infty}^{\infty} c(\hat{\tau}, t)e^{-j2\pi f\hat{\tau}} d\hat{\tau} \quad (4.2.4)$$

If the system is slowly time-varying, then the concepts of frequency response and bandwidth can be applied to $C_{\hat{\tau}}(f, t)$. Whereas the LTI system is characterized by a single impulse response function and a single transfer function, the LTV system is characterized by a family of impulse response functions and transfer functions, one function for each value of t , as shown in Figure 4.33. In fact, $C_{\hat{\tau}}(f, t)e^{2\pi ft}$ is the response to $\exp(j2\pi ft)$.

The variable t in $c(\hat{\tau}, t)$ and $C_{\hat{\tau}}(f, t)$ describes the time-varying nature of the system. Strong dependence on t and fast changes associated with t indicate a rapidly time-varying system. Usually, the time-varying nature of the system is modeled as a random phenomenon and is treated as a random process in t (see multipath channels in Section 9.1.3). If the process is stationary, then the time variations can be modeled by an appropriate autocorrelation function in the time domain or by a corresponding power spectral density in the frequency domain. The time constant of the autocorrelation function or the bandwidth of the power spectral density are key parameters that describe whether $c(\hat{\tau}, t)$ is slowly or rapidly time-varying.

The inverse transform is given by

$$c(\hat{\tau}, t) = \int_{-\infty}^{\infty} C_{\hat{\tau}}(f, t)e^{2\pi f\hat{\tau}} df \quad (4.2.5)$$

The output $y(t)$ of a system $c(\hat{\tau}, t)$ with the input $x(t)$ can be determined by

$$y(t) = \int_{-\infty}^{\infty} C_{\hat{\tau}}(f, t)X(f)e^{2\pi ft} df \quad (4.2.6)$$

where $X(f)$ is the Fourier transform of $x(t)$. The time-varying frequency response describes the system completely, as does the impulse response $c(\hat{\tau}, t)$.

4.2.3.1. Two-Dimensional Frequency Response

Kailath⁽¹⁵⁾ introduced a frequency-domain function

$$\begin{aligned} C_{\hat{\tau},t}(f, \nu) &= \int_{-\infty}^{\infty} C_{\hat{\tau}}(f, t) e^{-j2\pi\nu t} dt \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\hat{\tau}, t) e^{-j2\pi f \hat{\tau}} e^{-j2\pi\nu t} d\hat{\tau} dt \end{aligned} \quad (4.2.7)$$

The frequency ν is associated with the rate of change of the system and f is associated with the frequency of the system excitation.

In defining the above two-dimensional Fourier transform, the usual finite-energy assumption is made about the impulse response function. However, we will see later in Section 9.4 that for channel modeling, $c(\hat{\tau}, t)$ will be treated as a stationary random process in t , in which case the Fourier transform of $c(\hat{\tau}, t)$ may not exist with respect to t . The appropriate procedure here is to define the autocorrelation of $c(\hat{\tau}, t)$ with respect to the t variable and then take the Fourier transform of the autocorrelation function to obtain the frequency-domain representation in terms of the power spectral density of the random process.

From (4.2.5) and (4.2.6) we get the system output frequency response

$$Y(\nu) = \int_{-\infty}^{\infty} C_{\hat{\tau},t}(f, \nu - f) X(f) df \quad (4.2.8)$$

Equation (4.2.8) is the frequency-domain convolution of the input and system frequency-domain characteristics. In the double integral of Equation (4.2.7), the frequency variable f is associated with the tune variable $\hat{\tau}$ and it may be viewed as analogous to the frequency variable f in the transfer function $H(f)$ of linear time-invariant systems. However, the input-output relationship for LTV systems in the frequency domain given in Equation (4.2.8) involves a convolution in the frequency domain in the second variable of the transfer function $C(f, \nu)$. This convolution accounts for the effect of the time-varying nature of the system in the frequency domain.

4.2.3.2. Bandwidth Relations in Time-Varying Systems

The time-varying character of systems is usually manifested by a frequency spread, or a frequency shift, or both. Thus, the response of a LTV system to a single frequency f_0 can be a frequency spectrum or a frequency spectrum centered about a different frequency than f_0 (see Figure 4.34). The width of the spectrum (or frequency spread) about f_0 , appropriately defined, can be regarded as a measure of the rate of variation of the system. This frequency spread will depend on f_0 . An input signal normally has some nonzero bandwidth. Hence the frequency spread may differ for every spectral component. However, we can reasonably define the largest spread B_s as an appropriate measure of the time variability of the system. This spreading is usually encountered in multipath channel models, where it is usually referred to as Doppler spread, and the shift of the center frequency is referred to as the Doppler shift.

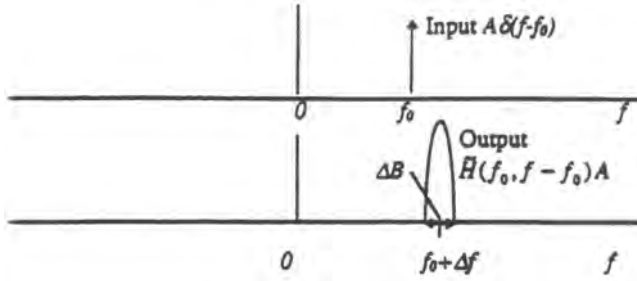


Figure 4.34. Illustration of frequency shift and spectral spreading due to motion of the transmitter, receiver, or medium.

4.2.3.3. Sampling Rate

From (4.2.8) we notice that an input with bandwidth B_i to a linear time-varying system characterized by B_s results in an output bandwidth not greater than $B_i + B_s$. Hence, to eliminate aliasing distortion in simulations of time-varying linear systems, the sampling rate f_s has to satisfy the sampling theorem (see Section 3.5), so that

$$f_s = \frac{1}{T_s} \geq 2(B_i + B_s) \quad (4.2.9)$$

In many cases $B_i \gg B_s$ and there is little impact on f_s , but when that is not the case, as in very low data rate systems, the simulation user should make sure that f_s is large enough to take the spread into account.

4.2.4. Properties of Linear Time-Varying Systems

4.2.4.1. Introduction

The properties of convolution for LTI systems were described in Section 2.2. Similar to the time-invariant case, the superposition possesses the properties of *associativity*

$$h_1(t, \tau) * [h_2(t, \tau) * h_3(t, \tau)] = [h_1(t, \tau) * h_2(t, \tau)] * h_3(t, \tau) \quad (4.2.10a)$$

and *distributivity*

$$h_1(t, \tau) * [h_2(t, \tau) + h_3(t, \tau)] = h_1(t, \tau) * h_2(t, \tau) + h_1(t, \tau) * h_3(t, \tau) \quad (4.2.10b)$$

However, generally it does not possess the *commutative* property

$$h_1(t, \tau) * h_2(t, \tau) \neq h_2(t, \tau) * h_1(t, \tau) \quad (4.2.10c)$$

4.2.4.2. Interconnections of Linear Time-Varying Systems

As in the case of LTI systems described in Section 3.6.5, the interconnections of LTV systems can be simplified using block diagram equivalents that are similar to the ones shown in Figure 3.25. However, the simplification of block diagrams for LTV systems is much more complicated than for LTI systems since transform methods are not applicable. Therefore, for the time-varying case the operations indicated in Figure 3.25 are not simply additions and multiplications of transfer functions as in the time-invariant case. Rather, the block-diagram

equivalents apply formally to the time-varying case by replacing H_1 by Γ_1 , H_2 by Γ_2 , and H by Γ , where the Γ 's are time-dependent linear operators whose properties may not always allow the usual simplification of block diagrams.

Cascade Interconnection of Systems. As an example of the complexity of reducing block diagrams of a time-varying system, consider a cascade connection of LTV systems Γ_1 and Γ_2 . Let the impulse responses of systems Γ_1 and Γ_2 be $h_1(\tau, t)$ and $h_2(\tau, t)$, respectively.

Then

$$\begin{aligned} z(t) &= \int_{-\infty}^t h_1(t, \tau) x(\tau) d\tau \\ y(t) &= \int_{-\infty}^t h_2(t, \xi) z(\xi) d\xi \end{aligned} \quad (4.2.11)$$

Combining these equations, we get

$$\begin{aligned} y(t) &= \int_{-\infty}^t d\xi h_2(t, \xi) \int_{-\infty}^{\xi} h_1(\xi, \tau) x(\tau) d\tau \\ &= \int_{-\infty}^t d\tau x(\tau) \int_{\tau}^t h_2(t, \xi) h_1(\xi, \tau) d\xi \end{aligned} \quad (4.2.12)$$

Therefore the equivalent system $\Gamma = \Gamma_1 \Gamma_2$ has the impulse response

$$h_1 * h_2 = h(t, \tau) = \int_{\tau}^t h_2(t, \xi) h_1(\xi, \tau) d\xi \quad (4.2.13)$$

A cascade interconnection of systems is equivalent to the superposition of their impulse responses. It should be noted, however, that the cascade operation is not commutative. Indeed, if we reverse the order of the cascade connection, the equivalent system is given by

$$h_2 * h_1 = \int_{\tau}^t h_1(t, \xi) h_2(\xi, \tau) d\xi \quad (4.2.14)$$

■ *Example 4.2.1.* The system in Figure 4.35 has

$$h_1(t, \tau) = u(t - \tau); \quad h_2(t, \tau) = t\delta(t - \tau)$$

The superposition for the cascade connection, using (4.2.13), is given by

$$h_1 * h_2 = \int_{\tau}^t t\delta(t - \xi) u(\xi - \tau) d\xi = tu(t - \tau)$$

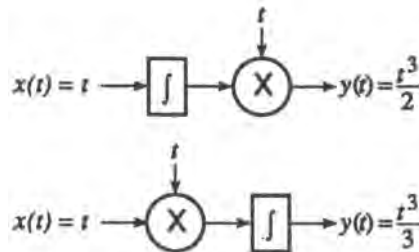


Figure 4.35. Illustration of the noncommutativity of time-varying linear systems (for Example 4.2.1).

while the reverse cascade connection yields

$$h_2 * h_1 = \int_{\tau}^t u(t - \xi) \xi \delta(\xi - \tau) d\xi = \tau u(t - \tau)$$

Applying the input $x(t) = t$ for $t \geq 0$ and using (4.2.2a) yields $y(t) = t^3/2$ in the first instance and $t^3/3$ in the second. ■

Other interconnection simplifications that involve a system function inverse, as in Figure 3.25c and especially the elimination of feedback interconnections in Figure 3.25e, would greatly simplify the simulation of communication systems. However, the evaluation of the inverse of time-varying systems is usually very difficult and its use in simulations is therefore limited.⁽²⁰⁾

4.2.5. Models for LTV Systems

4.2.5.1. Linear Differential Equation with Time-Varying Coefficients

Some LTV systems are described by linear differential equations with time-varying coefficients (LTV/DE) of the form

$$a_n(t) \frac{d^n y(t)}{dt^n} + a_{n-1}(t) \frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_0(t) y(t) = x(t) \quad (4.2.15a)$$

If the system is slowly varying, i.e., the coefficients of the LTV/DE are varying much more slowly than the signal rate, the system can be regarded as *quasistatic*. The LTV/DE can then be modeled as a recursive IIR filter described in Section 4.1.

For rapidly varying systems we now find the impulse response of the system, which is defined to be the solution of (4.2.15a) when $x(t) = \delta(t - \tau)$. It is well known^(17,18) that the solution of (4.2.15a) when $x(t) = \delta(t - \tau)$ is $u(t - \tau)$ times the solution $y(t, \tau)$ of the initial value problem

$$a_n(t) \frac{d^n y(t)}{dt^n} + a_{n-1}(t) \frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_0(t) y(t) = 0 \quad (4.2.15b)$$

with the initial conditions

$$y(\tau) = y'(\tau) = \cdots = y^{(n-2)}(\tau) = y^{(n)}(\tau) = 0, \quad y^{(n-1)}(\tau) = \frac{1}{a_n(\tau)}$$

Therefore the impulse response is

$$h(t, \tau) = y(t, \tau) u(t - \tau) \quad (4.2.16)$$

If a LTV/DE has an analytical solution, its impulse response has a separable form

$$h(t, \tau) = \sum_{i=1}^n p_i(t) q_i(\tau), \quad t \geq \tau \quad (4.2.17)$$

The above represents one of the most desirable types of models (known as a *separation model*) to be used both in simulation and analysis of time-varying linear systems.

Unfortunately, only first-order LTV/DEs have a general solution. Therefore, most “rapidly varying” LTV/DEs are modeled and simulated using the same methods that are used for nonlinear differential equations in Section 5.1.

As an illustration of an analytical evaluation of an LTV/DE, we evaluate the impulse response for a first-order equation

$$a_1(t) \frac{dy(t)}{dt} + a_0(t)y(t) = \delta(t - \tau) \quad (4.2.18)$$

The first step is to find a solution to the homogeneous equation

$$a_1(t) \frac{dy(t)}{dt} + a_0(t)y(t) = 0$$

Rearranging terms, we get

$$\frac{dy(t)}{y(t)} = -\frac{a_0(t)}{a_1(t)} dt$$

and therefore

$$y(t, \tau) = C \exp \left[- \int_0^t \frac{a_0(\xi)}{a_1(\xi)} d\xi \right]$$

Imposing the initial condition $y(\tau, \tau) = 1/a_1(\tau)$ allows us to determine C , and we find

$$h(t, \tau) = \frac{1}{a_1(\tau)} \exp \left[\int_0^\tau \frac{a_0(\xi)}{a_1(\xi)} d\xi - \int_0^t \frac{a_0(\xi)}{a_1(\xi)} d\xi \right] u(t - \tau) \quad (4.2.19)$$

An example of a first-order impulse response is shown in the following example.

■ **Example 4.2.2.** Determine the impulse response of the LTV system defined by

$$(t + 1) \frac{dy(t)}{dt} + (t^2 - 1)y(t) = \delta(t - \tau)$$

From (4.2.19)

$$h(t, \tau) = \frac{1}{\tau + 1} \exp \left[\int_0^\tau \frac{\xi^2 - 1}{\xi + 1} d\xi - \int_0^t \frac{\xi^2 - 1}{\xi + 1} d\xi \right] u(t - \tau)$$

which yields

$$h(t, \tau) = \frac{1}{\tau + 1} \exp \left[\frac{\tau^2}{2} - \tau \right] \exp \left[-\frac{t^2}{2} + t \right] u(t - \tau)$$

The impulse response $h(t, \tau)$ for four values of τ is shown in Figure 4.36. ■

4.2.5.2. Separable Models

As mentioned before, among the most desirable types of models to be used both in simulation and analysis of time-varying linear systems are the ones that have separable impulse responses.^{(21) †} An impulse response is termed separable if

$$h(t, \tau) = \sum_{i=1}^n p_i(t) q_i(\tau), \quad t \geq \tau$$

† It can be shown⁽²²⁾ that all impulse responses of linear systems characterized by a scalar linear time-varying differential equation as in (4.2.15a) are separable. However, as noted earlier, solutions to differential equations other than first order are generally not available.

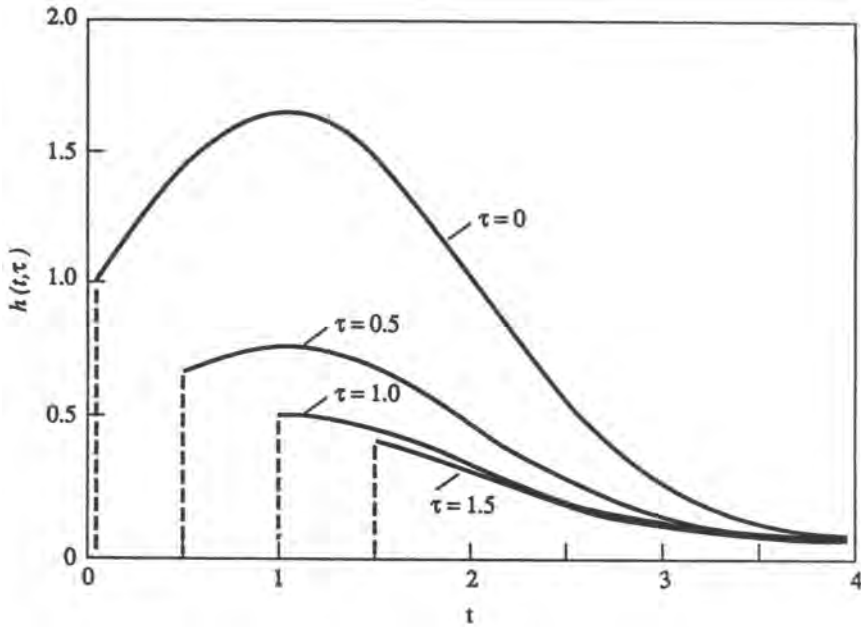


Figure 4.36. Equivalent linear time-varying system (for Example 4.2.2).

The system output for a separable impulse response is thus

$$y(t) = \int_0^t h(t, \tau) x(\tau) d\tau = \sum_{i=1}^N p_i(t) \int_0^t x(\tau) q_i(\tau) d\tau \quad (4.2.20)$$

The realization of such a system is shown in Figure 4.37.

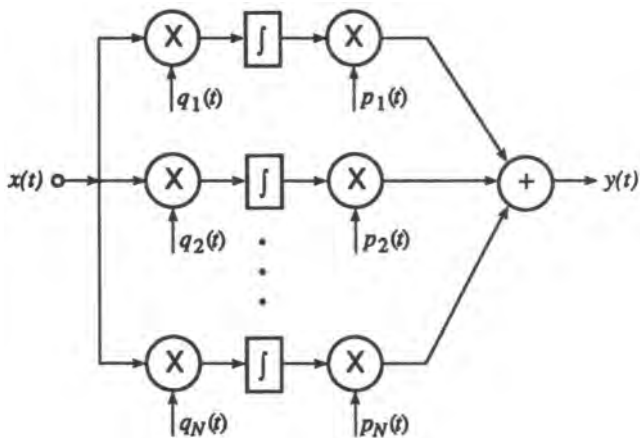


Figure 4.37. Structure of separable models for linear time-varying systems.

4.2.5.3. Tapped Delay-Line Channel Models

A variety of so-called *tapped delay-line* (TDL) models have been developed by Kailath based on the sampling theorem⁽¹⁵⁾. These models differ depending upon whether one assumes the input to the channel to be bandlimited or the output to be bandlimited, and in addition, whether or not one assumes the channel itself to be bandlimited. For our purposes, probably the most useful of the models is the one derived from assuming that the input signal is bandlimited.

From this formulation, a tapped delay-line model can be derived either for lowpass or bandpass channels. We confine ourselves here to a lowpass channel since it can be applied directly to the lowpass-equivalent case. The transmitted signal is modeled as

$$x(t) = s(t) * h(t) \quad (4.2.21)$$

where $*$ represents convolution, $s(t)$ is the modulator output, and $h(t)$ is the impulse response of an ideal lowpass output filter with transfer function $H(f) = 1$ for $|f| \leq B_i$ and $H(f) = 0$ for $|f| > B_i$. Thus, the channel input is bandlimited to B_i ; of course, if we assume $s(t)$ itself to be bandlimited with bandwidth $\leq B_i$, then the lowpass filter is superfluous.

There are a number of ways to arrive at the desired result. It is interesting to consider two different approaches. One of them will be given in Chapter 9, in conjunction with *randomly* time-varying channels. Here, we take a slightly different path, as follows.

First, based on the bandlimited assumption on $x(t)$, we can write it in the form of the sampling expansion. As we discussed earlier, there are two variables, $\hat{t}$ and t , the first associated with signal variations, and the second with the system or channel variations. It will be convenient, therefore, to use the $\hat{t}$ variable to expand $x(\cdot)$. Thus, from (3.5.9),

$$x(\hat{t}) = \sum_{n=-\infty}^{\infty} x\left[\frac{n}{2B_i}\right] \text{sinc}(2B_i\hat{t} - n) \quad (4.2.22)$$

Now, the Fourier transform of $x(\hat{t})$ is bandlimited by definition:

$$X(f) = F[x(\hat{t})] = 0 \quad \text{for } |f| > B_i$$

Hence, the system output (in the frequency domain)

$$Y(f, t) = C(f, t)X(f) \quad (4.2.23)$$

is also bandlimited to $|f| \leq B_i$. Therefore, (4.2.23) is equivalent to

$$Y(f, t) = G(f, t)X(f) \quad (4.2.24)$$

where $G(f, t)$ is the channel's effective transfer function with respect to the signal:

$$G(f, t) = C(f, t)H(f) \quad (4.2.25)$$

where $H(f)$ is an ideal filter with bandwidth B_i . Thus, $G(f, t)$ is also bandlimited, and its transform $g(\hat{t}, t)$ is also representable in terms of the sampling expansion:

$$g(\hat{t}, t) = \sum_{m=-\infty}^{\infty} g\left[\frac{m}{2B_i}, t\right] \text{sinc}(2B_i\hat{t} - m) \quad (4.2.26)$$

Applying (4.2.2b), we find that (4.2.24) implies

$$y(t) = \int_{-\infty}^{\infty} g(\hat{t}, t)x(t - \hat{t}) d\hat{t} \quad (4.2.27)$$

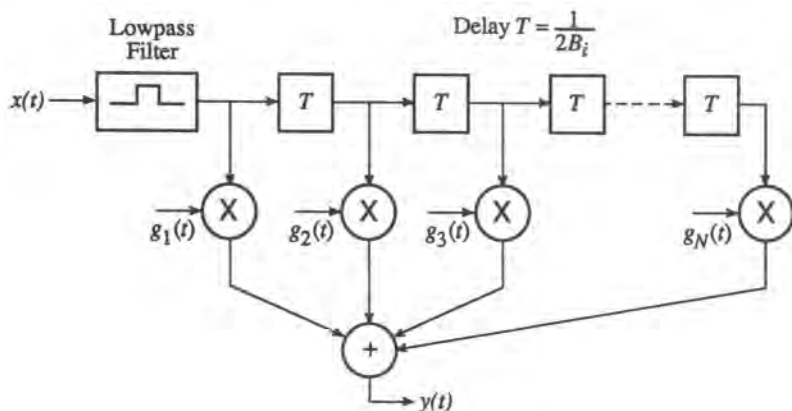


Figure 4.38. Sampling model for linear time-varying systems, in the form of a tapped delay-line with time-varying coefficients.

and upon substituting (4.2.22) and (4.2.26), it is straightforward to show that (4.2.27) is equivalent to

$$y(t) = \frac{1}{2B_i} \sum_{m=-\infty}^{\infty} g\left[\frac{m}{2B_i}, t\right] x\left[t - \frac{m}{2B_i}\right] \quad (4.2.28)$$

The representation (4.2.28) can be synthesized by a tapped delay-line with taps having time-varying gains $g_n(t) = g(n/(2B_i), t)$ and tap delay $T = (2B_i)^{-1}$, as shown in Figure 4.38.

Theoretically, the delay line should have an infinite number of taps. In practice, we use a finite length delay line at the cost of an error that can be made arbitrarily small. These basic notions underlie the tapped-delay-line random multipath channel models discussed in Chapter 9. In the random case, $c(\tau, t)$ is a random process in t and must be given a statistical characterization; the tap gains $g_n(t)$ may then be correlated.

As mentioned, the tapped delay-line model (4.2.22) forms the basis for modeling multipath radio channels. For simulation application we note that a quasistatic approach is feasible if the channel Doppler spread (fading rate) is much smaller than the signal bandwidth, i.e., $B_s \ll B_i$. In this case, the tap gains can be regarded as constant for some period of time that is nevertheless long with respect to a signaling interval (or reciprocal of the bandwidth). The simulation of the system under fading conditions can then be approximated by a series of simulations in each of which the tap gains are assumed to be constant. (See Case Study I in Chapter 12.)

4.3. Summary

In this chapter we have discussed how to go about simulating the effect of a linear system on an input signal. The synonym almost universally used for linear system is *filter*, and likewise the effect in question is commonly called *filtering*. In most systems, filters are linear and time-invariant (LTI), but they can also be linear and time-varying (LTV), as in adaptive structures. While not system elements as such, certain wireless channels are also properly modeled as (randomly) time-varying. LTI systems are much easier to deal with than LTV systems, and by far the preponderance of analytical and simulation techniques that have been developed apply to the former.

Although the modeling of a filter as such and of the filtering process will evidently be closely connected, it is useful for a clearer view of the subject to separate the two initially. Thus, we began the chapter with an outline of the ways in which LTI filters can be described or specified in practical applications and how continuous filter structures can be approximated in discrete time. Along the way, we presented the particulars for an important practical class of filters, the “classical” filters, which include the Butterworth, Chebyshev, and elliptic filters.

For modeling purposes, two major approaches present themselves, one where a filter is represented (or approximated) by a finite impulse response (FIR), and the other where the impulse response is infinite (IIR). For FIR filters, two filtering techniques are possible, referred to as time-domain and frequency-domain, respectively. Time-domain filtering is equivalent to a sliding (finite) convolution sum, and frequency-domain filtering is simply another term for using the discrete Fourier transform, normally implemented with the computationally efficient fast Fourier transform (FFT). Although it is possible to design FIR filters in the form of digital signal processors, in which case there can be no modeling error, FIR filter *models* often arise from truncating the impulse response of the actual filter. In this case errors arise both from truncation and aliasing. Both errors can be controlled (at some cost), the first by judicious use of windows, and the second by sufficiently frequent sampling.

If we wish to retain the “infinite” character of a filter’s impulse response, a different set of techniques must be used both for the filter representation and the filtering itself. Here, two basic approaches are common. In the first, it is desired to duplicate the actual impulse response, subject only to the limitation of discrete sampling. This approach is called the *impulse-invariant transformation*. It is obvious, however, that even if we have a way of calculating the impulse response samples, the infinite memory of the filter cannot be literally reproduced. If the simulation starts at $t = 0$, however, we can in principle keep going indefinitely. This can be visualized as an ever-increasing convolution sum, but this will quickly become computationally burdensome, and perhaps also problematic with respect to precision. These difficulties can be mitigated, though not completely, by expressing the filter output recursively, if this is possible. For filters whose transfer functions are rational functions of s or z , this is indeed possible, as was shown.

The second approach to simulating IIR filtering is to first model the filter through what we called a transform-domain approximation. This approach is intended to apply to filters whose transfer function is defined as a ratio of polynomials in s , or equivalently a ratio of polynomials in $1/s$. The nature of the approximation is to visualize $1/s$ as integration and replace it by a discrete integration formula. The most widely used such formula is called the *bilinear transformation*, which amounts to the trapezoidal integration formula. For such filters as we are considering, replacing $1/s$ by a ratio of polynomials in z (the bilinear transformation being a specific instance of this) leads to the implementation of the filtering process as a linear (recursive) difference equation. Standard alternative forms for realizing such filter structures (both for hardware and computation) were presented.

General hints were given for approaching the simulation of filtering both for FIR and IIR representations.

As indicated above, LTV systems are more difficult to deal with, and the options for simulating them much more limited. We provided an introductory discussion of the properties of LTV systems. From the simulation standpoint, one of the potentially more significant properties is the bandwidth expansion induced by LTV behavior, captured by the term *Doppler broadening*. Depending on the degree of bandwidth expansion relative to the data rate, this will require an increase in the simulation sampling rate. Under the reasonable assumption that the input signal is bandlimited, we also showed that an LTV system can be represented as a tapped delay-line with time-varying coefficients. This structural model is of

the most useful one for simulating LTV systems, and in Chapter 9 we shall apply it to the mobile (multipath) channel, for which case the tap coefficients become *randomly* time-varying.

4.4. Appendix: Biquadratic Factors for Classical Filters

This appendix lists biquadratic factors for normalized lowpass Butterworth filters (Table 4.A.1), normalized lowpass Bessel filters (Table 4.A.2), normalized lowpass Chebyshev filters (Table 4.A.3), and normalized lowpass elliptic filters (Table 4.A.4).

Table 4.A.1. Biquadratic Factors for Normalized Lowpass Butterworth Filters

$$H(s) = \frac{H(0)}{D(s)}; \omega_b = 1$$

n	Denominator $D(s)$
1	$s + 1$
2	$s^2 + \sqrt{2}s + 1$
3	$(s + 1)(s^2 + s + 1)$
4	$(s^2 + 0.765s + 1)(s^2 + 1.848s + 1)$
5	$(s + 1.0000)[(s + 0.3090)^2 + 0.9511^2][(s + 0.8090)^2 + 0.5878^2]$
6	$[(s + 0.2588)^2 + 0.9659^2][(s + 0.7071)^2 + 0.7071^2][(s + 0.9659)^2 - 0.2588^2]$
7	$(s + 1.0000)[(s + 0.2225)^2 + 0.9749^2][(s + 0.6235)^2 + 0.7818^2][(s + 0.9010)^2 + 0.4339^2]$
8	$[(s + 0.1951)^2 + 0.9808^2][(s + 0.5556)^2 + 0.8315^2][(s + 0.8315)^2 - 0.5556^2]$ $\times [(s + 0.9808)^2 + 0.1951^2]$
9	$(s + 1.000)[(s + 0.1737)^2 + 0.9848^2][(s + 0.5000)^2 + 0.8660^2][(s - 0.7660)^2 + 0.6428^2]$ $\times [(s + 0.9397)^2 + 0.3420^2]$
10	$[(s + 0.1564)^2 + 0.9877^2][(s + 0.4540)^2 + 0.8910^2][(s + 0.7071)^2 - 0.7071^2]$ $\times [(s + 0.8910)^2 + 0.4540^2][(s + 0.9877)^2 + 0.1564^2]$

Table 4.A.2. Biquadratic Factors for Normalized Lowpass Bessel Filters

$$H(s) = \frac{d_0}{D(s)}; \tau_0 = 1$$

n	Denominator $D(s)$
1	$s + 1$
2	$s^2 + 3s + 3$
3	$(s + 2.322)(s^2 + 3.678s + 6.460)$
4	$(s^2 + 5.792s + 9.140)(s^2 + 4.208s + 11.488)$
5	$(s + 3.6467)[(s + 3.3520)^2 + 1.7427^2][(s + 2.3247)^2 + 3.5710^2]$
6	$[(s + 4.2484)^2 + 0.8675^2][(s + 3.7356)^2 + 2.6263^2][(s + 2.5159)^2 + 4.4927^2]$
7	$(s + 4.9718)[(s + 4.7583)^2 + 1.7393^2][(s + 4.0701)^2 + 3.5172^2][(s + 2.6857)^2 + 5.4207^2]$
8	$[(s + 5.5879)^2 + 0.8676^2][(s + 2.8390)^2 + 6.3539^2][(s + 4.3683)^2 + 4.1444^2]$ $\times [(s + 5.2048)^2 + 2.6162^2]$
9	$(s + 6.2970)[(s + 6.1294)^2 + 1.7378^2][(s + 5.6044)^2 + 3.4982^2][(s + 4.6384)^2 + 5.3173^2]$ $\times [(s + 2.9793)^2 + 7.2915^2]$
10	$[(s + 6.9220)^2 + 0.8677^2][(s + 3.1089)^2 + 8.2327^2][(s + 6.6153)^2 + 2.6116^2]$ $\times [(s + 5.9675)^2 + 4.3850^2][(s + 4.8862)^2 + 6.2250^2]$

Table 4.A.3. Biquadratic Factors for Normalized Lowpass Chebyshev Filters

$$H(s) = H_0/D(s); \quad \omega_p = 1$$

n	Denominator $D(s)$
$\alpha_p = 0.5 \text{ dB } (\epsilon = 0.3493)$	
1	$s + 2.863$
2	$s^2 + 1.425s + 1.516$
3	$(s + 0.626)(s^2 + 0.626s + 1.142)$
4	$(s^2 + 0.351s + 1.064)(s^2 + 0.845s + 0.356)$
5	$(s + 0.3623)[(s + 0.1120)^2 + 1.0116^2][(s + 0.2931)^2 + 0.6252^2]$
6	$[(s + 0.0777)^2 + 1.0085^2][(s + 0.2121)^2 + 0.7382^2][(s + 0.2898)^2 - 0.2702^2]$
7	$(s + 0.2562)[(s + 0.0570)^2 + 1.0064^2][(s + 0.1597)^2 + 0.8001^2][(s - 0.2308)^2 + 0.4479^2]$
8	$[(s + 0.436)^2 + 1.0050^2][(s + 0.1242)^2 + 0.8520^2][(s + 0.1859)^2 + 0.5693^2]$ $\times [(s + 0.2193)^2 + 0.1999^2]$
9	$(s + 0.1984)[(s + 0.0345)^2 + 1.0040^2][(s + 0.0992)^2 + 0.8829^2][(s - 0.1520)^2 + 0.6553^2]$ $\times [(s + 0.1864)^2 + 0.3487^2]$
10	$[(s + 0.0279)^2 + 1.0033^2][(s + 0.0810)^2 + 0.9051^2][(s + 0.1261)^2 + 0.7183^2]$ $\times [(s + 0.1589)^2 + 0.4612^2][(s + 0.1761)^2 + 0.1589^2]$
$\alpha_p = 1 \text{ dB } (\epsilon = 0.5080)$	
1	$s + 1.965$
2	$s^2 + 1.098s + 1.103$
3	$(s + 0.494)(s^2 + 0.490s + 0.994)$
4	$(s^2 + 0.279s + 0.987)(s^2 + 0.674s + 0.279)$
5	$(s + 0.2895)[(s + 0.0895)^2 + 0.9901^2][(s + 0.2342)^2 + 0.6119^2]$
6	$[(s + 0.0622)^2 + 0.9934^2][(s + 0.1699)^2 + 0.7272^2][(s + 0.2321)^2 + 0.2662^2]$
7	$(s + 0.2054)[(s + 0.0457)^2 + 0.9953^2][(s + 0.1281)^2 + 0.7982^2][(s + 0.1851)^2 + 0.4429^2]$
8	$[(s + 0.0350)^2 + 0.9965^2][(s + 0.0997)^2 + 0.8448^2][(s + 0.1492)^2 + 0.5644^2]$ $\times [(s + 0.1759)^2 + 0.1982^2]$
9	$(s + 0.1593)[(s + 0.0277)^2 + 0.9972^2][(s + 0.0797)^2 + 0.8769^2][(s + 0.1221)^2 + 0.6509^2]$ $\times [(s + 0.1497)^2 + 0.3463^2]$
10	$[(s + 0.0224)^2 + 0.9978^2][(s + 0.1013)^2 + 0.7143^2][(s + 0.0651)^2 + 0.9001^2]$ $\times [(s + 0.1277)^2 + 0.4586^2][(s + 0.1415)^2 + 0.1580^2]$
$\alpha_p = 2 \text{ dB } (\epsilon = 0.7648)$	
1	$s + 1.308$
2	$s^2 + 0.804s + 0.637$
3	$(s + 0.402)(s^2 + 0.369s + 0.886)$
4	$(s^2 + 0.210s + 0.928)(s^2 + 0.506s + 0.221)$
5	$(s + 0.1283)[(s + 0.0675)^2 + 0.9735^2][(s + 0.1766)^2 + 0.6016^2]$
6	$[(s + 0.0470)^2 + 0.9817^2][(s + 0.1283)^2 + 0.7187^2][(s + 0.1753)^2 + 0.2630^2]$
7	$(s + 0.1553)[(s + 0.0346)^2 + 0.9867^2][(s + 0.0968)^2 + 0.7912^2][(s + 0.1399)^2 + 0.4391^2]$
8	$[(s + 0.0265)^2 + 0.9898^2][(s + 0.0754)^2 + 0.8391^2][(s + 0.1129)^2 + 0.5607^2]$ $\times [(s + 0.1332)^2 + 0.1969^2]$
9	$(s + 0.1206)[(s + 0.0209)^2 + 0.9919^2][(s + 0.0603)^2 + 0.8723^2][(s + 0.0924)^2 + 0.6474^2]$ $\times [(s + 0.1134)^2 + 0.3445^2]$
10	$[(s + 0.0170)^2 + 0.9935^2][(s + 0.0767)^2 + 0.7113^2][(s + 0.0493)^2 + 0.8962^2]$ $\times [(s + 0.0967)^2 + 0.4567^2][(s + 0.1072)^2 + 0.1574^2]$

Table 4.A.4. Biquadratic Factors for Normalized Lowpass Elliptic Filters^a

$$H(s) = C \frac{N(s)}{D(s)}; \omega_p = 1$$

n	α_r	Constant C	Numerator $N(s)$	Denominator $D(s)$
$\alpha_p = 0.5\text{dB}$				
(A) $\omega_r/\omega_p = 1.1$				
2	2.3	$7.6262 \cdot 10^{-1}$	$s^2 + 1.71408$	$s^2 + 0.42987s + 1.38465$
3	8.5	$9.0441 \cdot 10^{-1}$	$s^2 + 1.37031$	$(s^2 + 0.20787s + 1.14221)$ $\times (s + 1.112286)$
4	17.6	$1.3176 \cdot 10^{-1}$	$(s^2 + 1.2909)(s^2 + 4.34993)$	$(s^2 + 0.12448s + 1.04556)$ $\times (s^2 + 0.99508s + 0.74958)$
5	27.2	$1.6244 \cdot 10^{-1}$	$(s^2 + 1.25932)(s^2 + 2.19309)$	$(s^2 + 0.08164s + 1.02258)$ $\times (s^2 + 0.49230s + 0.76553)$ $\times (s + 0.57311)$
(B) $\omega_r/\omega_p = 1.5$				
2	8.3	$3.8540 \cdot 10^{-1}$	$s^2 + 3.92705$	$s^2 + 1.03153s + 1.60319$
3	21.9	$3.1410 \cdot 10^{-1}$	$s^2 + 2.80601$	$(s^2 + 0.45286s + 1.14917)$ $\times (s + 0.766952)$
4	36.3	$1.5397 \cdot 10^{-2}$	$(s^2 + 2.53555)(s^2 + 12.09931)$	$(s^2 + 0.25496s + 1.06044)$ $\times (s^2 + 0.92001s + 0.47183)$
5	50.6	$1.9197 \cdot 10^{-2}$	$(s^2 + 2.42551)(s^2 + 5.43764)$	$(s^2 + 0.16346s + 1.03189)$ $\times (s^2 + 0.57023s + 0.57601)$ $\times (s + 0.42597)$
(C) $\omega_r/\omega_p = 2.0$				
2	13.9	$2.0133 \cdot 10^{-1}$	$s^2 + 7.4641$	$s^2 + 1.24504s + 1.59179$
3	31.2	$1.5424 \cdot 10^{-1}$	$s^2 + 5.15321$	$(s^2 + 0.53787s + 1.14849)$ $\times (s + 0.69212)$
4	48.6	$3.6987 \cdot 10^{-3}$	$(s^2 + 4.59326)(s^2 + 24.22720)$	$(s^2 + 0.30116s + 1.06258)$ $\times (s^2 + 0.88456s + 0.41032)$
5	66.1	$4.6205 \cdot 10^{-3}$	$(s^2 + 4.36495)(s^2 + 10.56773)$	$(s^2 + 0.19255s + 1.03402)$ $\times (s^2 + 0.58054s + 0.52500)$ $\times (s + 0.392612)$
$\alpha_p = 1.0\text{dB}$				
(A) $\omega_r/\omega_p = 1.1$				
2	4.02	$6.2911 \cdot 10^{-1}$	$s^2 + 1.71408$	$s^2 + 0.45820s + 1.20993$
3	11.48	6.2086	$10^{-1} s^2 + 1.37031$	$(s^2 + 0.19530s + 1.04241)$ $\times (s + 0.81616)$
4	20.83	$9.0870 \cdot 10^{-2}$	$(s^2 + 1.29092)(s^2 + 4.34993)$	$(s^2 + 0.10896s + 1.00968)$ $\times (s^2 + 0.79846s + 0.56704)$
5	30.47	$1.1151 \cdot 10^{-1}$	$(s^2 + 1.25932)(s^2 + 2.19309)$	$(s^2 + 0.06924s + 1.00164)$ $\times (s^2 + 0.40429s + 0.68854)$ $\times (s + 0.44656)$
(B) $\omega_r/\omega_p = 1.5$				
2	11.19	$2.7562 \cdot 10^{-1}$	$s^2 + 3.92705$	$s^2 + 0.87941s + 1.2144$
3	25.17	$2.1562 \cdot 10^{-1}$	$s^2 + 2.80601$	$(s^2 + 0.37539s + 1.02371)$ $\times (s + 0.59101)$
4	39.5	$1.0570 \cdot 10^{-2}$	$(s^2 + 2.53555)(s^2 + 12.09930)$	$(s^2 + 0.20881s + 0.99881)$ $\times (s^2 + 0.72998s + 0.36428)$
5	53.9	$1.3178 \cdot 10^{-2}$	$(s^2 + 2.42551)(s^2 + 5.43764)$	$(s^2 + 0.13308s + 0.99495)$ $\times (s^2 + 0.45774s + 0.51706)$ $\times (s + 0.33784)$

Table 4.A.4 (continued)

n	α_r	Constant C	Numerator $N(s)$	Denominator $D(s)$
			(C) $\omega_r/\omega_p = 2.0$	
2	17.09	$1.3971 \cdot 10^{-1}$	$s^2 + 7.46410$	$s^2 + 0.99894s + 1.17007$
3	34.45	$1.0589 \cdot 10^{-1}$	$s^2 + 5.15320$	$(s^2 + 0.43406s + 1.01059)$ $\times (s + 0.53995)$
4	51.9	$2.5391 \cdot 10^{-3}$	$(s^2 + 4.59326)(s^2 + 24.2271)$	$(s^2 + 0.24295s + 0.99322)$ $\times (s^2 + 0.70254s + 0.31919)$
5	69.36	$4.5634 \cdot 10^{-5}$	$(s^2 + 4.36495)(s^2 + 10.56773)$	$(s^2 + 0.15524s + 0.99190)$ $\times (s^2 + 0.46467s + 0.47186)$ $\times (s + 0.31259)$

"Note: $\alpha_p = 10 \log(1 + \epsilon^2)$, $\alpha_r = 10 \log(A^2)$, for ϵ and A as defined in Figure 4.5. Extensive tables can be found in the literature—e.g., K. L. Su, *Handbook of Tables for Elliptic-Function Filters*, Kluwer Academic Press, Dordrecht (1990).

References

1. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1975).
2. A. V. Oppenheim and R. W. Schaffer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1977).
3. B. Gold and C. M. Rader, *Digital Processing of Signals*, McGraw-Hill, New York (1969).
4. A. Antoniou, *Digital Filters: Analysis and Design*, McGraw-Hill, New York (1979).
5. A. Papoulis, *Signal Analysis*, McGraw-Hill, New York (1977).
6. A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*, Prentice-Hall, Englewood Cliffs, New Jersey (1983).
7. E. Christian and E. Eisenmann, *Filter Design Tables and Graphs*, Wiley, New York (1966).
8. M. E. Van Valkenburg, *Analog Filter Design*, Holt, Rinehart and Winston, New York (1982).
9. M. S. Ghaussi and K. R. Laker, *Modem Filter Design*, Prentice-Hall, Englewood Cliffs, New Jersey (1981).
10. A. Papoulis, *The Fourier Integral and Its Applications*, McGraw-Hill, New York (1962).
11. R. S. Blum and M. C. Jeruchim, A note on windowing in the simulation of continuous-time communication systems, *IEEE Trans. Commun.* **45**(8), 889–892 (1997).
12. T. W. Parks and J. H. McClellan, Chebyshev approximation of nonrecursive digital filters with linear phase, *IEEE Trans. Circuit Theory* **CT-19**, 189–194 (1972).
13. A. Peled and B. Liu, *Digital Signal Processing: Theory, Design, and Implementation*, Wiley, New York (1976).
14. J. B. Knowles and E. M. Olcayto, Coefficient accuracy and digital filter response, *IEEE Trans. Circuit Theory* **CT-15**, 31–41 (1968).
15. T. Kailath, Channel characterization: Time-variant dispersive channels, in *Lectures on Communication System Theory*, E. J. Baghdady (ed.), McGraw-Hill, New York (1961).
16. R. J. Schwarz and B. Friedland, *Linear Systems*, McGraw-Hill, New York (1965).
17. W. Kaplan, *Operational Methods for Linear Systems*, Addison-Wesley, Reading, Massachusetts (1962).
18. L. Schwartz, *Mathematics for the Physical Sciences*, Addison-Wesley, Reading, Massachusetts (1966).
19. L. A. Zadeh, Frequency analysis of variable networks, *Proc. IRE* **38**, 291–299 (1950).
20. J. B. Cruz and M. E. Van Valkenburg, The synthesis of models for time-varying linear systems, in *Proceedings of the Symposium on Active Networks and Feedback Systems*, Polytechnic Press, Polytechnic Institute of Brooklyn, New York (1960), pp. 527–544.
21. H. D'Angelo, *Linear Time-Varying Systems: Analysis and Synthesis*, Allyn and Bacon, Boston (1970).

This page intentionally left blank

Modeling and Simulation of Nonlinear Systems

Perhaps the single greatest justification for the simulation approach to the study of communication systems is the presence of nonlinear elements. In principle, the performance of linear systems can be attacked by analytical means, but the study of nonlinear systems by such means is by and large intractable. Although there have been many analytical studies of one or another nonlinear device, the system context is usually idealized or much simplified compared to realistic scenarios. In addition, it is typical for such analytical formulations to be of such complexity as to require numerical evaluation, a situation which negates the values of analysis—insight and generality. Hence, simulation is generally the appropriate tool for perhaps most nonlinear communication systems because simulation of such systems is no more difficult than for linear systems, given the model. Indeed, the main difficulty in nonlinear system simulation is obtaining the model itself. Hence, much of the focus in this chapter will be on modeling methodology and on associated measuring techniques for determining parameters of models.

In general, a nonlinear system consists of linear and nonlinear elements. Some of the linear elements may be described by linear differential equations, as discussed in the previous chapter, or by other relationships. The nonlinear elements, which are normally relatively few in number, are described by nonlinear functions or nonlinear differential equations relating the input and output of the element. The transform methods described in the previous chapters cannot strictly be applied to nonlinear systems since superposition does not hold in the latter case. The nonlinear system, therefore, generally has to be simulated in the time domain. This applies to the nonlinear part of the system only. The rest of the system can be simulated either in the time domain or the frequency domain (see Figure 5.1), except when it is part of a feedback system. In contrasting linear and nonlinear system simulation, we should make another point. In the linear case, we are generally satisfied that we will tend to the correct answer under certain limiting conditions (assuming no finite-precision effects), such as letting the sampling interval become smaller and smaller or increasing the duration of a (truncated) finite impulse response. In nonlinear systems, there is no clear set of limiting conditions under which we can assert that the model comes closer to reality, because it is so much more difficult to ascertain the closeness of a nonlinear model to the functioning of the actual device. This reinforces the point made in Chapter 2 about the need to validate simulation models against hardware counterparts.

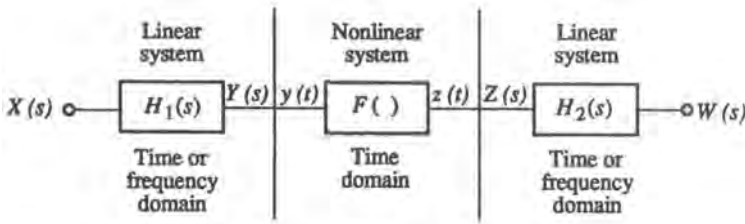


Figure 5.1. Simulation of a nonlinear communication system.

5.1. Modeling Considerations for Nonlinear Systems

Usually devices used in communication systems such as traveling-wave tube amplifiers (TWTAs) and solid-state power amplifiers (SSPAs) are described by analytical models that typically involve solving a set of simultaneous, nonlinear partial differential equations by numerical methods. Unless some suitable simplifications can be made (and this is sometimes possible, as in the optical amplifier discussed later), such detailed models are far too time-consuming to be useful in a system-level simulation where the nonlinear device is just one of many subsystems. Hence, a higher level model is needed, one that converts an input waveform to (nearly) the correct output waveform without necessarily resorting to the fundamental physics of the device. For linear systems, the transfer function is such a model.

In order to speak about nonlinear system models, it will be useful to introduce a number of model-type descriptors:

1. Memoryless models
2. Models with memory
3. Baseband models
4. Bandpass models
5. Block (input/output) models
6. Analytical models
7. Nonlinear differential equation models

These terms capture one aspect or another of the model, but they are not all intended to be mutually exclusive. For example, a model can be both a bandpass and a memoryless model, while its mode of description is an analytical form. Evidently, some of the classifiers *are* mutually exclusive: a model is either of the baseband or bandpass variety, and a model is either memoryless or with memory. We now outline some of these distinctions.

Perhaps the most significant categorical distinction is between a memoryless (or zero-memory) nonlinearity and those with memory. For future reference we shall sometimes abbreviate memoryless models as ZMNL (for zero-memory nonlinearity) and those with memory as NLWM (for nonlinearity with memory). We will also use the term *instantaneous* more or less interchangeably with memoryless. The term “memoryless” implies that the output of a device is a function of the input signal at the present instant only, or perhaps a function of the input at some fixed delay. This implies also that the transfer characteristics are frequency independent. These models are very popular and they fairly accurately characterize a great variety of devices, generally under the condition that the input signal have a bandwidth sufficiently smaller than that of the device. We will discuss memoryless models and different ways of developing them in Section 5.2. In that section we will also examine the primary

consequence of dealing with nonlinearities in the simulation context, which is a required increase in the simulation sampling rate in order to maintain a desired aliasing error.

Memoryless models are an idealization: no physical device is truly (input) frequency independent. Rather, as the bandwidth of an input signal increases, we can expect filtering effects to become more manifest until they are clearly visible. If we allow ourselves only this modeling dichotomy, the decision whether to use a memoryless model or one with memory lies in the art, rather than the science of simulation. Models with memory are considerably more difficult to obtain and, as one would expect, more computationally demanding. Several approaches to developing such models will be elaborated in Section 5.3.

The designations *baseband* and *bandpass* simply refer to the nature of the input signal, although there are generally consequent impacts on the structure of the nonlinearity model itself. An important distinction in this regard is that while the term *bandpass* connotes a modulated carrier, the term *bandpass model* implies one that connects directly the *complex envelopes* of the input and output carriers, as defined in Chapter 3.[†] This, of course, is the type of model called for if the purpose of the simulation is to extract a performance estimate. Furthermore, in this case, a model that effectively establishes a transfer characteristic for the complex envelope is clearly the most computationally efficient. Not all possible models need be of this type. The term *baseband* model here is intended to connote merely that the input signal's energy is concentrated around zero frequency and we are interested in the nonlinearity's output also in that general vicinity.

Since the goal of communication system simulation is typically to obtain a performance estimate, we are primarily interested in complex envelope models (for bandpass inputs). However, in system design we are also interested in estimating harmonic levels that might be generated, for example, by mixers and limiters. These levels are generally constrained by regulatory agencies. By construction, however, complex envelope models cannot deal with harmonic generation. Different types of models are required for this purpose. While it is not within our intended scope to deal with these models, we shall comment on them at appropriate places.

Returning to our categorization, the descriptors *block models*, *analytical models*, and *nonlinear differential equations* all basically refer to the structure of models—*how* the models are constructed. Any one of these modalities can be combined with any of the other model descriptors. A block model is one that consists of a set of “blocks” (in the sense of block diagram) connected in series or parallel or a combination thereof. While an analytical model can be presented in block form, the distinction we make here is that the blocks in a block model consist of conventional system elements like filters and memoryless nonlinearities. An analytical model is one based on an analytic construct, which may be, for example, a postulated functional form relating input and output. An analytical model may be memoryless or implicitly with memory: power series are of the first type, while Volterra series are examples of the second. Nonlinear differential equations (NLDEs) are, of course, analytical descriptions, and imply a model with memory. Because NLDEs are an important special case, we shall discuss them separately in Section 5.4.

Virtually any generic type of model will have parameters to be specified or some aspect of its behavior to be determined before it can be applied in simulation to a specific device or class of devices. These undetermined properties must in practice be obtained from measurements which may or may not be simple to make. Some of the model constructions

[†]Generally, the input and output carrier frequencies need not be the same.

require nonconventional measurements. Hence the practicality of one type of model or another may well hinge on the implied measurements. Thus, in Section 5.5 we spend a little time reviewing these measurement aspects.

Finally, we make the point that we have variously spoken of nonlinear “devices,” “blocks,” “subsystems,” etc. While we are interested in characterizing the most irreducible of these elements, which we usually call a device or a component, the constraints of real hardware may make the terminals available for measurement those of a larger assembly of components. In fact, the methodology associated with constructing models based on measurements, which we will discuss later, does allow us to do so irrespective of what lies between the terminals. Thus, at least in principle, we have the attractive possibility (from a complexity reduction viewpoint) of synthesizing a model for any nonlinear system. However, we might expect that the larger the aggregation between the terminals, the less reliable the model is likely to become.

5.2. Memoryless Nonlinearities

In this section we look at various aspects of modeling memoryless nonlinearities. First we discuss baseband nonlinearities, following which we consider the (generally) bandwidth-expanding effects of a memoryless nonlinearity. In the subsections following, we expose two approaches for developing complex envelope models. The first of these is based on an analytical construction, and the resulting type of model is often referred to as a *bandpass nonlinearity*. The second type of bandpass model we shall refer to as a *bandpass amplifier*, whose modeling evolves from empirical observations. This latter type of model serves as the springboard for some of the models with memory addressed in Section 5.3.

5.2.1. Memoryless Baseband Nonlinearities

The model for a memoryless baseband nonlinearity is characterized by a simple functional relationship of the form

$$y(t) = F[x(t)] \quad (5.2.1)$$

Recall that our definition of a baseband nonlinearity simply means that $x(t)$ is a baseband signal, meaning that its power (energy) is spectrally concentrated around zero frequency, and generally this will also be true of $y(t)$. Although the operator F is independent of that fact, we will for convenience refer to it as a baseband nonlinearity in this context. A relationship such as (5.2.1) is used for most solid-state devices.

Certain nonlinear characteristics can be given in analytical form. An example of such a baseband nonlinearity is a diode. The typical volt–ampere characteristic is

$$I = I_s(e^{\lambda V} - 1) \quad (5.2.2)$$

Here I_s is a device parameter and $\lambda = q/kT$, where q is the charge of an electron, k is Boltzmann’s constant, and T is the absolute temperature. The exponential in (5.2.2) can be expanded into a power series, which we can suppose could be truncated after a moderate number of terms with ensuing acceptable error. This suggests that the function F might also be representable by a power series, or by an orthogonal function expansion, or perhaps

reasonably well approximated over the range of $x(t)$ by a polynomial in x . For now we will assume the latter, as it will facilitate later discussion. Thus we can write

$$y(t) = F[x(t)] \approx \sum_{n=0}^N a_n x^n(t) \quad (5.2.3)$$

In a practical application, the coefficients a_n would be obtained by fitting a polynomial of degree N to a measured characteristic. It may be that to obtain a good fit would require a large value of N , which would reduce the efficiency of such a model. An alternative, in this case, might be to try a fit using a different type of fitting function. Often, the most efficient and accurate model is to use the raw experimental data themselves in a lookup table, using appropriate interpolation between tabulated data points.

We note that the complex envelope can be interpreted as a (complex) baseband signal. Hence, with an appropriate interpretation of F (which can vary, depending upon the nature of the nonlinearity), most statements that can be made for baseband signals can be suitably carried over to complex envelopes.

5.2.2. Estimating the Sampling Rate for Nonlinear Systems

Given a nonlinear model, the most important implication in simulation is that the sampling rate must be increased. This is because a nonlinearity generally (but not always) increases the bandwidth of a signal, that is, the output bandwidth is greater than the input bandwidth. Since bandwidth is uniquely defined only for a bandlimited signal, let us suppose that the input $x(t)$ is bandlimited to $\pm B/2$. To illustrate the point more easily, let us assume that (5.2.3) holds and that $x(t)$ has finite energy, so that it has a Fourier spectrum $X(f)$. Then, it is easy to see (Problem 5.1) that

$$Y(f) = a_0 \delta(f) + \sum_{n=1}^N a_n [X(f) \overset{n-1}{*} X(f)] \quad (5.2.4)$$

where $\overset{n-1}{*}$ indicates $(n-1)$ -fold convolution. Hence the term $x^n(t)$ has bandwidth nB (i.e., $\pm \frac{1}{2} nB$).

The situation is not so straightforward for finite-power random processes. But the following result illustrates the theme. For a Gaussian random process $X(t)$ with power spectral density $S_X(f)$ and a large class of nonlinearities, it can be shown⁽¹⁾ that the output PSD is given by

$$S_Y(f) = \sum_{k=1}^{\infty} (b_k^2 / \sigma^2) [S_X(f) \overset{k-1}{*} S_X(f)] \quad (5.2.5)$$

where σ^2 is the average power of $X(t)$ and $\{b_k\}$ is a square-summable sequence of coefficients. Again, if we consider $X(t)$ to be essentially bandlimited to a bandwidth B , the k th term in (5.2.5) has a bandwidth equal to kB .

As we know, the sampling rate for any bandlimited system has to be at least twice the bandwidth in order not to introduce aliasing distortion. Thus, in order to avoid aliasing error for a nonlinear system such as that given by (5.2.3), it would seem that we must have $f_s > 2NB$. This is really overstating the difficulty, however, for two reasons. First, for many

nonlinear systems the coefficients a_n decrease with n . Thus, the contribution of $a_n x^n(t)$ to the energy of $y(t)$ becomes relatively minor for the larger values of n . Hence, the aliasing error in this component alone would make a comparatively smaller contribution to the overall error. Second, the magnitude of the spectrum of $x^n(t)$ is not uniformly distributed over nB . Indeed, as n increases, we can generally expect the spectrum of $x^n(t)$ to be increasingly concentrated around zero frequency and to decrease more rapidly toward zero value near the band edge. Hence, if the sampling rate is less than $2NB$, only the relatively low valued tails of the spectrum are aliased, and this error may well be quite small. The point is that an N -fold increase in sampling rate is almost certainly not required, but *some* increase in sampling rate will be required to keep the aliasing error at an acceptably low level.

Because a realistic nonlinear system will not in general be representable by such a simple form as (5.2.3), or if it could, it would probably not be manifest on the surface, the determination of a proper sampling rate can best be made experimentally, by running some relatively short test simulations with different sampling rates. This, of course, applies not only to the memoryless models discussed here, but to those with memory as well. In fact, at least for the variety of NLWM models that consist of cascades of filters and memoryless nonlinearities (Section 5.3.1, for example), it is clear that the bandwidth expansion takes place in the memoryless nonlinearities, for filters cannot do so.

To anticipate results which will become clear in the next section, we note that the output of a nonlinear system with a bandpass input is more complicated and includes, in addition to spreading of the bandwidth of the signal, spectral components centered around the frequencies $\pm n f_c$, $n = 0, 1, \dots$ (Figure 5.2), where f_c is the carrier frequency. In the majority of applications in which we are interested the components of the output spectrum center around $\pm f_c$. This partial signal, usually called the first or principal zone output, is obtained by filtering the output with a bandpass filter. This will be discussed in detail shortly.

The increase in the output bandwidth is a phenomenon sometimes referred to as *spectral regrowth*. It was apparently first taken note of in conjunction with satellite applications.⁽²⁾ In that context, the final power amplification, which was done with a TWT operated into saturation, was seen to raise the level of adjacent channel interference well above the preamplified levels.

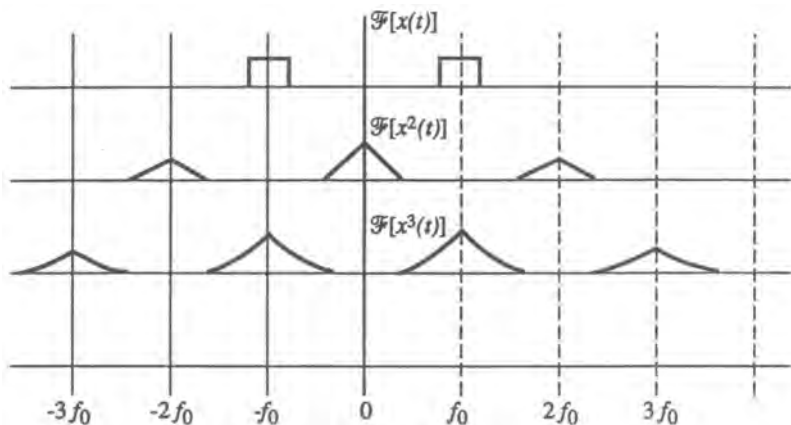


Figure 5.2. Spectral components of the function $x^n(t)$; $\mathcal{F}[\cdot]$ is the Fourier transform.

We add a final note here that nonlinearities do not always increase bandwidth. In fact, they can increase, decrease, or leave unchanged the bandwidth of an input signal, depending on the nonlinearity and the process. A simple example is a random binary waveform input to a square-law device. Here the spectrum collapses to a spectral line at zero frequency.

5.2.3. Memoryless Bandpass Nonlinearities: Analytically Based Models

We define a bandpass nonlinearity as one whose output spectrum is concentrated around a carrier frequency if the input spectrum is itself concentrated around a carrier frequency. A standard model (Figure 5.3) for a bandpass nonlinearity is a memoryless nonlinearity followed by a filter, usually called a *zonal* filter. In much of the work involving nonlinearities one often starts with a postulated instantaneous characteristic which relates the input voltage $x(t)$ to the output voltage $y(t)$, as in (5.2.1), which we repeat here:

$$y(t) = F[x(t)] \quad (5.2.1a)$$

where F is an arbitrary memoryless operator. Consider now an input bandpass signal

$$x(t) = A(t) \cos[2\pi f_c t + \theta(t)] \quad (5.2.6)$$

and make the substitution

$$\alpha = 2\pi f_c t + \theta(t) \quad (5.2.7)$$

so that, considered as a function of α , the nonlinearity output (suppressing t for the moment)

$$z = F(A \cos \alpha)$$

is a periodic function and hence can be expanded in a Fourier series

$$z = a_0 + \sum_{k=1}^{\infty} (a_k \cos k\alpha + b_k \sin k\alpha) \quad (5.2.8)$$

Recalling the definition of α , we see that Equation (5.2.8) makes explicit the *harmonics* of the carrier implied by the sketch of Figure 5.2. Hence, although the model so far is somewhat simplistic, we can see the possibility of predicting harmonic levels with a realistic model for F and a model for a filter characteristic at each harmonic kf_c . As will be seen below, the coefficients $\{a_k\}$ and $\{b_k\}$ are generally functions of time.

For performance evaluation, we are usually interested in the *first-zone* output of the memoryless nonlinearity: the first zone is that existing near the band of the operating frequency f_c and is *defined* as the $k = 1$ term in (5.2.8). An implicit assumption in retaining only this term is that the terms centered around kf_c for $k > 1$ are sufficiently removed in

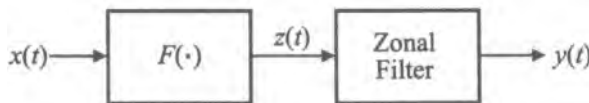


Figure 5.3. Block diagram representation of bandpass nonlinearity.

frequency that their contributed spectral energy to the region around f_c is completely negligible. The coefficients in (5.2.8) are given by the standard formulas, for $k \geq 1$ (we can ignore the dc component in this application),

$$a_k \equiv g_{k1}(A) = \frac{1}{\pi} \int_0^{2\pi} F(A \cos \alpha) \cos k\alpha \, d\alpha \quad (5.2.9a)$$

and

$$b_k \equiv g_{k2}(A) = \frac{1}{\pi} \int_0^{2\pi} F(A \cos \alpha) \sin k\alpha \, d\alpha \quad (5.2.9b)$$

For $k = 1$ we will use the abbreviations $g_{11} = g_1$ and $g_{12} = g_2$. Thus, reinserting (5.2.7) and making time explicit, we can write the first-zone output $y(t)$ as

$$y(t) = g_1[A(t)] \cos[2\pi f_c t + \theta(t)] + g_2[A(t)] \sin[2\pi f_c t + \theta(t)] \quad (5.2.10)$$

The complex function

$$N(A) = \frac{1}{A} [g_1(A) + jg_2(A)]$$

has long been known in the control systems literature as the describing function.⁽³⁾ The coefficient $a_1 \equiv g_1$ has also been called the (first-order) *Chebyshev transform* of $F(x)$ by Blachman,^(4,5) who also considered the inversion of $g_1(A)$. (See also Problem 5.4.)

Clearly the complex envelope of the first-zone output is given by

$$\tilde{y}(t) = \{g_1[A(t)] - jg_2[A(t)]\}e^{j\theta(t)} \quad (5.2.11)$$

so that, while the full output of the nonlinearity does not have a complex envelope representation, the output of the first zone, and indeed any zone, does have, which makes the use of such a model possible in simulations based on the usual lowpass equivalents.

Equation (5.2.10) implies that the signal can undergo both amplitude and phase distortions. In fact, it can be shown that for any “ordinary” function F the coefficient g_2 will *always* be zero (Problem 5.3). Hence the model under consideration leads to only the first term of (5.2.10),

$$y(t) = g_1[A(t)] \cos[2\pi f_c t + \theta(t)] \quad (5.2.12)$$

and thus can produce only amplitude distortion. Similarly, the complex envelope reduces to the first term of (5.2.11), namely

$$\tilde{y}(t) = g_1[A(t)]e^{j\theta(t)} \quad (5.2.13)$$

which is the form in which it would be implemented in simulation. This is clearly a “time-domain” model, which has a straightforward implementation. Given the (sampled) complex envelope sequence $\{A(kT_s), \theta(kT_s)\}$, the output complex envelope sequence is simply $\{g_1[A(kT_s)], \theta(kT_s)\}$.

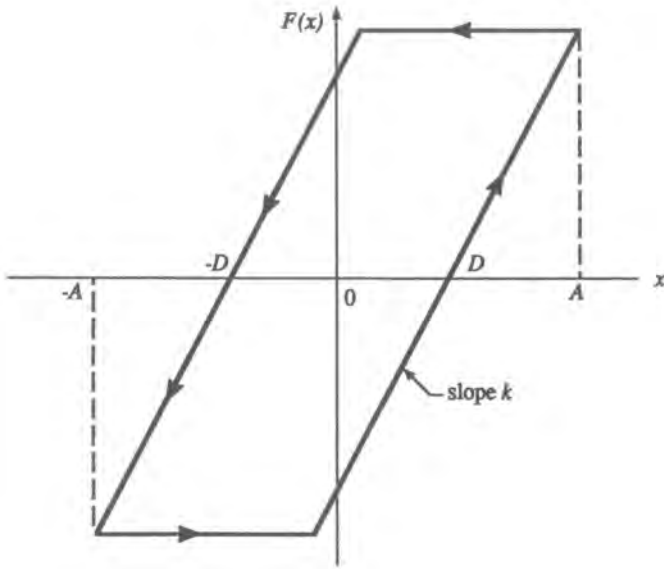


Figure 5.4. A nonlinearity with hysteresis.

In order to retain both terms of (5.2.10) using this general modeling approach, we have to invoke more “unusual” nonlinearities such as the hysteresis characteristic shown in Figure 5.4. Such a nonlinearity might represent certain types of magnetic memories or nonlinear dielectrics. It can be argued that this characteristic cannot properly be labeled “memoryless,” though in this instance the term “instantaneous” would be apt. Indeed the notation $F(x)$ when used in conjunction with (5.2.9) must be interpreted in accordance with the arrow convention in Figure 5.4. That is, one must “travel” around the loop in a time-dependent manner, which means that over a cycle, the nonlinearity output takes on the different values dictated by the figure. For this particular nonlinearity, one has (Problem 5.6)

$$\frac{g_1(A)}{A} = \frac{k}{\pi} \left\{ \frac{\pi}{2} + \sin^{-1} \left(1 - \frac{2D}{A} \right) + 2 \left(1 - \frac{2D}{A} \right) \left[\frac{D}{A} \left(1 - \frac{D}{A} \right) \right]^{1/2} \right\} \quad (5.2.14a)$$

$$\frac{g_2(A)}{A} = \frac{4kD}{\pi A} \left(1 - \frac{D}{A} \right) \quad (5.2.14b)$$

for $A \geq D$. Many more instances of describing functions can be found in the literature, e.g., Ref. 3.

For arbitrary instantaneous nonlinearity $F(x)$ the envelope transfers g_1 and g_2 are obtained in general from (5.2.9). In specific cases, however, it may be possible to proceed in a more direct fashion to obtain the first-zone output. In the next two subsections we consider first the family of limiter characteristics for which we will have to use (5.2.9) to obtain g_1 , and in the subsection following we consider the “power series” model[†] of (5.2.3), which yields results more directly.

[†]This term is somewhat of a misnomer, but is appropriately suggestive.

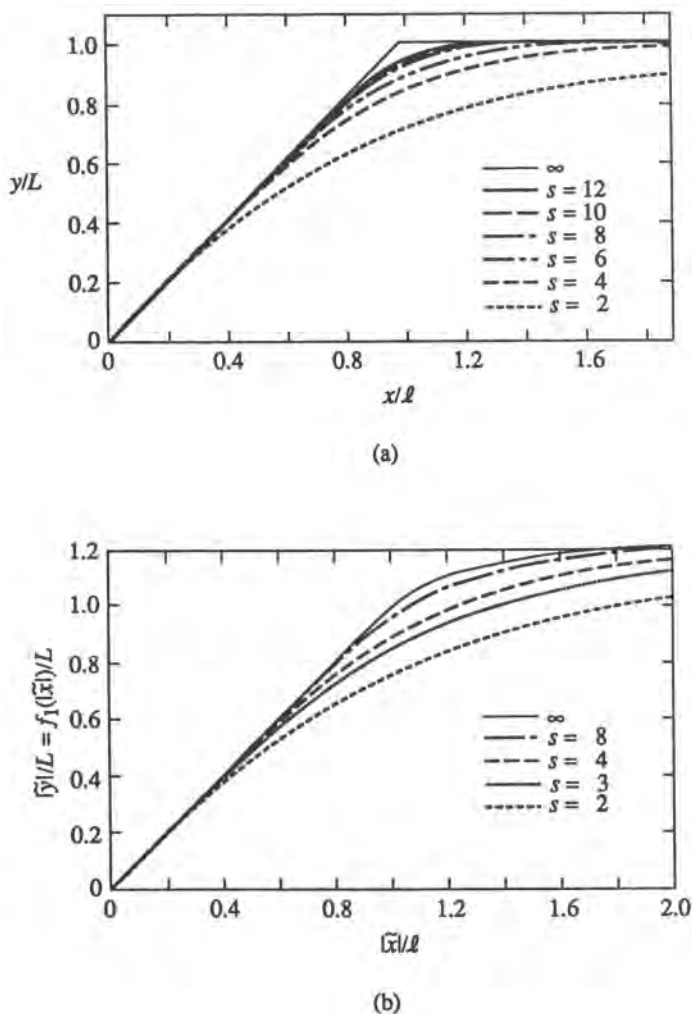


Figure 5.5. Transfer characteristics of lowpass and bandpass limiters. (a) Transfer characteristics of a limiter family; (b) first-zone output for a bandpass limiter. (From Ref. 6, ©IEEE, 1980.)

5.2.3.1. The Limiter Family

A class of memoryless nonlinear devices which can be analyzed using the previous development is the limiter family, members of which are used in a wide variety of communication circuits both at baseband and carrier frequency. The following useful parametric form has been suggested⁽⁶⁾ for the memoryless nonlinearity:

$$y = F(x) = \frac{L \operatorname{sgn}(x)}{[1 + (l/z)^s]^{1/s}} \quad (5.2.15)$$

This family has a great deal of flexibility due to the parameter s . Several examples are shown in Figure 5.5a. Equation (5.2.15) can be considered to describe a baseband limiter. That is, the

instantaneous output y is as given when x is a baseband input. In (5.2.15) the symbols have the following meaning: $z = |x|$; L is the asymptotic output level as $z \rightarrow \infty$, and also serves to scale the output; l is the input “limit” level; and s is the “knee sharpness” parameter. Equation (5.2.15) reduces to familiar forms when the parameters are set to the proper values, as will be seen shortly.

A bandpass limiter consists of a limiter per se, followed by a bandpass (zonal) filter, in accordance with the preceding construct. Hence, with $x(t)$ as in (5.2.6), the output is given by the first term on the right side of (5.2.10). The phase is unaffected, but the output envelope, which is the modulus of the complex envelope, $|\tilde{y}|$, is given by

$$|\tilde{y}| = g_1(|\tilde{x}|) \quad (5.2.16)$$

where $|\tilde{x}|$ is the input envelope [the same as $A(t)$]. Here g_1 is calculated from (5.2.9) with F defined by (5.2.15). For the set of F characteristics shown in Figure 5.5a, the corresponding Chebyshev transforms g_1 are given in Figure 5.5b.

Special Cases. (a) Hard-limiter: Here, one only needs to set $l = 0$; s can be arbitrarily set, but for convenience could be set to unity. The resulting transfer characteristics are (Problem 5.7)

$$y = L \operatorname{sgn}(x), \quad \text{baseband} \quad (5.2.17a)$$

$$|\tilde{y}| = \frac{4L}{\pi}, \quad \text{bandpass} \quad (5.2.17b)$$

(b) Soft-limiter or limiting amplifier: Here we let $s \rightarrow \infty$, which results in

$$\text{baseband} \quad \begin{cases} y = L \operatorname{sgn}(x), & z > l \\ y = (L/l)x, & z < l \end{cases} \quad (5.2.18a)$$

$$\text{bandpass} \quad \begin{cases} |\tilde{y}| = g_1(|\tilde{x}|), & |\tilde{x}| > l \\ |\tilde{y}| = (L/l)|\tilde{x}|, & |\tilde{x}| < l \end{cases} \quad (5.2.18b)$$

where g_1 is given graphically in Figure 5.5b and explicitly by the following (Problem 5.10), with $A = |\tilde{x}|$:

$$\frac{g_1(A)}{A} = \frac{2L}{\pi l} \left[\sin^{-1} \frac{l}{A} + \frac{l}{A} \left(1 - \frac{l^2}{A^2} \right)^{1/2} \right], \quad A \geq l \quad (5.2.19a)$$

$$\frac{g_1(A)}{A} = \frac{L}{l}, \quad A < l \quad (5.2.19b)$$

It is clear that (5.2.17) are special cases of (5.2.18) as $l \rightarrow 0$ in the latter. The particular form (5.2.18) has a sharp knee, but by choosing a finite value of s , we can synthesize a characteristic with rounded knee, as is clear from Figure 5.5. We note that the family (5.2.15) can be used for nonlinearities other than limiters. In particular, the nature of the curves in Figure 5.5 is due to the fact that $s > 1$: other types can be created for $s < 1$. Finally, we note that practical limiters do have some amplitude-to-phase conversion, but to accommodate that behavior, we need a more general model.

5.2.3.2. Power Series Model

As mentioned before, this is a frequently assumed model. It has the nice property of being able to expose relatively clearly the effect that it has on an input signal. Again, we repeat the model, from (5.2.3):

$$y(t) = F[x(t)] \approx \sum_{n=0}^N a_n x^n(t)$$

with the same input signal as in (5.2.6). Expressing the latter in terms of the complex envelope

$$\begin{aligned} x(t) &= \text{Re}[\tilde{x}(t)e^{j2\pi f_c t}] \\ &= \frac{1}{2}[\tilde{x}(t)e^{j2\pi f_c t} + \tilde{x}^*(t)e^{-j2\pi f_c t}] \end{aligned} \quad (5.2.20)$$

and using the binomial expansion for $x^n(t)$, we obtain

$$\begin{aligned} x^n(t) &= \left\{ \frac{1}{2}[\tilde{x}(t)e^{j2\pi f_c t} + \tilde{x}^*(t)e^{-j2\pi f_c t}] \right\}^n \\ &= \frac{1}{2^n} \sum_{k=0}^n \binom{n}{k} [\tilde{x}(t)]^k [\tilde{x}^*(t)]^{n-k} e^{j2\pi f_c (2k-n)t} \end{aligned} \quad (5.2.21)$$

For the first-zone output we see that only terms where n is odd and $2k - n = \pm 1$ can contribute. The first-zone contribution of (5.2.21) is then

$$x_{1z}^n(t) = \frac{1}{2^{n-1}} \binom{n}{(n+1)/2} |\tilde{x}^*(t)|^{n-1} x(t) \quad \text{for } n \text{ odd} \quad (5.2.22)$$

Hence the complex envelope of the first-zone component of $y(t)$ is

$$\tilde{y}(t) = \tilde{x}(t) \sum_{m=0}^{(N-1)/2} \frac{a_{2m+1}}{2^{2m}} \binom{2m+1}{m+1} |\tilde{x}^*(t)|^{2m} \quad (5.2.23)$$

The bandpass output signal can now be written (see Problem 5.12)

$$\begin{aligned} y(t) &= \text{Re}[\tilde{y}(t)e^{j2\pi f_c t}] \\ &= \left[\sum_{m=0}^{(N-1)/2} \frac{a_{2m+1}}{2^{2m}} \binom{2m+1}{m+1} |\tilde{x}^*(t)|^{2m+1} \right] \cos[2\pi f_c t + \theta(t)] \\ &= g_1[A(t)] \cos[2\pi f_c t + \theta(t)] \end{aligned} \quad (5.2.24)$$

so that in this case we obtain directly the envelope transfer characteristic. The power series model also permits us to obtain directly the harmonic levels in terms of the coefficients, which then allows us to compute what those coefficients should be in order to control those levels.

5.2.4. Memoryless Bandpass Amplifiers: Empirically Based Models

We now consider the second class of memoryless bandpass nonlinearities, which we referred to as bandpass amplifiers. This terminology is intended to connote a device with an *inherently* passband behavior; that is, only bandpass inputs can produce any measurable output. Nevertheless, spurious signals (harmonics) are also produced, even though the output around the original carrier frequency is the desired signal. Bandpass amplifiers are normally characterized by measuring them, and it is these measurements that form a basis for almost all of the models that will be discussed and that are in use, at least for simulation purposes. In Section 5.5 we will discuss these measurements in more detail.

It is an experimental observation⁽⁷⁾ that an input

$$x(t) = A \cos(2\pi f_c t + \theta) \quad (5.2.25)$$

into a bandpass amplifier produces an output of the form

$$y(t) = g(A) \cos[2\pi f_c t + \theta + \Phi(A)] \quad (5.2.26)$$

If we were to plot $g(A)$ and $\Phi(A)$ as a function of A , we would obtain curves more or less similar to those shown in Figure 5.6, depending on the specific device being measured. The (generally) nonlinear gain function $g(A)$ is conventionally referred to as the *AM/AM characteristic* and the (generally) nonlinear phase function $\Phi(A)$ is called the *AM/PM characteristic*, and also often referred to as AM/PM conversion. Sometimes, for brevity, we will refer to both AM/AM and AM/PM as the nonlinear characteristics.

The conditions under which (5.2.26) would be measured are such that A is varied very slowly. This is sometimes called a static or quasistatic measurement. It is natural to ask, therefore, whether the form (5.2.26) can be formally extended if now we modulate the carrier, that is, $A \rightarrow A(t)$, such that the bandwidth is now nonzero. We would feel justified doing so if the bandwidth in question is relatively small compared to the inherent bandwidth of the

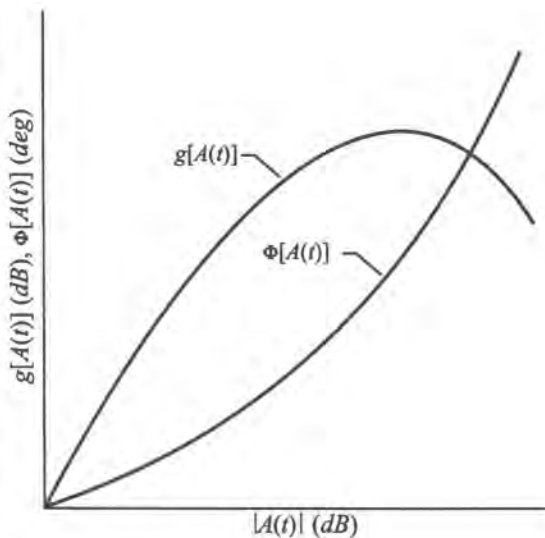


Figure 5.6. Illustration of the amplitude and phase transfer characteristics of an envelope nonlinearity.

device. We can ask the same question as before if we modulate the carrier by letting the phase be time-varying, $\theta \rightarrow \theta(t)$, and again we would answer the same way. Thus, under conditions which can only be qualitatively phrased, we will suppose that the formal extension of (5.2.26) is permitted when the carrier is amplitude- and/or phase-modulated, so that if

$$x(t) = A(t) \cos[2\pi f_c t + \theta(t)] \quad (5.2.27)$$

then

$$y(t) = g[A(t)] \cos[2\pi f_c t + \theta(t) + \Phi[A(t)]] \quad (5.2.28)$$

There is experimental evidence⁽⁸⁾ that (5.2.28) applies relatively well to solid-state amplifiers, and again we can expect it to be a reasonably good approximation for “narrowband” signals in any device. This model has long been used in analysis and simulation, but begins to fail when the input signal bandwidth becomes large enough. This is when we need to consider the use of a model with memory, which we will discuss in the next section.

We have presented the AM/AM and AM/PM characteristics as quantities which are defined and measured in a definite way. In fact, the way in which one can define (hence measure) these curves is not unique. In Section 5.2.4.3 we will give an example of one variation on the traditional measurement technique that uses a two-tone stimulus. For the moment, however, so long as we retain the memoryless assumption, the model of (5.2.28) applies, no matter how the specific AM/AM and AM/PM curves are obtained.

In terms of our earlier definitions, Equation (5.2.28) is a block model, which can be presented as in Figure 5.7. Figure 5.7a is a symbolic block diagram, while Figure 5.7b is a simulation block diagram showing explicitly what must be done to implement the model in

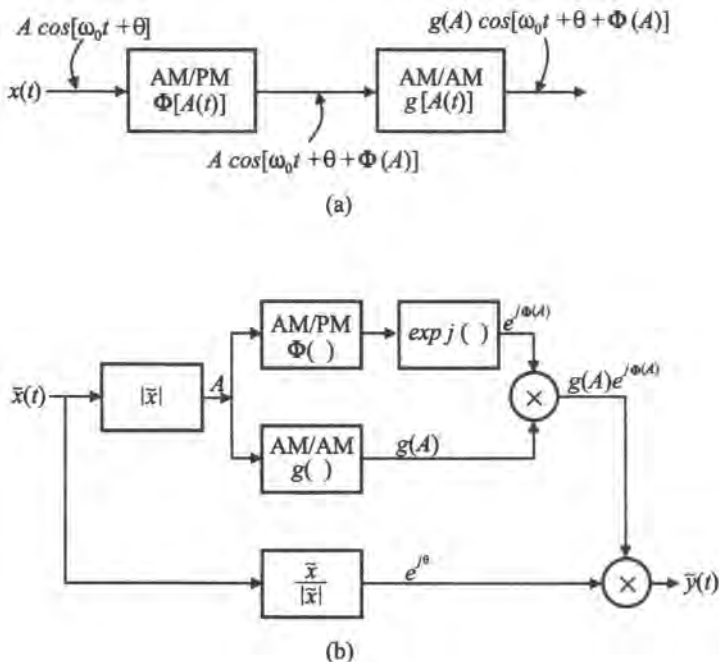


Figure 5.7. Block model for AM/AM and AM/PM envelope (bandpass) nonlinearity; both A and θ are functions of time. (a) Symbolic model at the carrier frequency level; (b) explicit simulation model at the complex envelope level.

code. Because of the structure of Figure 5.7a, we will refer to that particular form as a serial model or, because of the signal representation, as a polar model.

By a standard trigonometric identity, we can expand (5.2.28) into the equivalent form

$$y(t) = S_p(t) \cos[2\pi f_c t + \theta(t)] - S_q(t) \sin[2\pi f_c t + \theta(t)] \quad (5.2.29)$$

where

$$S_p(t) = g[A(t)] \cos \Phi[A(t)] \quad (5.2.30a)$$

and

$$S_q(t) = g[A(t)] \sin \Phi[A(t)] \quad (5.2.30b)$$

In analogy with the previous section, we can identify $S_p(t)$ and $S_q(t)$ with $g_1[A(t)]$ and $g_2[A(t)]$, respectively, so that we can conceive of the amplifier output as the first zone generated by some hypothetical memoryless nonlinearity. The alternative model (5.2.30) is shown in Figure 5.8. Figure 5.8a is again a symbolic representation, while Figure 5.8b shows an explicit structure for purposes of implementing the model. We will refer to this model as a

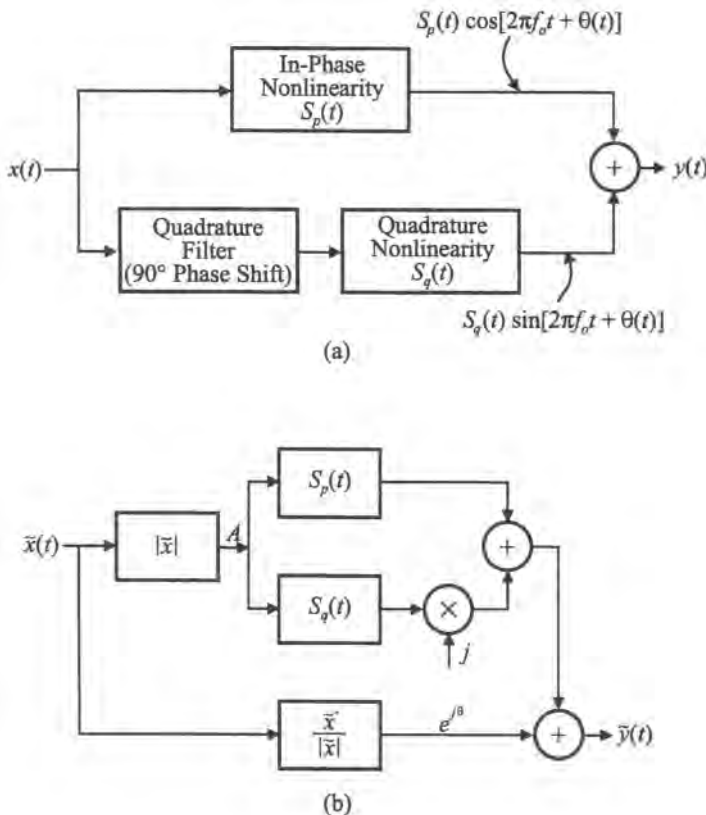


Figure 5.8. Quadrature model of envelope nonlinearity. (a) Symbolic model at the carrier frequency level; (b) explicit model at the complex envelope level.

parallel or quadrature model. At this point the parallel model and the serial model are totally equivalent. However, they do not remain so when they are extended in certain ways to model nonlinearities with memory: this will be seen in the next section.

■ *Remark.* It has been pointed out⁽⁹⁾ that a truly memoryless nonlinear system cannot affect the phase, as was already concluded in the previous section. Therefore, the models (5.2.28) and (5.2.29) cannot strictly be called memoryless. However, to make the same distinction as we did in conjunction with the hysteresis nonlinearity in Figure 5.4, we can call the models at hand instantaneous, since they rely only on the present value of the envelope. Following Ref. 9, we can also call this type of model effectively memoryless, which carries the implication that we have already pointed out, that the bandwidth of the input signal is sufficiently smaller than that of the device. A particular structure with memory is considered in Ref. 9 to show how an effectively memoryless model can be extracted under certain limiting conditions (see Problem 5.13). ■

5.2.4.1. Description and Interpretation of AM/AM and AM/PM Characteristics for Simulation

Two questions arise with respect to the AM/AM and AM/PM characteristics relative to their use in simulation. One is the proper interpretation of the related measurements into the abstraction which is the simulation model, and the other is simply the manner of description of these functions for simulation use. With respect to the first point, recall that the construction of the nonlinear characteristics is made for different values of A under conditions where each value used can be considered fixed for a long time with respect to all time constants involved. Thus, the average input power is $P_{\text{in}} = A^2/2$ and the average output power is $P_{\text{out}} = g^2(A)/2$. The main point about interpretation of the AM/AM and AM/PM curves in simulation is that while they are obtained under static measurements, they are applied to dynamic situations. That is, we usually interpret $g(\cdot)$ and $\Phi(\cdot)$ as *instantaneous* transfer characteristics, whereas, as just discussed, the corresponding measurements imply an averaging. In fact, the notion of instantaneous, as applied to simulation, must be interpreted to mean no faster than the simulation sampling rate. With this interpretation it can be shown (see Problem 5.10) that the power scale in the measured characteristics can be meaningfully interpreted as *instantaneous power* (a seeming oxymoron!), by which we mean the square of the instantaneous envelope. Thus, to obtain $g(A)$ and $\Phi(A)$ from measurements, we can interpret the power axes as one-half the square of the instantaneous envelope, the latter being the modulus of the complex envelope.

In the typical measurement and display of the nonlinear characteristics, as in Figure 5.6, the abscissa is graduated in units of $10 \log P_{\text{in}}$, and similarly for the output power scale. In order to apply these curves in simulation, the axes with log(power) units must first be converted to numeric units, e.g., via $A = \sqrt{2} \cdot 10^{P(\text{dB})/20}$, where $P(\text{dB})$ is the input or output power in dB units.

The second question relates to how g and Φ are represented. Since these curves are typically obtained experimentally, the most obvious way is simply to construct tables and look up values as the simulation proceeds. Of course, this will necessitate interpolation. Linear interpolation will be adequate if obtained points are sufficiently closely spaced. An alternative is to use an analytical representation. In the absence of measurements we might want to use some such representation which is more or less typical, but even with available measurements we might try to fit these with perhaps an easy-to-evaluate function. If the fit is good, the error will be minimal and the model could thereby be more efficient, and an analytical form is at

least usually helpful for visualization. Many different kinds of forms have been used in the literature. For example, the form

$$\Phi(A) = c_1(1 - e^{c_2 P_{in}}) + c_3 P_{in} + \phi_0$$

has been suggested,⁽⁷⁾ where the coefficients c_1, c_2, c_3 are determined by a best fit. Because of the exponential, however, a simple table lookup is probably more efficient.

As an illustration of a computationally efficient type of fit, consider the following forms, which have been proposed by Saleh⁽¹⁰⁾:

$$g(A) = \frac{\alpha_g A}{1 + \beta_g A^2} \quad (5.2.31a)$$

and

$$\Phi(A) = \frac{\alpha_\Phi A^2}{1 + \beta_\Phi A^2} \quad (5.2.31b)$$

for the serial model, and the alternative forms

$$S_p(A) = \frac{\alpha_p A}{1 + \beta_p A^2} \quad (5.2.32a)$$

and

$$S_q(A) = \frac{\alpha_q A^3}{(1 + \beta_q A^2)^2} \quad (5.2.32b)$$

for the parallel model. The coefficients in (5.2.31)–(5.2.32) are determined by a least-squares fit, and it has been shown⁽¹⁰⁾ that the resulting functions provide excellent agreement with several sets of measured data. Still, in terms of computational efficiency there is probably not much to choose between these analytical forms and table lookup. Probably the greatest utility of analytical models is their application to sets of g and Φ measured with frequency as a parameter, which leads naturally to block models for nonlinearities with memory, as will be seen.

5.2.4.2. Lowpass Equivalent of a Bandpass Amplifier

The lowpass-equivalent representation of the model we have been discussing is inherent in the modeling construct and evident on the surface. For the serial model, (5.2.28), it is clear that the output complex envelope is given by

$$\tilde{y}(t) = g[A(t)]e^{j\theta(t)}e^{j\Phi(A(t))} \quad (5.2.33a)$$

$$= A(t)e^{j\theta(t)} \frac{g[A(t)]}{A(t)} e^{j\Phi(A(t))} \quad (5.2.33b)$$

$$= \tilde{x}(t)G(|\tilde{x}(t)|) \quad (5.2.33c)$$

with an obvious identification of the complex gain G , which can be interpreted as the lowpass-equivalent transfer function of the nonlinearity. In actuality, the form (5.2.33a) is simpler to calculate, and would be used.

An alternative representation can be extracted for the parallel form (5.2.29), for which it is evident that

$$\tilde{y}(t) = \{S_p[A(t)] + jS_q[A(t)]\}e^{j\theta(t)} \quad (5.2.34a)$$

$$= A(t)e^{j\theta(t)} \left\{ \frac{S_p[A(t)]}{A(t)} + j \frac{S_q[A(t)]}{A(t)} \right\} \quad (5.2.34b)$$

$$= \tilde{x}(t)N(|\tilde{x}(t)|) \quad (5.2.34c)$$

where we recognize N as the describing function defined in the preceding section. Again, these alternative forms provide different insights into the action of the nonlinearity, but the form (5.2.34a) would be used in an actual simulation.

5.2.4.3. Alternative Approaches to Defining AM/AM and AM/PM Characteristics

Classically, the AM/AM and AM/PM conversions have been defined with respect to a CW signal as discussed in conjunction with (5.2.25) and (5.2.26). (We will also refer to this kind of characterization as *static*, because the envelope and phase are not time-varying.) However, the definition of these conversions can be generalized in several ways by using *dynamic* rather than static signals.⁽¹¹⁾ The motivation for this generalization is to improve the accuracy of the resulting model for operational signals that are not necessarily “narrowband,” as just discussed. The reasoning here is that because of the natural continuity of the nonlinear device’s input/output operator with respect to its input signal, the output predictions of a model constructed from basis signals more complex and “closer” to the operational should be more accurate than those from the classical static model.

Perhaps the simplest basis-signal generalization is that of a two-tone stimulus consisting of a large tone and a small tone at some offset frequency Δf above or below the large tone’s frequency f_c (termed *high-side* or *low-side injection*, respectively). In particular, the basis signal can be written in the form of (5.2.27) with (see Problem P5.15)

$$A(t) = A[1 + \alpha^2 + 2\alpha \cos(\pm 2\pi\Delta f t + \theta_a - \theta_A)]^{1/2} \quad (5.2.35a)$$

and

$$\theta(t) = \tan^{-1} \left[\frac{\sin \theta_A + \alpha \sin(\pm 2\pi\Delta f t + \theta_a)}{\cos \theta_A + \alpha \cos(\pm 2\pi\Delta f t + \theta_a)} \right] \quad (5.2.35b)$$

where A and a represent the static amplitudes of the large and small tones, respectively, their corresponding phases θ_A and θ_a are arbitrary, and

$$\alpha \equiv \frac{a}{A} \ll 1 \quad (5.2.36)$$

Condition (5.2.36) is maintained as the amplitude A is swept from a small-signal to a large-signal value that saturates the device being measured. In this case, the dynamic AM/AM and

AM/PM conversions are in general a function of two frequency parameters (f_c and Δf) instead of the single parameter f_c as in the static case. Assuming that the basis signal has its large-tone frequency fixed and only the offset frequency is varied, the output signal can again be represented by expression (5.2.28) with

$$g = g[A(t), \pm\Delta f], \quad \Phi = \Phi[A(t), \pm\Delta f] \quad (5.2.37)$$

where $A(t)$ and $\theta(t)$ are as in (5.2.35), and the sign on Δf indicates whether high-side or low-side injection is taking place (since, in general, the device's response to these two stimuli need not be identical). By expanding the output (5.2.28) in a McClaurin series with respect to the small parameter α , and assuming high-side injection (low-side injection is entirely similar), it can be shown that, to order α^2 (see Problem P5.16),

$$y(t) = g(A, \Delta f) \cos[2\pi f_c t + \theta_A + \Phi(A, \Delta f)] \\ + \alpha g(A, \Delta f) \{S_+ \cos(2\pi f_+ t + \Phi_+) - S_- \cos(2\pi f_- t + \Phi_-)\} + O(\alpha^2) \quad (5.2.38a)$$

where

$$S_{\pm} \equiv S_{\pm}(A, \Delta f) = \frac{1}{2} \left\{ \left[1 \pm \frac{A g'(A, \Delta f)}{g(A, \Delta f)} \right]^2 + [A \Phi'(A, \Delta f)]^2 \right\}^{1/2} \quad (5.2.38b)$$

are the amplitudes of the *two* sidebands present at the output of the nonlinearity at the frequencies $f_{\pm} \equiv f_c \pm \Delta f$, with similar expressions for the sideband phase components $\Phi_{\pm}$. The primes in (5.2.38b) represent derivatives of the dynamic conversions (with $\alpha = 0$) with respect to A . Observe from (5.2.38b) that the differential equations for the dynamic AM/AM and AM/PM conversion curves $g(A, \Delta f)$ and $\Phi(A, \Delta f)$ can be obtained from just the amplitude measurements of the two output sidebands, and these in turn can be solved using the appropriate small-signal conditions. It can be seen that, as developed so far, $\alpha \rightarrow 0$ is a necessary condition for the first part of the expression on the right-hand side of (5.2.38a) to be a good approximation for the output. Further discussion of the measurement procedures and model construction involved in this approach can be found in Section 5.5 below. The result of such a construction for a representative satellite communications helix TWT is shown in Figure 5.9. There is clearly a difference between the curves obtained by the method just described and the one conventionally obtained.

By employing the Hilbert transform described in Chapter 3, a dynamic AM/AM and AM/PM conversion can be defined for more general bandpass basis signals, such as a more general two-tone input for which the smallness condition (5.2.36) does not necessarily hold; a triangular-wave AM signal that in essence represents a multitone signal with a distinct set of amplitudes; or a more general, digitally modulated signal such as QAM.

5.2.4.4. Multiple Carriers and Intermodulation Products

Although the preceding development is phrased in terms of an amplitude- and/or phase-modulated input carrier, the lowpass-equivalent model described is equally applicable to multiple modulated carriers (Problem 5.13). This is because the form (5.2.27) can represent one or several carriers at the input to the amplifier. To see this, consider a multiple carrier input

$$x(t) = \sum_{i=1}^N A_i(t) \cos[(\omega_c + \omega_i)t + \theta_i(t)]$$

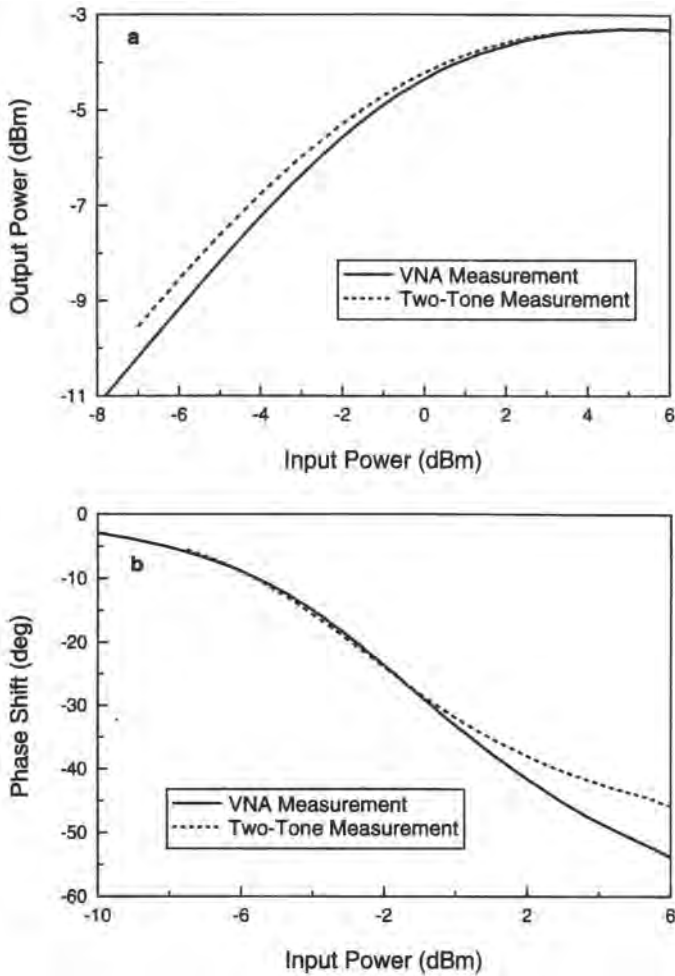


Figure 5.9. Comparison of amplifier characteristics measured with VNA and a two-tone technique. (a) AM/AM characteristic; (b) AM/PM characteristic.

which can be written as

$$x(t) = A(t) \cos[\omega_c t + \theta(t)] \quad (5.2.39)$$

where

$$A(t) = [x_I^2(t) + x_Q^2(t)]^{1/2} \quad (5.2.40a)$$

$$\theta(t) = \tan^{-1}[x_I(t)/x_Q(t)] \quad (5.2.40b)$$

$$x_I(t) = \sum_{i=1}^N A_i(t) \cos[\omega_i t + \theta_i(t)] \quad (5.2.40c)$$

$$x_Q(t) = \sum_{i=1}^N A_i(t) \sin[\omega_i t + \theta_i(t)] \quad (5.2.40d)$$

Thus, it is clear that the output of the envelope nonlinearity can be expressed in the form (5.2.28) or (5.2.29), where $A(t)$ and $\theta(t)$ are functions of the several input carriers. The multiple carrier case is of obvious importance in multiple access scenarios that use frequency-division multiplexing. Here, *intermodulation (IM) products* are a major potential source of degradation, which induce a typical tradeoff between performance (IM product levels) and power efficiency (amplifier backoff). It should be emphasized that the model at hand is a lowpass equivalent, so that the IM products that can be revealed are only those existing in the first zone, i.e., in the general vicinity of the carrier.

We note that this subject has received a great deal of attention in the literature, using various analytic techniques: Refs. 12–16 are just a small sample. For example, one form of the complex envelope of the nonlinearity output is given by⁽¹⁵⁾

$$\tilde{y}(t) = \sum_{k_1+\dots+k_N} = 1M(k_1, \dots, k_N, A_1, \dots, A_N) \exp \left\{ j \sum_{p=1}^N k_p [\omega_p t + \theta_p(t)] \right\} \quad (5.2.41)$$

where

$$M = \frac{1}{(2\pi)^2} \int_0^\infty \int_0^\infty u \left[\prod_{p=1}^N J_{k_p}(uA_p) \right] J_1(u\rho) \rho g(\rho) e^{j\Phi(\rho)} du d\rho \quad (5.2.42)$$

and here ρ represents the multicarrier envelope (5.2.40a). Although these equations give a clear qualitative picture of the interaction among the carriers, the net effect upon any one of them is not so obvious and requires numerical evaluation. While this can reasonably be done for the A_i constant, it becomes problematic when the envelopes of the individual carriers are not constant, for then M becomes a function of time. Even if the individual carriers arrive as constant envelope, filtering at the input to the receiver will generally alter that fact. Thus, in this case, a practical alternative is to let the simulation “solve” the problem.

We should also reiterate a point made earlier, namely, that while we have been considering here a memoryless nonlinearity, the discussion applies also to certain models with memory in which (as will be seen) memoryless nonlinearities appear sandwiched by linear filters. For, in this case, intermodulation can be brought about only by the memoryless nonlinearities. We also note that although the bandwidth expansion (hence sampling rate increase) is fundamentally caused by the same mechanism as before, here it manifests itself differently. With respect to the simulation sampling rate, it is the IM products that fall outside the original band covered by the input carriers that need to be anticipated, so that they do not fold back onto the spectrum of interest.

Regarding the sampling rate, aside from the bandwidth expansion just mentioned, it should also be clear that the sampling rate for N equal bandwidth carriers is at least N times that for a single such carrier (including guard bands). This point is discussed further in Chapter 8. Typically, we are interested in the performance of a “typical” carrier, but we must nevertheless account for the presence of all. If indeed we want to concentrate on a typical single carrier, it is possible to reduce the run time to something like that needed if that carrier were by itself. The reader is referred to Ref. 17 for the details.

5.2.4.5. Setting the Operating Point of a Memoryless Nonlinearity

The operating point (OP) of a nonlinear amplifier is the value of average input power that is desired in a particular application. If the input envelope is constant, $A(t) = A$, then the average input power $P_{in} = A^2/2$ is well defined and the maximum output power would be

extracted by setting the operating point to correspond to the peak of the input/output power curve (usually referred to as saturation).

Let us suppose that there is a preamplifier preceding the nonlinear amplifier itself, and let P_{in} be the average power of the arriving signal $x(t)$ at the input to the preamplifier,

$$P_{\text{in}} = \frac{1}{T} \int_0^T \frac{1}{2} A^2(t) dt$$

where T is some averaging period and, as usual, $A(t)$ is the envelope (equivalently, the modulus of the complex envelope). If we want the average input power to the nonlinear amplifier (i.e., the operating point) to be, say, P_o , then we merely need to set the preamplifier gain $G = P_o/P_{\text{in}}$.

We can identify two ways of setting G . In the first case, we might use a calibrating signal like an unmodulated carrier, in which case P_{in} is a constant, irrespective of the averaging interval (which of course cannot be too small). Then G is easily established from the AM/AM curve. Generally, the envelope is time-varying, so that, for a given T , so is P_{in} . In practice, of course, the preamplifier is really an automatic gain control (AGC), which adjusts the gain in accordance with the “current” input power. The latter can be defined as a sliding time average

$$P_{\text{in}}(t) = \frac{1}{T} \int_{t-T}^t \frac{1}{2} A^2(t) dt$$

so that, with T reasonably defined, the AGC gain is simply $G(t) = P_o/P_{\text{in}}(t)$. Since, in simulation, time is incremented by T_s , the possible update rate of G is evidently some multiple of T_s .

5.3. Nonlinearities with Memory (NLWM)

In the preceding section we discussed models for memoryless (or perhaps instantaneous) nonlinearities. These models represent fairly accurately a variety of devices driven by narrowband inputs, and are commonly used in analysis and simulation. The suitability of a memoryless model for narrowband signals stems from the fact that many radiofrequency amplifiers have indeed a wide bandwidth, and over any relatively small portion of that band the transfer characteristic does look nearly frequency independent about the center of that portion.

When, however, we attempt to send “wideband” signals, by which we mean that the bandwidth of the signal is comparable to the inherent bandwidth of the device (no matter what the actual bandwidth), then we are bound to encounter some frequency-dependent behavior. That such behavior exists is also revealed experimentally through a generalization of Figure 5.6. Suppose that instead of using a single tone at f_c , we measure AM/AM and AM/PM characteristics using a set of tones of different frequencies, spanning the bandwidth of the device. The resulting characteristics might have an appearance generally like that shown in Figure 5.10[†], which would be *prima facie* evidence of frequency dependence. If, on the other hand, all the curves at the different frequencies overlay one another, or nearly so, we would

[†]For the moment we are not concerned whether these curves are “realistic.” The main point is that they are not identical.

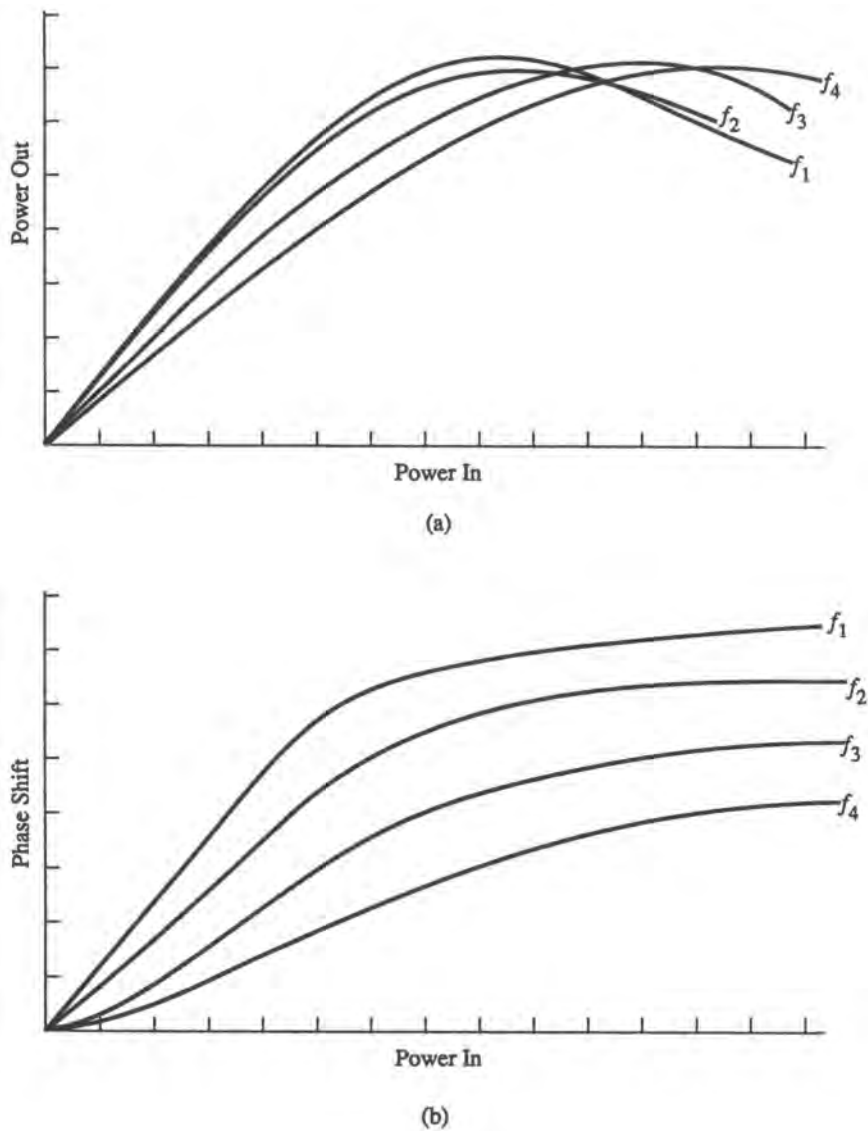


Figure 5.10. Illustration of transfer characteristics of a nonlinear amplifier with memory, as measured by a “static” test (swept-frequency measurements). (a) Gain transfer function with frequency as a parameter; (b) phase transfer function with frequency as a parameter.

have good grounds to believe that a memoryless model would be adequate. Of course, just as with the memoryless model, we are not limited to making measurements with a single tone, and the resulting measured characterization would very likely be different.

In Section 5.1, we began a discussion on the different modeling options for nonlinearities. Here we will continue that discussion focusing only on nonlinearities with memory. As a general statement, a nonlinearity with memory can be defined as one whose output, at any time t , depends on a segment (finite or infinite) of the input signal prior to time t . Memory

implies a filtering effect that is in part manifested by the frequency-dependent nonlinear characteristics mentioned above. It will therefore not be surprising that nonlinear block models will now include blocks that will be identified as filters.

To place NLWM models in perspective, it is useful to define (roughly following Palmer and Lebowitz⁽¹⁸⁾) three broad classes of characterization: *macroscopic*, *microscopic*, and *analytical* models. By macroscopic models we mean models constructed on the basis of input/output measurements using an appropriate probing signal. In other words, we treat the nonlinear device (subsystem) as a black box, and use the measurements to “identify” a topology that seems to account for the measurements.[†] This type of model will typically be a block model, as defined earlier. Because it is the highest level type of modeling, it is computationally the most efficient. Such models also lend themselves best to practical system-level specifications. The drawback to such models is their variable accuracy. Nevertheless, they constitute the standard approach for system simulation, and below we shall concentrate on this type of model.

By microscopic models, we mean models that are *exact* at the intended level of representation. Just as the term macroscopic implies only external access to the box, our use of microscopic implies some degree of visibility inside the box. We can identify three types of microscopic models.

- *Microscopic Models I.* In this modeling construct, subsystems are comprised of some set of well-defined functions operating at the waveform level (for baseband signals) or at the complex envelope level (for bandpass signals). An example of such a subsystem would be one consisting of a cascade of elements some of which are memoryless nonlinearities. If we accept that this set of elements is a perfect representation of the subsystem in question, the model is exact at the *intended* level of representation. Another example of this type of model is a phase-locked loop. The nonlinear differential equation representing the loop is treated as an exact representation of its functioning. This type of model is suitable for communication system simulation.

- *Microscopic Models II.* This type of model applies to subsystems with nonlinear devices containing filtering or energy storage elements for which an equivalent circuit can be developed. By and large, this is the case for solid-state devices.⁽²⁰⁾ Assuming such circuit equivalents are exact, larger circuits that include these (nonlinear) solid-state devices can be considered exact at the intended level of representation. However, circuit-level models have to be simulated at the actual carrier frequency, making them generally unsuitable for system-level simulation. One way⁽¹⁸⁾ to use circuit-level models is to derive complex envelope properties (such as AM/AM and AM/PM characteristics) and port these results to a system-level waveform simulator. The availability of a circuit model notwithstanding, it may be that the most practical course would be to synthesize a macroscopic type of model.

- *Microscopic Models III.* These include subsystems or devices containing inseparable nonlinear and filtering behavior for which equivalent circuits cannot be readily formulated. The most important example is the traveling-wave tube. Here, the fundamental mode of description is a physical model^(21,22) consisting of systems of partial differential equations, which is not suitable for inclusion in a system-level simulation. Here, the usual course is to derive a block model based on black-box characterization.

[†]An interesting taxonomy of identification has been proposed by Sjöberg *et al.*,⁽¹⁹⁾ who defined “white-,” “gray-,” and “black”-box modeling, depending upon the degree to which prior knowledge or understanding could be brought to bear in the modeling process. For nonlinear amplifiers, it would probably be fair to say that our procedures fall under the “gray”-box category.

The third general modeling category for NLWM, analytical models, is meant here to refer to models such as the Volterra series that are able, under some circumstances, to represent exactly the functioning of a NLWM. In a sense, such models stand at the intersection of macroscopic and microscopic models because they are both fed by external measurements and at the same time are capable of predicting the device's behavior to the same degree of accuracy as a microscopic model. We shall take a brief look at Volterra series models as well as a less general version, referred to as polyspectral models.

The modeling approaches we shall discuss will be “fed” by one set of measurements or another. The role of measurements is crucial to define the model in a quantitative, not qualitative, sense. That is, the structure of the model is defined independently of the measurements themselves, although a particular modeling idea may well completely specify what measurements must be taken. It may also be that several structures are defined by the same set of measurements, but that these measurements are not used in the same way. For example, the structure may be preconceived according to some heuristic construct, or it may “fall out” from some reasoning process. One or another of the blocks may be directly defined by a particular measurement, or the blocks may be made to fit some set of measurements as best they can within the limitations of that structure. There are many variations on this theme, and in the following we will illustrate some, but by no means all, of the modeling possibilities, which we will group (somewhat loosely) into a few categories.

5.3.1. NLWM Modeling I: Fitting Swept-Tone AM/AM and AM/PM Measurements

In this approach, the measurements are of the type illustrated in Figure 5.10: swept-tone, swept-amplitude measurements are made (Section 5.5) resulting generally in two sets of frequency-dependent curves. The objective in this category of modeling is to construct a block model which, under the same input conditions as the measurements, will reproduce as closely as possible the measured AM/AM and AM/PM curves. While this property of the model is satisfying to some degree, there is no guarantee of closeness for inputs other than single tones. In fact, since superposition does not hold in a nonlinearity, we know that such a model must be an approximation,⁽²³⁾ though it may be a reasonably good one. We note that in this modeling methodology the structure of the model is not predetermined, but emerges as a consequence of the basic objective. We will present three versions of this approach.

5.3.1.1. The Poza–Sarkozy–Berger (PSB) Model⁽²⁴⁾

It will be convenient to discuss AM/AM and AM/PM models separately, and then combine the results.

5.3.1.1.1. AM/AM Characteristics. The basis for this model construction is the observation that AM/AM (input/output power) characteristics at different frequencies for TWTs, when displayed in decibel units for both input and output scales, have a very “similar” appearance. While similarity is somewhat subjective, the authors postulated that the similarity was in fact complete. In other words, it was assumed, for the purpose of model construction, that every AM/AM curve could be obtained from any other such curve by some combination of horizontal and vertical displacement. Below, we will refer to this idea as the *fundamental assumption*.

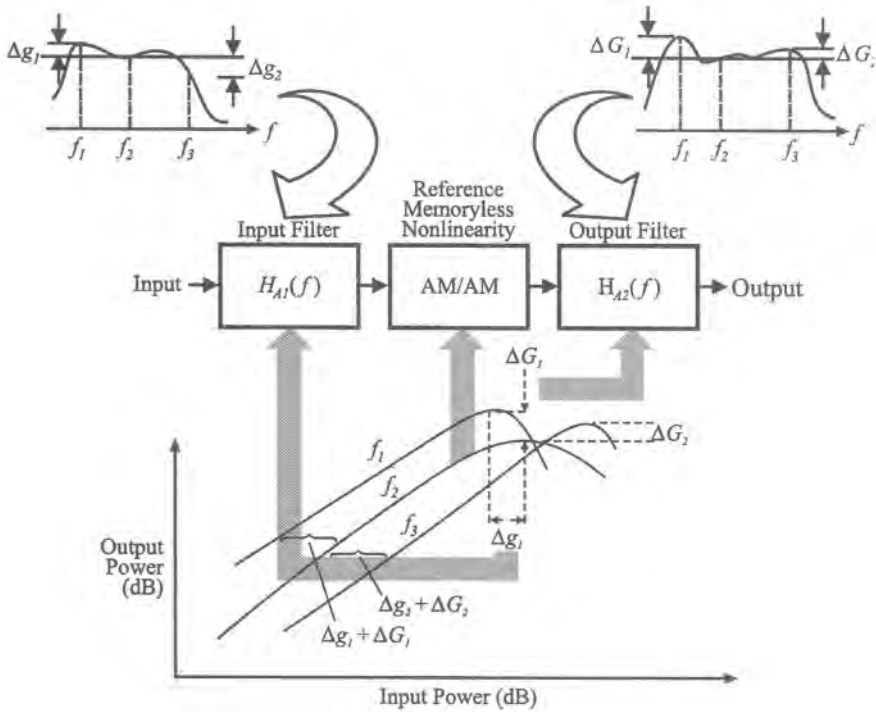


Figure 5.11. The synthesis procedure for the AM/AM portion of the PSB model.

To develop the procedure, we will refer to the simplified sketch in Figure 5.11. Given a set of AM/AM characteristics, we choose one such curve as a *reference* curve; the appropriate one to use is that for the center frequency since this frequency becomes the zero frequency in the lowpass equivalent. In Figure 5.11, the reference curve is the one for $f = f_2$.[†] In the absence of frequency dependence, that curve would give the input/output relationship for an input sinusoid of any frequency. Suppose now that we input a tone at $f = f_1$. If we were to use the reference curve to project the output level (for a given input level), that output level would be lower than the true value. Hence, if we “preemphasized” the tone at f_1 by the proper amount and then used the reference curve, the corresponding output power would be correct. What is the proper amount of preemphasis? By the fundamental assumption, the curves at f_1 and f_2 differ only by some amount of horizontal and vertical displacement. It is intuitive that the proper amount of horizontal displacement is that needed to align the peaks of the curves, and then the proper amount of vertical displacement is that needed to bring these peaks together. Hence, for each frequency $\neq f_2$ we can interpret the needed amount of horizontal displacement as the required amount of preemphasis, as indicated in Figure 5.11. This, in effect, defines a filter, $H_{A1}(f)$, as shown in the figure (there, the filter’s transfer function is defined in the power domain).

If the different AM/AM curves had the same peak power, the input filter $H_{A1}(f)$ would complete the model. However, to account for the difference in the peaks that does exist, we

[†]For visual clarity, Figure 5.11 shows only three curves. In practice, to get usable resolution we would want measurements at something like 100 frequencies across the band.

now need to postemphasize (or deemphasize) each curve, in accordance with the difference between its peak and that of the reference curve. As shown in Figure 5.11, this effectively generates another filter function, labeled $H_{42}(f)$, placed after the reference nonlinearity. Thus, to account for the frequency dependence of input/output power, the model consists of a filter-memoryless nonlinearity-filter cascade. In actuality, the fundamental assumption is not quite true, so to define these boxes requires a sophisticated numerical fitting to define in some best way the horizontal and vertical translations.

5.3.1.1.2. AM/PM Characteristics. For these curves, a fundamental assumption analogous to that previously described is supposed to hold, namely, that any one AM/PM curve can be obtained from any other by some combination of vertical and horizontal translation, when the ordinate is measured in degrees (or radians) and the abscissa in decibel units. Again, we eventually have to deal with the fact that this assumption is not quite true and have to construct some associated fitting procedure. But, under the conditions of the fundamental assumption, we can conceive of a procedure similar to that above. Figure 5.12 illustrates the procedure. As before, we take the curve at f_2 as the reference nonlinearity, and construct an input filter that reflects the amount of horizontal movement and an output filter that accounts for the vertical movement. Note that the latter filter has a phase-only transfer function, whereas all other filters are amplitude-only. In the case at hand, the procedure for defining the filters is a little more subtle. The determination of a certain amount of horizontal or vertical displacement requires the identification of a unique feature on the curve with respect to which the displacements can be measured. In the AM/AM case, the saturation point provides such a reference. Typical AM/PM curves, however, such as displayed in Figure 5.12 do not have such a readily obvious feature. One possible procedure to address this problem is to identify a reference point on the basic curve by its slope. Assuming the value of that slope to exist only at one point on the input power axis, we can use it to find the vertical or horizontal movement of the various curves (for different frequencies) with respect to the curve for the reference frequency. This procedure is illustrated in Figure 5.12, where some value of the slope s has been chosen to mark the reference point.

5.3.1.1.3. Combined Model. The final frequency-dependent model is now obtained by cascading the blocks of the two previous figures back to back, as shown in Figure 5.13. The ordering of the blocks follows that previously shown for the memoryless model. Since the connecting blocks are both filters, they can be joined together as one filter, with the proper normalization (Problem 5.14). Although the filters in Figures 5.11 and 5.12 are labeled with actual frequencies ($f_1, f_2, f_3, \dots$), it should be clear that their use in the model of Figure 5.13 is as lowpass-equivalent filters, which is accomplished simply by translating the transfer functions from the actual center frequency to zero frequency. It can be seen that the PSB model is an extension of the serial or polar topology defined for memoryless nonlinearities.

It is useful to make the point that while the fundamental assumptions above imply a filter-single memoryless nonlinearity-filter cascade, the converse is also true. That is, a postulated cascade of that sort implies that the resulting swept-tone, swept-amplitude curves will be horizontally and vertically shifted versions of one another (Problem 5.24). This observation has consequences for the properties of certain models.

5.3.1.2. The Saleh Model⁽¹⁰⁾

This particular model is an extension of the parallel or quadrature topology defined for memoryless nonlinearities. It begins with its own fundamental assumption, that the forms given by (5.2.32a) and (5.2.32b) for a memoryless nonlinearity hold with more or the less the

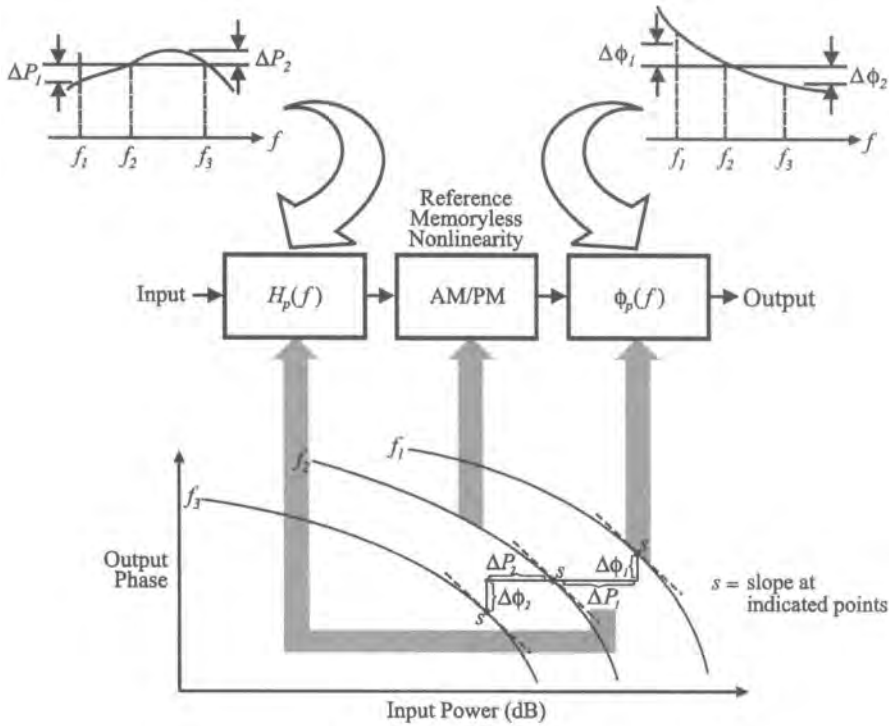


Figure 5.12. The synthesis procedure for the AM/PM portion of the PSB model.

same degree of accuracy for the functions $S_p(A)$ and $S_q(A)$ extracted from the AM/AM and AM/PM curves that apply to any given frequency. If we were to perform the fit implied by (5.2.32a) and (5.2.32b) for any measured nonlinear characteristics at some frequency f , the resulting functions and the fitting coefficients would in general be functions of f . However, it turns out that the rational forms in question provide a better fit if we first remove the “small-signal phase,” defined as follows. Let $g(A, f)$ represent any AM/AM curve as a function of A for any given f , and let $\Phi(A, f)$ similarly be any measured AM/PM curve as a function of A for any given f . The small-signal phase characteristic $\Phi_0(f)$ is defined as

$$\Phi_0(f) = \lim_{A \rightarrow 0} \Phi[A, f] \quad (5.3.1)$$

Let us return to (5.2.32a) and (5.2.32b) and express modified functions S_p and S_q as functions of (A, f) , suppressing the t dependence:

$$S_p(A, f) = g[A, f] \cos\{\Phi[A, f] - \Phi_0(f)\} \quad (5.3.2a)$$

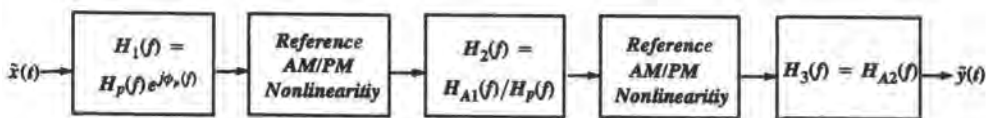


Figure 5.13. The complete PSB model.

and

$$S_q(A, f) = g[A, f] \sin\{\Phi[A, f] - \Phi_0(f)\} \quad (5.3.2b)$$

It is these last two equations that are assumed to be well fitted by equations of the form (5.2.32a) and (5.2.32b), respectively, which can now be generalized to

$$S_p(A, f) = \frac{\alpha_p(f)A}{1 + \beta_p(f)A^2} \quad (5.3.3a)$$

and

$$S_q(A, f) = \frac{\alpha_q(f)A^3}{[1 + \beta_q(f)A^2]^2} \quad (5.3.3b)$$

Equations (5.3.3) are the analytical form of the frequency-dependent model, which can be interpreted as the block model shown in Figure 5.14. The connection between the experimentally derived functions α_p , α_q , β_p , and β_q and the block properties are as given in the legend of Figure 5.14 (Problem 5.22). Notice also that the small-signal phase has been reinserted as an all-pass filter, since it has been removed from the fitting (Problem 5.23).

It is interesting that the block model derived from (5.3.3) consists of (in this case) parallel filter-memoryless amplitude nonlinearity-filter cascades, as was the case with the PSB model. However, the block properties are not the same. Recall the point made earlier that

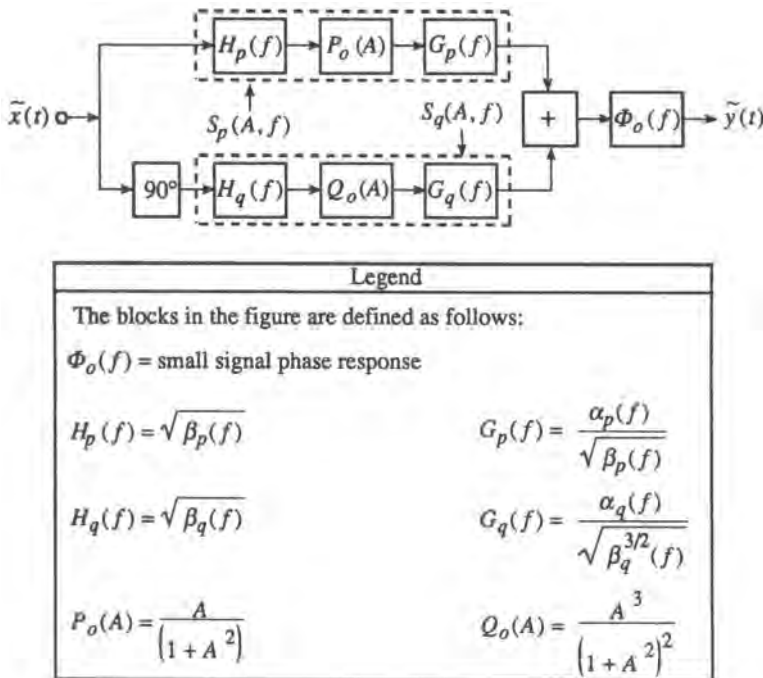


Figure 5.14. The frequency-dependent Saleh model (from Ref. 10, @IEEE, 1981).

a filter–nonlinearity–filter cascade necessarily implies a set of output frequency-dependent curves which are combinations of horizontal and vertical displacements of one another, when displayed in decibel units (Problem 5.17). Thus, this model structure has this inherent constraint. In fact, it can be shown⁽²³⁾ that constraining the amplitude and phase curves to maintain shape is fundamentally different than requiring the in-phase and quadrature nonlinearities to maintain shape. Thus, except in very specific circumstances (see Problem 5.18), the PSB and Saleh models are in fact different.

We note again that, as with the PSB model, the fitting of the relevant equations (5.3.3) would be performed on the measured curves at the actual frequencies. But, again, by simple translation, the resulting fits can be applied as lowpass-equivalent descriptions.

5.3.1.3. The Abuelma'atti Model⁽²⁵⁾

This model again uses the parallel or quadrature structure of the previous model, and it is also based on a numerical fitting of the functions S_p and S_q . However, as will be seen, the choice of fitting function will not impose a similarity of the input/output curves for different frequencies. The cost of this flexibility is added computational complexity. Suppose for the moment that the nonlinearity is memoryless. The beginning point for the modeling procedure is to assume for this nonlinearity a representation of the following form for the bandpass in-phase and quadrature amplitude nonlinearities:

$$S_p(A) = \sum_{n=1}^N \gamma_p(n) J_1 \left(\frac{n\pi A}{D} \right) \quad (5.3.4a)$$

and

$$S_q(A) = \sum_{n=1}^N \gamma_q(n) J_1 \left(\frac{n\pi A}{D} \right) \quad (5.3.4b)$$

where J_1 is the first-order Bessel function of the first kind, $2D$ is the assumed dynamic range of the input, and N is a selectable parameter that can be chosen to satisfy some criterion of closeness between the actual curve and the fit.

The Bessel sum actually arises naturally in the representation of a memoryless nonlinearity (see, e.g., Ref. 26). We give only a sketch of the idea, which reverts to the “bandpass nonlinearity” type of modeling described in Section 5.2.3. There, the bandpass nonlinearity g arises as the first-zone output of a memoryless nonlinearity F . In the present case, if we hypothesize such an F and expand it into a Fourier series in x (the input), a typical term in the series has the form $\sin(2\pi kx/2D)$, assuming odd symmetry for F , and $2D$ is the range of the input. (Incidentally, this is a limitation of the model, since a noisy input theoretically has no limit.) If we assume that N terms of the series is sufficient, this is the same N as in (5.3.4). Substituting $x(t) = \sqrt{2P} \cos[\omega_c t + \theta(t)]$ and using the Jacobi–Anger formula for $\sin(\beta \cos z)$ leads to a series of harmonics of the argument of the input, and if we extract only the first zone, the coefficient of $\cos[\omega_c t + \theta(t)]$ at the nonlinearity output has the form of either sum in (5.3.4). Since the quadrature model consists of two amplitude nonlinearities, each can be represented by the same generic form.

If now we assume the nonlinearity has memory, we have a set of curves $S_p(A, f)$ and $S_q(A, f)$, as before. For each such curve, we can use a fitting formula having the form (5.3.4), where the coefficients of the fit now become functions of frequency:

$$S_p(A, f) = \sum_{n=1}^N \gamma_p(n, f) J_1\left(\frac{n\pi A}{D}\right) \quad (5.3.5a)$$

and

$$S_q(A, f) = \sum_{n=1}^N \gamma_q(n, f) J_1\left(\frac{n\pi A}{D}\right) \quad (5.3.5b)$$

For future reference, it will be convenient to rewrite the coefficients as

$$\gamma_p(n, f) = \gamma_{p,0} G_p(f, n) \quad (5.3.6a)$$

$$\gamma_q(n, f) = \gamma_{q,0} G_q(f, n) \quad (5.3.6b)$$

where $\gamma_{p,0}$ and $\gamma_{q,0}$ are defined to be the values of the left side of (5.3.6) if $G_p = 1$ and $G_q = 1$. The functions $G_p(f, n)$ and $G_q(f, n)$ can be seen to incorporate filtering, while the Bessel functions embody memoryless nonlinearities. A block model representing (5.3.5)–(5.3.6) is given in Figure 5.15. As mentioned before, it is clear that this model will generally be considerably more computationally demanding than the two previous ones, depending on N . However, it can also be seen that, except in rare circumstances, the curves for $S_p(A, f)$ and

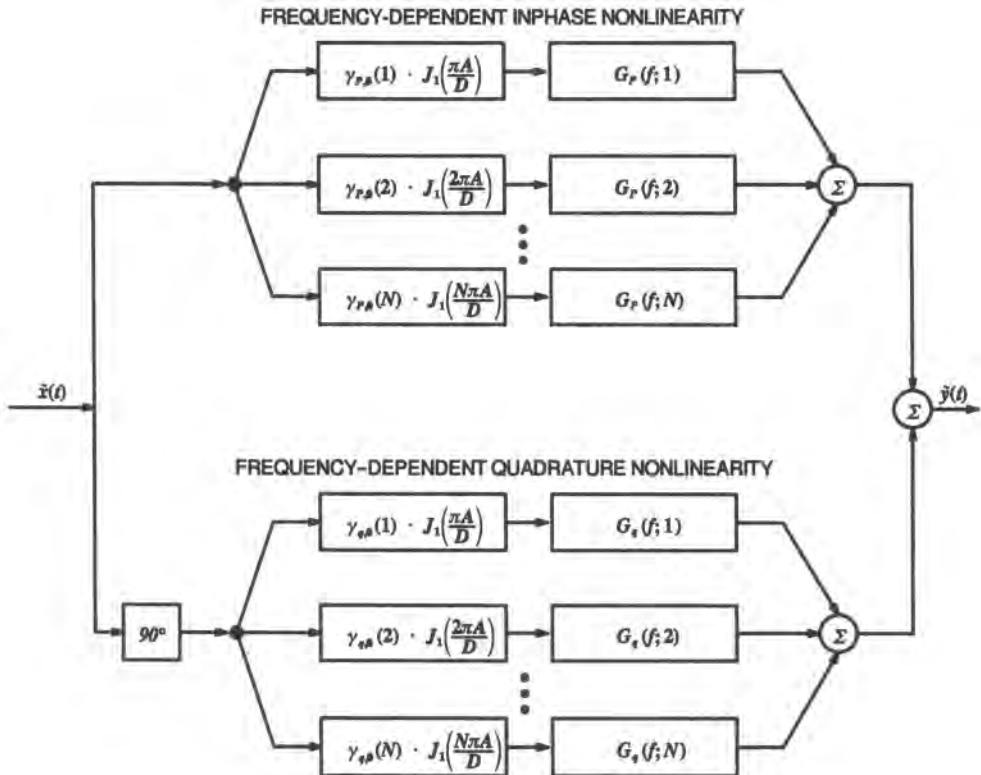


Figure 5.15. Structure of the Abuelma'atti model.

$S_g(A, f)$ issuing from the model will not be merely translates of one another in the decibel domain (Problem 5.26).

We should for clarity make once again the comment made about the two previous models, namely that the measurements, while made on the actual device, lead to lowpass-equivalent models by simple translation from the center frequency to zero frequency.

5.3.2. NLWM Modeling II: Fitting Preset Structures

As one might see from the preceding development, it is perhaps natural to conceive of nonlinearities with memory as combinations of filters and memoryless nonlinearities. But, rather than derive a topology from a given modeling objective, as above, one can start out with a preset block structure whose individual block characteristics are defined in some prescribed way. One prescription, for example, could be to define the block transfer characteristics through a (nonlinear) least-squares fit of certain measurements. Another might be simply to define a structure *and* its block properties in ad hoc fashion, hoping that the logic of the construction yields a reasonably good model. In this section we will give a few examples of this modeling methodology.

5.3.2.1. One Filter–One Nonlinearity (Two-Box) Models

In this modeling category, we simply assume that an adequate model can be synthesized from a single filter and a single (complex) nonlinearity. It would seem that this is the simplest possible model to represent both memory and nonlinearity. For brevity, we will sometimes refer to this combination as a two-box model.[†] Here, we present briefly several versions of the two-box model.

5.3.2.1.1. Filter–Nonlinearity with Least-Squares Fit. Figure 5.16a shows the simple topology of this model, where it is assumed the filter comes first. The idea⁽²⁷⁾ behind this model is to fit the assumed structure to a set of measured AM/AM and AM/PM

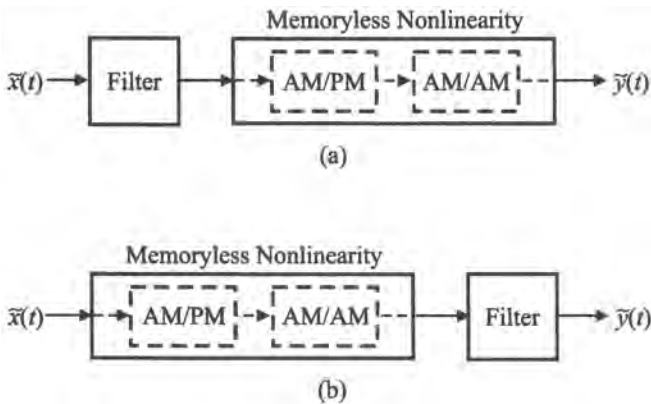


Figure 5.16. Two-box model topologies. (a) Filter followed by a memoryless nonlinearity; (b) memoryless nonlinearity followed by a filter.

[†]This terminology is convenient, but not totally consistent because in some of the earlier models, the complex nonlinearity is itself distributed into two distinct blocks.

characteristics, using the usual swept-tone technique. The criterion for the fit is in the least-squares sense, although, in principle, any reasonable criterion could be used. This, in general, is a quite complicated fitting problem. The basic approach is to describe the amplitude and phase characteristics of the filter as (truncated) power series, and likewise for the AM/AM and AM/PM characteristics. This sets up a set of simultaneous equations, the solution to which are the coefficients of the various series representations just mentioned. The reader is referred to Ref. 27 for further details.

5.3.2.1.2. Filter–Nonlinearity ARMA Model. This model consists of a classical *autoregressive moving average* (ARMA) filter followed by a memoryless nonlinearity, and is constructed using time-domain measurements.⁽²⁸⁾ The model topology is shown in Figure 5.17, where the filter is represented by a digital IIR structure and the instantaneous amplitude/phase nonlinearity is expanded in a Bessel series. The initial conditions for the model parameters are obtained from small-signal measurements (for the filter) and a bandcenter AM/AM, AM/PM measurement (for the nonlinearity). Optimization of the model parameters is also performed in a least-squares fashion using baseband measurements of time-domain pulses. The test pulses are chosen to cover the expected dynamic range and spectral bandwidth of the device's operational signal environment.

5.3.2.1.3. Filter–Nonlinearity with Small-Signal Transfer Function. This modeling construct (mentioned in passing on p. 698 of the first edition of this book) has the same topology as that in Figure 5.16a, but the box specifications are obtained very differently. The methodology is essentially based on intuition, or “engineering judgment.” The filter is specified to have the transfer function equal to the measured small-signal (large-backoff) characteristics. The nonlinearity is assumed to be given by the AM/AM and AM/PM curves obtained from a single-frequency measurement at center frequency. Thus, the model is easy to “populate,” certainly in comparison with the immediately preceding model. The intuitive reasoning supporting the model is as follows. In a moderate- to large-backoff condition, as might be the case with a multiple carrier input, the device behaves quasilinearly, and the small-signal transfer function is indeed what we might expect the signal to “see.” In saturated operation, as we might do with a single carrier, that signal will “see” essentially the saturated transfer function. An implicit assumption of the model, therefore, is that the small-signal transfer characteristics reflected through the nonlinearity will result in an adequate approx-

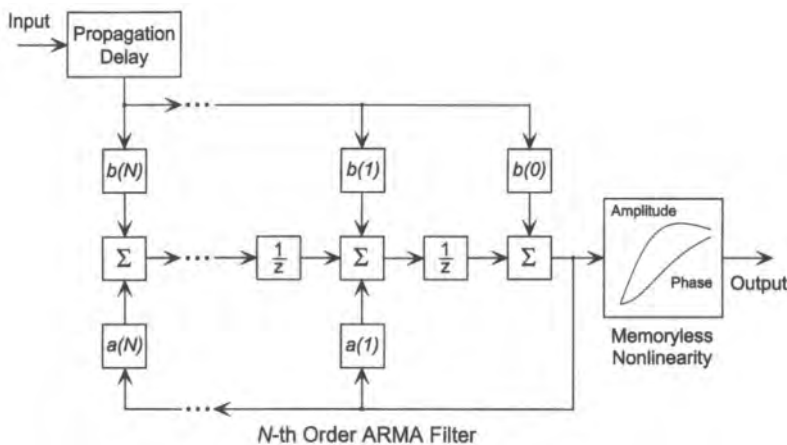


Figure 5.17. ARMA filter–nonlinearity model topology; the model coefficients are “optimized” using time-domain measurements. This bandpass model requires complex tap coefficients.

imation to the saturated transfer function. Simulations performed with this model have yielded surprisingly good results, but as always in this arena, validation of any model is highly recommended.

5.3.2.1.4. Nonlinearity–Filter with Least-Squares Fit. The topology for this model is shown in Figure 5.16b. We have seen in Section 5.3.1 that filters can both precede or follow nonlinearities, and in fact the Abuelma'atti model consists of nonlinearity–filter branches. So, in a two-box model, it is not clear *a priori* what the preferred order is. In the present case, the idea⁽²⁷⁾ as in Section 5.3.2.1.1, is to fit a set of measured AM/AM and AM/PM curves to the preset structure, which is represented also as several truncated power series. The fitting problem, while somewhat different, is also not straightforward. The reader is referred to Ref. 27 for further details. Although the basic idea for this model and that in Section 5.3.2.1.1 is essentially the same, it does not appear simple to choose one over the other without performing validation tests.

5.3.2.2. Filter–Nonlinearity–Filter (Three-Box) Models

Here we extend the preset modeling methodology to a model containing three boxes, which we will refer to as a three-box model. The model topology is preset to a filter–nonlinearity–filter cascade, as shown in Figure 5.18. By including a third box, we create an added dimension of flexibility which presumably can be used to improve on the two-box models without incurring the complexity of the models in the previous section. We again have a choice as to how we will define the boxes themselves, and below we outline two possibilities.

5.3.2.2.1. Three-Box Model with Least-Squares Fit. This model can be considered an extension of either of the two box models in in Sections 5.3.2.1.1 and 5.3.2.1.3. As mentioned, the additional filter lends another degree of freedom which should produce a better fit of the frequency-dependent AM/AM and AM/PM curves. To see that this possibility exists, we recall the observation that a filter–nonlinearity cascade must produce an output set of responses (to a single-tone input) that are horizontal translations of one another. Thus, in the corresponding two-box model, there is no possibility of vertical displacement. This possibility is materialized by an output filter. In the version at hand, we also suggest another modification that greatly simplifies the numerical fitting of the model to the measurements. In the two-box model as originally proposed, all the box functions had to be deduced from the fit. Here we propose to use for the memoryless nonlinearity the pair of AM/AM and AM/PM curves measured at center frequency. These particular curves are usually reasonably representative of the general behavior, and making this assumption removes from the fitting algorithm the burden of deducing a shape for the nonlinearities. To show the procedure, let $\{P_{\text{out}}(P_{\text{in}}, f)\}$ and $\{\Phi(P_{\text{in}}, f)\}$ be the set of measured AM/AM and AM/PM characteristics,[†] and let $P_o(P_{\text{in}}) \equiv P_{\text{out}}(P_{\text{in}}, f_c)$ and $\Phi_o(P_{\text{in}}) \equiv \Phi(P_{\text{in}}, f_c)$, with f_c being the carrier frequency. Then the fitting procedure is given by the following steps:

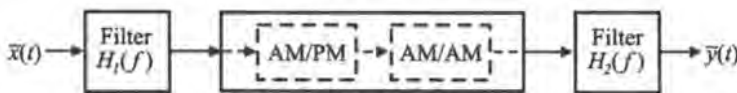


Figure 5.18. The three-box model configuration.

[†]We note that although in simulation we reinterpret input and output power axes as instantaneous envelope, in a fitting procedure we may as well use the raw measurements themselves. Here we also assume that P_{in} and P_{out} are given in dB units.

1. Find $\alpha(f_m), \beta(f_m)$ such that

$$\Delta_p(f_m) = \sum_{P_{in}} \{P_{out}(P_{in}, f_m) - P_o[P_{in} + \alpha(f_m)] - \beta(f_m)\}^2$$

is minimized for all measured frequencies $f_m, m = 1, \dots, M$.

2. Find $\xi(f_m)$ such that

$$\Delta_\Phi(f_m) = \sum_{P_{in}} \{\Phi_{out}(P_{in}, f_m) - \Phi_o[P_{in} + \alpha(f_m)] - \beta(f_m)\}^2$$

is also minimized for all f_m .

Thus, the model is given by $10 \log |H_1(f_m)|^2 = \alpha(f_m)$, $10 \log |H_2(f_m)|^2 = \beta(f_m)$ and $\angle H_1(f_m) + \angle H_2(f_m) = \xi(f_m)$. It can be seen that there is an ambiguity (not peculiar to this model alone) in the phase characteristics. That is, the procedure as given cannot uniquely identify the phase of the individual filters, only their sum. Further steps can be taken to resolve this ambiguity (see Problem 5.20).

5.3.2.2.2. Three-Box Model with Specified Characteristics. As was just seen, adding a filter to a two-box model should improve its ability to match some set of measurements. In the present case, the objective of the model is to extend the intuitively based construct in Section 5.3.2.1.2 so as to partly remove a seemingly obvious weakness in that model, namely what the signal “sees” in the saturated condition. We want to contrive to make the signal see the measured small-signal transfer function when the signal is indeed small, and to see the saturated transfer function when the signal is large. This leads naturally to defining the first filter as $H_1(f) = H_{ss}(f)/H_2(f)$ and $H_2(f) = |H_{sat}(f)|$, where $H_{ss}(f)$ and $H_{sat}(f)$ are, respectively, the small-signal and saturated responses. The nonlinearity is taken to be the AM/AM and AM/PM characteristics measured for the center frequency. Under small-signal conditions, the nonlinearity has transfer function $g(f_c) \exp[j\Phi(f_c)]$, so, except for these constants, which should be accounted for in the initial calibration, it can be seen that the signal sees $H_1(f)H_2(f) = H_{ss}(f)$, as desired. Under large-signal conditions, if we make the idealizing assumption that the amplifier acts like a limiter, the signal sees $H_2(f) = |H_{sat}(f)|$ as the amplitude function, which is correct; the phase function $\angle H_1(f) + \Phi(P_{sat}, f_c)$ still only approximates $\angle H_{sat}(f)$, though experimental observations indicate the approximation is fairly good.

5.3.3. NLWM Modeling III: Analytical Models

Our intended sense of *analytical* here is akin to its traditional use. Models are defined through an analytical formulation, though that formation may ultimately lead to interpretation in terms of some block structure, as opposed to beginning with some block structure which may then be described analytically. The “classic” analytical model is based on the Volterra series. A related approach, not as general, but easier to measure, uses so-called polyspectral techniques. We will discuss these two methods in this subsection.

5.3.3.1. Volterra Series Modeling

The *Volterra series approach*^(29–32) to modeling is appealing owing to its generality and its relatively intuitive decomposition of the response of a nonlinear system into that due to an

equivalent linear filter plus additional terms produced by nonlinear behavior. Thus, at least conceptually, this may offer a mechanism to describe the “mildness” of a nonlinearity. A Volterra series is described by

$$y(t) = \sum_{n=0}^{\infty} y_n(t) \quad (5.3.7)$$

where the term of order n is given by an n -fold convolution

$$y_n(t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \dots, \tau_n) x(t - \tau_1) \cdots x(t - \tau_n) d\tau_1 \cdots d\tau_n \quad (5.3.8)$$

Expressions (5.3.7)–(5.3.8) provide an input–output relationship for nonlinear time-invariant continuous systems. In simulation, we are of course forced to approximate the continuous system in discrete time. So in the above formulation, integrals are replaced with summations and t and τ are constrained to occur at discretely spaced intervals. The functions $h_0, h_1(t), h_2(t_1, t_2), \dots$ are called the *Volterra kernels* of the system. The system is thus completely characterized by its Volterra kernels. The zeroth-order term accounts for the response to a dc input. The first-order kernel is the impulse response of a linear system. The higher order kernels can be thought of as higher order impulse responses that are a manifestation of the nonlinear behavior of the system. A block-diagram interpretation of the model is shown in Figure 5.19. Without any assumptions on the nature of the kernels, it can be seen that the computation necessary to produce an output sample from the n th kernel box is the number of operations required for $n = 1$, raised to the n th power. Thus, it seems clear that unless the series can be truncated at some fairly small number of terms, the computational load will quickly become prohibitive. A simple example will be useful.

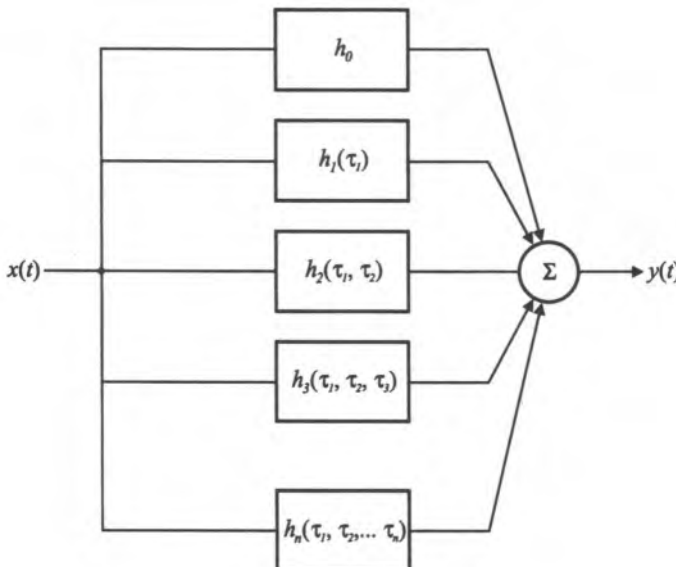


Figure 5.19. Block diagram interpretation of the Volterra series representation.

■ *Example 5.3.1.*⁽³²⁾ Consider the time-invariant nonlinear system shown in Figure 5.20, obtained by cascading a linear time-invariant system with impulse response $h(t)$ and a memoryless nonlinear system with a Taylor series expansion

$$y(t) = g[v(t)] = \sum_{n=0}^{\infty} a_n v^n(t)$$

The input–output relationship for the system can be expanded into

$$\begin{aligned} y(t) &= g \left[\int_{-\infty}^{\infty} h(\tau) x(t - \tau) d\tau \right] \\ &= a_0 + a_1 \int_{-\infty}^{\infty} h(\tau) x(t - \tau) d\tau \\ &\quad + a_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(\tau_1) h(\tau_2) x(t - \tau_1) x(t - \tau_2) d\tau_1 d\tau_2 + \cdots \end{aligned} \quad (5.3.9)$$

Comparing (5.3.9) with (5.3.7)–(5.3.8), we determine the Volterra kernels for the system as

$$\begin{aligned} h_0 &= a_0 \\ h_1(t) &= a_1 h(t) \\ h_2(t_1, t_2) &= a_2 h(t_1) h(t_2) \end{aligned} \quad (5.3.10)$$

The example is useful in illustrating how the Volterra kernels might be related to the system properties. It also illustrates the practical difference between knowing what the actual system is like (or having some engineering sense about its structure), and a black-box identification, which presumes no such insight. If we were to apply the actual identification procedure (described below for a particular case), we would simply obtain some set of kernels $h_n(t_1, \dots, t_n)$, $n = 1, 2, \dots$, as some functions of the arguments. Without seeking further possible relationships among the kernels, as in (5.3.10), we would incur a computational burden increasing as a power of n for the successive kernels. As we can see here, knowledge of the system topology would eliminate this increase since, in essence, we would have to compute only the response to the first-order kernel. ■

If the input $x(t)$ is a bandpass signal, (5.3.7)–(5.3.8) still apply, but if we are interested in the first-zone output of the system, a “bandpass” version of the above equations results, in a fashion similar to that for the power series nonlinearity in Section 5.2.3.2. By lengthy but straightforward computations it can be shown that, for the first-zone output, only terms with n odd can contribute. It can also be shown that for $k = 2m + 1$ there are $\binom{2m+1}{m}$ integrand factors of the type $x(t - \tau_i)$ that can contribute to the output. Thus, the low pass-equivalent output is

$$\tilde{y}(t) = \sum_{m=0}^{\infty} \tilde{y}_{2m+1}(t) \quad (5.3.11)$$

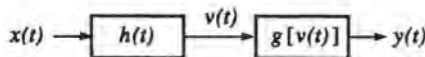


Figure 5.20. Time-invariant nonlinear system in Example 5.3.1.

where

$$\begin{aligned} \tilde{y}_{2m+1}(t) = & \frac{1}{2^{m+1}} \binom{2m+1}{m} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \tilde{h}_{2m+1}(\tau_1, \tau_2, \dots, \tau_{2m+1}) \\ & \times \prod_{i=1}^{m+1} \tilde{x}(t - \tau_i) \prod_{i=m+2}^{2m+1} \tilde{x}^*(t - \tau_i) d\tau_1 \cdots d\tau_{2m+1} \end{aligned} \quad (5.3.12)$$

The computation of the lowpass-equivalent Volterra kernels

$$\frac{1}{2^{m+1}} \binom{2m+1}{m} \tilde{h}_{2m+1}(\tau_1, \tau_2, \dots, \tau_{2m+1}), \quad m = 1, 2, \dots$$

is quite complex in general. Detailed derivations and examples can be found in the cited references.

Volterra series are a generalization of Taylor series, and have sometimes been referred to as “Taylor series with memory.” Just as the validity of a Taylor series imposes conditions on the function to be represented, systems to be represented by a Volterra series must also satisfy certain regularity conditions. In other words, not all nonlinear systems can be so represented. A simple example of a system that does not have a Volterra expansion is a hard-limiter. A certain type of smoothness is required for a system to be Volterra-expandable; conditions can be found, e.g., in Ref. 31. If a system satisfies these conditions, then, in principle at least, it is perfectly representable by a series of the type in (5.3.7)–(5.3.8). Assuming this is the case, two important questions arise: (1) Can we truncate the series to a reasonably small number of terms, a requirement for practical implementation? and (2) How do we identify the kernels $\tilde{h}_n$? These questions are interrelated, as will be seen.

As to the first, truncation is possible if the series is uniformly convergent in a certain sense, otherwise it is obvious that the series has no practical usefulness. In fact, it can be shown that for systems with a certain smoothness property,⁽³¹⁾ the error between the true output and that given by the truncated series can be made as small as desired. However, the number of terms needed in the partial sum may not be small for a given degree of closeness. With respect to the second point, the kernels can be identified with the procedure to be described. In this procedure one postulates that, say, K terms of the series are sufficient. The K corresponding kernels are determined from a set of measurements. One then compares the output predicted from the K identified kernels to the actual output, thus providing a measure of the model fidelity. Thus, the model is self-checking in this sense. If the calculated fidelity is inadequate, one can add another kernel and repeat the procedure.

With respect to the the second question posed above, another general comment should be made concerning the organization of the kernel identification procedure that applies to the classical Volterra series (VS) model, as well as to the generalized VS and the polyspectral models to be described below. In particular, the procedure for general (that is, actual) input/output signals will be provided first, followed by the derivation of the lowpass-equivalent (LPE) model from the generic model. The reasons for this sequence are that (1) no method of directly obtaining the LPE model from the measured input/output records exists, and (2) the treatment in Ref. 33, upon which the identification procedures given later were based, was not originally intended for LPE modeling.

The identification procedure inherently requires time-domain measurements of the input $x(t)$ and the output $y(t)$ of the device or system to be modeled, which we partition into a set of n_d pairs of input/output records, each of duration T . The number of records n_d and their duration T are determined by the desired frequency resolution of the model filters, and quantitative statistical error estimates for the various spectral quantities needed to determine

the model parameters. Details concerning these experimental design issues can be found in Ref. 33. To get a flavor of this procedure, we outline the necessary computations for a third-order model which is illustrated in Figure 5.21a. In this particular instance, we choose to identify the frequency-domain versions of the kernels, that is, H_1 is the Fourier transform of h_1 , H_2 is the two-dimensional Fourier transform of h_2 , and H_3 is the three-dimensional Fourier transform of h_3 . Define $X_i(f)$ as the finite-time Fourier transform of the i th record,[†]

$$X_i(f) = \int_0^T x_i(t) e^{-j2\pi ft} dt$$

with a similar definition for $Y_i(f)$ for the i th output record. Then:

1. Calculate the following autospectral, cross-spectral, and multispectral densities:

$$S_{xx}(f) = \frac{1}{T} E\{|X_i(f)|^2\} \quad (5.3.13a)$$

$$S_{xy}(f) = \frac{1}{T} E\{X_i^*(f) Y_i(f)\} \quad (5.3.13b)$$

$$S_{xxy}(f, g) = \frac{1}{T} E\{X_i^*(f) X_i^*(g) Y_i(f + g)\} \quad (5.3.13c)$$

$$S_{xxx}(f, g, h) = \frac{1}{T} E\{X_i^*(f) X_i^*(g) X_i^*(h) Y_i(f + g + h)\} \quad (5.3.13d)$$

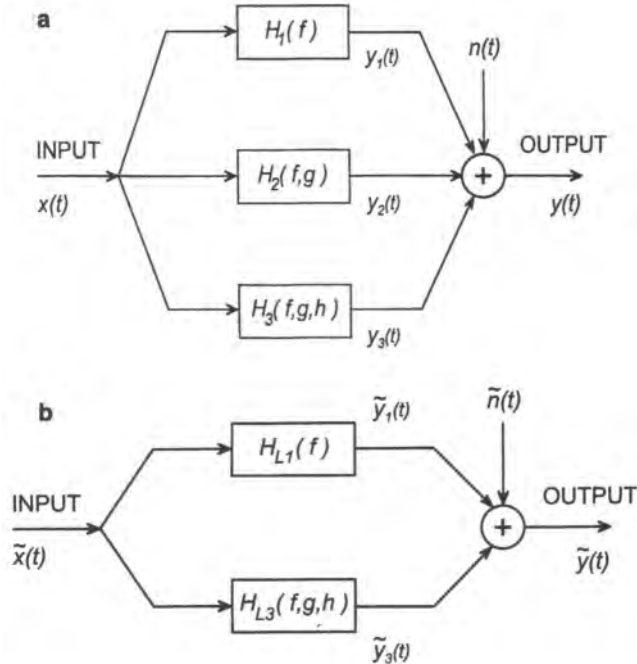


Figure 5.21. Topology of a third-order Volterra model. (a) Symbolic model at the carrier frequency level; (b) explicit model at the complex envelope level.

[†]The same quantity will be used in Chapter 10 in conjunction with spectral estimation.

where E represents the average over the different records (this can be thought of as a truncated version of an expectation).

2. The model quantities are then given as follows. The third-order kernel (transform) is

$$H_3(f, g, h) = \frac{S_{\text{xy}}(f, g, h)}{6S_{\text{xx}}(f)S_{\text{xx}}(g)S_{\text{xx}}(h)}$$

The second-order kernel is

$$H_2(f, g) = \frac{S_{\text{xy}}(f, g)}{2S_{\text{xx}}(f)S_{\text{xx}}(g)}$$

and the first-order kernel is given by

$$H_1(f) = H_o(f) - C(f)$$

where the optimal filter $H_o(f)$ is given by

$$H_o(f) = \frac{S_{\text{xy}}(f)}{S_{\text{xx}}(f)}$$

which represents the best linear approximation to the given input/output records, and

$$C(f) = 3 \int_{-\infty}^{\infty} H_3(w, -w, f) S_{\text{xx}}(w) dw$$

It is worthwhile pointing out that $H_1(f)$ is generally not the same as the small-signal response mentioned in connection with some of the earlier models. Calculation of the error in the model uses the same functions as already defined in (5.3.13). The reader is referred to Ref. 33 for details.

In a communications context the probing signal used to identify the model would typically be a bandpass-modulated signal with signaling rate appropriate to the bandwidth of interest. Hence, even though we may wish to obtain an LPE model (the first zone about the carrier), the actual input/output signals are bandpass functions about the carrier frequency (or some suitable intermediate frequency). The actual *records* or measurements of these functions could in principle also be measured directly at some RF or IF frequency, or obtained in a generally more accurate fashion using the time-domain measurement technique described in Section 5.5.3. In this case, the in-phase and quadrature components are the measured input/output records; that is, we measure $z_I(t)$ and $z_Q(t)$ ($z = x$ or y), which are described by the usual relationship

$$z(t) = z_I(t) \cos(2\pi f_c t) - z_Q(t) \sin(2\pi f_c t)$$

where f_c is again the carrier (or IF) frequency.

Figure 5.21b depicts the LPE model for the special third-order case. Here the linear filter denoted $H_{L1}(f)$ is the usual LPE of the bandpass filter $H_1(f)$ identified in the general model. In accordance with the statements made before (5.3.11) and (5.3.12), note that the second-order branch of the general model must be dropped in the LPE model since it does not

contribute to the first-zone output. Finally, in a generalization of the linear filter case, the LPE third-order transform is given by

$$H_{L3}(f, g, h) = (3/4)H_3(f + f_c, g + f_c, h - f_c)U_3(f + f_c, g + f_c, h - f_c)$$

where $U_3(\cdot, \cdot, \cdot)$ is a special three-dimensional version of the unit step function equal to unity when $f \geq 0$, $g \geq 0$, and $h \leq 0$ and zero otherwise. When using the LPE model in simulation, the third-order branch contribution is calculated in the frequency domain as follows:

$$\tilde{y}_3(f) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} H_{L3}(u, v - u, f - v) \tilde{X}(u) \tilde{X}(v - u) \tilde{X}^*(v - f) du dv$$

Note that this calculation requires only a double integral, whereas in the time domain we need to calculate a triple integral. As usual, an efficient and fast inverse FFT routine can be used to obtain the desired time-domain output component $\tilde{y}_3(t)$.

As we have noted, unless we can have a reasonably faithful model with very few kernels, the Volterra series model will not be computationally practical. There is, however, another utility to the generality of the Volterra series representation, namely its ability to provide insight into the structure of a nonlinear compensator. Because the identification is a time-domain procedure, we can reverse the role of input, $x(t)$, and (measured) output, $y(t)$, and ask what structure would produce $x(t)$ at its output given $y(t)$ as the input.

The classical Volterra series just described can be thought of as a McClaurlin functional series for the device's input/output operation, that is, it is an expansion of the device's input/output operator (with memory) about the *zero* function. Consequently, the predicted response to any input signal that is significantly different from the zero function may be quite inaccurate and may not even exist. Since such input signals occur often in practice, a more general Taylor-type expansion about a nonzero input basis signal can be proposed. By choosing the basis signal to be "close" to the signals of interest, the resulting model can be expected to possess superior predictive fidelity. This is the essence of the so-called *generalized Volterra series* model.⁽³⁴⁻³⁶⁾

One manifestation of this generalization develops a classical Volterra series model for the input/output *deviation signals*

$$\Delta x(t) \equiv x(t) - x_s(t), \quad \Delta y(t) \equiv y(t) - y_s(t) \quad (5.3.14)$$

where $y_s(t)$ denotes the response of the device to the input signal $x_s(t)$, with the latter chosen to lie close to the signals of interest. As a consequence, the series is described by [compare with (5.3.7) and (5.3.8)]

$$y(t) = y_s(t) + \Delta y(t) = y_s(t) + \sum_{n=1}^{\infty} \Delta y_n(t) \quad (5.3.15a)$$

where

$$\Delta y_n(t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \dots, \tau_n) \prod_{i=1}^n \Delta x(t - \tau_i) d\tau_i \quad (5.3.15b)$$

Note that the Volterra kernels $\mathbf{h}_n$ in (5.3.15b) will *not* be the same as the ones in (5.3.8) since they describe the device's deviation behavior about the $\mathbf{x}_s - \mathbf{y}_s$ input/output pair. Most of the model details discussed above for the classical Volterra series carry over to the generalized case by making the appropriate substitutions of $\Delta \mathbf{x}(t)$, $\Delta \mathbf{y}(t)$ for $x(t)$, $y(t)$, respectively. For example, the bandpass first-zonal form of (5.3.15) would be as in (5.3.11) and (5.3.12) with the appropriate substitutions. A similar statement holds for the general model identification procedure that results in the transforms $H_1(f)$, $H_2(f, g)$, and $H_3(f, g, h)$. Note that these transforms will be different from the ones obtained using the original input/output records.

There are two observations about convergence and accuracy that can be deduced from comparing (5.3.7) and (5.3.8) with (5.3.15):

1. For a given input signal that is local to $\mathbf{x}_s(t)$, the classical series may not converge or provide a reasonable approximation to $y(t)$, while the generalized series is more likely to converge and be accurate, especially if the basis signal is chosen intelligently.

2. Because the deviation $\Delta \mathbf{x}(t)$ will be smaller than $x(t)$ in a norm sense, and since both of these terms essentially appear to the n th power in the n th term of the series, it is clear that for moderately similar Volterra kernels, the generalized series will produce a given level of accuracy with much fewer terms than the classical series. Because of this, the former series will be able to handle a much larger degree of nonlinearity in the device, dispensing with the common weak nonlinearity assumption that is required for the classical case. This is a very important advantage because of the well-known fact that the experimental determination of the Volterra kernels becomes essentially impractical—in terms of the number of measurements, data processing complexity, and maintenance of error levels—very quickly with respect to order.

A third-order instance of this model for general inputs is shown in Figure 5.22a, which can be contrasted with the earlier classical model illustrated in Figure 5.21a. As in the latter figure, the quantity $n(t)$ represents the modeling error whose power spectral density $S_{nn}(f)$ can be calculated from the input/output time-domain records. In particular, for both types of models, this spectral density is given by

$$S_{nn}(f) = \begin{cases} S_{yy}(f) - \sum_{i=1}^3 S_{y y_i}(f) & \text{(classical VS)} \\ S_{\Delta y \Delta y}(f) - \sum_{i=1}^3 S_{\Delta y_i \Delta y_i}(f) & \text{(generalized VS)} \end{cases} \quad (5.3.16)$$

Note from (5.3.14) that

$$S_{yy}(f) = S_{(y_s + \Delta y)(y_s + \Delta y)}(f) = S_{y_s y_s}(f) + S_{\Delta y \Delta y}(f) + 2 \operatorname{Re}[S_{y_s \Delta y}(f)] \quad (5.3.17)$$

Under most circumstances the PSD of $y_s(t)$ will dominate in (5.3.17), suggesting that the relative modeling error, defined through

$$\rho(f) \equiv \frac{S_{nn}(f)}{S_{yy}(f)}$$

will be substantially less for the generalized series, in view of (5.3.16). In other words, the generalized Volterra series model will provide increased predictive fidelity for the same model

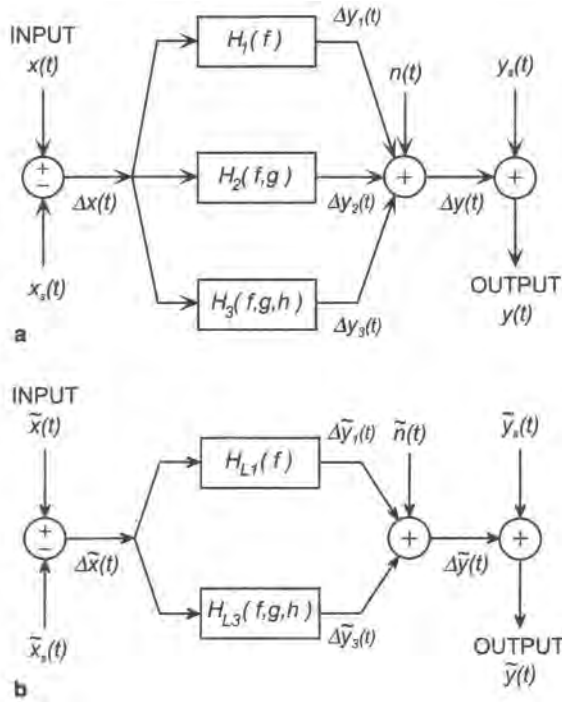


Figure 5.22. A third-order instance of the generalized Volterra series model that corresponds to the expansion of the system's input/output operator about a chosen signal $\mathbf{x}_s(t)$. (a) Symbolic model at the carrier frequency level; (b) explicit model at the complex envelope level.

order. Note that for good models, $\rho(f)$ should be small at frequencies of interest, corresponding to where the output power density $S_{yy}(f)$ is largest.

In a communications context where $x(t)$ is a bandpass-modulated signal, a natural choice for the basis signal $\mathbf{x}_s(t)$ is the carrier itself, with $y_s(t)$ then usually being another tone at the same frequency. Note that the deviation signals in (5.3.14) will still be bandpass, since the subtraction operation does not translate the spectrum. As with the classical VS model, accurate measurements will still involve input/output records of the I- and Q-signal components. The third-order LPE model corresponding to the general model is shown in Figure 5.22b, which can be compared with Figure 5.21b. The LPE filters $H_{L1}(f)$ and $H_{L3}(f, g, h)$, as well as the calculation of the third-order branch contribution to the model output deviation, are obtained in the same manner as for the classical third-order VS model. For the case in which $\mathbf{x}_s$ and $\mathbf{y}_s$ are unmodulated tones at the same frequency, the LPE signals $\tilde{\mathbf{x}}_s(t)$ and $\tilde{\mathbf{y}}_s(t)$ will be just complex constants.

5.3.3.2. Polyspectral Models

Perhaps the greatest source of difficulty with the model construction and execution of the Volterra series approaches is their inherent multidimensionality both in time and frequency. This section will present two one-dimensional cases of the Volterra series that result in a much more manageable—but still “analytical”—*polyspectral model* that possesses several desirable features. First, the model construction requires simple input/output random time-domain

datasets and standard random data processing to arrive at the model parameters. There is a systematic procedure to design the probing signals to use in the construction process, and commercial software is available to automate the model development.⁽³⁷⁾ All spectral calculations in this technique involve only a *single* frequency variable instead of several, significantly reducing computational complexity and measurement/modeling error. The basic structure of the model consists of parallel signal paths, one of which is linear, while the remainder contain user-chosen memoryless nonlinearities and filters whose responses are calculated from the input/output measurements (and thus serve as the principal variables for the model). The method also provides a means to quantify the model fidelity just from the input/output records, which can be further enhanced by the addition of more signal paths. As with the Volterra series, this approach is well suited for compensation design since a reverse model can easily be obtained with a structure that is natural for hardware implementation. A thorough treatment of this modeling methodology (as well as the classical Volterra series approach up to third order), including its several variations, data record design and reduction, identification procedures, error analysis, and representative applications, can be found in Ref. 33.

5.3.3.2.1. Nonlinearity–Filter Polyspectral Model. A general form of this type of model is shown in Figure 5.23a, consisting of a linear path with undetermined filter response $H_1(f)$, one nonlinear path with a specified, but arbitrary memoryless nonlinearity $g(\cdot)$, followed by the undetermined filter response $H_2(f)$ (thus giving rise to the name of the model). One possible choice for $g(\cdot)$ would be the measured static AM/AM and AM/PM conversion curves. The outputs $y_h(t)$ and $y_v(t)$ of the two paths can be correlated. The optimal filter responses $H_i(f)$ ($i = 1, 2$) are determined from the measured input/output data, and $n(t)$ embodies the error noise of the model, which is also quantitatively determined and corresponds to the model's predictive fidelity. The direct *multiple-input, single-output* (MISO) approach then transforms this model into an equivalent linear one that has two uncorrelated inputs and one output. If the predictive fidelity is not sufficient, another

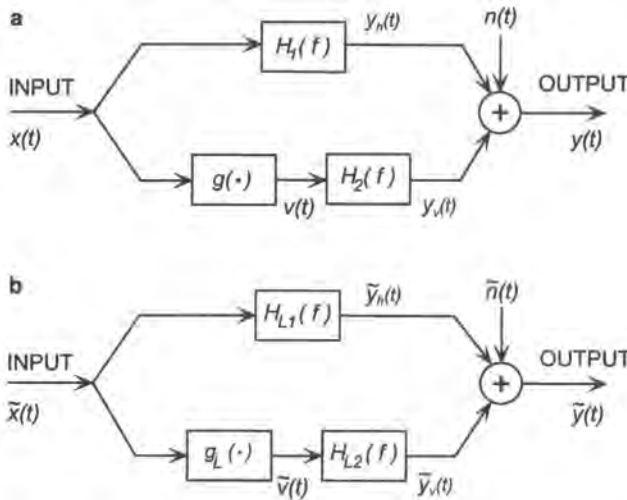


Figure 5.23. An instance of a polyspectral model with the nonlinearity–filter topology, (a) Symbolic model at the carrier frequency level; (b) explicit model at the complex envelope level, where $g_L(\cdot)$ depends on the nature of the memoryless nonlinearity $g(\cdot)$.

candidate for $g(\cdot)$ can be chosen (such as the dynamic AM/AM and AM/PM conversions) or additional chosen nonlinearity–undetermined filter paths can be included.

The general identification procedure for this model class is quite similar to the one outlined for the classical Volterra series in the previous section. If necessary, additional measurements are made to arrive at the memoryless nonlinearity $g(\cdot)$ (such as the dynamic AM/AM and AM/PM curve measurements outlined in Section 5.5.2). The following successive steps would then be taken to arrive at the model parameters:

1. Choose the memoryless nonlinearity $g(\cdot)$ from measurements, physical intuition, or other means.
2. Calculate the output $v(t)$ of the memoryless nonlinearity from the input $x(t)$ using the procedure presented in Section 5.2.4.
3. Calculate the following autospectral and cross-spectral densities using the records for $x(t)$, $v(t)$, and $y(t)$

$$S_{ab}(f) := \frac{1}{T} E\{A^*(f)B(f)\}, \quad (a, b) \in \{(x, x), (x, y), (v, v), (v, x), (v, y), (y, y)\} \quad (5.3.18a)$$

where E again represents the average value taken over the n_d records of length T , $A(f)$ denotes the finite Fourier transform of $a(t)$, and similarly for $B(f)$. For the case $a \neq b$, the complex cross-spectral density $S_{ab}(f)$ satisfies the relation

$$S_{ba}(f) = [S_{ab}(f)]^* \quad (5.3.18b)$$

4. Calculate the real linear (ordinary) *coherence function*

$$\gamma_{xv}^2(f) := \frac{|S_{xv}(f)|^2}{S_{xx}(f)S_{vv}(f)} \in [0, 1] \quad (5.3.19)$$

which measures the degree of linear relationship between the $x(t)$ and $v(t)$ records, with perfect linearity indicated by a unity value for all f .

5. Calculate the following spectral densities for the output signal $u(t)$ of a revised memoryless nonlinearity that is completely uncorrelated with $x(t)$:

$$S_{uu}(f) = S_{vv}(f)[1 - \gamma_{xv}^2(f)], \quad S_{uy}(f) = S_{vy}(f) - \frac{S_{vx}(f)S_{xy}(f)}{S_{xx}(f)} \quad (5.3.20)$$

6. The two filters in the model are then given by

$$H_2(f) = \frac{S_{vy}(f)}{S_{uu}(f)} \quad (5.3.21a)$$

and

$$H_1(f) = H_o(f) - C(f) \quad (5.3.21b)$$

where the *optimal* linear filter $H_o(f)$ is as for the classical Volterra series case, while

$$C(f) = \frac{S_{xy}(f)}{S_{xx}(f)} H_2(f) \quad (5.3.21c)$$

instead. Again observe that the filter $H_1(f)$ is not necessarily equal to the small-signal measurement through the nonlinear device, as for some of the n -box models ($n \geq 2$).

The fidelity of the model is again measured by the ratio $\rho(f)$ defined in the preceding subsection. The details for determining this quantity and explicit statistical error estimates for the two derived filters can be found in Ref. 33. The expressions given here can be contrasted with the ones presented earlier for the classical third-order Volterra series. The main difference is in the appearance of only one frequency here, as opposed to the three frequencies that have to be handled there, a much more complex task that requires many more time-domain input/output records to achieve the same model fidelity. To be more concrete, suppose the memoryless nonlinearity $g(\cdot)$ is taken to be the finite power series

$$v(t) = g[x(t)] = \sum_{n=0}^N a_n x^n(t)$$

Then it can be shown that (see Problem 5.28) the nonlinear branch will have Volterra kernels of order zero to N with the n th-order kernel given by

$$h_0 = a_0 H_2(0), \quad h_n(\tau_1, \dots, \tau_n) = \begin{cases} a_1 h(\tau_1), & n = 1 \\ a_n h(\tau_1) \prod_{m=2}^n \delta(\tau_1 - \tau_m), & n = 2, \dots, N \end{cases} \quad (5.3.22a)$$

where $h(\cdot)$ denotes the ordinary impulse response of the filter $H_2(f)$, and $\delta(\cdot)$ represents the Dirac delta function. The corresponding results in the frequency domain are

$$H_0(f) = a_0 H_2(0) \delta(f), \quad H_n(f_1, \dots, f_n) = \begin{cases} a_1 H_2(f), & n = 1 \\ a_n H_2\left(\sum_{m=1}^n f_m\right), & n = 2, \dots, N \end{cases} \quad (5.3.22b)$$

We thus see from (5.3.22) how such a nonlinearity-filter branch can result in an N th-order Volterra system, but one whose identification is much easier since it is done all at once using a single frequency as outlined above. In addition, any general Volterra system that has order greater than three becomes impractical very quickly.

The general comments made for the VS model in a communications context apply here as well. In contrast, however, the LPE model here (see Figure 5.23b) depends on the type of nonlinearity chosen for $g(\cdot)$. If the nonlinearity is chosen to be of the envelope type (as suggested earlier), then the LPE model follows almost immediately from the general model since the nonlinearity remains unchanged [that is, $g_L(\cdot) \equiv g(\cdot)$], while the two linear filters are replaced with their standard LPE versions. If the nonlinearity is chosen to be the finite power

series given earlier, then it can be shown that (Problem P5.29) the LPE model is the same as the previous one except that g_L is given by

$$g_L(\tilde{x}) = \sum_{m=0}^{M-1 \text{ or } M} \binom{2m+1}{m} \frac{a_{2m+1}}{2^{2m}} \tilde{x}^{m+1} (\tilde{x}^*)^m$$

where the upper limit of the sum is $M - 1$ (M) when $N = 2M$ ($2M + 1$). In this case the LPE nonlinearity output $\tilde{v}(t)$ would be calculated in the time domain using the given expression before being transformed to the frequency domain in order to pass through the filter H_{L2} . If the nonlinearity is of a more general form, the most straightforward approach to obtain the LPE model would be to approximate it with a finite power series and use the results just presented.

5.3.3.2.2. Filter–Nonlinearity Polyspectral Model. This form of the model may possess enhanced predictive power afforded by a filter–nonlinearity arrangement (which mimics the structure naturally found in TWT amplifiers, for example), but this must be traded off against the increased complexity of its measurement and construction and its reduced generality in the specified branch nonlinearities. A third-order instance of this type of model is illustrated in Figure 5.24a, consisting again of a linear path with undetermined filter response $H_1(f)$, two nonlinear paths with a specified squarer and cuber memoryless nonlinearity, followed by an undetermined filter response $H_2(f)$ and $H_3(f)$, respectively (thus giving rise to the name of the model). In this case, the outputs $y_i(t)$, $i = 1, 2, 3$, can be correlated; the outputs $y_i(t)$, $i = 2, 3$, of the memoryless nonlinearities are calculated trivially from their general inputs $u_i(t)$, $i = 2, 3$; the optimal filters $H_i(f)$, $i = 1, 2, 3$, are again determined from the measured input/output data; and $n(t)$ is as for the nonlinearity–filter model. The modeling approach transforms this model into an equivalent one that has three uncorrelated path outputs. If the predictive fidelity is not sufficient, it is conceivable that more nonlinear paths could be added, although the complexity of the undetermined response calculations quickly escalates. A form of the model in which there is a single general memoryless nonlinearity instead of the two monomial nonlinearities (as in the previous model) has not been developed to date. We will only cover the three-path case chosen here for the sake of simplicity in describing the identification procedure.

The general identification procedure for this model class is quite similar to the one outlined for the nonlinearity–filter polyspectral model. In fact, the initial comments about the probing signal, experimental design, and measurement recommendations are as given there. The following successive steps then lead to the model parameters:

1. Calculate the following autospectral, cross-spectral, and polyspectral densities using the measured $x(t)$ and $y(t)$ data records [compare with (5.3.13) for the classical Volterra series]:

$$S_{xx}(f) = \frac{1}{T} E[|X(f)|^2] \quad (5.3.23a)$$

$$S_{xy}(f) = \frac{1}{T} E[X^*(f)Y(f)] \quad (5.3.23b)$$

$$S_{xxy}(f) = \frac{1}{T} E[X^*(f)X^*(f)Y(2f)] \quad (5.3.23c)$$

$$S_{xxxy}(f) = \frac{1}{T} E[X^*(f)X^*(f)X^*(f)Y(3f)] \quad (5.3.23d)$$

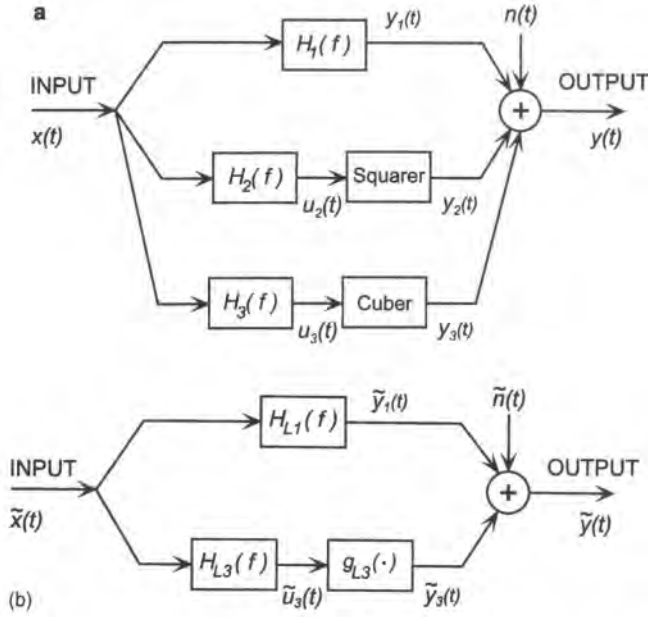


Figure 5.24. An instance of a polyspectral model with two filter–nonlinearity branches. (a) Symbolic model at the carrier frequency level; (b) explicit model at the complex envelope level.

with the quantities as previously defined.

2. The three filters in the model are then calculated as follows. First,

$$H_3(f) = \frac{[S_{\text{xxx}}(f)/6]^{1/3}}{S_{\text{xx}}(f)} \quad (5.3.24a)$$

and then

$$H_2(f) = \frac{[S_{\text{xy}}(f)/2]^{1/2}}{S_{\text{xx}}(f)} \quad (5.3.24b)$$

so that

$$H_1(f) = H_o(f) - 3\sigma_3^2 H_3(f) \quad (5.3.24c)$$

where the *optimal* linear filter $H_o(f)$ is as for the classical Volterra series case, and

$$\sigma_3^2 \equiv \int_{-\infty}^{\infty} |H_3(w)|^2 S_{\text{xx}}(w) dw \quad (5.3.24d)$$

is the variance of the output $u_3(t)$ of $H_3(f)$.

The same comments about $H_1(f)$, comparison with classical n -box models and the classical Volterra series models, and the fidelity of the model made previously here as well. Using the results of Example 5.3.1 above, it is easy to show that the n th branch of the

filter–nonlinearity model ($n = 2, \dots$) is the following special case of the Volterra series model:

$$h_n(\tau_1, \dots, \tau_n) = \prod_{m=1}^n h(\tau_m) \quad (5.3.25a)$$

where $h(t)$ is the ordinary impulse response of the filter in the branch. In the frequency domain, this translates to the relation

$$H_n(f_1, \dots, f_n) = \prod_{m=1}^n H(f_m) \quad (5.3.25b)$$

We also see from (5.3.25) how $N - 1$ such filter–nonlinearity branches, plus the linear path, can result in an N th-order Volterra system, but one whose identification is still substantially easier since it involves only a single frequency as outlined above.

Again the comments made for the VS model in a communications context apply here also. For example, if this model is to be used for systems containing bndlimited RF signals, then the squarer branch would not contribute to the first zone and can be dropped (as can any even-powered term). Under these circumstances, the LPE model shown in Figure 5.24b can be derived as follows. First, the linear path would be replaced straightforwardly with its LPE model in the usual way. The filter–cuber branch is not so straightforward, but can be shown to become a special third-order LPE Volterra kernel $\frac{3}{8}\tilde{h}_3(\tau_1, \tau_2, \tau_3)$, where (Problem 5.30)

$$\tilde{h}_3(\tau_1, \tau_2, \tau_3) = \frac{1}{4}\tilde{h}(\tau_1)\tilde{h}(\tau_2)\tilde{h}^*(\tau_3)$$

and $\tilde{h}(\tau)$ corresponds to the impulse response of $H_3(f)$. The LPE third-order transform is then given by

$$H_{L3}(f, g, h) = \frac{3}{4}H_{L3}(f)H_{L3}(g)H_{L3}^*(-h)$$

where H_{L3} is the LPE of H_3 . The output $\tilde{y}_3(t)$ of the resulting LPE branch is given by

$$\tilde{y}_3(t) = \frac{3}{8} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{h}_3(\tau_1, \tau_2, \tau_3) \tilde{x}(t - \tau_1) \tilde{x}(t - \tau_2) \tilde{x}^*(t - \tau_3) d\tau_1 d\tau_2 d\tau_3$$

Proceeding in a manner similar to that for the nonlinearity–filter polyspectral model, we can show the LPE equivalent of the filter–cuber branch to be (Problem P5.23) as in Figure 5.24b with $H_{L3}(f)$ as above, and

$$g_{L3}(\tilde{x}) = \frac{3}{4}\tilde{x}^2\tilde{x}^*$$

Again the execution of this model is best done in the time domain. These expressions readily generalize to higher odd-powered monomial nonlinearities. However, the determination of the bandpass filter $H_{2m+1}(f)$ in front of the nonlinearity becomes increasingly difficult and inaccurate, since it requires the calculation of cross-spectral densities up to order $2(m + 1)$.

Because both of the polyspectral models presented here can be thought of as special cases of the classical Volterra series, it follows that they both also can be considered functional expansions about the *zero* input function. Similar to the generalized Volterra series, both of the polyspectral models could be extended as expansions about a chosen nonzero input signal $x_s(t)$.

5.3.4. NLWM Modeling IV: Miscellaneous Models

There is a large number of papers on the subject of nonlinear models. The preceding classification attempts to organize them into conceptually useful groupings. There are many variations of two-box models and three-box models. For example, in all such models we have to somehow define a pair of memoryless nonlinear functions (the AM/AM and AM/PM characteristics). The way these characteristics are obtained, i.e., the test signal used, could generate a significant number of variations of such models. For example, the nonlinear characteristics could be determined by single-tone measurements under small signal conditions, or under saturated-signal conditions; they could be determined by two-tone test signals (as discussed in Section 5.4.2.3); or by any number of test signal variations. The 2-box or 3-box construct could be extended into an n -box model: for example, the PSB model discussed earlier could be characterized as a five-box model. Not all models fit neatly (or even not so neatly) into particular categories. For example, there are some models that employ feedback. These might be characterized as feedback models. On the other hand, all such models comprise some set of “boxes” of one type or another, hence might also be classed as n -box models. Of course, it is not the creation of categories that is a goal in itself; that is merely an aid in visualization. In any case, in the remainder of this section we briefly describe three modeling constructs that are not easily placed into one of the previously defined categories. They present alternative ways of constructing NLWM models.

5.3.4.1. Power-Dependent Transfer Function Model

As is apparent from the previous subsections, most of the NLWM models are constructed from swept-tone, swept amplitude measurements. As we have already remarked, such models are likely to be inaccurate to some degree because superposition does not hold in nonlinear devices. Hence a model built on a more complex test signal holds the possibility of greater accuracy, at least in principle. We already alluded to such a possibility in Section 5.4.2.3 where we discussed a two-tone test for measuring nonlinear characteristics. A further step in this direction can be taken by using a test signal which more closely resembles the operational signal(s) that might be used. The model about to be discussed as well as the next one, though structurally different, are based on waveform inputs.

For systems transmitting digital signals by QAM modulation, a more natural test signal is a binary PN sequence having an appropriate period and chip rate. Such a signal has, of course, a multitone (line) spectrum $S_i(f)$. Since the NLWM is frequently selective it follows that the output spectrum $S_o(f)$ will be different. This defines an effective transfer function $H(f) = S_o(f)/S_i(f)$. However, we know that the response of an NLWM also depends on the operating point, i.e., the input power P . Thus, in general we can conceive of characterizing the device two-dimensionally with a set of transfer functions $\{H(f; P)\}$. The input power P must be interpreted dynamically here. As with any other frequency selective device, the NLWM has an effective memory T_M . Hence, the response at any time should be minimally affected by inputs older than T_M . An implementation of this model for an actual input signal (which could have fluctuations associated with it) uses a running DFT technique (see, e.g., Ref. 1 in Chapter 3) and takes the following form.⁽²⁴⁾ The input signal power is measured over intervals T_M in duration on a sliding basis; in other words successive T_M -long intervals overlap and are advanced by one simulation sample step T_s . The tune samples corresponding to the i th T_M -long interval, with measured power P_i , are discrete Fourier transformed, multiplied by the appropriate $H(f; P_i)$ and inverse transformed. Because the signal is processed on a sliding

block basis only a *single* output sample is retained (the most recent one) from each transform operation. Thus, for this seemingly closer approach to “reality” a price is paid in computational complexity.

5.3.4.2. Nonlinear Parametric Discrete-Time Models

We have already discussed in the previous section the use of the Volterra series to characterize nonlinear continuous systems. This series can also be used to represent nonlinear discrete systems wherein the construction can be performed adaptively⁽³⁸⁾ or nonadaptively.⁽³⁹⁾ In the Volterra series approach, the system’s nonlinear input–output operator is expanded as a series with memory. However, other nonanalytic forms of nonlinear system identification exist wherein a general nonlinear difference equation or related structure is chosen with the goal of parametrically capturing the given system’s nonlinear behavior. This class of nonlinear models is perhaps a natural extension of the digital filter structures that we have seen in the previous chapter for modeling linear systems. Numerous parameter estimation algorithms have been developed for linear models. However, this is not yet the case for nonlinear models. The model structures for these approaches can be either nonrecursive or recursive, and can contain multiple inputs and outputs. The representations using recursive structures can often model nonlinear systems more efficiently than the Volterra series approach⁽⁴⁰⁾; however stability must be considered due to feedback. A general, formal treatment of the existence and validity of recursive models can be found in Ref. 41. We will restrict this discussion to single input–output models because of their simplicity and wide use in communication systems.

In general, nonlinear discrete-time parametric models are most easily identified directly from time-domain measurements. This often simplifies the data acquisition task when compared to commonly used frequency-domain approaches. The primary drawback to this approach is the complexity of model parameter computation, whether performed adaptively or nonadaptively. For successful modeling, engineering judgment is therefore critical in (1) the initial selection of the model configuration, (2) the design of the signals used to identify the parameters, and (3) determining the parameter extraction strategy (initial parameter values and optimization algorithm). In principle, these models can be represented in either baseband or bandpass form. Nonlinear parametric discrete-time models are usually constructed from the following basic elements: (1) adders, (2) linear and nonlinear gain, (3) linear and nonlinear time delays, and (4) multipliers. Among the many possible structures, Figures 5.25 and 5.26 show representative nonrecursive and recursive nonlinear parametric models which

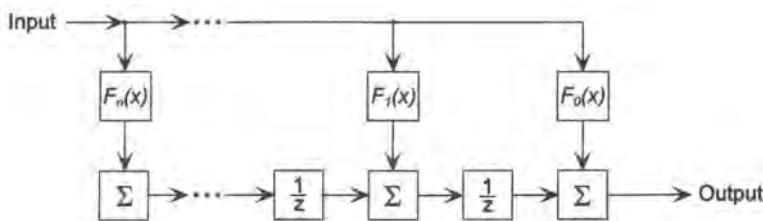


Figure 5.25. General structure of a nonrecursive nonlinear parametric model. The structure holds for both the baseband and bandpass models, except that the branch functions F_i are complex in the latter case.

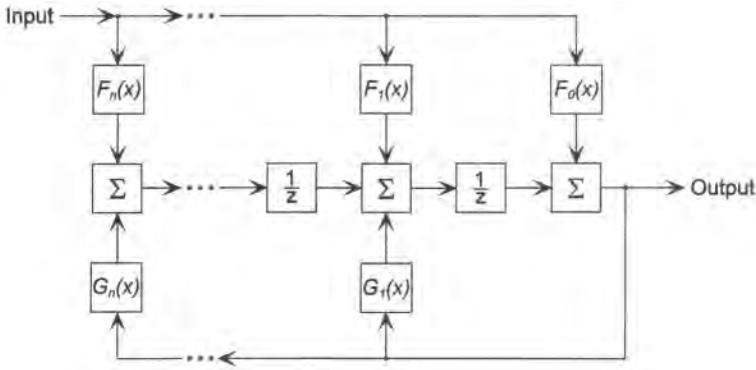


Figure 5.26. General structure of a recursive nonlinear parametric model. The structure holds for both the baseband and bandpass models, except that the branch functions F_i and G_i are complex in the latter case.

are derived from traditional linear FIR and IIR structures, respectively. The output of the models at the discrete times $t = kT$, where T is the chosen sampling interval, is given by

$$y(k) = \sum_{i=0}^n F_i[x(k-i)] \quad (\text{Nonrecursive}) \quad (5.3.26a)$$

$$y(k) = \sum_{i=0}^n F_i[x(k-i)] + \sum_{i=1}^n G_i[y(k-i)] \quad (\text{Recursive}) \quad (5.3.26b)$$

where $F_i(\cdot)$ and $G_i(\cdot)$ are nonlinear functions. In both cases, note that the (constant) tap coefficients normally associated with linear filters have been replaced with more general nonlinear functions, allowing for such common nonlinear effects as AM/AM and AM/PM conversion. The models can be thought of as input-dependent filters that naturally capture device memory through their feedforward and feedback structures. An example of both nonrecursive and recursive models for nonlinear amplifiers is given next.

Probably the first form of a nonlinear nonrecursive model was the baseband model proposed in Ref. 42. In this model, a power-series expansion of amplitude nonlinearities is extended to include phase nonlinearities through the use of order-dependent time delays. For commensurate delays, this continuous-time model can be considered a special case of Figure 5.25 when converted to a discrete-time model. This is due to the fact that only a single power nonlinearity is used on each tap. In this early approach, the model parameters were extracted from single-tone AM/AM and AM/PM measurements. A model for a TWT amplifier was developed with reasonable accuracy in predicting intermodulation effects. One enhancement of this model would be to use a more general tap nonlinearity (for example, a full power series) and time-domain measurements for model construction. This form is a general characterization which would be more predictive for typical communications signals.

A nonlinear recursive model developed for a microwave solid-state amplifier is discussed in Ref. 43. In this case, the hyperbolic tangent function, which resembles the nonlinear saturation characteristic, is chosen as the nonlinear function. For identifying the model parameters, a pulsed incident waveform is used for optimal accuracy and efficiency. The pulse duration is chosen such that its usable spectrum covers the input operating frequency of the device. The range of pulse amplitudes is chosen to cover linear through saturated operation. Since the measured responses to large-signal positive and negative inputs are different,

different nonlinear functions are required for positive and negative inputs. The nonlinear functions are therefore given by

$$F_i[x(k)] = \begin{cases} b_{ip} \tanh[k_{p1}x(k)], & x(k) \geq 0 \\ b_{in} \tanh[k_{n1}x(k)], & x(k) < 0 \end{cases} \quad (5.3.27a)$$

and

$$G_i[x(k)] = \begin{cases} a_{ip} \tanh[k_{p2}x(k)], & x(k) \geq 0 \\ a_{in} \tanh[k_{n2}x(k)], & x(k) < 0 \end{cases} \quad (5.3.27b)$$

where a_{ip} , a_{in} , b_{ip} , b_{in} , k_{p1} , k_{n1} , k_{p2} and k_{n2} are constants that are identified using measured pulse response time-domain data along with (5.3.26b). The resulting model was used to accurately predict measured gain compression and harmonic distortion of the solid-state amplifier.

5.3.4.3. Instantaneous Frequency Model

The basic idea behind this model is to interpret the swept-tone, swept-amplitude measurements (AM/AM and AM/PM) in a dynamic sense. These measurements, as displayed in Figure 5.10, provide amplitude and phase versus input power for a set of frequencies. With a modulated signal, we can visualize that signal traversing these curves back and forth as time evolves. This vision requires that we view the signal in terms of its frequency content, specifically its instantaneous frequency $f_i(t)$, where t must of course be taken as discrete time in our context. Certainly, if $f_i(t)$ dwells at any particular value for a relatively long period of time, we might equate that condition with the one under which the curves were originally obtained. The crux of the matter is quantifying “relatively long.” But, as with many of the models discussed before, we can make a modeling leap and assume that this conception will work well enough, an assumption which can ultimately be checked only by comparison with other types of measurements.

Thus, to calculate $f_i(kT_s)$, we would look up the corresponding output amplitude and phase (and interpolate when necessary) for each succeeding discrete-time instant. The instantaneous frequency is, by definition, the derivative of the phase. In discrete time, the derivative can be a somewhat unreliable quantity because it involves differences of nearly equal numbers divided by a small number. There are several options for calculating the discrete-time derivative, which are discussed briefly in Section 8.8.2, to which the reader is directed. In order to look up the amplitude or phase, given the instantaneous frequency, we have to assign a value to the independent variable—the input power or envelope. It is again not entirely obvious how to define this quantity. But since the discrete-time derivative involves at least two discrete-time instants, it seems reasonable for this purpose to take the input power to be the average of the square of the modulus of the complex envelope, the number of points in that average being the number of points involved in the calculation of the instantaneous frequency.

A version of this model is shown in block diagram form in Figure 5.27. Here, we combine the instantaneous frequency idea with the appealing notion used earlier that the model produce the small-signal transfer function at the corresponding input condition. Thus,

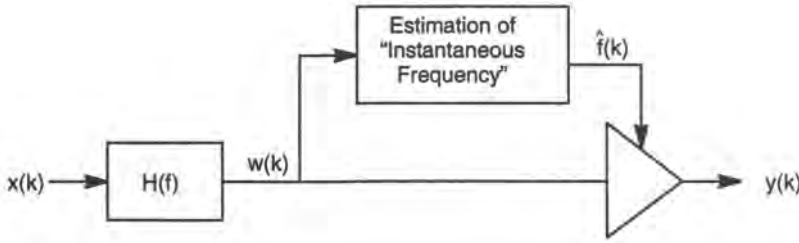


Figure 5.27. Topology of the instantaneous frequency model.

the first box in Figure 5.27 is again $H_{ss}(f)$. If we call the output of that filter $w(k) = A(k) \exp[j\theta(k)]$ at discrete time k , the output of the device is expressible as

$$y(k) = g\left[A(k), \hat{f}(k)\right] \exp\left\{j\theta(k) + j\Phi\left[A(k), \hat{f}(k)\right]\right\}$$

where $\hat{f}(k)$ is the estimate of the instantaneous frequency at discrete time k . The nonlinearity is characterized by a two-dimensional complex table, which contains a modified version of the measurements. This modification is necessary because of the presence of $H_{ss}(f)$ as the first block and the fact that $H_{ss}(f)$ is already inherent in the measurements. Thus, we define the modified measurements to be stored in the table as

$$\begin{aligned} P'_{\text{out}}(P_{\text{in}}, f) &= P_{\text{out}}(P_{\text{in}} - 10 \log |H_{ss}(f)|^2, f) \\ \Phi'_{\text{out}}(P_{\text{in}}, f) &= \Phi_{\text{out}}(P_{\text{in}} - 10 \log |H_{ss}(f)|^2, f) - \angle H_{ss}(f) \end{aligned}$$

This model concludes our brief survey of modeling ideas for nonlinearities with memory. In the chapter summary, we shall try to put this discussion in perspective.

5.3.5. Setting the Operating Point for a Nonlinearity with Memory

In simulation we need to scale the input signal power so that the nonlinearity is set at some desired operating point. We have discussed how to do this for a memoryless nonlinearity in Section 5.2.4.4, and largely the same ideas apply here. In the NLWM case, there is a bit more subtlety because the power can be modified by more than one box in the model. Indeed, in some models, like the Volterra-based model, there does not exist a specific box, such as an AM/AM characteristic, to which the power setting can be meaningfully ascribed. Nevertheless, one can obtain an effective AM/AM characteristic through the simulation itself, assuming that all (previously) unspecified constants or unknowns have been determined. This is done simply by setting a tone as the input (say at the center frequency), and measuring the corresponding output level. By varying the level of the input tone, one can then develop an output versus input power curve that will be useful, if not ideal, for setting the input power level at the appropriate operating point. As with the memoryless nonlinearity, we can maintain the desired input power level by an AGC.

One can also generate an input/output power transfer curve for a more or less arbitrary input signal structure, presumably one that is close to the signals of interest. For example, a pseudonoise (PN) sequence can be used as the modulating signal to imitate many types of modulation schemes.

5.4. Nonlinear Differential Equations

Many devices and subsystems in communications are governed by nonlinear differential equations (NLDE), notably phase-locked loops (PLL) and synchronization circuits employing PLLs, and for that matter any subsystem that contains nonlinearities interspersed between elements described by linear differential equations (such as the classical filters). It should be clear that an NLDE represents a nonlinear system with memory. There are two basic options for modeling and simulating such subsystems or devices, which we refer to as *stand-alone* and *assembled* models or approaches. If a nonlinear subsystem arises physically from the assemblage of lower level boxes each of which can be separately modeled, we can simply string these boxes together to obtain a simulation realization of that subsystem. That *assembled* simulation model then represents a nonlinear differential equation, for which the simulation in effect provides an approximate solution. On the other hand, if we accept that the NLDE is a good model of the subsystem, we could opt to treat the NLDE as a self-contained mathematical entity, and attempt to “solve” it, in the usual sense of that word. This is the stand-alone approach. That approach is also the natural one if we have a device which is not decomposable into simpler blocks and whose operation is naturally described by an NLDE. Solving an NLDE generally requires numerical methods, as closed-form solutions are generally not available for arbitrary inputs. There exists a variety of possible numerical techniques, but the most appropriate choice depends to some extent on the simulation context, by which we mean whether the nonlinear subsystem is embedded within a larger system or is simulated by itself, as we may want to do if we are doing design studies on the subsystem itself. Part of this choice is a methodological issue which we have alluded to elsewhere. The most efficient methods produce samples of the solution at irregular intervals, depending on the apparent rate of variation. On the other hand, in a typical simulation, samples of the signal are generated at regular intervals, which would require interpolation from an NLDE solution that did not do so. We will discuss this point again later.

In this section, we consider only the stand-alone solutions for an NLDE. The assembled model approach is discussed in Chapter 8 relative to simulation of the second-order phase-locked loop. Our objective here is to present only a very brief introduction to the numerical solutions of ordinary differential equations (ODEs) and to highlight the considerations that the practitioner should be aware of in the simulation context. As a concrete illustration of the discussion, an example application to an optical amplifier is given at the end of this section. The subject of numerical solutions of ODEs is a discipline in itself. The interested reader is referred to the many fine texts and publications on numerical methods, e.g., Refs. 44–51, and to their applications, e.g., Refs. 52–55.

5.4.1. Outline of Numerical Methods

There is a rich body of numerical techniques dealing with the solutions of the scalar first-order differential equation

$$\dot{y} = f(t, y, x) \quad (5.4.1)$$

where $x(t)$ is a given driving function (or stimulus, or input). The solution defines a differentiable function $y(t)$ on some interval $[t_0, t_0 + T]$, for which $\dot{y}(t) = f(t, y(t), x(t))$ is an identity, satisfying an initial condition $y(t_0)$; the dot represents derivative with respect to t (time), and (5.4.1) can be either a linear or nonlinear differential equation. It should be clear

that a numerical technique cannot provide a solution for all t on an **interval**,[†] but rather at discretely spaced times $\dots, t_1, t_2, \dots, t_n, \dots$. To motivate the later development, consider the following heuristic derivation of a numerical method. Consider a Taylor expansion of the solution $y(t_{n+1})$ to (5.4.1):

$$y(t_{n+1}) = y(t_n + h_n) = y(t_n) + h_n \dot{y}(t_n) + R(\xi) \quad (5.4.2)$$

where the remainder is $R(\xi) = h_n^2 \ddot{y}(\xi)/2$ for some ξ in the interval $t_n \leq \xi \leq t_{n+1}$. If h_n is sufficiently small, we can construct an approximate recursive solution

$$y_{n+1} = y_n + h_n \dot{y}_n = y_n + h_n f_n \quad (5.4.3a)$$

$$f_n = f(t_n, y_n, x(t_n)) \quad (5.4.3b)$$

starting with the given initial condition $y(t_0)$. In (5.4.2) and (5.4.3) we have used notation that we shall maintain (with a variation, later), namely $y(t_n)$ denotes the true solution at t_n , and y_n the approximate solution at the same instant. Since we are “solving” a differential equation, formulas such as (5.4.3) are referred to as integration formulas. Equation (5.4.3) is called the forward Euler method. Some other possibilities will be introduced below.

In general, the step size h_n can be different for different n . The most efficient techniques use a variable step size, automatically determined by the behavior of the solution and the numerical method itself. In tune intervals where the solution is very well behaved, large step sizes may be used, thus resulting in potentially significant computational reduction. A simpler version of (5.4.3) results if we use a fixed size step h , i.e., $t_{n+1} = t_n + h$. The implications of the size of h and of using variable step size in the context of simulation will be addressed later.

Equation (5.4.3) is a recursive formula. The availability of a recursive method is central to its implementation in the time domain for two primary reasons. The solution “bootstraps” on itself, needing only the most recent values of the solution and the current value of the input to obtain the (approximate) solution at the next time step, and so on. Second, the input function $x(t)$ is the sum of signal plus noise, both of which are in general stochastic processes. Thus, $x(t)$ is generally not given either in analytical form or in its entirety before the fact. This means that in simulation we must also generate $x(t)$ time step by time step as we go along, which is compatible with a recursive procedure.

In this connection it should be mentioned that there are certain technical difficulties connected to the solution of (5.4.1) (whether the solution is analytical or numerical) when the driving function contains white noise.⁽⁵⁶⁾ The nature of the difficulty has to do with the fact that “true” white noise is a process of unbounded variation, which can fluctuate arbitrarily often over any nonzero interval h_n . Such properties are incompatible with an implied slow enough rate of change of the input processes for a statement like (5.4.3) to be meaningful. While white noise is a convenient idealization for many purposes, in situations like the present the difficulties alluded to can be avoided by reverting to a physically more realistic model. Specifically, we can always consider the inputs (signal or noise) to be bandlimited, to within any desired degree of accuracy. In fact, such an assumption is inevitable in simulation, as we have already discussed. It is known that bandlimited functions have derivatives of all orders, and “almost” bandlimited functions that characterize real processes can be expected

[†]We are assuming, of course, the absence of a closed-form solution.

to have a degree of smoothness that will ensure that the solutions obtained from the methods to be discussed are meaningful and close to what would be observed in the physical system.

A higher order nonlinear differential equation can often be cast as a vector first-order differential equation, to which the techniques alluded to apply generally without change. The only requirement is to be able to isolate the highest order derivative. Thus, if a k th-order NLDE given by

$$\phi[t, y, \dot{y}, \ddot{y}, \dots, y^{(k)}, x] = 0$$

can be written in the form

$$y^{(k)} = f[t, y, \dot{y}, \ddot{y}, \dots, y^{(k-1)}, x] \quad (5.4.4)$$

then (5.4.4) can be written as a system of k first-order differential equations. In analogy with (5.4.1), the solution $y(t)$ is a k -times differentiable function for which

$$y^{(k)}(t) = f[t, y(t), \dot{y}(t), \ddot{y}(t), \dots, y^{(k-1)}(t), x(t)]$$

is an identity, given $x(t)$ and initial conditions $y(t_0), \dot{y}(t_0), \dots, y^{(k)}(t_0)$. Equation (5.4.4) also facilitates the representation of the NLDE as a feedback loop (Figure 5.28), which provides an alternative solution approach as will be discussed later. A natural but not unique choice of converting (5.4.4) into a set of first-order differential equations is to define a set of *state variables* as follows:

$$\begin{aligned} q_1 &= y \\ q_2 &= \dot{y} \\ &\vdots \\ q_k &= y^{(k-1)} \end{aligned} \quad (5.4.5)$$

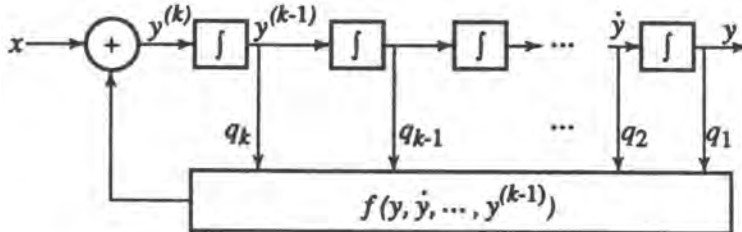


Figure 5.28. Feedback model of a nonlinear differential equation.

from which it follows that

$$\begin{aligned}
 \dot{q}_1 &= q_2 \\
 \dot{q}_2 &= q_3 \\
 &\vdots \\
 \dot{q}_{k-1} &= q_k \\
 \dot{q}_k &= f(t, q_1, q_2, \dots, q_k, x)
 \end{aligned} \tag{5.4.6}$$

or, in generalized vector form, making t explicit,

$$\dot{\mathbf{q}} = \mathbf{f}(t, \mathbf{q}, \mathbf{x}) \tag{5.4.7}$$

which is analogous to (5.4.1). As mentioned, the techniques applicable to solving (5.4.1) apply generally to (5.4.7) (also with the obvious requirement of sufficient initial conditions) and so for notational simplicity we will usually deal with the scalar version. An application of the vector version to the phase-locked loop will be made in Chapter 8. A brief example will be helpful.

■ *Example 5.4.1.* As an illustration of the preceding discussion, consider the (arbitrarily manufactured) differential equation

$$\ddot{y} - ty^2 + \log y - x(t) = 0$$

Rewriting this in normal form, we have

$$\ddot{y} = ty^2 + \log y + x(t)$$

and defining

$$q_1 = y, \quad q_2 = \dot{y}$$

we have

$$\dot{q}_1 = q_2, \quad \dot{q}_2 = tq_2^2 - \log q_1 + x$$

so that $\dot{\mathbf{q}} = (\dot{q}_1, \dot{q}_2)$ and $\mathbf{f}(t, \mathbf{q}, \mathbf{x}) = (q_2, tq_2^2 - \log q_1 + x)$. Notice that we can represent the NLDE as a feedback system, as in Figure 5.29. This is always possible for an NLDE that can be placed in normal form. We can think of this as an “assembled” model of an NLDE, where

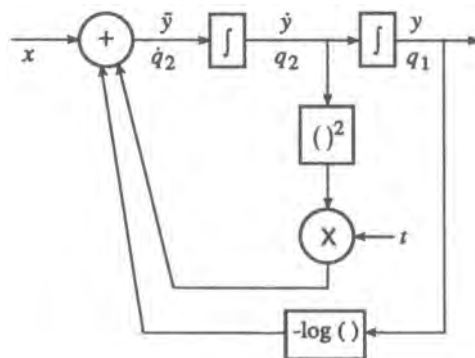


Figure 5.29. Feedback model of the nonlinear differential equation of Example 5.4.1.

the blocks in this case are mathematical operations. Of course, as it is, this loop would produce computational deadlock. But a small delay in the feedback branches would get the loop running. The goodness of the solution, however, is another matter, which we shall address below. ■

5.4.2. Families of Numerical Methods

Equation (5.4.3) is a special case of a fairly general formulation for the numerical solution of a first-order ODE, namely⁽⁴⁸⁾

$$y_{n+p} = a_{p-1}y_{n+p-1} + \cdots + a_0y_n + h\Phi(t_{n+p}, \dots, t_n; y_{n+p}, \dots, y_n; x(t_{n+p}), \dots, x(t_n); h) \quad (5.4.8)$$

where Φ is a given function which depends on f , though this dependence is not explicit here. The initial values $y_0, \dots, y_{k-1}$ are assumed known. The parameter p is an important descriptor: if $p = 1$, then (5.4.8) is called a one-step method, otherwise it is a multistep method. If Φ is independent of y_{n+p} , the method is called *explicit*, otherwise it is *implicit*. These two options have important consequences, as will be seen below.

Equation (5.4.8) establishes a recursion that allows us to proceed forward in steps of h , assuming for now a fixed step size. Most of the known methods are special cases of (5.4.8). Some of these are as follows.

(a) The method of Euler (backward Euler),

$$y_{n+1} = y_n + hf_n, \quad n = 0, 1, \dots \quad (5.4.9a)$$

where $f_n = f(t_n, y_n, x(t_n))$; here $\Phi = f$. This is of course the same as (5.4.3).

(b) Another method of Euler (forward Euler),

$$y_{n+1} = y_n + hf_{n+1} \quad (5.4.9b)$$

(c) Method of Heun, or second-order Runge-Kutta,

$$y_{n+1} = y_n + \frac{1}{2}h[f_n + f(t_{n+1}, y_n + hf_n + x(t_{n+1}))] \quad (5.4.9c)$$

Here,

$$\Phi(t, y, x, h) = \frac{1}{2}[f(t, y, x) + f(t + h, y + hf(t, y, x), x(t + h))]$$

(d) Fourth-order Runge-Kutta method,

$$y_{n+1} = y_n + \frac{1}{6}[\Gamma_1 + \Gamma_2 + \Gamma_3 + \Gamma_4] \quad (5.4.9d)$$

where

$$\begin{aligned}\Gamma_1 &= hf_n \\ \Gamma_2 &= hf(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}\Gamma_1, x(t_n + \tfrac{1}{2}h)) \\ \Gamma_3 &= hf(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}\Gamma_2, x(t_n + \tfrac{1}{2}h)) \\ \Gamma_4 &= hf(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}\Gamma_3, x(t_n + \tfrac{1}{2}h))\end{aligned}$$

(e) Trapezoidal rule

$$y_{n+1} = y_n + \tfrac{1}{2}h[f_n + f_{n+1}] \quad (5.4.9e)$$

(f) Second-order Adams–Bashforth,

$$y_{n+1} = y_n + \tfrac{1}{2}h(3f_n - f_{n-1}) \quad (5.4.9f)$$

(g) Fourth-order Adams–Bashforth,

$$y_{n+4} = y_{n+3} + (h/24)[55f_{n+3} - 59f_{n+2} + 37f_{n+1} - 9f_n] \quad (5.4.9g)$$

(h) Fourth-order Adams–Moulton,

$$y_{n+3} = y_{n+2} + (h/24)[9f_{n+3} + 19f_{n+2} - 5f_{n+1} + f_n] \quad (5.4.9h)$$

(i) Method of Milne (“predictor”),

$$y_{n+3} = y_{n-1} + (4h/3)[2f_{n+2} - f_{n+1} + 2f_n] \quad (5.4.9i)$$

(j) Method of Milne (“corrector”),

$$y_{n+2} = y_n + (h/3)[f_{n+2} + 4f_{n+1} + f_n] \quad (5.4.9j)$$

(k) General linear multistep method. Many of the popular cases of (5.4.8) themselves can be subsumed under the category of linear multistep methods,

$$y_{n+p} = a_{p-1}y_{n+p-1} + \cdots + a_0y_n + h[b_p f_{n+p} + \cdots + b_0 f_n]$$

or, in more compact form,

$$y_{n+p} = hb_p f_{n+p} + \sum_{i=0}^{p-1} a_i y_{n+i} + h \sum_{i=0}^{p-1} b_i f_{n+i} \quad (5.4.9k)$$

Note that methods (a), (c), (d), (f), (g), and (i) are all explicit methods, while methods (b), (e), (h), and (j) are implicit. Method (k) is implicit if $b_p \neq 0$; otherwise it is explicit.

5.4.2.1. Solution Using Explicit Methods

Explicit formulas are very straightforward to use. Starting from initial conditions, the solution at the next step is easily obtained from the values at the current step. Thus, these methods are computationally efficient, though their properties are generally not as good as those of the implicit formulas.

■ **Example 5.4.2.** Let us return to the differential equation of Example 5.4.1, and apply to it method (a). We then have

$$\begin{aligned} q_{1,n+1} &= q_{1,n} + h q_{2,n} \\ q_{2,n+1} &= q_{2,n} + h(t_n q_{2,n}^2 - \log q_{1,n} + x(t_n)) \end{aligned}$$

where the subscripts indicate the vector component followed by the time step. The recursive solution of these two algebraic equations, given h and the initial conditions $q_1(0)$ and $q_2(0)$, is quite straightforward. ■

5.4.2.2. Solution Using Implicit Methods

Implicit formulas generally yield better accuracy and stability at the expense of increased computation. The latter is due to the fact that at each time step t_{n+1} the solution will involve not only the quantities computed at the previous time step t_n , but also quantities to be computed at the current time step. This inherently requires iteration to be performed at each time step. To illustrate this point, consider method (e) above, trapezoidal integration. In expanded form, this rule is

$$y_{n+1} = y_n + \frac{h}{2} [f(t_n, y_n, x(t_n)) + f(t_{n+1}, y_{n+1}, x(t_{n+1}))] \quad (5.4.10)$$

When f is nonlinear, it is not possible to place y_{n+1} on one side of the equation, to be solved in terms of the other variables. Hence, to satisfy the equality in (5.4.10), we must iterate on y_{n+1} starting from a good initial guess, until the two sides agree to within some desired closeness. Notice that the term $x(t_{n+1})$ does not pose a problem in this regard since $x(t)$ is either known or generated independently. Usually, an explicit method is used to provide the initial “guess,” which we will call the zeroth iterate. An explicit method used in this fashion is referred to as a predictor, and the implicit method as a corrector. There are different possible ways to perform the iteration. Here, we illustrate two approaches.

5.4.2.2.1. Iterated Predictor–Corrector Method. Suppose that we have previously calculated y_n . The zeroth iterate $y_{n+1}^{(0)}$ is calculated with an explicit method. For method (a), for example, we have

$$y_{n+1}^{(0)} = y_n + h f(t_n, y_n, x(t_n)) \quad (5.4.11)$$

Using the trapezoidal rule as the corrector, we calculate the first iterate $y_{n+1}^{(1)}$ from

$$y_{n+1}^{(1)} = y_n + \frac{h}{2} [f(t_n, y_n, x(t_n)) + f(t_{n+1}, y_{n+1}^{(0)}, x(t_{n+1}))] \quad (5.4.12)$$

which can be extended to any number of iterations as

$$y_{n+1}^{(r+1)} = y_n + \frac{h}{2} [f(t_n, y_n, x(t_n)) + f(t_{n+1}, y_{n+1}^{(r)}, x(t_{n+1}))] \quad (5.4.13)$$

where r is the iteration index. Usually, methods of equal order (defined below) are used in the predictor–corrector pair, with the tacit assumption that the corrector is inherently more accurate.

5.4.2.2.2. Root Finding Using Newton–Raphson Method. An alternative to the above method is to recast the problem in terms of finding a root of an algebraic equation. There are many root-finding techniques. We illustrate the approach here with the Newton–Raphson (NR) technique.⁽⁵¹⁾ Let us rearrange (5.4.10) in the following way:

$$F(y_{n+1}) = y_{n+1} - y_n - \frac{h}{2} [f(t_n, y_n, x(t_n)) + f(t_{n+1}, y_{n+1}, x(t_{n+1}))] = 0 \quad (5.4.14)$$

so that the desired solution y_{n+1} is seen to be a root of F . Notice that in (5.4.14) all other terms are known, so that y_{n+1} is the only variable to consider. We can visualize $F(y_{n+1})$ versus y_{n+1} as in Figure 5.30, the objective being to find a value of y_{n+1} where F crosses the abscissa. The NR iterative procedure is as follows.

We assume again that y_n has been calculated, and the zeroth iterate $y_{n+1}^{(0)}$ computed through an explicit technique. With reference to Figure 5.30, define the tangent to $F(y_{n+1})$ at $y_{n+1}^{(0)}$ by

$$F'(y_{n+1}^{(0)}) \equiv \frac{dF}{dy_{n+1}}(y_{n+1}^{(0)})$$

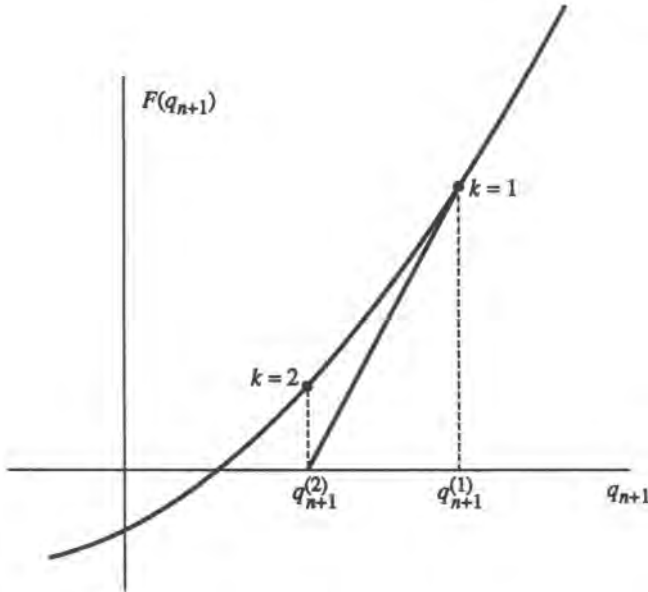


Figure 5.30. Illustration of the Newton–Raphson method.

Hence,

$$F'(y_{n+1}^{(0)}) = \frac{F(y_{n+1}^{(0)})}{y_{n+1}^{(0)} - y_{n+1}^{(1)}} \quad (5.4.15)$$

where $y_{n+1}^{(1)}$ is the next iterate, which clearly comes closer to the zero crossing. Rearranging (5.4.15) as

$$y_{n+1}^{(1)} = y_{n+1}^{(0)} - \frac{F(y_{n+1}^{(0)})}{F'(y_{n+1}^{(0)})}$$

places it in a form that can be generalized to find the $(r + 1)$ th iterate,

$$y_{n+1}^{(r+1)} = y_{n+1}^{(r)} - \frac{F(y_{n+1}^{(r)})}{F'(y_{n+1}^{(r)})} \quad (5.4.16)$$

The procedure is repeated until F is close enough to zero. Although we started with a particular integration formula, it is clear that the procedure is general. However, it may not always work; we generally need a good starting point and a relatively well-behaved F .

In the vector case, (5.4.16) needs to be generalized. Using the notation in (5.4.7), we obtain the result analogous to (5.4.16) as

$$\mathbf{q}_{n+1}^{(r+1)} = \mathbf{q}_{n+1}^{(r)} - \mathbf{J}^{-1}(\mathbf{q}_{n+1}^{(r)}) \mathbf{F}(\mathbf{q}_{n+1}^{(r)}) \quad (5.4.17)$$

where $\mathbf{J}^{-1}(\mathbf{q}_{n+1}^{(r)})$ is the inverse of the Jacobian of $\mathbf{F}$ computed at $\mathbf{q}_{n+1}^{(r)}$, i.e.,

$$\mathbf{J}(\mathbf{q}_{n+1}^{(r)}) = \begin{bmatrix} \frac{\partial F_1(\mathbf{q}_{n+1}^{(r)})}{\partial q_{1,n+1}} & \dots & \frac{\partial F_1(\mathbf{q}_{n+1}^{(r)})}{\partial q_{k,n+1}} \\ \vdots & & \vdots \\ \frac{\partial F_k(\mathbf{q}_{n+1}^{(r)})}{\partial q_{1,n+1}} & \dots & \frac{\partial F_k(\mathbf{q}_{n+1}^{(r)})}{\partial q_{k,n+1}} \end{bmatrix} \quad (5.4.18)$$

where the indices are $n \rightarrow n$ th time step, $r \rightarrow r$ th iteration, and k is the dimension of the vector $\mathbf{q}$. It is assumed, of course, that $\mathbf{J}$ is invertible.

■ **Example 5.4.3.** Using the same differential equation as in the previous examples, let us illustrate the numerical solution using an implicit method. Let us assume that $\mathbf{q}_n$ has been previously calculated. Using, for example, the explicit method in Example 5.4.2, we obtain the zeroth iterate $\mathbf{q}_{n+1}^{(0)}$ to begin the iteration (5.4.18). For simplicity let us use method (b), the backward Euler formula, which gives for the case at hand

$$q_{1,n+1} = q_{1,n} + h q_{2,n+1}$$

or

$$F_1(\mathbf{q}_{n+1}) = q_{1,n+1} - q_{1,n} - hq_{2,n+1} = 0$$

and

$$q_{2,n+1} = q_{2,n} + h(t_{n+1}q_{2,n+1}^2 - \log q_{1,n+1} + x(t_{n+1}))$$

or

$$F_2(\mathbf{q}_{n+1}) = q_{2,n+1} - q_{2,n} - h(t_{n+1}q_{2,n+1}^2 - \log q_{1,n+1} + x(t_{n+1})) = 0$$

The Jacobian is then

$$\mathbf{J}(\mathbf{q}_{n+1}^{(r)}) = \begin{bmatrix} 1 & -h \\ h/q_{1,n+1}^{(r)} & 1 - 2ht_{n+1}q_{2,n+1}^{(r)} \end{bmatrix}$$

and its inverse is

$$\mathbf{J}^{-1}(\mathbf{q}_{n+1}^{(r)}) = \frac{1}{D} \begin{bmatrix} 1 - 2ht_{n+1}q_{2,n+1}^{(r)} & h \\ -h/q_{1,n+1}^{(r)} & 1 \end{bmatrix}$$

where

$$D = (1 - 2ht_{n+1}q_{2,n+1}^{(r)}) + (h^2/q_{1,n+1}^{(r)})$$

and now we can proceed with the iteration until an r is reached for which $\mathbf{F}(\mathbf{q}_{n+1}^{(r)})$ is close enough to zero. ■

5.4.3. Properties of Numerical Methods: Accuracy and Stability

The recipes given by the integration formulas will produce some sequence of numbers, and the obvious question arises as to the closeness of that sequence to the true solution. We take it for granted that the underlying differential equation has a unique solution for the class of input functions of interest. Therefore, we are interested in characterizing, and ultimately controlling, the *global discretization error*, defined as

$$\epsilon = \max_{0 \leq t_n \leq T} |y(t_n) - y_n| \quad (5.4.19)$$

where T specifies the range of the interval examined. It would seem to be a basic requirement on any method that $\lim_{h \rightarrow 0} \epsilon = 0$. Conditions for this to be true will be stated later. It turns out that an easier error criterion to analyze is the *local discretization error*, to which the behavior of the global error can be related.

Let us return to (5.4.8) and assume that instead of the approximate values $y_{n+p-1}, \dots, y_n$, we have the exact solution values $y(t_n + (p-1)h), \dots, y(t_n)$. The estimate of the left side of (5.4.8) given by evaluating the right side is then

$$\begin{aligned} \hat{y}(t_n + ph) &= a_{p-1}y(t_n + (p-1)h) + \dots + a_0y(t_n) \\ &\quad + h\Phi(t_{n+p}, \dots, t_n; y(t_n + ph), \dots, y(t_n); x(t_n + ph), \dots, x(t_n); h) \end{aligned} \quad (5.4.20)$$

where the caret here indicates the estimate obtained under the conditions that the previous p estimates are perfect. [For brevity we will now abbreviate by $y(t_{n+p-1})$ such terms as $y(t_n + (p-1)h)$.] In other words, (5.4.20) is as perfect an estimate of $y(t_{n+p})$ as the method is

capable of providing, since it is operating on true, rather than approximate, values. We will call the difference between the true value and its most perfect estimate the local error, $E(t_{n+p}, h)$,[†]

$$E(t_{n+p}, h) = y(t_{n+p}) - \hat{y}(t_{n+p}) \quad (5.4.21)$$

Thus,

$$\begin{aligned} E(t_{n+p}, h) = & y(t_{n+p}) - a_{p-1}y(t_{n+p-1}) + \cdots + a_0y(t_n) \\ & - h\Phi(t_{n+p}, \dots, t_n; y(t_{n+p}), \dots, y(t_n); x(t_{n+p}), \dots, x(t_n); h) \end{aligned} \quad (5.4.22)$$

In general, for any t_n , the local discretization error (LDE)[‡] is defined as⁽⁴⁸⁾

$$\tau(t_n, h) = h^{-1}E(t_n, h) \quad (5.4.23)$$

A method is called *consistent*⁽⁴⁸⁾ with the differential equation if

$$\lim_{h \rightarrow 0} \tau_*(h) = 0 \quad (5.4.24a)$$

where

$$\tau_*(h) = \max_n |\tau(t_n, h)| \quad (5.4.24b)$$

To paraphrase Ortega,⁽⁴⁸⁾ it is tempting to assume that consistency of a method will ensure that the *global* discretization error tends to zero with h . This turns out to be true for one-step methods, but not necessarily for multistep methods.

A numerical method is *convergent* if the global discretization error tends to zero as $h \rightarrow 0$. It can be shown that a method is convergent if it is both (a) consistent and (b) stable. A numerical method is *stable* if the roots $\lambda_1, \dots, \lambda_p$ of the polynomial

$$p(\lambda) = \lambda^p - a_{p-1}\lambda^{p-1} - \cdots - a_0 \quad (5.4.25)$$

satisfy $|\lambda_i| \leq 1$, $i = 1, \dots, p$, and any root of modulus 1 is simple.

It turns out that convergence as just defined does not necessarily imply a satisfactory numerical method. (See, e.g., Ref. 48, among others, for a discussion.) A well-behaved method should satisfy the stronger requirement called strong stability, which is that only one of the roots of $p(\lambda)$ in (5.4.25) should have value unity: say, $\lambda_1 = 1$, and $|\lambda_i| < 1$, $i = 2, \dots, p$. Any consistent one-step method is strongly stable, as are methods (g) and (h) of Section 5.4.2, for example, while methods (i) and (j) are not.

[†]The local error is also referred to as the truncation error, for reasons that will be seen below.

[‡]Terminology is not entirely consistent throughout the literature. For example, the LDE is called truncation error in Ref. 51.

5.4.3.1. Order of a Method: Computation of Local or Truncation Error

The fact that the local or global discretization errors tend to zero as $h \rightarrow 0$ is not entirely descriptive of the utility of a numerical method. Since in practice, we are forced to use a nonzero value of h , no matter how small, we wish to know how rapidly the error diminishes as a function of h . This leads to the concept of order. A method is at least of *order* m if, for a differential equation with an m -times continuously differentiable solution, $\tau(h) = O(h^m)$, this notation meaning $h^{-m}\tau(h)$ is bounded as $h \rightarrow 0$. Although order is defined with respect to the local discretization error, the key point that can be shown⁽⁴⁸⁾ is that if a method is at least of order m , then the global discretization error is also $O(h^m)$.

We now illustrate the determination of the order of a method. One typical technique for determining the order of a method is to form $\tau(t_n, h)$ as given by (5.4.23), use a Taylor expansion appropriately, and observe the highest remaining power of h . Let us illustrate first with the simple case of the forward Euler method, for which

$$\begin{aligned}\hat{y}(t_{n+1}) &= y(t_n) + hf(t_n, y(t_n), x(t_n)) \\ &= y(t_n) + h \dot{y}(t_n)\end{aligned}$$

Now expand $y(t_{n+1}) = y(t_n + h)$ about t_n :

$$y(t_{n+1}) = y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(\xi_n)$$

where $t_n \leq \xi_n \leq t_n + h$. By (5.4.22),

$$\begin{aligned}E(t_{n+1}, h) &= y(t_{n+1}) - \hat{y}(t_{n+1}) \\ &= \frac{h^2}{2}\ddot{y}(\xi_n)\end{aligned}$$

The error can be attributed to the fact that the numerical method in effect uses a truncated version of the Taylor expansion, hence the alternative terminology as the truncation error, although, as mentioned in the previous footnote, this term is also used to mean the LDE $\tau(t_n, h)$ given by (5.4.23). This method is $O(h)$, that is, of order 1.

As a second example, consider method (f), the second-order Adams–Bashforth method, for which

$$\begin{aligned}\hat{y}(t_{n+1}) &= y(t_n) + \frac{h}{2}\{3f(t_n, y(t_n), x(t_n)) - f(t_{n-1}, y(t_{n-1}), x(t_{n-1}))\} \\ &= y(t_n) + \frac{h}{2}[3\dot{y}(t_n) - \dot{y}(t_{n-1})]\end{aligned}\tag{5.4.26}$$

Expanding $\dot{y}(t_{n-1}) = \dot{y}(t_n - h)$ about t_n , we have

$$\dot{y}(t_n - h) = \dot{y}(t_n) - h\ddot{y}(t_n) + \frac{h^2}{2}\ddot{y}'(t_n) + O(h^3)\tag{5.4.27}$$

and similarly for $y(t_{n+1})$,

$$y(t_n + h) = y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) + \frac{h^3}{3!}\dddot{y}(t_n) + O(h^4) \quad (5.4.28)$$

Substituting (5.4.27) into (5.4.26) and subtracting the result from (5.4.28) yields

$$E(t_{n+1}, h) = \frac{5h^3}{12}\ddot{y}(t_n) + O(h^4)$$

which shows that this method is of order 2.

More systematic approaches can be devised for computing the local error⁽⁵¹⁾ for different families of multistep methods. Other things being equal, methods of higher order are more accurate, but two methods of equal order may have quite different coefficients.

5.4.3.2. Absolute Stability

From the definition of stability above, it can be seen that the polynomial whose roots define stability is not related to the specific differential equation being solved, that connection appearing only in the argument of the function Φ in (5.4.8). Indeed, since convergence is defined as $h \rightarrow 0$, the latter term would thereby disappear. Stability, therefore, as defined above, is a property of the method rather than the ODE at hand. But, as indicated above, in practice we are constrained to using nonzero values of h , however small. The question arises, therefore, as to the influence of the specific ODE on the nature of the approximate solution. In particular, an ODE may depend on a parameter set, say $\dot{y} = f(t, y, \mathbf{A})$, where $\mathbf{A}$ is a vector in a finite-dimensional parameter space. For given h , there is generally a region of $\mathbf{A}$ outside of which the approximate solution will diverge from the true solution. This region is referred to as the *absolute stability* region. The size of this region for particular integration formulas is usually investigated through simple test equations. Typically, the following⁽⁴⁴⁾ test equation is used, which has a one-dimensional parameter space labeled by λ :

$$\dot{y} = \lambda y, \quad y(0) = 1 \quad (5.4.29)$$

whose solution is well known to be $y = e^{\lambda t}$.

In order to demonstrate the determination of the absolute stability region, consider again Euler's (forward) method

$$y_{n+1} = y_n + h\dot{y}_n$$

Substituting for $y_n = \lambda y_n$ from (5.4.29) produces the homogeneous difference equation

$$y_{n+1} - y_n(1 + h\lambda) = 0 \quad (5.4.30)$$

which has the solution

$$y_n = (1 + h\lambda)^n \quad (5.4.31)$$

If $\lambda > 0$, (5.4.31) increases indefinitely, as does the true solution. But if $\lambda < 0$, the solution will diverge unless

$$|1 + h\lambda| < 1$$

which defines the region of absolute stability in terms of $h\lambda$. Since λ can be complex, this condition represents the region inside a unit circle centered at -1 in the complex plane.

As an example, absolute stability regions for several of the methods defined previously are shown as the shaded areas in Figure 5.31. Because of symmetry, only half the regions are displayed. It can be seen that the implicit formulas enjoy a much larger absolute stability region than do the explicit formulas. For example, if λ is negative, *any* value of h is admissible with respect to this stability criterion, though for accuracy it is another matter.

5.4.4. Computational Considerations: Methods of Quality Control

As a numerical method advances from step to step, over a long interval it is difficult to know if the solution is acceptably accurate, i.e., if the global discretization error is small enough for the application. While we know that the error is proportional to h^{m+1} , we generally do not know the proportionality constant, which of course makes all the difference. In explicit methods in particular, the solution is running “open-loop,” as it were; that is, there is no mechanism for checking or controlling the error. There are a number of techniques for doing so. One such technique, for example, employs two different step sizes, at each of which the LDE is estimated and compared. If not greatly different, the larger step size might be acceptable. Otherwise a further subdivision of the interval could be carried out, and so on. Evidently, this technique implies variable step size. An alternative might be to use two different techniques of different orders, settling eventually on one which appears to have sufficient power. It is beyond our scope to discuss these details here, but good discussions can be found, e.g., in Refs. 46, 50, and 51. It is nevertheless necessary to understand the basic procedural aspects of a chosen technique as it will affect and be affected by the simulation

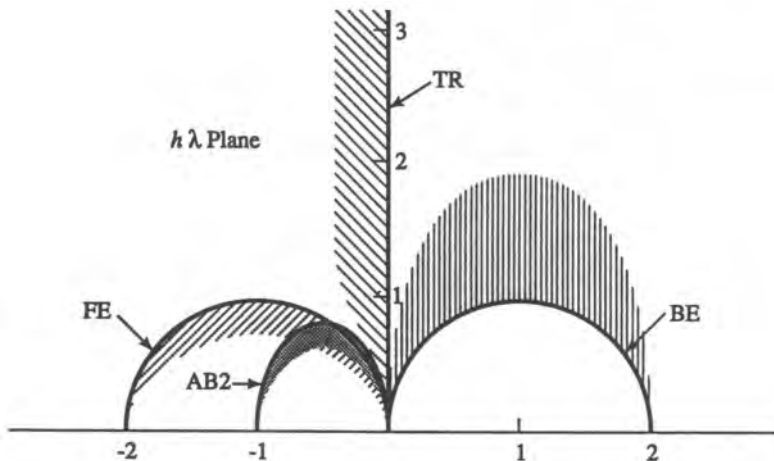


Figure 5.31. Stability regions (shaded) for forward Euler (FE), backward Euler (BE), second-order Adams–Bashforth (AB2), and trapezoidal (TR) integration formulas for a first-order linear differential equation.

within which it is embedded. These considerations are evidently of lesser import if the particular subsystem represented by the NLDE is to be simulated by itself. Whether to use any of the techniques just alluded to, and which to use, will in general depend on the size of the sampling interval that satisfies the sampling theorem for the signal relative to the smallest h that is required by the differential equation.

Suppose a stand-alone model of a nonlinear subsystem is embedded within a larger simulation and that we want to provide the solutions y_n to other blocks of the simulation that are running synchronously with T_s . If the integration method uses a variable step size h_n , we will have to interpolate the solutions. In fact, in order to use such a technique at all, we will also generally have to interpolate the *input function* as well, for, as we have seen, the value $x(t_n)$ enters into the integration methods. But since we generate values of the input only at intervals T_s and t_n may in general be arbitrary, it may very well be necessary to supply an interpolation routine to standard “packaged” algorithms (which usually assume knowledge of the input function in closed form).

It may often happen, by the nature of the system, that T_s is much smaller than the smallest h_n that would likely be used by the NLDE solver. In this case, we could simply set $h = T_s$, but this might result in computational inefficiency if T_s is much smaller than necessary for the NLDE itself. Here, we might constrain a variable-step-size algorithm to using values defined only on a grid separated by T_s . In this case, interpolation of the solution would still be required if $h \neq T_s$. But this would at least avoid interpolation of the input function. Using an h or h_n that is larger than T_s can be regarded as a kind of multirate technique. Different problems have to be examined individually to assess whether there is sufficient computational gain to warrant using values of h not equal to T_s .

5.4.5. Application of Numerical Methods

5.4.5.1. Introduction

If one has to model a device or subsystem representable by a NLDE that is to be used repeatedly, such as a specific phase-locked loop, the better choice would probably be to develop a stand-alone model. A proper choice of integration method would produce a model with better accuracy and one that is likely to be computationally efficient. There is, of course, an investment to be made in developing the model which will probably be cost-effective only if the given design is to be used in a relatively large number of simulation runs. It is, however, the natural modeling methodology if the device is *defined* by the differential equation, as is the case in the optical amplifier example below.

If a stand-alone model is not available, the simulator is constrained to “assemble” the overall model of the device represented by an NLDE from “off-the-shelf” modules. Even in the case when a device is defined by an NLDE, we can construct an assembled model in the sense of mathematical rather than physical modules, as was discussed in conjunction with Figure 5.29. For an assembled model the solution of the NLDE is essentially equivalent to an explicit numerical method even if individual blocks of the assemblage are themselves represented implicitly. The implication is that an assembled model cannot be unconditionally stable. The further implication is that, if a choice is available, blocks of the assembled model should use well-behaved and optimized explicit integration formulas, such as the Adams–Bashforth methods, rather than implicit integration formulas, which impose an artificial delay.^(54,55)

The above considerations will be developed in Section 8.12, where we will model a PLL using both stand-alone and assembled approaches. In the remainder of this section, we present an example of the stand-alone methodology applied to an optical amplifier which is modeled by a nonlinear differential equation with *time-varying* coefficients.

5.4.5.2. Stand-Alone Model for a Traveling-Wave Semiconductor Amplifier

Semiconductor optical amplifiers will play an important role in future optical systems, both for optical fiber transmission and for optical data processing. They provide high gain, with low power consumption and compact size, and their single-mode waveguide structure makes them particularly suitable for use with single-mode fiber. Semiconductor optical amplifiers can be classified into two types: Fabry–Perot (FP) and traveling-wave (TW), according to the values of their residual facet reflectivities. For FP amplifiers, the facet reflectivities are larger than about 0.1. For TW amplifiers, the facet reflectivities are typically less than 10^{-3} . In the following we develop a model for the traveling-wave semiconductor optical amplifier that is a nonlinear differential equation which can be solved using the techniques of this section. The simulation model is preferably of the stand-alone type.

Using the approach described by Saleh^(57,59) and others,⁽⁵⁸⁾ we can describe the behavior of a traveling-wave semiconductor optical amplifier by a pair of coupled partial differential equations, the wave equation and the rate equation, which relate the propagating optical field and the electron carrier density within the active region of the amplifier. By making some judicious approximations and manipulations, we can obtain a model in terms of a single first-order ordinary nonlinear differential equation, which can be solved practically by numerical methods.

Let the electric field of the optical wave propagating through the traveling-wave amplifier in the z direction be denoted by $E(z, t) = \text{Re}[\tilde{E}(z, t) \exp[j(\omega t - kz)]]$, where $\tilde{E}(z, t)$ is the complex envelope, $\omega = 2\pi\nu$ is the optical angular frequency (ν being the frequency), and $k = \omega n/c$ is the nominal propagating constant (n is the intrinsic refractive index and c the speed of light). The wave equation is given by

$$\nabla^2 E(z, t) = \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \epsilon_r E(z, t) \quad (5.4.32)$$

where ϵ_r is the dielectric constant of the amplifier medium,

$$\epsilon_r = n_m^2 = (n' + jn'')^2 \quad (5.4.33)$$

and where n' and n'' are the real and imaginary parts of the refractive index of the medium, n_m , under current excitation.

The imaginary part of the refractive index n'' is given by

$$n'' = \frac{c}{2\omega} g(z, t) \quad (5.4.34)$$

where $g(z, t)$ is the incremental power gain of the amplifier at any z, t . Then the real part of the refractive index n' is given by

$$n' = n - \alpha n'' \quad (5.4.35)$$

where α is the linewidth enhancement factor.

Then the wave equation becomes (assuming $n''^2 \ll 1$)

$$\frac{\partial^2}{\partial z^2} E(z, t) = \frac{\partial^2}{\partial t^2} \left[\frac{n^2}{c^2} + \frac{jn}{\omega c} (1 + j\alpha) g(z, t) \right] E(z, t) \quad (5.4.36)$$

Because the optical period is much shorter than the photon transit time through the amplifier, which, in turn, is much smaller than the time scale of the envelope and gain variations, one can make the approximate operator substitutions $\partial^2/\partial t^2 \rightarrow (j\omega)^2$ and $\partial^2/\partial z^2 \rightarrow (-jk)^2 - (j2k)\partial/\partial z$, where the new operators are now interpreted as acting upon the complex envelope. Then, upon integration of (5.4.36) with respect to z , we obtain

$$\tilde{E}(z, t) = \tilde{E}_{\text{in}}(t) \exp \left[G(z, t) \frac{1 + j\alpha}{2} \right] \quad (5.4.37a)$$

where $G(z, t) = \int_0^z g(z', t) dz'$, and $\tilde{E}_{\text{in}}(t) \equiv \tilde{E}(0, t)$ is the input field. In the particular instance where $z = L$, the amplifier's length, (5.4.37a) takes the form

$$\tilde{E}_{\text{out}}(t) = \tilde{E}_{\text{in}}(t) \exp \left[G(t) \frac{1 + j\alpha}{2} \right] \quad (5.4.37b)$$

where $G(t) \equiv G(L, t)$ and $\tilde{E}_{\text{out}} \equiv \tilde{E}(L, t)$ is the output field. The total power gain is given by $\exp[G(t)]$.

The partial differential rate equation that governs the electron density $N(z, t)$ within the active region of the amplifier can be reduced to the following ordinary differential equation:

$$\frac{dN(z, t)}{dt} = \frac{I(t)}{eV} - \frac{N(z, t)}{\tau_c} - \Gamma a [N(z, t) - N_0] \frac{|\tilde{E}(z, t)|^2}{Ah\nu} \quad (5.4.38)$$

where $\Gamma a [N(z, t) - N_0] = g(z, t)$, $I(t)$ is the amplifier bias current, e is the electron charge, V is the active volume, τ_c is the carrier lifetime, Γ is the confinement factor, a is the gain constant, N_0 is the carrier density at transparency, A is the active area ($V = AL$), and h is Planck's constant. It may be noted that, in customary operation, $I(t)$ is constant. It may be possible to vary $I(t)$ so as to linearize the amplifier. If $I(t)$ is a binary on/off waveform, then the amplifier acts as a switch. Substituting (5.4.37a) into (5.4.38) and integrating with respect to z yields

$$\frac{dG(t)}{dt} = \Gamma a N_0 L \frac{I(t)/I_0 - 1}{\tau_c} - \frac{G(t)}{\tau_c} - \frac{P_{\text{in}}(t)}{\tau_c P_{\text{sat}}} (e^{G(t)} - 1) \quad (5.4.39)$$

where $P_{\text{sat}} = (h\nu/a\tau_c)(A/\Gamma)$ is the saturation output power of the amplifier and $I_0 = eVN_0/\tau_c$. The computer model of the amplifier is obtained as follows: The differential equation (5.4.39) for the time-dependent gain exponent $G(t)$ is solved numerically in response to an arbitrary input optical power $P_{\text{in}}(t) = |\tilde{E}(0, t)|^2$ and an arbitrary bias current waveform $I(t)$. The resulting output field from the amplifier is obtained using (5.4.37b). The amplifier model is shown in Figure 5.32.

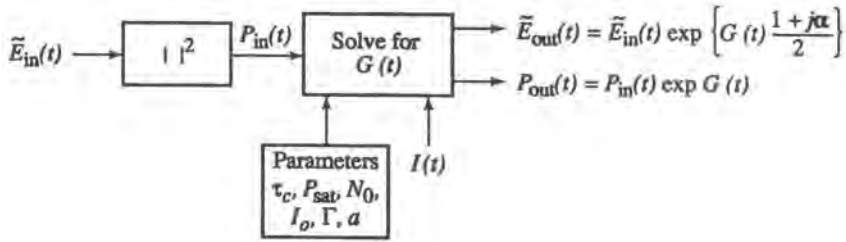


Figure 5.32. Symbolic diagram for a traveling-wave optical semiconductor amplifier.

As we noted earlier, a differential equation (for which the highest derivative can be isolated) can be represented in the form of a feedback loop. Such a representation for (5.4.39) is shown in Figure 5.33. This diagram can be regarded as the implementation of a simulation method. The nature of this model depends strongly on the implementation of the integrator. Both the explicit and implicit solution methods can be used successfully. If one used an implicit integration formula not coupled with a root-finding technique such as Newton–Raphson, the simulation scheduler would introduce a delay in the feedback loop so as to avoid a computational deadlock; this is the situation that often results with an assembled model, as will be discussed in Section 8.12. Such a solution is equivalent to an explicit method.

To apply one of the methods discussed earlier, we need to put (5.4.39) in the standard form, (5.4.7) and apply an appropriate integration method. The application of integration methods to (5.4.39) is left as an exercise.

The optical amplifier model has been presented here as an example of the application of solving nonlinear differential equations in the context of simulation. Of course, such an amplifier model is useful in its own right, but as presented thus far, the model is not quite

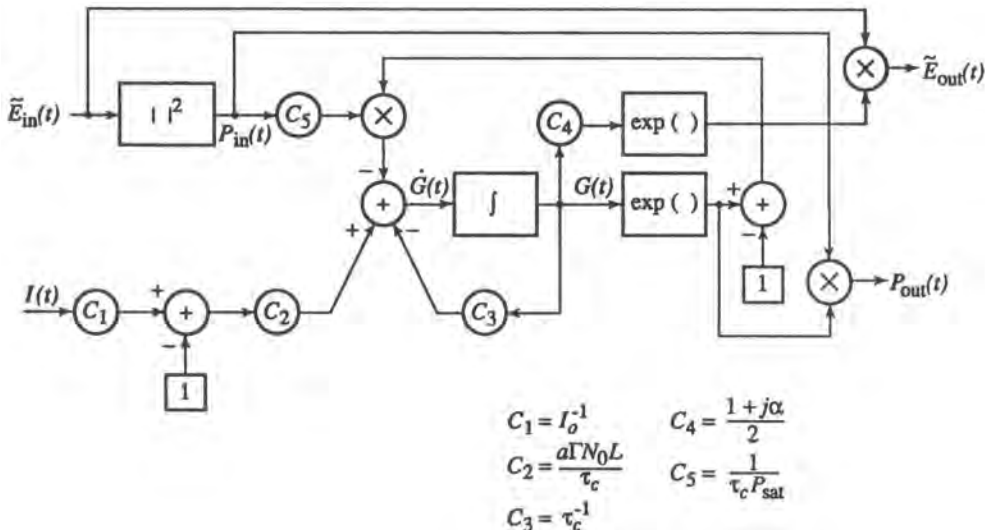


Figure 5.33. Simulation model for a nonlinear differential equation representing a traveling-wave semiconductor optical amplifier.

complete. A fuller model takes into account the presence and the effect of amplified spontaneous emission (ASE) noise. This noise can be modeled as an additive white Gaussian noise source at the output of the amplifier [added to $\tilde{E}_{\text{out}}(t)$ in Figure 5.32; see, e.g., Ref. 59, which also gives the formula for the power spectral density. The ASE noise also changes the amplifier model itself in a minor way, namely, the values of the parameters P_{sat} and τ_c are modified; the reader is referred to Ref. 59 for details. Applications of the model may also be found in Ref. 58. (See also Problem P5.32.)

5.5. Measurement Techniques for Nonlinear Components

As we saw earlier, virtually every model for a nonlinear subsystem requires some type of measurement to “feed” one or another of its defined blocks. This section describes the measurement of nonlinear components (principally amplifiers) to construct and verify models of them. To provide a practical understanding of nonlinear characterization, a basic description of a select group of measurement techniques is presented. The *vector network analyzer* (VNA) single-tone measurement is presented first since it is the most widely accepted technique in laboratories. This VNA technique is commonly used to obtain gain (AM/AM) and phase (AM/PM) transfer curves at a fixed frequency for the memoryless bandpass amplifier model. As we discussed, such a measurement suffices if the component response is relatively independent of frequency over the signal bandwidth.

As the bandwidth of a signal increases, memory effects will begin to manifest themselves. This manifestation will become apparent in the frequency dependence of the gain and phase, as can be seen in Figure 5.10. These curves can also be measured using a VNA. The weakness of this modeling approach, mentioned earlier, is that superposition does not hold in a nonlinear device. We will refer to the measurement in this case as *static* because the test signal (a tone) has unchanging characteristics. Various techniques have been developed that can be considered *dynamic*, in the sense that they utilize a more complex test signal, which in general may be amplitude- and phase-modulated. For example, the simplest dynamic technique uses a two-tone signal, as will be described in Section 5.5.2. These techniques are not yet widely accepted, but they do have a better potential for generating accurate wideband models since the stimulus used for the measurements is closer to an operational signal. These techniques can also be used to estimate the range of signal bandwidths over which a memoryless model may be reasonably accurate.

Note that both the VNA and the dynamic measurement techniques are performed in the frequency domain. Traditionally, this measurement domain has been favored because of the accuracy and the relatively simple structure of the test signals. However, time-domain measurements can provide a more general characterization, though the test signal will for practical reasons have to be repetitive. This section will conclude with a discussion of time-domain measurement methods, including one that directly obtains the LPE waveforms.

5.5.1. The Vector Network Analyzer Single-Tone Measurement

The VNA single-tone measurement of gain and phase transfer curves is simple to implement due to the common availability of the necessary automated instrumentation. This technique is used routinely to measure both traveling-wave tube amplifiers (TWTAs) and solid-state power amplifiers (SSPA) operating at microwave frequencies. There are several

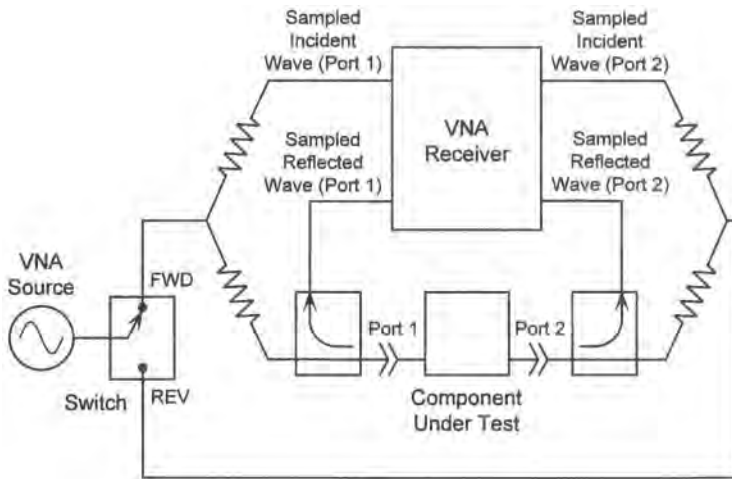


Figure 5.34. Simplified block diagram of a two-port VNA test set.

commercially available VNA system,[†] each having its own unique capabilities. The focus here, of course, is on the capabilities important to our modeling context. A simplified block diagram of the VNA is shown in Figure 5.34. The VNA includes a synthesized source that generates the sinusoid to be applied to the device[‡] under test (DUT). Directional couplers are used to sample both the reflected voltage wave from the component and the transmitted wave through the component. The VNA receiver synchronously detects⁶⁰ both the transmitted and reflected voltages as well as the incident voltage. By means of a microwave switch controlled by the VNA's computer, the incident voltage wave may be directed either to the component's input port or its output port.

The detected voltage waves are processed by the VNA's computer to yield the transmission and reflection responses. When the incident voltage wave is applied to the device's input port, the forward transmission response and the input reflection coefficient are measured. The forward transmission response S_{21} is equal to the standardly defined transfer function or frequency response $H(f)$, as defined in Section 3.3.4. The input reflection coefficient S_{11} is the ratio of the reflected voltage to the incident voltage. When the incident voltage wave is applied to the component's output port, the reverse transmission response and the output reflection coefficient can be measured in a similar manner.

Before performing the above measurements on the DUT, the frequency response of the VNA receiver, couplers, switches, and cables must first be determined by a calibration procedure. This procedure involves the measurement of known calibration standards and the derivation of an error-correction model.⁽⁶¹⁾ This correction is then applied to all measurements taken by the VNA to remove its own response, leaving (more) accurate estimates of the transmission responses and reflection coefficients for the DUT.

The forward transmission response is of primary importance to system modeling, while the reflection coefficients and the reverse transmission response are usually neglected. The reverse transmission response is neglected because communication systems are normally

[†]8510C Network Analyzer, Hewlett-Packard Co., Palo Alto, CA; 360B Network Analyzer, Anritsu-Wiltron Co., Microwave Measurements Division, Morgan Hill, CA.

[‡]For stylistic variation we shall refer to the measured unit as the component or the device.

designed to minimize the signal traveling in the reverse direction. Reflection coefficients are neglected because most communications components (e.g., filters, amplifiers, oscillators, etc.) are designed to have consistent source and load impedances, which minimizes reflected voltage. When a number of components or subsystems are cascaded, the effect of the reflections between them is to introduce amplitude and phase ripple in the overall transmission response. If system components are measured individually, the model of their cascade must include the effect of their interactions if these are not negligible. On the other hand, if the overall transmission response of a cascade of linear components is measured, the reflections between components are automatically included in the overall transmission response, so only the reflections at the input and output of the chain are unmodeled if the reflection parameters are neglected.

Other features of the VNA should also be noted to understand its operation and the scope of applications. First, the single-tone stimulus can be swept to cover the frequency band of interest. Second, the power level of the input tone can be swept to cover the dynamic range of interest. These measurements are referred to as swept-frequency and swept-amplitude, respectively. These features allow active devices (e.g., amplifiers) to be characterized at any operating point from linear through saturation. The typical VNA receiver is designed for an input power less than about 0.1 mW, so a power amplifier must have its output signal attenuated before it enters the receiver (thereby introducing another source of measurement uncertainty).

The VNA single-tone measurement is normally performed with the stimulus frequency set at the center of the signal band of interest. The input power must be stepped to cover not only the desired operating range, but the maximum excursions of the input signal envelope. The transmission amplitude and phase are recorded at each power level. The power at the input of the component must be calibrated with a power meter to take account of losses between the source and the component. This input power is multiplied by the amplitude response to derive the output power. The output power versus input power data are the gain transfer curve and the transmission phase versus input power data are the phase transfer curve. Since it is power that is measured, the measurements must be reinterpreted in terms of “instantaneous” envelope, as discussed in Section 5.2.4.1. In that section we also discussed the question of representing these measurements by some kind of analytical fit or by a simple table lookup. With respect to the latter, interpolation errors will be the smallest if finer power steps are used. In practice, the size of these steps cannot be too small due to limited accuracy of setting the input power level. A practical minimum step size is 0.25 dB, unless the power level is measured at each level setting with a power meter. An alternative and efficient technique involves the use of cubic splines,⁽⁶²⁾ which can be more accurate than the linear interpolation typically used in lookup tables (see also Chapter 3).

5.5.2. Dynamic AM/AM and AM/PM Measurement Techniques Using a Periodically Modulated Signal

In the VNA single-tone technique, the input power is stepped up or down slowly enough that the measurement can be considered quasistatic from the standpoint of temperature and dc bias. This is true for most components because there is adequate time to establish thermal and dc power equilibrium before a measurement is taken at a given power level.

The envelope or phase variations in a communication signal typically occur at dynamic rates that are significantly higher than the rate at which the power is stepped in a VNA

measurement. To apply the VNA measurement in modeling one assumes that the gain and phase transfer is the same for the quasistatic case as for the dynamic case. This is an adequate assumption for most power amplifiers operated with sufficiently narrowband input signals, i.e., where the signal bandwidth is much less than the component bandwidth as measured by a VNA. However, in some instances this assumption is violated even for “narrowband” signals in both SSPAs⁽⁶³⁾ and TWTAs⁽⁶⁴⁾ due to both dc bias circuit resonances and thermal effects.

To find the dynamic gain and phase transfer curves, one can use an input signal to the nonlinear component that has a small amount of periodic AM induced upon it. The input signal has the form

$$S(t) = A(1 + m \cos \omega_m t) \cos \omega_c t$$

whose average power is given by $P_{in} = A^2(1 + m^2/4)$.

The period of the AM should be comparable to the typical period of envelope variations of the operational signal. This input AM will induce output PM proportional to the component's AM/PM, and output AM proportional to the component's AM/AM. In a direct implementation of this concept⁶⁴ one measures the input and output AM by means of power detectors, and the output PM by means of a bridge technique. The input power P_{in} is stepped over the entire operating range (by varying A) of the nonlinear component and, at each power level, the AM/PM and AM/AM transfer are measured in the manner indicated. It is important to note that in the typical instrumentation, what is reported are the so-called AM/AM and AM/PM coefficients, measured, respectively, in units of “dB per dB” and “degrees per dB,” which are in effect the derivatives of the phase and gain transfer curves at the operating point set by the average power of the input signal. These two quantities are then numerically integrated over power to yield the dynamic gain and phase transfer curves.

In a second implementation of this concept^(11,64) the input AM is induced not by directly modulating the carrier, but rather by adding to it a small tone, offset from it by Δf . That is, the input signal is

$$S(t) = A \cos(\omega_c t) + a \cos[(\omega_c + \Delta\omega)t]$$

An advantage of this implementation is that the phase of the input or output signal need not be measured directly. One can obtain the AM/AM and AM/PM solely from amplitude measurements. At the input, one must measure the power in the carrier and in the small tone, while at the output one measures the power at the same two frequencies and the power in the intermodulation product at frequency $f_c - \Delta f$. The tone at frequency $f_c + \Delta f$ is typically set at 20–30 dB below the tone at f_c . A block diagram of a system to perform this measurement is shown in Figure 5.35.

To derive the AM/AM and AM/PM, a reference measurement is first made where the input tones are reduced in power to the point where the component is in linear operation. Four reference measurements are made: the input voltage V_{ic}^r at f_c , the input voltage V_{i+}^r at $f_c + \Delta f$, the output voltage V_{oc}^r at f_c , and the output voltage V_{o+}^r at $f_c + \Delta f$. The input tones are then increased in power to the operating point (keeping the ratio fixed). Five additional measurements are made: the input voltage V_{ic} at f_c , the input voltage V_{i+} at $f_c + \Delta f$, the output voltage V_{oc} at f_c , the output voltage V_{o+} at $f_c + \Delta f$, and the output voltage V_{o-} at $f_c - \Delta f$.

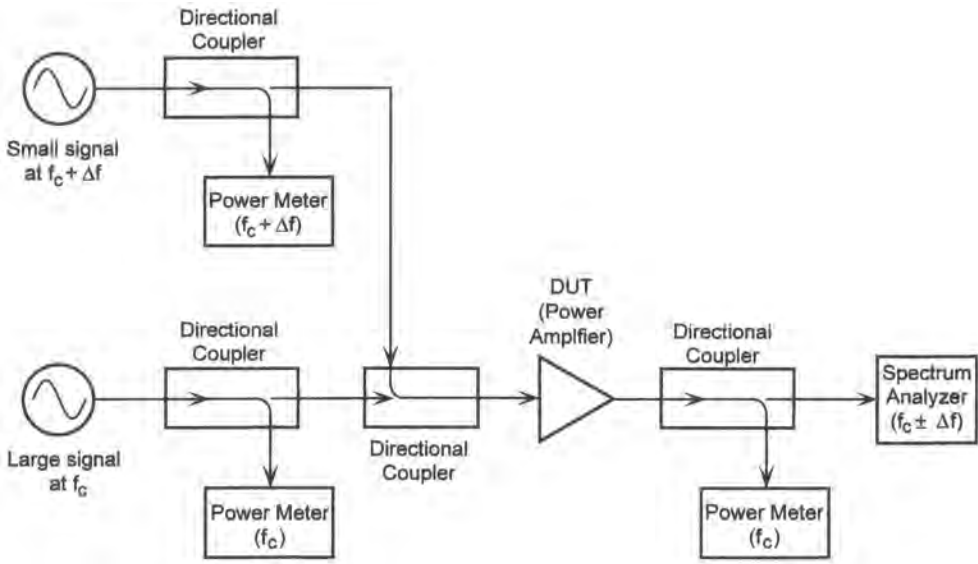


Figure 5.35. Dynamic AM/AM, AM/PM test configuration.

From these measurements, the normalized output amplitudes of the sidebands, denoted by S_+ and S_- at the frequencies $f_c + \Delta f$ and $f_c - \Delta f$, respectively, can be defined⁶⁴ as follows:

$$S_+ = S_+(V_{ic}) \equiv \left(\frac{V_{i+}^r}{V_{o+}^r} \right) \left(\frac{V_{oc}^r}{V_{ic}^r} \right) \left(\frac{V_{o+}}{V_{i+}} \right) \left(\frac{V_{ic}}{V_{oc}} \right) \quad (5.5.1)$$

and

$$S_- = S_-(V_{ic}) \equiv \left(\frac{V_{i+}^r}{V_{o+}^r} \right) \left(\frac{V_{oc}^r}{V_{ic}^r} \right) \left(\frac{V_{o-}}{V_{i+}} \right) \left(\frac{V_{ic}}{V_{oc}} \right) \quad (5.5.2)$$

As a result of the normalization, S_+ and S_- are independent of the sideband input voltage, and are functions only of the operating point, which is set by the input voltage V_{ic} at f_c . The AM/AM (dB/dB) and AM/PM (deg/dB) are then calculated, respectively, from

$$\frac{dV_{oc}}{dV_{ic}} = S_+^2 - S_-^2 \quad (5.5.3)$$

and

$$\frac{d\theta_{oc}}{dV_{ic}} \approx \frac{18 \ln 10}{\pi} \left[S_+^2 - \left(\frac{1 + S_+^2 - S_-^2}{2} \right)^2 \right]^{1/2} \quad (5.5.4)$$

where the AM/AM expression is exact, and the AM/PM expression is within a 0.5% error when the difference between the two input tone levels exceeds 20 dB. These expressions must be integrated to arrive at the standard amplitude and phase transfer curves.

A third implementation of the dynamic envelope technique utilizes either one or two VNAs and a mixer to generate a two-tone stimulus.^(63,65) A second mixer is used to coherently detect the envelope of this two-tone signal, and the VNA receiver then finds the gain and phase of the envelope. By comparing the input envelope to the output envelope of the nonlinear component, the dynamic gain and phase transfer curves may be deduced.

The second and third implementations just discussed may be termed dynamic two-tone techniques. An advantage of such techniques is that the AM frequency can be set to MHz or even GHz rates simply by changing the offset frequency between the two tones. By varying the tone spacing as well as the tone power, a set of gain and phase transfer curves can be obtained as a function of tone spacing. If the nonlinearity is truly memoryless, the transfer curves are independent of modulation frequency (Δf). If the only deviations from the memoryless assumption consist of linear filters before or after the nonlinearity, their effects can be analytically removed from the two-tone measurements to give the response of the nonlinear element alone. In practice one would expect the transfer curves of the nonlinear element to vary with modulation frequency. At very low modulation rates one finds the quasistatic transfer curves, which include the quasistatic thermal and bias effects. As the modulation frequency is increased, the thermal effects decrease, since heat flow is a relatively slow process, while the dc bias circuitry may go through a resonance where it has a strong effect on the transfer curves. At modulation frequencies well above the inverse of the thermal and power supply time constants, and well below the RF circuit bandwidth, the transfer curves are independent of modulation frequency. For modulation frequencies above the RF circuit bandwidth, the memoryless model breaks down.

By means of such measurements of transfer curves as a function of modulation frequency, one can determine the bandwidth over which the memoryless model is valid. Note that this bandwidth cannot be determined by VNA measurements alone, because the VNA can only measure the static transfer curves. A necessary, but not sufficient, condition for the validity of the memoryless model is that the static transfer curves be independent of frequency. Static measurements determine the bandwidth of the RF circuit, but they are insensitive to thermal and power supply effects that can affect dynamic transfer curves. The latter, of course, better represent the component's response to operational signals.

5.5.3. Time-Domain Measurement Techniques

Time-domain measurements provide a way to measure the component's response to a stimulus having characteristics very similar to those of an operational wideband signal. Such a measurement provides a direct and complete description of the action of the component (for the input in question) as opposed to the partial description that the traditional swept-frequency, swept-amplitude observations supply. The latter measurements are partial because superposition does not apply in the nonlinear device, and the interactions between frequency components in a realistic signal are not captured. All such interactions are automatically incorporated in a time-domain measurement. Thus, time-domain measurements can provide a means of validating models generated by other techniques, and perhaps even to determine the optimal parameters for such models by "tweaking" them for best agreement between modeled and measured waveforms.²⁸

In order to provide a relatively full characterization of the nonlinearity, the test signal should emulate an operational signal, which means it should be fairly rich in frequency components. This is somewhat constrained by the practical requirement that this stimulus be repetitive, because most digital recording instrumentation have a narrow real-time bandwidth.

The sampling rate of the recording instruments is typically too low to capture a microwave signal in real time; thus, precise incrementing is used to achieve a high equivalent bandwidth for a periodic signal. Typical waveforms that are adequate for the purpose are pulse RF or microwave carriers or carriers digitally modulated by PN sequences.

Assuming the bandwidth of the recording instrument is sufficient, the ideal procedure is to record directly the RF or microwave input and output waveforms of the nonlinear component. These waveforms should be recorded at a number of input power levels throughout the operating range of the component. Time-domain instrumentation can record the waveforms digitally and store the data on a controlling computer.

Typical instruments useful for recording waveform data are a *digital storage oscilloscope* (DSO), a *microwave transition analyzer* (MTA), or the recently developed *nonlinear vector network analyzer* (NVNA).[†] The accuracy of these instruments for measuring high-frequency signals is limited by linear amplitude and phase distortion. The MTA and high-bandwidth DSOs start to have significant phase and amplitude distortion at about 15 GHz.⁽⁶⁶⁾ The NVNA is based on an MTA, but it comes with calibration standards and an extensive calibration routine so that phase and amplitude distortion and other errors are removed from the measurements analytically. Hence the NVNA can provide accurate waveforms up to 50 GHz. The NVNA also has the advantage that it can measure the reflections from components as well as the transmission response.

Recently, a method has been developed to measure microwave signals directly at baseband.⁶⁷ In this approach the LPE waveforms used in simulation are recorded by converting the microwave signals to baseband with a calibrated downconverter before being captured by a DSO or an MTA. Since LPE signals are complex, two waveform measurements are required.

Measuring signals at baseband has several advantages compared to measuring them directly at the carrier frequency. One is that the sample rate can be reduced by the ratio of half the signal bandwidth to the carrier frequency, thereby allowing for a longer time record or higher time resolution. Another advantage is that the phase noise of the carrier is eliminated since the downconverting receiver uses a local oscillator (LO) which is coherent with the carrier. Also, with an appropriate downconverter, measurements can be done at high carrier frequencies, where the MTA or DSO would yield inaccurate waveforms.

The basic measurement system shown in Figure 5.36 consists of a baseband source, an upconverting transmitter, and a downconverting receiver followed by a waveform recording instrument (DSO or MTA). The microwave signal to be measured may have any arbitrary repetitive phase or amplitude modulation imposed upon it. In practice, any signal source (e.g., a modulator) may be used in place of the transmitter in Figure 5.36 as long as it can furnish an unmodulated carrier for use by the downconverting receiver as a coherent LO signal.

To record the LPE input waveforms, the equipment is configured as shown in Figure 5.36, while to record output waveforms, the nonlinear component is inserted between the transmitter and the receiver. In either case, the downconverted signal is first recorded at arbitrary relative carrier phase between the microwave signal source and the receiver. The waveform recorded is the in-phase component of the LPE signal. The phase shifter is then adjusted by 90° and the downconverted signal is again recorded to form the quadrature component of the LPE signal. Note that, at this point, the recorded waveforms include the response of the downconverting receiver, which is not negligible. These waveforms must

[†]Nonlinear Vector Network Analyzer (NVNA) System, Hewlett-Packard Co., Palo Alto, CA.

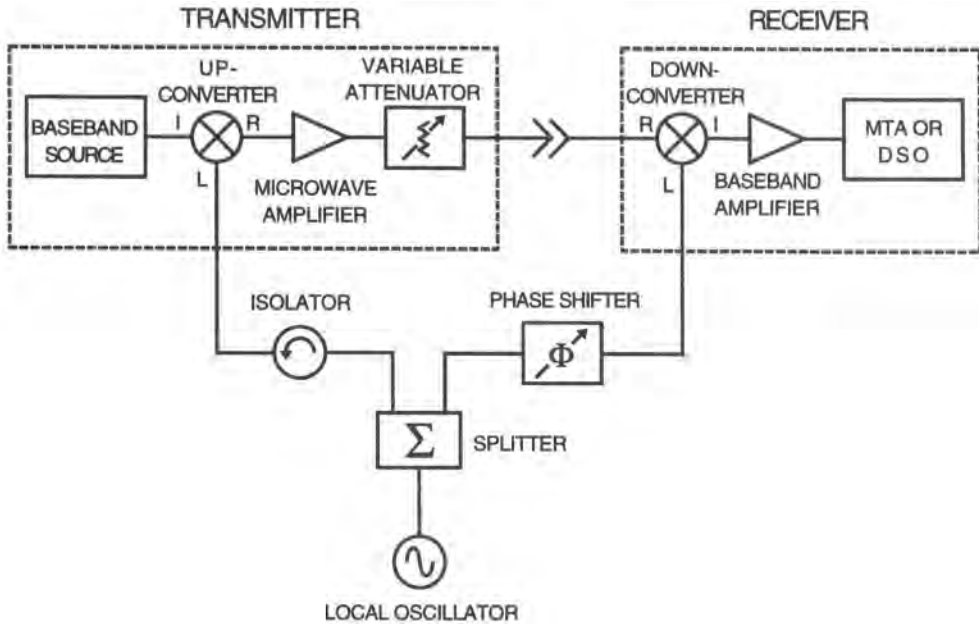


Figure 5.36. Simplified block diagram of the baseband time-domain measurement setup.

obviously be corrected for the receiver response before they can be applied to the modeling task.

The receiver response may be removed analytically by means of the *baseband-double-sideband frequency-translating device* (BB-DSB-FTD) characterization method.⁶⁸ In this procedure, measurements of the frequency-domain LPE responses of all possible pairwise combinations of the transmitter, receiver, and an additional test mixer are used to derive the downconverting receiver response so that it can be removed from the recorded waveform data. A VNA is most conveniently used to perform these measurements. This procedure requires a total of six different measurements of the three possible configurations. The first test configuration is the transmitter–receiver combination as shown in Figure 5.36, with the VNA source connected to the baseband input in place of the baseband source, and the VNA receiver connected to the baseband output in place of the MTA. The second and third configurations are similar to the first except that the test mixer replaces the transmitter in the second configuration, and the test mixer replaces the receiver in the third configuration. The method requires the test mixer to have the same frequency response whether it is used as an upconverter or a downconverter. In practice, commonly available double-balanced mixers exhibit this reciprocal response if a low VSWR is provided on all ports by means of fixed attenuators. For each of the three configurations, the LPE response of the pairwise combination is derived from two measurements at a relative LO phase setting of 90° . The LPE response of each configuration is the sum of the responses of the upconverter and the downconverter used in each configuration. Hence the LPE response of the receiver alone may be obtained by mathematically combining the LPE responses of the three configurations. The receiver response can then be removed analytically from the measurements. Note that the calibration procedure described facilitates the accurate measurement not only of nonlinear components, but also other frequency-translating systems⁽⁶⁸⁾ such as a satellite transponder.

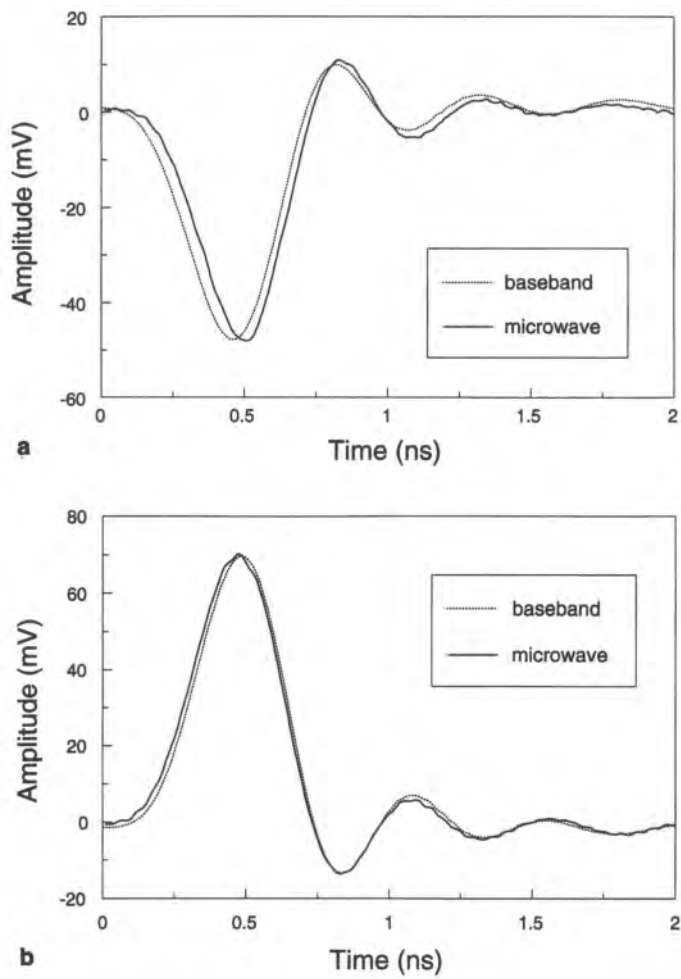


Figure 5.37. Time-domain measurements of a microwave pulse. The baseband and the direct microwave measurements are compared by converting the microwave measurement to baseband in software. (a) In-phase waveform; (b) quadrature waveform.

Figure 5.37 shows a simple example of a time-domain measurement of a subnanosecond-amplitude pulse. This waveform was generated by applying a 0.35-ns wide, 0.2-V amplitude baseband pulse to the upconverting transmitter shown in Figure 5.36, using an LO frequency of 20 GHz. This signal was measured by means of the baseband time-domain technique described above, in this case using an MTA as the waveform recording instrument. The same signal was also measured directly at the 20-GHz carrier frequency, using again an MTA with a sampling frequency of four samples per cycle of the carrier frequency. The direct microwave measurement was then downconverted in software to baseband for comparison with the baseband measurement. The in-phase and quadrature waveforms generated by the two techniques are compared in Figure 5.37. The excellent agreement indicates the consistency between the two measurement techniques. The response of the MTA has not been

removed from the data shown. The increased amplitude and phase distortion of the MTA in the direct microwave measurement may account for the slight differences between the results of the two techniques.

5.6. Summary

Nonlinear elements or subsystems are present in many types of communication systems, notably in the form of power amplifiers and synchronization circuits. We have shown that one implication of nonlinearity is to require (generally) an increase in the simulation sampling rate. However, probably the main issue from the simulation standpoint is the construction of an accurate model, or one accurate enough for the application. Given the model, there are no difficulties carrying out the simulation.

We defined different classes of models that may form a useful starting point for approaching the task of developing an explicit model for use in simulation. Unless a model is already given in complete form, such as a nonlinear differential equation, we are generally reduced to synthesizing a block model for a nonlinearity. A block model is an abstraction that purports to capture, at least approximately, the action of the nonlinearity on the signal or its complex envelope. Although this type of model is a “black-box” type of characterization, the model topology is itself usually guided by some kind of intuitive argument about the device’s behavior. There are variations on this argument, which lead to corresponding variations on the actual form of the model. A number of these variations were presented. In this context, it is important to note that the block models discussed for nonlinearities with memory form only a small subset of possible modeling approaches, and are intended only to provide an inkling into ways of thinking. Although the ideas underlying these model constructions are all plausible on the surface, it must be recalled that the synthesis of the blocks’ functionalities is generally based on a restricted set of measurements, such as swept-tone, which does not fully characterize a device’s behavior under operational conditions. For this reason, among others, it is quite difficult to predict *a priori* the relative accuracy of different modeling constructs, and it should be pointed out that modeling complexity in itself will not guarantee a more accurate model. The very existence of a large number of models is indicative of the caution that must be exercised in accepting a particular one.

An entirely different sort of nonlinear model is a nonlinear differential equation. One can arrive at such a representation through different means, perhaps through fundamental physical principles (as in the optical amplifier example), or perhaps as a direct consequence of the structure of a block diagram (as for a phase-locked loop, as will be seen later). However the NLDE may arise, we have focused in this chapter on the numerical means through which we can produce a solution “on the fly,” in other words, a numerical process that becomes just another block of the simulation. We outlined various considerations related to this process.

Whatever the model structure, measurements play an important role in this context, in two different ways. First, virtually every modeling construct has parameters or characteristics that eventually must be extracted from some set of measurements, either directly or through some sort of fitting procedure. Second, given the model, measurements must ultimately be used to validate the model. This is especially true for nonlinear systems, where intuition often fails and where alternative checks through analytical means are not usually available. Thus, we concluded the chapter with a short discussion of relevant measurement techniques.

References

1. G. L. Wise, A. P. Traganitis, and J. B. Thomas, The effect of a memoryless nonlinearity on the spectrum of a random process, *IEEE Trans. Inf. Theory* **IT-23**(1), 84–89 (1977).
2. C. L. Devieux, Analysis of PSK signal power spectrum spread with a Markov chain model, *COMSAT Tech. Rev.* **5**(2), 225–252 (1975).
3. D. D. Siljak, *Nonlinear Systems*, Wiley, New York (1969).
4. N. M. Blachman, Bandpass nonlinearities, *IEEE Trans. Inf. Theory* **IT-10**(1), 162–164 (1964).
5. N. M. Blachman, Detectors, bandpass nonlinearities and their optimization: Inversion of the Chebyshev transform, *IEEE Trans. Inf. Theory* **IT-17**(4), 398–404 (1971).
6. A. J. Cann, Nonlinearity model with variable knee sharpness, *IEEE Trans. Aerospace Electron. Syst.* **16** (6), 874–877 (1980).
7. A. L. Berman and C. E. Mahle, Nonlinear phase shift in traveling wave tube as applied to multiple access communication satellites, *IEEE Trans. Commun. Technol.* **COM-19**(1), 37–48 (1970).
8. J. B. Minkoff, Wideband operation of nonlinear solid state power amplifiers—Comparison of calculations and measurements, *AT T Bell Lab. Tech. J.* **63**(2), 231–248 (1984).
9. J. B. Minkoff, Intermodulation noise in solid-state power amplifiers for wideband signal transmission, in *Proc. 9th AIAA International Communications Satellite System Conference* (1982); pp. 304–313.
10. A. A. M. Saleh, Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers, *IEEE Trans. Commun.* **COM-29**(11), 1715–1720 (1981).
11. A. A. Moulthrop, C. J. Clark, C. P. Silva, and M. S. Muha, A dynamic AM/AM and AM/PM measurement technique, *IEEE MTT-S Int. Microwave Symp. Digest* **1997**, 1455–1458 (1997).
12. O. Shimbo, Effects of intermodulation, AM-PM conversion and additive noise in multicarrier TWT systems, *Proc. IEEE* **59**(2), 230–238 (1971).
13. J. C. Fuenzalida, O. Shimbo, and W. L. Cook, Time-domain analysis of intermodulation effects caused by nonlinear amplifiers, *COMSAT Tech. Rev.* **3** (1), 89–143 (Spring 1973).
14. A. A. M. Saleh, Intermodulation analysis of FDMA satellite systems employing compensated and uncompensated TWTs, *IEEE Trans. Commun.* **COM-30**(5), 1233–1242 (1982).
15. O. Shimbo, *Transmission Analysis in Communication Systems*, Vol. 2, Computer Science Press, Rockville, Maryland (1988).
16. N. J. Frigo, A model of intermodulation distortion in nonlinear multicarrier systems, *IEEE Trans. Commun.* **COM-35** (2/3/4), 1216–1222 (1994).
17. K. W. Schneider and W. H. Tranter, Efficient simulation of multicarrier digital communication systems in nonlinear environments, *IEEE J. Select. Areas Commun.* **11**(3), 328–339 (1993).
18. L. C. Palmer and S. Lebowitz, Computer simulation of solid-state amplifiers, *COMSAT Tech. Rev.* **8**(2), 371–404 (1978).
19. J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, Nonlinear black-box modeling in system identification: A unified overview, *Automatica* **31**(12), 1691–1724 (1995).
20. G. D. Vendelin, A. M. Pavio, and U. L. Rohde, *Microwave Circuit Design Using Linear and Nonlinear Techniques*, Wiley, New York (1990).
21. J. E. Rowe, *Nonlinear Electron-Wave Interaction Phenomena*, Academic Press, New York (1965).
22. M. K. Sherba and J. E. Rowe, Characteristics of multisignal and noise-modulated high-power microwave amplifiers, *IEEE Trans. Electronic Devices* **ED-18**(1), 11–34 (1971).
23. H. B. Poza, Z. A. Sarkozy, and H. L. Berger, A wideband data link computer simulation model, in *Proc. NAECON Conference* (1975).
24. R. Blum and M. C. Jeruchim, Modeling nonlinear amplifiers for communication simulation, in *Conference Record, IEEE International Conference on Communications, ICC '89*, Boston (June 1989).
25. M. T. Abuelma'atti, Frequency-dependent nonlinear quadrature model for TWT amplifiers, *IEEE Trans. Commun.* **COM-32** (8), 982–986 (1984).
26. A. R. Kaye, D. A. George, and M. J. Eric, Analysis and compensation of bandpass nonlinearities for communications, *IEEE Trans. Commun. Technol.* **COM-20**(10), 965–972 (1972).
27. X. T. Vuong and A. F. Guibord, Modelling of nonlinear elements exhibiting frequency-dependent AM/AM and AM/PM transfer characteristics, *Can. Elect. Eng. J.* **9**(3), 112–116 (1984).
28. G. Chrisikos, C. J. Clark, A. A. Moulthrop, M. S. Muha, and C. P. Silva, A nonlinear ARMA model for simulating power amplifiers, *IEEE MTT-S Int. Microwave Symp. Digest* **1998**, 733–736 (1998).

29. E. Bedrosian and S. O. Rice, The output properties of Volterra systems (nonlinear systems with memory) driven by harmonic and Gaussian inputs, *Proc. IEEE* **59**(12), 1688–1707 (1971)
30. G. Palm and T. Poggio, The Volterra representation and the Wiener expansion: Validity and pitfalls, *SIAM J. Appl. Math.* **33**(2), 195–216 (1977).
31. S. P. Boyd, Volterra series: Engineering fundamentals, Ph.D. dissertation, University of California, Berkeley (1985).
32. S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
33. J. S. Bendat, *Nonlinear System Analysis and Identification*, Wiley, New York (1990).
34. E. Van den Eijnde, Steady-state analysis of strongly nonlinear circuits, Ph.D. thesis, Vrije Universiteit Brussel, Brussels, Belgium, (1989).
35. V. Krozer, K. Fricke, and H. L. Hartnagel, Novel analytical approach for the nonlinear microwave circuits and experimental characterisation of the nonlinear behavior of a new MESFET device structure, *IEEE MTT-S Int. Microwave Symp. Digest* **1989** (June), 351–354 (1989).
36. G. Vandersteen, Nonlinear characterization of a mixer, Thesis, Université Catholique de Louvain, Louvain, Belgium (1992).
37. *MAC/RAN IV: Time Series and Spectral Analysis System*, University Software Systems, Los Angeles, California [Available From e-mail: taghabian@usc.edu].
38. T. Koh and E. J. Powers, Second-order Volterra filtering and its application to nonlinear system identification, *IEEE Trans. ASSP* **33**(6), 1445–1455 (1985).
39. W. J. Rugh, *Nonlinear System Theory: The Volterra–Weiner Approach*, Johns Hopkins University Press, Baltimore, Maryland (1981).
40. H. Diaz and A. A. Desrochers, Modeling of nonlinear discrete-time systems from input–output data, *Automatica* **24**(5), 629–641 (1988).
41. J. Leontaritis and S. A. Billings, Input–output parametric models for nonlinear systems Part I: Deterministic nonlinear systems and Part II: Stochastic non-linear systems, *Int. J. Control* **41** (2), 303–344 (1985).
42. G. L. Heiter, Characterization of nonlinearities in microwave devices and systems, *IEEE Trans. Microwave Theory Tech.* **MTT-21**(12), 797–805 (1973).
43. M. I. Sobhy, E. A. Hosny, M. W. R. Ng, and E. A. Bakkar, Nonlinear system and subsystem modeling in time domain, *IEEE Trans. Microwave Theory Tech.* **44**(12), 2571–2579 (1996).
44. P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations*, Wiley, New York (1962).
45. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, Berlin (1980).
46. S. D. Conte and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw-Hill, New York (1980).
47. D. Quinney, *An Introduction to the Numerical Solution of Differential Equations*, Research Studies Press, Lechworth, England (1985).
48. J. M. Ortega, *Numerical Analysis*, SIAM, Philadelphia, Pennsylvania (1990).
49. F. S. Acton, *Numerical Methods that (Usually) Work*, Mathematical Association of America, Washington, D.C. (1990).
50. W. H. Press *et al.*, *Numerical Recipes in Fortran: The Art of Scientific Programming*, 2nd ed., Cambridge University Press, Cambridge (1992).
51. W. Gautschi, *Numerical Analysis: An Introduction*, Birkhäuser, Boston (1997).
52. J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, New York (1983).
53. W. J. Mc Calla, *Fundamentals of Computer Aided Circuit Simulation*, Kluwer, Norwell, Massachusetts (1989).
54. V. Raman and K. Abbas, Simulation of closed-loop systems: Computability vs. stability, in *Proc. Miami IEEE Technicon* (1987), pp. 114–117.
55. V. Raman and K. Abbas, Explicit multistep methods for system simulation, in *Symposium Proceedings, ISCAS, Portland, Oregon* (May 1989), pp. 1792–1795.
56. N. J. Kasdin, Discrete simulation of colored noise and stochastic processes and $1/f^\alpha$ power law noise generation, *Proc. IEEE* **83**(5), 802–827 (1995).
57. A. A. M. Saleh, Nonlinear models of traveling-wave optical amplifiers, *IEEE Electron. Lett.* **24**(14), 835–837 (1989).
58. A. Elrefaie and C. Lin, Performance degradations of multigigabit-per-second NRZ/RZ lightwave systems due to gain saturation in traveling-wave semiconductor optical amplifiers, *IEEE Photon. Technol. Lett.* **1**(10), 300–302 (1989).
59. A. A. M. Saleh, Modeling of nonlinearity in semiconductor optical amplifiers, in *Conference Record, IEEE Global*

Telecommunications Conference, Dallas, Texas (1989).

60. G. H. Bryant, *Principles of Microwave Measurements*, Peter Peregrinus, London (1988).
61. J. Fitzpatrick, Error models for systems measurements, *Microwave J.* **21**(5), 63–66 (1978).
62. J. Staudinger, Applying the quadrature modeling technique to wireless power amplifiers, *Microwave J.* **40**(11), 66–86 (1997).
63. W. Bosch and G. Gatti, Measurement and simulation of memory effects in predistortion linearizers, *IEEE Trans. Microwave Theory Tech.* **MTT-37**(12), 1885–1890 (1989).
64. J. P. Laico, H. L. McDowell, and C. R. Moster, A medium power traveling-wave tube amplifier for 6000-Mc radio relay, *Bell Syst. Tech. J.* **35**(6), 1285–1346 (1956).
65. A. Katz and R. Dorval, Evaluation and correction of time-dependent amplifier nonlinearity, *IEEE MTT-S Int. Microwave Symp. Digest* **1996**, 839–842 (1996).
66. J. Verspecht and K. Rush, Individual characterization of broadband sampling oscilloscopes with a nose-to-nose calibration procedure, *IEEE Trans. Measure.* **IM-43**(2), 347–354 (1994).
67. A. A. Moulthrop, M. S. Muha, C. P. Silva, and C. J. Clark, A new time-domain measurement technique for microwave devices, *IEEE MTT-S Int. Microwave Symp. Digest* **1998**, 945–948.
68. C. J. Clark, A. A. Moulthrop, M. S. Muha, and C. P. Silva, Transmission response measurements of frequency-translating devices using a vector network analyzer, *IEEE Trans. Microwave Theory Tech.* **44**(12) 2724–2737 (1996).

This page intentionally left blank

Fundamentals of Random Variables and Random Processes for Simulation

6.1. Introduction

In electrical and electronic systems we use voltage or current waveforms as signals for collecting, transmitting, and processing information, as well as for controlling and providing power to a variety of devices. Signals, whether they are voltage or current waveforms, are functions of time and they can be classified as deterministic or random. Deterministic signals can be described by functions in the usual mathematical sense with time t as the independent variable. In contrast to a deterministic signal, a random signal always has some element of uncertainty associated with it, and hence it is not possible to determine its value with certainty at any given point in time.

Models in which there is uncertainty or randomness play an important role in the analysis and design of communication systems and hence they are used extensively in the simulation of communication systems. These models are used in a variety of applications in which the signals, as well as the system parameters, may change randomly. As an example, consider the waveforms that occur in a typical data communication system such as the one shown in Figure 6.1, in which a number of terminals are sending information in binary format over noisy transmission links to a central computer. A transmitter in each link converts the binary data to an electrical waveform by mapping binary digits to pulses of duration T and amplitudes ± 1 . The received waveform in each link is a distorted and noisy version of the transmitted waveform. From the received waveform, the receiver attempts to extract the transmitted binary digits. As shown in Figure 6.1, distortion and noise cause the receiver to make occasional errors in recovering the transmitted binary digit sequence.

As we examine the collection or “ensemble” of waveforms shown in Figure 6.1, randomness is evident in all of these waveforms. By observing one waveform, or one member of the ensemble, say $x_i(t)$, over the time interval $[t_1, t_2]$, we cannot with certainty predict the value of $x_i(t)$ for any other value of t outside the observation interval. Furthermore, knowledge of one member function $x_i(t)$ will not enable us to know the value of another member function $x_j(t)$.

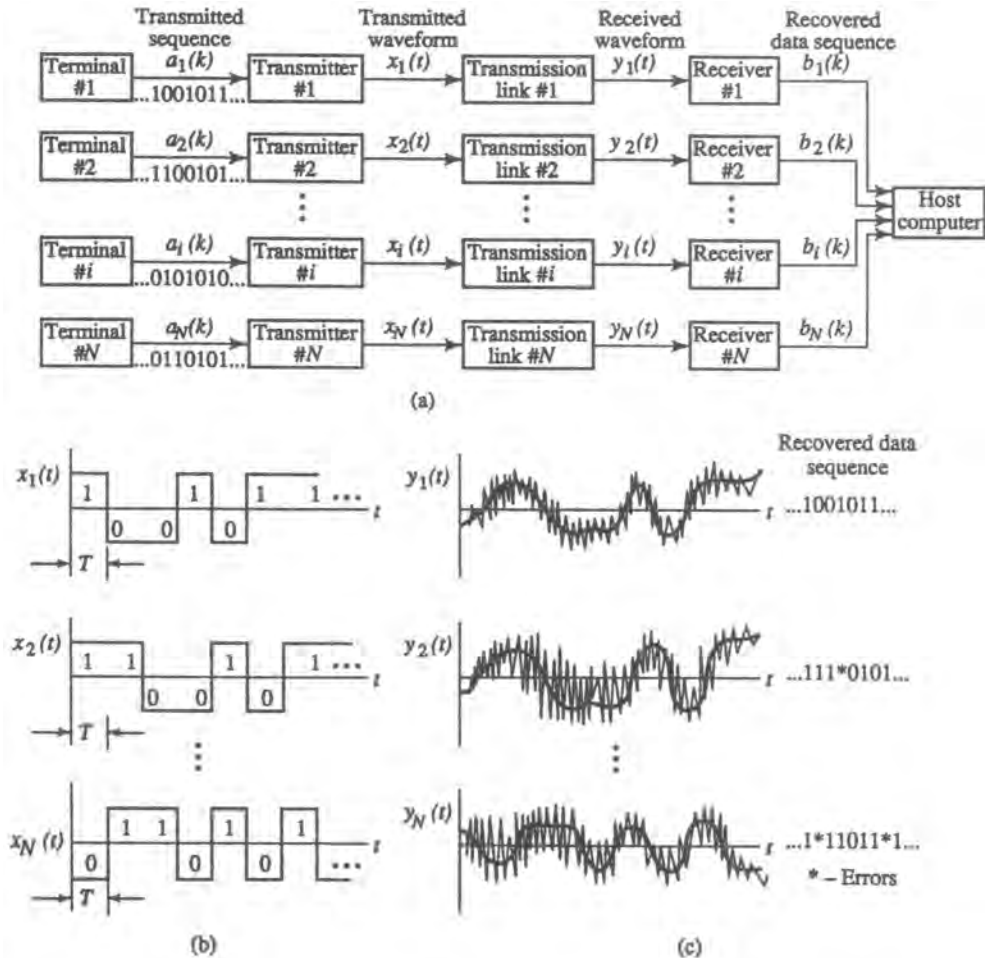


Figure 6.1. Example of random processes and sequences. (a) Data communication system; (b) ensemble of transmitted waveforms; (c) ensemble of received waveforms. (From K. Sam Shanmugan and A. M. Breipohl, *Random Data: Detection, Estimation and Data Analysis*, © John Wiley and Sons, 1988, reproduced with permission.)

Ensembles of waveforms exhibiting random variations are modeled using random variables and random processes. For analysis purposes, the ensemble of transmitted waveforms shown in Figure 6.1 can be modeled by a random process $X(t)$, whereas the ensemble of binary digits can be modeled by a random sequence $\{A(k)\}$. For simulation, we will use sampled values of the random processes, and the system shown in Figure 6.1 can be reduced to the models shown in Figures 6.2a, and 6.2b.

While the random process model characterizes an ensemble of functions of time, the instantaneous value of an ensemble can be modeled by a random variable. For example, the instantaneous value of $X(t)$ at $t = t_0$ can be modeled as a binary-valued random variable with probabilities p_1 and p_2 , where p_1 is the probability that $X(t)$ at $t = t_0$ has a value of +1 and p_2 is the probability that $X(t)$ at $t = t_0$ has the value of -1. The instantaneous amplitude distribution of the noise $N(t)$ can be modeled as a continuous random variable with an

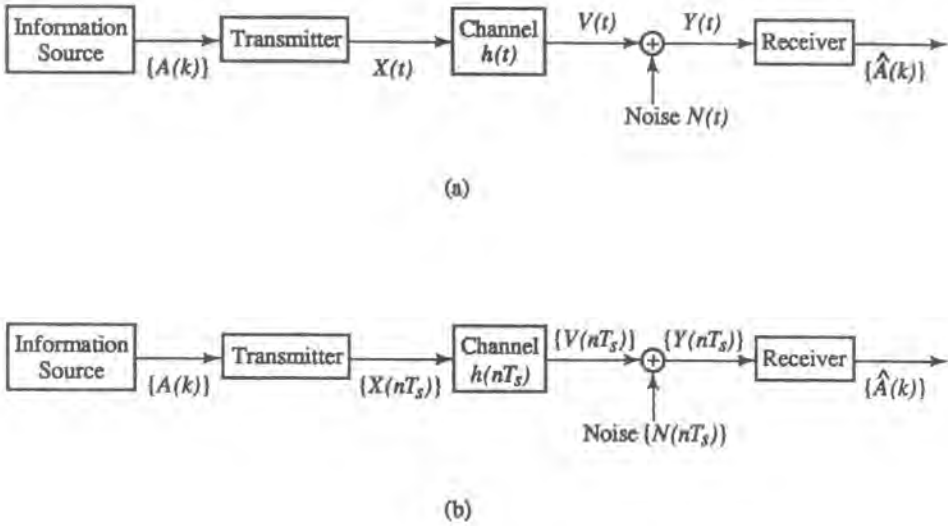


Figure 6.2. Models for the system shown in Figure 6.1. (a) Continuous-time model for the system; (b) discrete-time simulation model for the system.

appropriate probability density function. Furthermore, the joint distribution of the amplitude of $N(t)$ at $t = t_1, \dots, t_n$ can be characterized by a random vector with an appropriate n -dimensional density function.

Random variables and random processes are the major topics covered in this chapter. The coverage is aimed at providing the background for simulation-based analysis and design of communication systems. With reference to Figure 6.2b, simulation involves generating sampled values of the input random processes, computing sampled values of various output random processes, and estimating performance measures from samples of the output processes that were computed during simulation.

Models of random variables and random processes, their properties, as well as the response of systems driven by random inputs are covered in this chapter. Generation of sampled values of random variables and random processes are covered in Chapter 7. Techniques for estimating performance measures are described in Chapters 10 and 11.

For general background as well as details of the material covered in this chapter, the reader can refer to a number of textbooks listed at the end of this chapter (Refs. 1–7).

6.2. Random Variables

6.2.1. Basic Concepts, Definitions, and Notations

6.2.1.1. Introduction

The mathematical theory of probability is built around the simple notions of a random experiment, the outcomes of the random experiment, and a rule for assigning probabilities to the individual outcomes of the random experiment. Probabilities of random events associated with the random experiment are then calculated by extending the probability measure to sets

of outcomes (i.e., events) and to various combinations of sets (unions, intersections, complements, etc.).

A univariate random variable X maps the outcomes of a random experiment to points on the real line $\mathbf{R}_1$ (Figure 6.3). Such a random variable induces a probability measure on the real line by assigning a probability to each set of points in the real line according to (or equal to) the probability of the set of all outcomes of the random experiment which maps to the set on the real line. For example, the probability that X has a value less than or equal to 0.5, or $P(X \leq 0.5)$, is computed as $P(X \leq 0.5) = P(\text{set of all outcomes that map to } X \leq 0.5)$ (see Figure 6.3a).

A random variable X is completely characterized by the set of values it can have and a probability distribution function $F_X(x)$, which is defined as

$$F_X(x) = P(X \leq x)$$

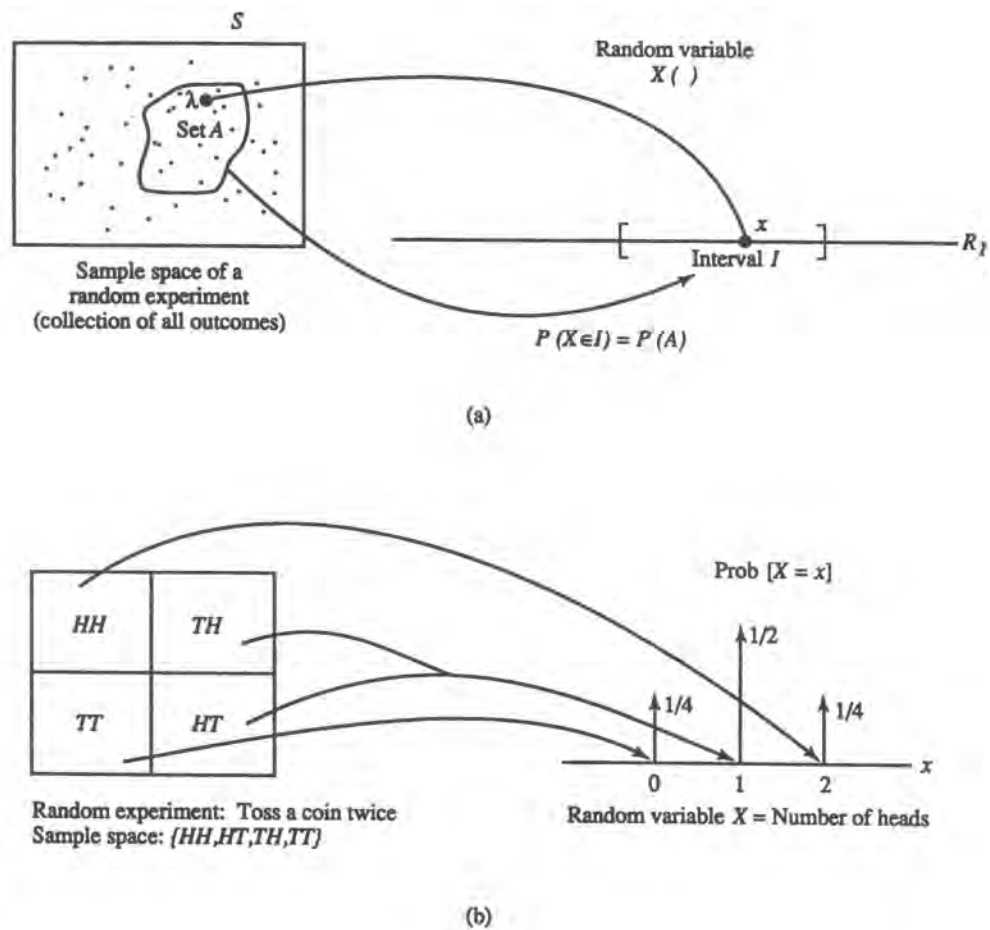


Figure 6.3. (a) Concept of a random variable; $X(\cdot)$ maps each outcome in S to a point on $\mathbf{R}_1$; the set $A \subset S$ maps to the interval $I \subset \mathbf{R}_1$. (b) Example of a random variable.

Note that we use upper case X to denote the random variable and lower case x to denote a particular value of the random variable.

If X can have only a countable number of values (finite or infinite) $x_1, x_2, \dots$, then X is called a discrete random variable and it is characterized by a probability mass function $P(X = x_i)$, $i = 1, 2, \dots$. The relationship between the probability distribution function and the probability mass function is

$$F_X(x) = P(X \leq x) = \sum_{\text{all } x_i \leq x} P(X = x_i) \quad (6.2.1)$$

If the random variable X has an uncountable number of values in one or more sets on the real line, then X is called a continuous random variable and it is described by a probability density function $f_X(x)$, where

$$f_X(x) = \frac{dF_X(x)}{dx}, \quad -\infty < x < \infty \quad (6.2.2)$$

or

$$F_X(x) = \int_{-\infty}^x f_X(a) da \quad (6.2.3)$$

The distribution function, when it is known, provides a complete description for both discrete and continuous random variables.

6.2.1.2. Statistical Averages or Expected Values

Random variables are sometimes described by a few summary measures called statistical averages, or expected values. The expected value of a function of a random variable is defined as

$$E\{g(X)\} = \sum_i g(x_i)P(x = x_i) \quad (6.2.4a)$$

for the discrete case and as

$$E\{g(X)\} = \int_{-\infty}^{\infty} g(x)f_X(x) dx \quad (6.2.4b)$$

for the continuous case.

Two expected values or moments that are most commonly used for characterizing a random variable X are its mean μ_X and variance σ_X^2 ,

$$\begin{aligned} \mu_X &= E\{X\} \\ \sigma_X^2 &= E\{(X - \mu_X)^2\} \end{aligned} \quad (6.2.5)$$

The square root of variance is called the standard deviation. The mean of a random variable is its average value and the variance of a random variable is the measure of the “spread” of the

values of the random variable. When the probability distribution function is not known, then the mean and variance can be used to arrive at bounds on probabilities via Chebyshev's inequality, which has the form (see Section 6.6.1)

$$P[|X - \mu_X| > k] \leq \frac{\sigma_X^2}{k^2} \quad (6.2.6)$$

The Chebyshev inequality (which applies to discrete as well as continuous random variables) can be used to obtain bounds on the probability of finding values of X outside of an interval $\mu_X \pm k\sigma_X$.

6.2.2. Multidimensional Random Variables (Random Vectors)

Whereas a scalar-valued or univariate random variable maps the outcomes of a random experiment to points on the real line, vector-valued random variables map the outcomes of one or more random experiments to points in m -dimensional space ($\mathbf{R}_m$). An example of a two-dimensional random vector is the amplitude and phase of bandlimited noise.

The probability law for vector-valued random variables is specified in terms of a joint distribution function

$$F_{X_1, \dots, X_m}(x_1, \dots, x_m) = P[(X_1 \leq x_1), \dots, (X_m \leq x_m)]$$

The probability law may also be specified by a joint probability mass function (discrete case) or a joint probability density function (continuous case). We treat the continuous case in this section, leaving details of the discrete case for the reader.

The joint probability density function (pdf) of an m -dimensional random vector is the partial derivative of the distribution function with respect to $x_1, x_2, \dots, x_m$ and is denoted by

$$f_{X_1, X_2, \dots, X_m}(x_1, x_2, \dots, x_m)$$

From the joint pdf, we can obtain the marginal pdf's as

$$f_{X_1}(x_1) = \underbrace{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{m-1 \text{ integrals}} f_{X_1, X_2, \dots, X_m}(x_1, x_2, \dots, x_m) dx_2 \dots dx_m \quad (6.2.7a)$$

and

$$\begin{aligned} & f_{X_1, X_2}(x_1, x_2) \\ &= \underbrace{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{m-2 \text{ integrals}} f_{X_1, X_2, \dots, X_m}(x_1, x_2, x_3, \dots, x_m) dx_3 dx_4 \dots dx_m \end{aligned} \quad (6.2.7b)$$

Note that the marginal pdf of any subset of the m variables is obtained by “integrating out” the variables not in the subset.

The conditional density functions are defined as (using $m = 4$ as an example)

$$f_{X_1, X_2, X_3 | X_4}(x_1, x_2, x_3 | x_4) = \frac{f_{X_1, X_2, X_3, X_4}(x_1, x_2, x_3, x_4)}{f_{X_4}(x_4)} \quad (6.2.8a)$$

and

$$f_{X_1, X_2 | X_3, X_4}(x_1, x_2 | x_3, x_4) = \frac{f_{X_1, X_2, X_3, X_4}(x_1, x_2, x_3, x_4)}{f_{X_3, X_4}(x_3, x_4)} \quad (6.2.8b)$$

Expected values are evaluated using multiple integrals. For example,

$$\begin{aligned} E\{g(X_1, X_2, X_3, X_4)\} \\ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x_1, x_2, x_3, x_4) \\ \times f_{X_1, X_2, X_3, X_4}(x_1, x_2, x_3, x_4) dx_1 dx_2 dx_3 dx_4 \end{aligned} \quad (6.2.9)$$

where g is a scalar-valued function. Conditional expected values are defined, for example, as

$$\begin{aligned} E\{g(X_1, X_2, X_3, X_4) | X_3 = x_3, X_4 = x_4\} \\ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x_1, x_2, x_3, x_4) f_{X_1, X_2 | X_3, X_4}(x_1, x_2 | x_3, x_4) dx_1 dx_2 \end{aligned} \quad (6.2.10)$$

Important parameters of the joint distribution are the means, the covariances, and the correlation coefficients, respectively:

$$\mu_{X_i} = E\{X_i\} \quad (6.2.11)$$

$$\sigma_{X_i X_j} = E\{X_i X_j\} - \mu_{X_i} \mu_{X_j} \quad (6.2.12)$$

$$\rho_{X_i X_j} = \frac{\sigma_{X_i X_j}}{(\sigma_{X_i X_i} \sigma_{X_j X_j})^{1/2}} \quad (6.2.13)$$

Note that $\sigma_{X_i X_i}$ is the variance of X_i . We will use both $\sigma_{X_i X_i}$ and $\sigma_{X_i}^2$ to denote the variance of X_i . Sometimes the notations E_{X_i} , $E_{X_i X_j}$, and $E_{X_i | X_j}$ are used to denote expected values with respect to the marginal distribution of X_i , the joint distribution of X_i and X_j , and the conditional distribution of X_i given X_j , respectively. We will use subscripted notation for the expectation operator only when there is ambiguity with the use of unsubscripted notation.

The probability law for random vectors can be specified in a concise form using vector notation. Suppose we are dealing with the joint probability law for m random variables

$X_1, X_2, \dots, X_m$. These m variables can be represented as components of an $m \times 1$ column vector $\mathbf{X}$,

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \quad \text{or} \quad \mathbf{X}^T = (X_1, X_2, \dots, X_m)$$

where T indicates the transpose of a vector (or matrix). The values of $\mathbf{X}$ are points in the m -dimensional space R_m . A specific value of $\mathbf{X}$ is denoted by

$$\mathbf{x}^T = (x_1, x_2, \dots, x_m)$$

Then, the joint pdf is denoted by

$$f_{\mathbf{x}}(\mathbf{x}) = f_{X_1, X_2, \dots, X_m}(x_1, x_2, \dots, x_m)$$

The mean vector is defined as

$$\mu_{\mathbf{x}} = E(\mathbf{X}) = \begin{bmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_m) \end{bmatrix} \quad (6.2.14)$$

and the “covariance-matrix” $\Sigma_{\mathbf{x}}$, an $m \times m$ matrix, is defined as

$$\Sigma_{\mathbf{x}} = E\{\mathbf{X}\mathbf{X}^T\} - \mu_{\mathbf{x}}\mu_{\mathbf{x}}^T = \begin{bmatrix} \sigma_{X_1 X_1} & \sigma_{X_1 X_2} & \cdots & \sigma_{X_1 X_m} \\ \sigma_{X_2 X_1} & \sigma_{X_2 X_2} & \cdots & \sigma_{X_2 X_m} \\ \vdots & & & \vdots \\ \sigma_{X_m X_1} & \sigma_{X_m X_2} & \cdots & \sigma_{X_m X_m} \end{bmatrix}_{m \times m} \quad (6.2.15)$$

The covariance matrix describes the second-order relationship between the components of the random vector $\mathbf{X}$. The components are said to be “uncorrelated” when

$$\sigma_{X_i X_j} = \sigma_{ij} = 0, \quad i \neq j$$

and independent if

$$f_{X_1, X_2, \dots, X_m}(x_1, x_2, \dots, x_m) = \prod_{i=1}^m f_{X_i}(x_i) \quad (6.2.16)$$

6.2.3. Complex Random Variables

A complex random variable Z is defined in terms of the real random variables X and Y by

$$Z = X + jY$$

The expected value of $g(Z)$ is defined as

$$E\{g(Z)\} \triangleq \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(z) f_{X,Y}(x, y) dx dy$$

Thus the mean μ_Z of Z is

$$\mu_Z = E\{Z\} = E\{X\} + jE\{Y\} = \mu_X + j\mu_Y$$

the variance σ_Z^2 is defined as

$$\sigma_Z^2 \triangleq E\{|Z - \mu_Z|^2\}$$

The covariance of two complex random variables Z_m and Z_n is defined by

$$C_{Z_m, Z_n} \triangleq E\{(Z_m - \mu_{Z_m})^*(Z_n - \mu_{Z_n})\} \quad (6.2.17)$$

where the asterisk denotes complex conjugate.

6.3. Univariate Models

In the simulation of communication systems we will use both univariate and multivariate distributions. Univariate distributions will usually be associated with the distribution of individual parameters or their estimators, whereas multivariate distributions are used to jointly describe the distributions of sampled values of a random process at multiple instances of time. Multivariate distributions are also used in cases where we have multiple random processes.

Discrete random variables are used in the simulation of communication systems to model and simulate such quantities as number of bits in a message (i.e., message length) and number of transmission errors. Continuous random variables represent, for example, the interarrival time between messages, the instantaneous value of noise, and attenuation in a communication channel.

In this section we present several examples of univariate random variables. Generation of sampled values of these random variables for use in simulations is presented in Chapter 7.

Table 6.1 Example of a Probability Mass Function

Value of X	0	1	2	3
Probability mass function	1/8	3/8	3/8	1/8

6.3.1. Univariate Models—Discrete

The probability mass function of a discrete random variable can be specified in empirical form (as a table) or in closed form by a formula. For example, we can specify the probability mass function of a random variable in table form as shown in Table 6.1, or in closed form as

$$P(X = k) = \frac{3!}{k!(3 - k)!} \left(\frac{1}{2}\right)^3, \quad k = 0, 1, 2, 3$$

where $m! = (m)(m - 1)(m - 2) \dots (3)(2)(1)$.

The empirical form is usually derived from data and the closed form is usually obtained from a description of the underlying random experiment and the definition of the random variable. The closed form is very useful in simulation and analysis, and we present below several examples of probability mass functions.

The models given below are based on a number of assumptions about the underlying random experiment and the definition of the random variable. While we omit the detailed derivation of the formula for the probability mass functions given below, we state the assumptions and present typical applications for each model.

Before using any of these models for a given simulation application, one has to make sure that the assumptions made in deriving the probability mass function are consistent with the modeling assumptions for the application.

6.3.1.1. Uniform

Assumptions: All values of the random variable are equally likely to occur.

Probability Mass Function:

$$P(X = i) = \frac{1}{n}, \quad i = 1, 2, \dots, n \quad (6.3.1)$$

Parameter(s): n .

Mean and Variance:

$$\mu_X = \frac{n + 1}{2} \quad (6.3.2a)$$

$$\sigma_X^2 = \frac{n^2 - 1}{12} \quad (6.3.2b)$$

6.3.1.2. Binomial

Assumptions: X is the number of successes in n independent trials of a random experiment in which p is the probability of success in a single trial.

Probability Mass Function:

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, \dots, n \quad (6.3.3)$$

where

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}$$

Mean and Variance:

$$\mu_X = np \quad (6.3.4a)$$

$$\sigma_X^2 = np(1-p) \quad (6.3.4b)$$

Applications: Number of ones in a block of n binary digits; number of random transmission errors in a block of n digits transmitted over a noisy communication channel.

6.3.1.3. Negative Binomial

Assumptions: X is the number of trials of a random experiment until the r th success where p is the probability of success in a single trial.

Probability Mass Function:

$$P(X = x) = \binom{x-1}{r-1} p^r (1-p)^{x-r}, \quad x = r, r+1, r+2, \dots \quad (6.3.5)$$

Parameters: p and $r, p > 0, r > 0$.

Mean and Variance:

$$\mu_X = r/p \quad (6.3.6a)$$

$$\sigma_X^2 = r(1-p)/p^2 \quad (6.3.6b)$$

Applications: Number of repeated telephone call attempts before getting through; number of transmissions over a noisy channel before a message gets through.

[Note: $r = 1$ is called a geometric variable.]

6.3.1.4. Poisson

Assumptions: X is the number of events that take place in a unit interval of time under the following assumptions.

- (i) Times of occurrences of events are distinct.
- (ii) Any finite interval contains only a finite number of occurrences.
- (iii) Any infinite interval contains an infinite number of occurrences.
- (iv) Events do not occur at predetermined times.
- (v) Number of occurrences in nonoverlapping intervals are independent.

Probability Mass Function:

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k = 0, 1, 2, \dots \quad (6.3.7)$$

Parameter: λ .

Mean and Variance:

$$\mu_X = \lambda \quad (6.3.8a)$$

$$\sigma_X^2 = \lambda \quad (6.3.8b)$$

Applications: Traffic models in communication systems (e.g., number of message arrivals in a given interval of time), emission of particles such as electrons, photons, etc., and shot noise.

6.3.2. Univariate Models—Continuous

Several models for continuous random variables that are used for simulating communication systems are presented below. In addition to the probability density functions, we also list the probability distribution functions for the cases where they can be expressed in closed form.

6.3.2.1. Uniform

Assumptions: Probability that the random variable has a value in a given interval is proportional to interval length.

Density Function:

$$f_X(x) = \frac{1}{b-a}, \quad a < x < b \quad (6.3.9)$$

Distribution Function:

$$F_X(x) = \frac{x-a}{b-a}, \quad a < x < b \quad (6.3.10)$$

Parameters: a, b ($a < b$).

Mean and Variance:

$$\mu_X = \frac{b+a}{2} \quad (6.3.11a)$$

$$\sigma_X^2 = \frac{(b-a)^2}{12} \quad (6.3.11b)$$

Applications: In simulation, samples from the uniform distribution are used to generate samples from other probability distributions (see Section 7.2.2).

6.3.2.2. Gaussian (Normal)

Assumptions: By virtue of the central limit theorem, a random variable that is determined by the sum of a large number of independent causes tends to have a Gaussian pdf.

Probability Density Function:

$$f_X(x) = \frac{1}{(2\pi)^{1/2} \sigma_X} \exp \left[\frac{-(x - \mu_X)^2}{2\sigma_X^2} \right] \quad -\infty < x < \infty \quad (6.3.12)$$

Parameters:

mean: μ_X

variance: σ_X^2

Distribution Function: The Gaussian density function cannot be integrated in closed form; tabulated values of the distribution function (obtained via numerical integration of the pdf) are given in Appendix B to this book and in Ref. 8. Several good assumptions for the tail probability under a Gaussian pdf are available.⁽⁹⁾ One simple form is

$$\begin{aligned} Q(x) &= \int_x^\infty \frac{1}{(2\pi)^{1/2}} e^{-t^2/2} dt \\ &\approx \left[\frac{1}{(1 - \alpha)x + \alpha(x^2 + b)^{1/2}} \right] \frac{1}{(2\pi)^{1/2}} e^{-x^2/2} \end{aligned}$$

where $\alpha = 1/\pi$ and $b = 2\pi$.

Applications: Model for the amplitude distribution of thermal noise; also used as an approximation for a number of other distributions.

6.3.2.3. Exponential

Assumptions: None

Probability Density Function:

$$f_X(x) = \lambda e^{-\lambda x}, \quad x > 0 \quad (6.3.13)$$

Parameters: λ ($\lambda > 0$).

Mean and Variance:

$$\mu_X = \frac{1}{\lambda} \quad (6.3.14a)$$

$$\sigma_X^2 = \frac{1}{\lambda^2} \quad (6.3.14b)$$

Distribution Function:

$$F_X(x) = 1 - e^{-\lambda x}, \quad x > 0 \quad (6.3.15)$$

Applications: Message length and interarrival time distribution in data communication systems; reliability models (time of failure).

6.3.2.4. Gamma

Assumption: None.

Density Function:

$$f_X(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta}, \quad 0 < x < \infty \quad (6.3.16)$$

where the gamma function $\Gamma(\cdot)$ is defined as

$$\Gamma(\alpha) = \int_0^\infty y^{\alpha-1} e^{-y} dy = (\alpha - 1)\Gamma(\alpha - 1) \quad (6.3.17)$$

$\Gamma(\alpha) = (\alpha - 1)!$ when α is an integer

Parameters: α, β ($\alpha, \beta > 0$).

Mean and Variance:

$$\mu_X = \alpha\beta \quad (6.3.18a)$$

$$\sigma_X^2 = \alpha\beta^2 \quad (6.3.18b)$$

Distribution Function: No closed form; tables are given in Ref. 8.

Application: General model for waiting times. For example, if X is the time interval needed to obtain exactly k arrivals of a Poisson process with parameter λ (which is the arrival rate), then X has a gamma distribution with $\alpha = k$ and $\beta = 1/\lambda$. When $k = 1$, X has an exponential distribution. When α is limited to the integer set $\{1, 2, \dots\}$, the resulting gamma distribution is called the Erlang distribution.

6.3.2.5. Rayleigh

Assumptions: $X = (X_1^2 + X_2^2)^{1/2}$, where X_1 and X_2 are independent Gaussian random variables each with zero mean and variance σ^2 .

Density Function:

$$f_X(x) = \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right), \quad x > 0 \quad (6.3.19)$$

Parameter(s): σ^2 ($\sigma^2 > 0$).

Mean and Variance:

$$\mu_X = \sigma(\pi/2)^{1/2} \quad (6.3.20a)$$

$$\sigma_X^2 = \sigma^2(2 - \pi/2) \quad (6.3.20b)$$

Distribution Function:

$$F_X(x) = 1 - \exp(-x^2/2\sigma^2), \quad x > 0$$

Applications: Fading in communication channels; envelope of narrow band Gaussian noise.

6.3.2.6. Chi-Square

Assumptions: $X = \sum_{i=1}^n X_i^2$, where the X_i are independent Gaussian random variables with $\mu_i = 0$ and $\sigma_i^2 = 1$.

Density Function:

$$f_X(x) = \frac{1}{2^{m/2}\Gamma(m/2)} x^{m/2-1} e^{-x/2}, \quad x > 0 \quad (6.3.21)$$

where the $\Gamma(\cdot)$ function is given in (6.3.17).

Parameters: m (degrees of freedom).

Mean and Variance:

$$\mu_X = m \quad (6.3.22a)$$

$$\sigma_X^2 = 2m \quad (6.3.22b)$$

Distribution Function: Cannot be expressed in closed form. For tabulated values, see Ref. 8 and Appendix E of this book.

Applications: Used in analyzing the variance of estimators and in goodness-of-fit tests.

■ *Remarks:* (1) The chi-square variable is a special case of the gamma variable with $\alpha = m/2$ and $\beta = 2$ (2) If $\mu_i \neq 0$, then $\sum X_i^2$ has a “noncentral” chi-square distribution.⁽⁴⁾ ■

There are a number of other probability density functions that are useful for statistical analysis of estimators of parameters and performance measures. We list these pdf's below; the reader can find detailed derivation of these pdf's and their properties in Refs. 1–6.

6.3.2.7. Student's t

Let $T_m = Z/(Y_m/m)^{1/2}$, where Z is Gaussian with $\mu_z = 0$ and $\sigma_z^2 = 1$, and Y_m is an independent chi square with m degrees of freedom; T_m then has Student's t distributions with pdf

$$f_{T_m}(t) = \frac{\Gamma((m+1)/2)}{(\pi m)^{1/2} \Gamma(m/2) (1 + t^2/m)^{(m+1)/2}}, \quad -\infty < t < \infty \quad (6.3.23)$$

The t distribution is used to set confidence limits on estimates of mean when the variance is not known (see Chapters 10 and 11). When $\mu \neq 0$, the resulting t distribution is called noncentral t ⁽⁴⁾.

6.3.2.8. *F Distribution*

Let U and V be two independent chi-square densities with m_1 and m_2 degrees of freedom, respectively, and

$$F = \frac{U/m_1}{V/m_2} \quad (6.3.24a)$$

Then F had an “ F ” distribution with the pdf

$$f_F(\lambda) = \frac{(m_1/m_2)^{m_1/2} \lambda^{(m_1/2-1)}}{\Gamma(m_1/2)\Gamma(m_2/2)(m_1\lambda/m_2 + 1)^{(m_1+m_2)/2}} \Gamma\left(\frac{m_1 + m_2}{2}\right), \quad \lambda \geq 0 \quad (6.3.24b)$$

The F distribution is used for analyzing the properties of estimators of variances, power levels, and signal-to-noise ratios as explained in Chapter 10. If the variable U in (6.3.24a) is noncentral chi square, then the resulting F distribution is called a noncentral F .⁽⁴⁾

6.3.2.9. *Generalized Exponential*

This class of pdf's can be used as a model for Gaussian variables that are modified by a mild nonlinearity:

$$f_X(x) = \frac{\alpha}{2\sqrt{2}\sigma\Gamma(1/\alpha)} \exp\left(-\left|\frac{x-\mu}{\sqrt{2}\sigma}\right|^\alpha\right), \quad -\infty < x < \infty \quad (6.3.25)$$

$$\mu_X = \mu \text{ and } \sigma_X^2 = 2\sigma^2\Gamma(3/\alpha)/\Gamma(1/\alpha).$$

Note that when $\alpha = 2$, the generalized pdf reduces to the Gaussian pdf.

6.4. Multivariate Models

Two multivariate distributions that are quite useful in analysis and simulation are the multinomial and the multivariate Gaussian distributions, which are generalizations of the univariate cases.

6.4.1. Multinomial

Let a random experiment be repeated n independent times. On each repetition, the experiment terminates in one of k mutually exclusive and exhaustive events $A_1, A_2, \dots, A_k$ with probabilities $p_1, p_2, \dots, p_k$. Define X_i to be equal to the number of times event A_i occurs in n trials. Then the joint probability mass function of $X_1, X_2, \dots, X_k$ is

$$P(X_1 = x_1, X_2 = x_2, \dots, X_k = x_k) = \frac{n!}{x_1!x_2!\dots x_k!} p_1^{x_1} p_2^{x_2} \dots p_k^{x_k} \quad (6.4.1)$$

where $x_1, x_2, \dots, x_k$ are nonnegative integers, $0 < p_i < 1$, and

$$x_1 + x_2 + \dots + x_k = n$$

$$p_1 + p_2 + \dots + p_k = 1$$

This distribution can be used, for example, to model the occurrence of various symbol sequences from a k -ary alphabet and also for modeling error patterns in a block of n symbols.

The mean, variance, and covariance of multinomial variables are, respectively,

$$\mu_{x_i} = np_i, \quad i = 1, 2, \dots, k \quad (6.4.2a)$$

$$\sigma_{x_i}^2 = np_i(1 - p_i), \quad i = 1, 2, \dots, k \quad (6.4.2b)$$

$$\sigma_{x_i x_j} = -np_i p_j \quad (6.4.2c)$$

6.4.2. Multivariate Gaussian

Gaussian random processes are used extensively for modeling noise and for approximating some types of signals. Sampled values of Gaussian random processes have a multivariate Gaussian distribution, and the study of multivariate Gaussian random variables is an important step in simulating and analyzing the performance of systems in which Gaussian signals and noise exist.

A random vector $\mathbf{X}$ is multivariate Gaussian if it has a pdf of the form

$$f_{\mathbf{x}}(\mathbf{x}) = \left[(2\pi)^{m/2} |\mathbf{\Sigma}_{\mathbf{x}}|^{1/2} \right]^{-1} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^T \mathbf{\Sigma}_{\mathbf{x}}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) \right] \quad (6.4.3)$$

where $\boldsymbol{\mu}_{\mathbf{x}}$ is the mean vector, $\mathbf{\Sigma}_{\mathbf{x}}$ is the covariance matrix, $\mathbf{\Sigma}_{\mathbf{x}}^{-1}$ is its inverse, $|\mathbf{\Sigma}_{\mathbf{x}}|$ is the determinant of $\mathbf{\Sigma}_{\mathbf{x}}$, and $\mathbf{X}$ is of dimension m .

6.4.2.1. Properties of the Multivariate Gaussian Distribution

We state next some of the important properties of the multivariate Gaussian distribution. Proofs of these properties are given in Ref. 3.

1. Suppose $\mathbf{X}$ has an m -dimensional multivariate Gaussian distribution. If we partition $\mathbf{X}$ as

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix}, \quad \mathbf{X}_1 = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} X_{k+1} \\ X_{k+2} \\ \vdots \\ X_m \end{bmatrix}$$

and

$$\boldsymbol{\mu}_{\mathbf{x}} = \begin{bmatrix} \mu_{x_1} \\ \mu_{x_2} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

where $\boldsymbol{\mu}_{\mathbf{x}_1}$ is $k \times 1$ and $\boldsymbol{\Sigma}_{11}$ is $k \times k$, then $\mathbf{X}_1$ has a k -dimensional multivariate Gaussian distribution with a mean $\boldsymbol{\mu}_{\mathbf{x}_1}$ and covariance $\boldsymbol{\Sigma}_{11}$.

2. If $\boldsymbol{\Sigma}_{\mathbf{x}}$ is a diagonal matrix, that is,

$$\boldsymbol{\Sigma}_{\mathbf{x}} = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_m^2 \end{bmatrix}$$

then the components of $\mathbf{X}$ are independent (i.e., uncorrelatedness implies independence; however, this property does not hold for other distributions).

3. If $\mathbf{A}$ is a $k \times m$ matrix of rank k , then $\mathbf{Y} = \mathbf{AX}$ has a k -variate Gaussian distribution with

$$\boldsymbol{\mu}_{\mathbf{Y}} = \mathbf{A}\boldsymbol{\mu}_{\mathbf{x}} \quad (6.4.4a)$$

$$\boldsymbol{\Sigma}_{\mathbf{Y}} = \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{A}^T \quad (6.4.4b)$$

4. With a partition of $\mathbf{X}$ as in property 1, the conditional density of $\mathbf{X}_1$ given $\mathbf{X}_2 = \mathbf{x}_2$ is a k -dimensional multivariate Gaussian with

$$\boldsymbol{\mu}_{\mathbf{x}_1|\mathbf{x}_2} = E[\mathbf{X}_1|\mathbf{X}_2 = \mathbf{x}_2] = \boldsymbol{\mu}_{\mathbf{x}_1} + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_{\mathbf{x}_2}) \quad (6.4.5a)$$

and

$$\boldsymbol{\Sigma}_{\mathbf{x}_1|\mathbf{x}_2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \quad (6.4.5b)$$

Properties 1, 3, and 4 state that marginals, conditionals, as well as linear transformations derived from a multivariate Gaussian distribution all have multivariate Gaussian distributions.

■ *Example 6.4.1.* Suppose $\mathbf{X}$ is four-variate Gaussian with

$$\boldsymbol{\mu}_{\mathbf{x}} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

and

$$\Sigma_{\mathbf{x}} = \begin{bmatrix} 6 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 3 \end{bmatrix}$$

Let

$$\mathbf{x}_1 = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} X_3 \\ X_4 \end{bmatrix}$$

- (a) Find the distribution of $\mathbf{X}_1$.
 (b) Find the distribution of

$$\mathbf{Y} = \begin{bmatrix} 2X_1 \\ X_1 + 2X_2 \\ X_3 + X_4 \end{bmatrix}$$

- (c) Find the distribution of $\mathbf{X}_1$ given $\mathbf{x}_2 = (x_3, x_4)^T$.
Solution: (a) $\mathbf{X}_1$ has a bivariate Gaussian distribution with

$$\mu_{\mathbf{x}_1} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{and} \quad \Sigma_{\mathbf{x}_1} = \begin{bmatrix} 6 & 3 \\ 3 & 4 \end{bmatrix}$$

- (b) We can express $\mathbf{Y}$ as

$$\mathbf{Y} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \mathbf{A}\mathbf{X}$$

Hence $\mathbf{Y}$ has a trivariate Gaussian distribution with

$$\mu_{\mathbf{Y}} = \mathbf{A}\mu_{\mathbf{x}} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix}$$

and

$$\begin{aligned}
 \Sigma_Y &= A \Sigma_X A^T \\
 &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 6 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 3 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 24 & 24 & 6 \\ 24 & 34 & 13 \\ 6 & 13 & 13 \end{bmatrix}
 \end{aligned}$$

(c) $\mathbf{X}_1$ given $\mathbf{X}_2 = (x_3, x_4)^T$ has a bivariate Gaussian distribution with

$$\begin{aligned}
 \mu_{\mathbf{x}_1|\mathbf{x}_2} &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 3 & 3 \end{pmatrix}^{-1} \begin{pmatrix} x_3 - 1 \\ x_4 - 0 \end{pmatrix} \\
 &= \begin{bmatrix} x_2 - \frac{2}{3}x_4 + 1 \\ x_3 - \frac{1}{3}x_4 \end{bmatrix}
 \end{aligned}$$

and

$$\begin{aligned}
 \Sigma_{\mathbf{X}_1|\mathbf{X}_2} &= \begin{bmatrix} 6 & 3 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 3 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \\
 &= \begin{bmatrix} 14/3 & 4/3 \\ 4/3 & 5/3 \end{bmatrix}
 \end{aligned}$$

■

6.4.2.2. Moments of Multivariate Gaussian pdf

Although (6.4.4) gives the moments of a linear combination of multivariate Gaussian variables, there are many simulation applications where we need to compute moments such as $E\{X_1^2 X_2^2\}$ and $E\{X_1 X_3 X_4\}$. Two formulas that are quite useful are

$$E\{X_i^n\} = \begin{cases} (1)(3)(5) \dots (n-1) \sigma_{X_i}^2, & n > 2, \quad n \text{ even} \\ 0 & n \text{ odd} \end{cases} \quad (6.4.6a)$$

$$E\{X_1 X_2 X_3 X_4\} = \sigma_{12} \sigma_{34} + \sigma_{23} \sigma_{14} + \sigma_{24} \sigma_{13} \quad (6.4.6b)$$

where the X_i are assumed to have zero means (see Refs. 1 and 2 for proof) and $\sigma_{ij} = E\{X_i X_j\}$.

6.5. Transformations (Functions) of Random Variables

In simulation-based analysis of communication systems we are often interested in finding the properties of a signal after it has been “processed” by the system. Typical signal

processing operations include integration, weighted averaging, and limiting. These signal processing operations may be viewed as transformations of a set of input variables to a set of output variables. If the input is a set of random variables, then the output will also be a set of random variables. In this section, we develop techniques for obtaining the probability law (distribution) for the set of output random variables, given the transformation and the probability law for the set of input random variables.

If the response of a system or component can be obtained analytically, then there is no need to explicitly simulate that component. In such cases, the effect can be accounted for by including the equivalent representation at the output. If this can be done for several blocks in cascade, then we avoid simulating the response through each block in the chain and the net effect can be accounted for by including an equivalent representation at the output of the last block in the chain. This notion of equivalent representation is a powerful one which can lead to significant saving in simulation time (see Chapter 12 for an example).

The general type of problem we address in this section is the following. Assume that X is a random variable with ensemble $\mathcal{S}_X$ and a known probability distribution. Let g be a scalar function that maps each $x \in \mathcal{S}_X$ to $y = g(x)$. The expression

$$Y = g(X)$$

defines a new random variable as follows (see Figure 6.4). For a given outcome λ , $X(\lambda)$ is a number x , and $g[X(\lambda)]$ is another number specified by $g(x)$. This number is the value of the random variable Y , that is, $Y(\lambda) = y = g(x)$. The ensemble $\mathcal{S}_Y$ of Y is the set

$$\mathcal{S}_Y = \{y = g(x) : x \in \mathcal{S}_X\}$$

We are interested in finding the probability law for Y .

The method used for finding the probability law for Y is to equate the probabilities of equivalent events. Suppose $C \subset \mathcal{S}_Y$. Because the function $g(x)$ maps $\mathcal{S}_X$ to $\mathcal{S}_Y$, there is an equivalent subset B , $B \subset \mathcal{S}_X$, defined by

$$B = \{x : g(x) \in C\}$$

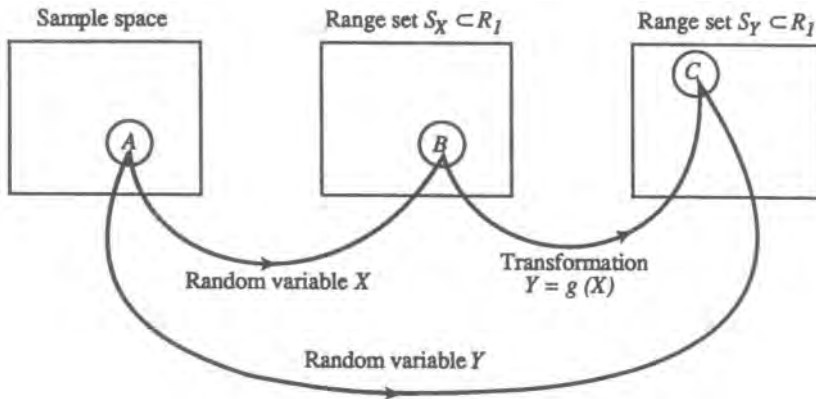


Figure 6.4. Transformation of a random variable.

Now, B corresponds to event A , which is a subset of the sample space S (see Figure 6.4). It is obvious that A maps to C and hence

$$P(C) = P(A) = P(B)$$

Now, suppose that g is a continuous function and $C = [-\infty, y]$. If $B = \{x : g(x) \leq y\}$, then

$$\begin{aligned} P(C) &= P(Y \leq y) = F_Y(y) \\ &= \int_B f_X(x) dx \end{aligned}$$

which gives the distribution function of Y in terms of the density function of X . The density function of Y (if Y is a continuous random variable) can be obtained by differentiating $F_Y(y)$.

As an alternative approach, suppose I_y is a small interval of length Δy containing the point y . Let $I_x = \{x : g(x) \in I_y\}$. Then, we have

$$\begin{aligned} P(Y \in I_y) &\approx f_Y(y) \Delta y \\ &\approx \int_{I_x} f_X(x) dx \end{aligned}$$

which shows that we can derive the density of Y from the density of X .

We will use the principles outlined in the preceding paragraphs to find the distribution of scalar-valued as well as vector-valued functions of random variables.

6.5.1. Scalar-Valued Function of One Random Variable

6.5.1.1. Discrete Case

Suppose X is a discrete random variable that can have one of n values $x_1, x_2, \dots, x_n$. Let $g(x)$ be a scalar-valued function. Then $Y = g(X)$ is a discrete random variable that can have one of m , $m \leq n$, values $y_1, y_2, \dots, y_m$. If $g(X)$ is a one-to-one mapping, then m will be equal to n . However, if $g(x)$ is a many-to-one mapping, then m will be smaller than n . The probability mass function of y can be obtained easily from the probability mass function of X as

$$P(Y = y_j) = \sum P(X = x_i)$$

where the sum is over all values of x_i that map to y_j .

6.5.1.2. Continuous Case

If X is a continuous random variable, then the pdf of $Y = g(X)$ can be obtained from the pdf of X as follows. Let y be a particular value of Y and let $x^{(1)}, x^{(2)}, \dots, x^{(k)}$ be the k roots of the equation $y = g(x)$. That is, $y = g(x^{(1)}) = g(x^{(2)}) = \dots = g(x^{(k)})$. For example, if $y = x^2$,

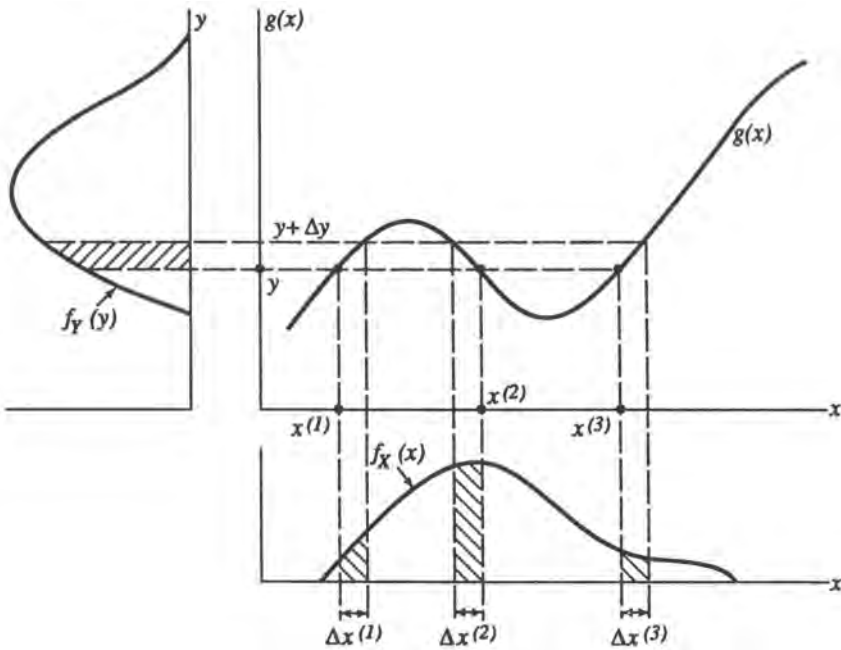


Figure 6.5. Transformation of a continuous random variable. (From K. Sam Shanmugan and A. M. Breipohl, *Random Data: Detection, Estimation and Data Analysis*, © John Wiley and Sons, 1988, reproduced with permission.)

then the roots (or the solutions) are $x^{(1)} = +\sqrt{y}$ and $x^{(2)} = -\sqrt{y}$ (see also Figure 6.5 for another example).

We know that

$$P[y < Y \leq y + \Delta y] = f_Y(y)\Delta y \quad \text{as } \Delta y \rightarrow 0$$

Now if we can find the set of values of x such that $y < g(x) \leq y + \Delta y$, then we can obtain $f_Y(y)\Delta y$ from the probability that X belongs to this set. That is,

$$P[y < Y \leq y + \Delta y] = P[y < g(X) \leq y + \Delta y]$$

For the example shown in Figure 6.5, this set consists of the following three intervals:

$$\begin{aligned} x^{(1)} < x \leq x^{(1)} + \Delta x^{(1)} \\ x^{(2)} + \Delta x^{(2)} < x \leq x^{(2)} \\ x^{(3)} < x \leq x^{(3)} + \Delta x^{(3)} \end{aligned}$$

where $\Delta x^{(1)} > 0$, $\Delta x^{(3)} > 0$, but $\Delta x^{(2)} < 0$. Hence,

$$\begin{aligned} P[y < Y < y + \Delta y] &= P[x^{(1)} < X < x^{(1)} + \Delta x^{(1)}] + P[x^{(2)} + \Delta x^{(2)} < X < x^{(2)}] \\ &\quad + P[x^{(3)} < X < x^{(3)} + \Delta x^{(3)}] \\ &\approx \sum_{i=1}^3 f_X(x^{(i)}) |\Delta x^{(i)}| \end{aligned}$$

Since the slope $g'(x)$ of $g(x)$ is $\Delta y / \Delta x$, we have

$$\Delta x^{(1)} = \Delta y / g'(x^{(1)})$$

$$\Delta x^{(2)} = \Delta y / g'(x^{(2)})$$

$$\Delta x^{(3)} = \Delta y / g'(x^{(3)})$$

Hence we conclude that, when we have three roots for the equation $y = g(x)$,

$$f_Y(y)\Delta y \approx \frac{f_X(x^{(1)})}{g'(x^{(1)})} \Delta y + \frac{f_X(x^{(2)})}{|g'(x^{(2)})|} \Delta y + \frac{f_X(x^{(3)})}{g'(x^{(3)})} \Delta y$$

Canceling the Δy and generalizing the result, we have

$$f_Y(y) = \sum_{i=1}^k \frac{f_X(x^{(i)})}{|g'(x^{(i)})|} \quad (6.5.1)$$

$g(x)$ is also called the Jacobian of the transformation and is often denoted by $J(x)$. Equation (6.5.1) gives the pdf of the transformed variable Y in terms of the pdf of X , which is known. The use of (6.5.1) is limited only by our ability to find the roots of the equation $y = g(x)$. If $g(x)$ is highly nonlinear, then the solutions of $y = g(x)$ can be difficult to find.

■ *Example 6.5.1.* Suppose X has a Gaussian distribution with mean 0 and variance of 1, and $Y = X^2 + 4$. Find the pdf of Y .

Solution: $y = g(x) = x^2 + 4$ has two roots,

$$x^{(1)} = +(y - 4)^{1/2}$$

$$x^{(2)} = -(y - 4)^{1/2}$$

and hence

$$g'(x^{(1)}) = 2(y - 4)^{1/2}$$

$$g'(x^{(2)}) = -2(y - 4)^{1/2}$$

The density function of Y is given by

$$f_Y(y) = \frac{f_X(x^{(1)})}{|g'(x^{(1)})|} + \frac{f_X(x^{(2)})}{|g'(x^{(2)})|}$$

With $f_X(x)$ given as

$$f_X(x) = \frac{1}{(2\pi)^{1/2}} e^{-x^2/2}$$

we obtain

$$f_Y(y) = \begin{cases} \frac{1}{(2\pi)^{1/2}(y-4)^{1/2}} e^{-(y-4)/2}, & y \geq 4 \\ 0, & y < 4 \end{cases}$$

Note that since $y = x^2 + 4$ and the domain of X is $(-\infty, \infty)$, the domain of Y is $[4, \infty)$. ■

6.5.2. Functions of Several Random Variables

The results derived in the preceding section can be generalized to find the joint distribution of n random variables $Y_1, Y_2, \dots, Y_n$, given the distribution of n related random variables $X_1, X_2, \dots, X_n$ and the relationship between the two sets of random variables

$$Y_i = g_i(X_1, X_2, \dots, X_n), \quad i = 1, 2, \dots, n$$

Using the vector notation, it can be shown that

$$f_Y(\mathbf{y}) = \sum_{i=1}^k \frac{f_X(\mathbf{x}^{(i)})}{|J(\mathbf{x}^{(i)})|} \quad (6.5.2a)$$

where $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]^T$ is the i th solution to $\mathbf{y} = \mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_n(\mathbf{x})]^T$, and the Jacobian J is defined by

$$J[\mathbf{x}^{(i)}] = \begin{vmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \dots & \frac{\partial g_1}{\partial x_n} \\ \vdots & & & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \dots & \frac{\partial g_n}{\partial x_n} \end{vmatrix}_{\text{at } \mathbf{x}^{(i)}} \quad (6.5.2b)$$

Suppose we have n random variables with known joint pdf, and we are interested in the joint pdf of $m < n$ function of them, say

$$y_i = g_i(x_1, x_2, \dots, x_n), \quad i = 1, 2, \dots, m$$

Now, we can define $n - m$ additional functions

$$y_j = g_j(x_1, x_2, \dots, x_n), \quad j = m + 1, \dots, n$$

in any convenient way so that the Jacobian is nonzero, compute the joint pdf of $Y_1, Y_2, \dots, Y_n$, and then obtain the marginal pdf of $Y_1, Y_2, \dots, Y_m$ by integrating out $Y_{m+1}, \dots, Y_n$. If the additional functions are carefully chosen, then the inverse can be easily found and the resulting integration can be handled, but often with some difficulty.

6.5.2.1. Special Case—Linear Transformation

One of the most frequently used types of transformation is the affine transformation, where each of the new variables is a linear combination of the old variables plus a constant. That is,

$$\begin{aligned} Y_1 &= a_{1,1}X_1 + a_{1,2}X_2 + \dots + a_{1,n}X_n + b_1 \\ Y_2 &= a_{2,1}X_1 + a_{2,2}X_2 + \dots + a_{2,n}X_n + b_2 \\ &\vdots \\ Y_n &= a_{n,1}X_1 + a_{n,2}X_2 + \dots + a_{n,n}X_n + b_n \end{aligned}$$

where the a_{ij} and b_i are all constants. In matrix notation we can write this transformation as

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{B} \quad (6.5.3a)$$

where $\mathbf{A}$ is $n \times n$, and $\mathbf{Y}$, $\mathbf{X}$, and $\mathbf{B}$ are $n \times 1$ matrices. If $\mathbf{A}$ is nonsingular, then the inverse transformation exists and is given by

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{Y} - \mathbf{A}^{-1}\mathbf{B}$$

The Jacobian of the transformation is

$$J = \begin{vmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{vmatrix} = |\mathbf{A}|$$

Substituting the preceding two equations into (6.5.2a), we obtain the pdf of $\mathbf{Y}$ as

$$f_Y(\mathbf{y}) = f_X(\mathbf{A}^{-1}\mathbf{y} - \mathbf{A}^{-1}\mathbf{B})|\mathbf{A}|^{-1} \quad (6.5.3b)$$

We can use this result to find analytically, without simulation, the distribution of the output of a linear component in a communication system whose input consists of sampled values of a random process.

6.5.2.2. Sum of Random Variables

We now consider $Y_1 = X_1 + X_2$, where X_1 and X_2 are independent random variables. As suggested before, let us introduce an additional function $Y_2 = X_2$ so that the transformation is given by

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

From (6.5.3b) it follows that

$$\begin{aligned} f_{Y_1, Y_2}(y_1, y_2) &= f_{X_1, X_2}(y_1 - y_2, y_2) \\ &= f_{X_1}(y_1 - y_2)f_{X_2}(y_2) \end{aligned}$$

since X_1 and X_2 are independent.

The pdf of Y_1 is obtained by integration as

$$f_{Y_1}(y_1) = \int_{-\infty}^{\infty} f_{X_1}(y_1 - y_2) f_{X_2}(y_2) dy_2 \quad (6.5.4a)$$

The relationship given in (6.5.4a) is the convolution of f_{X_1} and f_{X_2} , which is written symbolically as

$$f_{Y_1} = f_{X_1} * f_{X_2} \quad (6.5.4b)$$

Thus, the density function of the sum of two independent random variables is given by the convolution of their densities.

It can be shown that if X_1 and X_2 are Gaussian, then $Y_1 = X_1 + X_2$ will also be Gaussian. The mean and variance of Y_1 can be obtained using equation (6.4.4). This result cannot be generalized to non-Gaussian densities.

6.5.2.3. Order Statistics

Ordering, comparing, and finding the minimum and maximum are typical statistical data-processing operations. We can use the techniques outlined in the preceding sections for finding the distribution of minimum and maximum values within a group of independent random variables. The distribution of extreme values is useful for deriving techniques for estimating error rates in digital communication systems via simulations.

Let $X_1, X_2, X_3, \dots, X_n$ be a group of independent random variables having a common pdf $f_X(x)$ defined over the interval (a, b) . To find the distribution of the smallest and largest of these X_i , let us define the following transformation:

$$\begin{aligned} \text{Let } Y_1 &= \text{smallest of } (X_1, X_2, \dots, X_n) \\ Y_2 &= \text{next } X_i \text{ in order of magnitude} \\ &\vdots \\ Y_n &= \text{largest of } (X_1, X_2, \dots, X_n) \end{aligned}$$

That is, $Y_1 \leq Y_2 \leq \dots \leq Y_n$ represent $X_1, X_2, \dots, X_n$ when the latter are arranged in ascending order of magnitude. Then Y_i is called the i th-order statistic of the group. It can be shown that the joint pdf of $Y_1, Y_2, \dots, Y_n$ is given by

$$\begin{aligned} f_{Y_1, Y_2, \dots, Y_n}(y_1, y_2, \dots, y_n) &= n! f_X(y_1) f_X(y_2) \dots f_X(y_n) \\ a &< y_1 \leq y_2 \leq \dots \leq y_n < b \end{aligned} \quad (6.5.5a)$$

By integrating the joint pdf given in the preceding equation, we obtain the marginal pdf's of the largest and smallest values as

$$f_{Y_n}(y_n) = n[F_X(y_n)]^{n-1} f_X(y_n), \quad a < y_n < b \quad (6.5.6a)$$

and

$$f_Y(y_1) = n[1 - F_X(y_1)]^{n-1}f_X(y_1), \quad a < y_1 < b \quad (6.5.6b)$$

When n is very large, excellent approximations for the tails of these pdf's are available and these approximations from the basis of several techniques for estimating the error rates in digital communication systems via simulation (see Chapter 11).

6.5.3. Nonlinear Transformations

In some cases of nonlinear transformations such as $Y = g(X) = X^2$, the pdf of Y can be obtained easily using (6.5.1). In other cases, such as $Y = 1 + X^2 + e^X + \sin X$, it is impossible to find the pdf of Y directly using (6.5.1). Two techniques for handling this situation are given below.

6.5.3.1. Moment-Based Techniques

These techniques for finding the pdf of Y are based on finding the moments of Y . Once the moments of Y are found, they can be used in the Chebyshev or Chernoff inequalities (Section 6.6) to obtain bounds on probabilities of Y , or the moments can be used to obtain a series expansion of the pdf of Y and probabilities of Y can be obtained by integrating the series expansion of the pdf of Y (see Ref. 2 for a discussion of the Gram–Charlier expansion of pdf's).

The mean, variance, and the higher moments of Y can be obtained easily (at least conceptually) as follows. We start with

$$E\{h(Y)\} = \int_y h(y)f_Y(y) dy$$

However, $Y = g(X)$, and hence we can compute $E\{h(Y)\}$ as

$$E_Y\{h(Y)\} = E_X\{h(g(X))\}$$

Since the right-hand side is a function of X alone, its expected value is

$$E_X\{h(g(X))\} = \int_{-\infty}^{\infty} h(g(x))f_X(x) dx$$

While the preceding equation seems straightforward, the actual computation of the expected value might be difficult when g is highly nonlinear and the pdf of X is highly complex or not known explicitly. Approximate techniques for computing expected values are discussed in Section 6.6.5.

6.5.3.2. Monte Carlo Simulation Techniques

In Monte Carlo simulation, the distribution of $Y = g(X)$ is obtained by generating samples of the random variable X using the appropriate random number generator [based on $f_X(x)$] and producing samples of Y using the equation $Y = g(X)$. Estimates of the moments of

Y as well as the distribution of Y are obtained from the samples of Y generated. For example, estimates of the mean and variance of Y as well as $P(Y \leq a)$ can be obtained from

$$\begin{aligned}\hat{\mu}_Y &= \frac{1}{n} \sum_{i=1}^n Y_i = \frac{1}{n} \sum_{i=1}^n g(X_i) \\ \hat{\sigma}_Y^2 &= \frac{1}{n-1} \sum_{i=1}^n (Y_i - \hat{\mu}_Y)^2 \\ \hat{P}(Y \leq a) &= \frac{\text{(Number of samples of } Y \leq a)}{n}\end{aligned}$$

where $X_i, i = 1, 2, \dots, n$, are the n samples of X used to generate $Y_i = g(X_i), i = 1, 2, \dots, n$, and the caret notation is used to denote estimators. Properties of the estimators $\hat{\mu}_Y, \hat{\sigma}_Y^2$, and $\hat{P}(Y \leq a)$ and procedures for estimating other parameters as well as sample size n required for obtaining statistically accurate estimators are covered in Chapters 10 and 11.

6.6. Bounds and Approximations

In many applications requiring the calculations of probabilities we often face the following situations:

1. The underlying distributions are not completely specified; only the means, variances, and some of the higher order moments $E\{(X - \mu_X)^k\}, k \geq 2$, are known.
2. The underlying density function is known, but integration in closed form is not possible (example: the Gaussian pdf).
3. The underlying random variable is the weighted sum of a large number of independent random variables.

Several approximation techniques and bounds can be used for handling the situations mentioned above.

6.6.1. Chebyshev's Inequality

If only the mean and variance of a random variable X are known, we can obtain upper bounds on $P(|X| \geq \epsilon)$ using the Chebyshev inequality as^(1,2)

$$P(|X| \geq \epsilon) \leq \frac{1}{\epsilon^2} E[X^2] \quad (6.6.1a)$$

The Chebyshev inequality does not require any assumptions about the underlying distributions and hence is distribution-free. An alternate form of the Chebyshev inequality is

$$P(|Y - \mu_Y| \geq k\sigma_Y) \leq \frac{1}{k^2} \quad (6.6.1b)$$

The Chebyshev bound is a “loose” bound and it might provide an estimate of probability within one or two orders of magnitude as illustrated in an example presented later.

6.6.2. Chernoff Bound

The Chernoff bound, on the other hand, provides a “tighter” bound. To derive the Chernoff bound, define a random variable Y_ϵ as

$$Y_\epsilon = \begin{cases} 1, & X \geq \epsilon \\ 0, & X < \epsilon \end{cases}$$

Then for all $t \geq 0$, it must be true that

$$e^{tX} \geq e^{t\epsilon} Y_\epsilon$$

and hence

$$E\{e^{tX}\} \geq e^{t\epsilon} E\{Y_\epsilon\} = e^{t\epsilon} P[X \geq \epsilon]$$

or

$$P[X \geq \epsilon] = e^{-t\epsilon} E\{e^{tX}\}, \quad t \geq 0$$

Furthermore,

$$\begin{aligned} P[X \geq \epsilon] &\leq \min_{t \geq 0} e^{-t\epsilon} E\{e^{tX}\} \\ &\leq \min_{t \geq 0} \exp[-t\epsilon + \ln E\{e^{tX}\}] \end{aligned} \quad (6.6.2)$$

Equation (6.6.2) is the Chernoff bound. It is shown below in Example 6.6.1 that the Chernoff bound for a Gaussian variable with $\mu_X = 0$ and $\sigma_X^2 = 1$ is given by

$$P(X \geq \epsilon) \leq \exp(-\epsilon^2/2) \quad (6.6.3)$$

While the advantage of the Chernoff bound is that it is tighter than the Chebyshev bound, the disadvantage of the Chernoff bound is that it requires the evaluation of $E\{e^{tX}\}$ and thus requires more extensive knowledge of the distribution. The Chebyshev bound involves only the mean and variance.

6.6.3. Union Bound

This bound is very useful in approximating the probability of union of events, and it follows directly from

$$P(A \cup B) = P(A) + P(B) - P(AB) \leq P(A) + P(B)$$

since $P(AB) \geq 0$. This result can be generalized as

$$P\left(\bigcup_i A_i\right) \leq \sum_i P(A_i) \quad (6.6.4)$$

The union bound is used in the estimation of error probabilities in digital communication systems via simulation.

We now present an example to illustrate the use of these bounds.

■ *Example 6.6.1.* X_1 and X_2 are two independent Gaussian random variables with $\mu_{X_1} = \mu_{X_2} = 0$ and $\sigma_{X_1}^2 = 1$ and $\sigma_{X_2}^2 = 4$.

- Find the Chebyshev and Chernoff bounds on $P(X_1 \geq 3)$ and compare them with the exact value of $P(X_1 \geq 3)$.
- Find the union bound on $P(X_1 \geq 3 \text{ or } X_2 \geq 4)$ and compare it with the actual value.

Solution: (a) The Chebyshev bound on $P(X_1 \geq 3)$ is obtained using (6.6.1) as

$$P[X_1 \geq 3] \leq P(|X_1| \geq 3) \leq \frac{1}{9} \approx 0.111$$

To obtain the Chernoff bound, we start with

$$\begin{aligned} E\{e^{tX_1}\} &= \int_{-\infty}^{\infty} e^{tx_1} \frac{1}{(2\pi)^{1/2}} e^{-x_1^2/2} dx_1 \\ &= e^{t^2/2} \int_{-\infty}^{\infty} \frac{1}{(2\pi)^{1/2}} \exp\left[-\frac{(x_1 - t)^2}{2}\right] dx_1 \\ &= e^{t^2/2} \end{aligned}$$

Hence,

$$P[X_1 \geq \epsilon] \leq \min_{t \geq 0} \exp\left(-t\epsilon + \frac{t^2}{2}\right)$$

The minimum value of the right-hand side occurs with $t = \epsilon$ and hence

$$P[X_1 \geq \epsilon] \leq e^{-\epsilon^2/2}$$

Thus, the Chernoff bound on $P(X_1 \geq 3)$ is given by

$$P[X_1 \geq 3] \leq e^{-9/2} \approx 0.0111$$

From the tabulated values of the $Q(\cdot)$ function (Appendix B), we obtain the value of $P(X_1 \geq 3)$ as

$$P[X_1 \geq 3] = Q(3) = 0.0043$$

Comparison of the exact value with the Chernoff and Chebyshev bounds indicates that the Chebyshev bound is much looser than the Chernoff bound. This is to be expected since the Chebyshev bound does not take into account the functional form of the pdf.

$$\begin{aligned} \text{(b) } P(X_1 \geq 3 \text{ or } X_2 \geq 4) \\ &= P(X_1 \geq 3) + P(X_2 \geq 4) - P(X_1 \geq 3 \text{ and } X_2 \geq 4) \\ &= P(X_1 \geq 3) + P(X_2 \geq 4) - P(X_1 \geq 3)P(X_2 \geq 4) \end{aligned}$$

since X_1 and X_2 are independent. The union bound consists of the sum of the first two terms of the right-hand side of the preceding equation, and the union bound is “off” by the value of

the third term. Substituting the value of these probabilities, we have

$$\begin{aligned} P[X_1 \geq 3 \text{ or } X_2 \geq 4] &\approx (0.0013) + (0.0228) - (0.0013)(0.0228) \\ &\approx 0.02407 \end{aligned}$$

The union bound is given by

$$P[X_1 \geq 3 \text{ or } X_2 \geq 4] \leq P[X_1 \geq 3] + P[X_1 \geq 4] \approx 0.0241$$

The union bound is usually very tight when the probabilities involved are small and the random variables are independent. ■

6.6.4. Central Limit Theorem

If the value of a random variable X is determined by a large number of independent causes, then X tends to have a Gaussian distribution in the limit.⁽³⁾ This effect is stated formally as the central limit theorem:

■ *Central Limit Theorem.* Assume $X_1, X_2, \dots, X_n$ is a sequence of independent, identically distributed random variables with mean μ and variance σ^2 . Let

$$Z_n = \frac{1}{n} \sum_{i=1}^n \frac{X_i - \mu}{\sigma}, \quad n = 1, 2, 3, \dots$$

Then, for all z ,

$$\lim_{n \rightarrow \infty} P[Z_n \leq z] = \int_{-\infty}^z \frac{1}{(2\pi)^{1/2}} e^{-y^2/2} dy$$

(i.e., Z_n converges, in distribution, to a standard Gaussian distribution with zero mean and unit variance). The Gaussian approximation can be used for $n > 20$ or so. ■

There are several variations of the central limit theorem including the following:

- (i) The random variables $X_1, X_2, \dots, X_n, \dots$ in the original sequence are independent with the same mean and variance, but *not* identically distributed.
- (ii) $X_1, X_2, \dots$ are independent and have different finite variances $\sigma_1^2, \sigma_2^2, \dots$, $\epsilon < \sigma_i^2 < M$; and the variances are neither too small nor too large.

During the simulation of communication systems, we encounter $\{Z_n\}$ of the form

$$Z_n = \frac{1}{m} \sum_{k=n-m+1}^n X_k h_{n-k}$$

where $\{X_k\}$ is often a sequence of correlated random variables with arbitrary, non-Gaussian distributions. Whether we can approximate Z_n by a Gaussian distribution can be verified if we

can compute the “equivalent” number of independent samples of X_k that make up Z_n . This number is defined as

$$m^* = \frac{(1/m) \sum_{k=n-m+1}^n \text{var}[X_k]}{\text{var}[(1/m) \sum_{k=n-m+1}^n X_k h_{n-k}]}$$

If $m^* \gg 1$, say $m^* > 20$, then Z_n can be approximated by Gaussian. (Of course, Z_n will always have a Gaussian pdf if the X_k are Gaussian.)

6.6.5. Approximate Computation of Expected Values

In this section we describe two techniques for computing $E\{g(X)\}$ when $g(\cdot)$ is an explicitly known deterministic function, and X is a random variable whose distribution is highly complex or is not known explicitly. We will assume that the range of X is $[a, b]$, where a and b are finite, and that the moments of X (at least the first N moments) are known.

6.6.5.1. Series Expansion Technique

We can expand $g(x)$ in a Taylor series in the neighborhood of x_0 (usually the mean or mode) as

$$g(x) = g(x_0) + \sum_{k=1}^{\infty} \frac{(x - x_0)^k}{k!} g^{(k)}(x_0)$$

where $g^{(k)}(x_0)$ is the k th derivative of $g(\cdot)$. If we define

$$C_k = E\{(X - x_0)^k\} \quad (6.6.5a)$$

then we can express $E\{g(X)\}$ as

$$E\{g(X)\} = \sum_{k=0}^{\infty} \frac{g^{(k)}(x_0)}{k!} C_k, \quad C_0 = 1 \quad (6.6.5b)$$

An approximate value of $E\{g(X)\}$ can be obtained by truncating the series to N terms:

$$E\{g(X)\} \approx E\{g_N(X)\} = \sum_{k=0}^N \frac{g^{(k)}(x_0)}{k!} C_k \quad (6.6.5c)$$

The error of this approximation^(8,10) is

$$\begin{aligned} & E\{g(X)\} - E\{g_N(X)\} \\ &= \frac{1}{(N+1)!} E\{(X - x_0)^{N+1} g^{(N+1)}(x_0 + \theta(x - x_0))\} \end{aligned}$$

where $0 \leq \theta \leq 1$.

In many applications involving the estimation of error probabilities in digital communication systems, we encounter $E\{g(X)\}$, where $g(\cdot)$ is the tail probability of a Gaussian pdf, that is,

$$g(x) = \operatorname{erfc}\left(\frac{x+A}{\sqrt{2}\sigma}\right) = 2Q\left(\frac{x+A}{\sigma}\right)$$

where A and σ are the mean and standard deviation, respectively, of the Gaussian pdf. This function can be expanded in a Taylor series in the neighborhood of A by observing that

$$\frac{d^k}{dz^k} \operatorname{erfc}(z) = (-1)^k \frac{2}{\sqrt{\pi}} H_{k-1}(z) e^{-z^2}$$

where $H_{k-1}(\cdot)$ is the Hermite polynomial of degree $(k-1)$ given in Appendix C and Ref. 8. Thus, we have

$$g^{(k)}(0) = (-1)^k \frac{2}{\sqrt{\pi}(\sqrt{2}\sigma)^k} H_{k-1}\left(\frac{A}{\sqrt{2}\sigma}\right) \exp\left(-\frac{A^2}{2\sigma^2}\right)$$

and

$$\begin{aligned} E[g(X)] &= \operatorname{erfc}\left(\frac{A}{\sqrt{2}\sigma}\right) + \frac{2}{\sqrt{\pi}} \exp\left(-\frac{A^2}{2\sigma^2}\right) \\ &\quad \times \sum_{k=1}^{\infty} \frac{(-1)^k \mu_k}{(\sqrt{2}\sigma)^k k!} H_{k-1}\left(\frac{A}{\sqrt{2}\sigma}\right) \end{aligned} \quad (6.6.6)$$

where

$$\mu_k = E\{X^k\}, \quad k = 1, 2, \dots$$

are the central moments of the random variable X .

The proof that the series given in (6.6.6) is convergent as well as an upper bound on the error can be found using bounds on the values of Hermite polynomials given in Ref. 8. Note that the moments can be estimated via simulation. There are different ways to use simulation-derived moments as part of performance estimation procedures; one approach is described in Chapter 11 and another in Chapter 12.

6.6.5.2. Moments of Finite Sums of Random Variables

In simulating digital communication systems we often encounter random variables which are weighted sums of other random variables. (Intersymbol interference in digital communication systems is a typical example.) A recursive procedure for computing the moments of weighted sums of random variables can be derived as follows. Let

$$Z = \sum_{i=1}^N h_i X_i = \sum_{i=1}^N Y_i \quad (6.6.7)$$

where $\{h_i\}$ is a sequence of known constants and $\{X_i\}$ is a sequence of independent random variables whose moments are given. Assume that

$$\max |Y_i| \geq \max |Y_{i+1}|, \quad 1 \leq i \leq N-1 \quad (6.6.8)$$

The reordering is done so that the moments of the dominant terms are computed before those of others and the roundoff error is minimized.

Let Z_k be the partial sum

$$Z_k = \sum_{i=1}^k Y_i, \quad 1 \leq k \leq N \quad (6.6.9)$$

and

$$\theta^n(k) = E\{Z_k^n\}$$

with $\theta^0(k) = 1$, $1 \leq k \leq N$. From the definition of $\theta^n(k)$, we can obtain

$$\theta^n(k) = E\{(Z_{k-1} + Y_k)^n\}, \quad k > 1 \quad (6.6.10)$$

Using the binomial expansion and interchanging the operations of addition and expectation, we can express $\theta^n(k)$ as

$$\theta^n(k) = \sum_{p=0}^n \binom{n}{p} \theta^p(k-1) \alpha^{n-p}(k), \quad k > 1 \quad (6.6.11a)$$

where

$$\alpha^{n-p}(k) = E\{Y_k^{n-p}\}, \quad \alpha^0(k) = 1, \quad k \geq 1 \quad (6.6.11b)$$

Note that $\alpha^{n-p}(k)$ are the moments of $Y_k = h_k X_k$ which are assumed to be given. Thus, (6.6.11) gives the moments of the sum of k random variables recursively in terms of the moments of the sum of the first $(k-1)$ random variables and the moments of the k th term. Moments given in (6.6.11) can be used in (6.6.6) or quadrature formulas for computing performance measures such as error probabilities in digital communication systems.

6.6.5.3. Quadrature Approximations

These approximations can be used to compute $E\{g(X)\}$ when the pdf of X is not known explicitly or when closed-form integration is not possible. The Gaussian quadrature procedure is based on approximating X by a discrete random variable X_D with

$$P[X_D = x_i] = w_i, \quad i = 1, 2, \dots, N$$

Then,

$$E\{g(X)\} = \int_a^b g(x) f_X(x) dx \approx \sum_{i=1}^N w_i g(x_i)$$

The set of abscissas x_i and the weights w_i are usually referred to as the quadrature rule.^(8,10,24) The x_i and w_i are obtained by requiring the discrete approximation X_D to yield the same moments as X . Of course, this requires that the first $2N$ moments be known. For a number of well-known pdf's, the abscissas and weights have been computed and are given in Ref. 8. For example, if X is a Gaussian random variable with mean μ_X and variance σ_X^2 , the quadrature rule is

$$E\{g(X)\} \approx \frac{1}{\sqrt{\pi}} \sum_{i=1}^N w_i g(\sqrt{2}x'_i \sigma_X + \mu_X) \quad (6.6.12)$$

where (x'_i, w'_i) , $i = 1, 2, \dots, N$, are given in Appendix D and Ref. 8. The error of the approximation is given by

$$\varepsilon_N = \frac{N! \sigma^{2N}}{(2N)!} g^{(2N)}(\eta), \quad -\infty < \eta < \infty$$

■ *Example 6.6.2.* The probability of error in a binary communication system is given by

$$P_e = E\left\{Q\left(\frac{1+X}{\sigma}\right)\right\}, \quad Q(\alpha) = \frac{1}{2} \operatorname{erfc}\left(\frac{\alpha}{\sqrt{2}}\right)$$

where 1 is the signal amplitude, X is the intersymbol interference, Q is the tail probability under $N(0, 1)$, and $\sigma^2 = 1/4$ is the variance of the noise. The intersymbol interference (ISI) has the form

$$X = 0.2A_{i-1} + 0.1A_{i-2}$$

where A_{i-1} and A_{i-2} are independent binary variables with $P[A_k = \pm 1] = 1/2$.

- Find the exact value of P_e .
- Find an approximate value for P_e using a fourth-order Taylor series expansion for $Q(\alpha)$.

Solution:

- X has four possible values, 0.3, 0.1, -0.1, and -0.3, with equal probability of 1/4.

Hence, we have

$$\begin{aligned} P_e &= \frac{1}{4} \{Q(2(1+0.3)) + Q(2(1+0.1)) + Q(2(1-0.3)) + Q(2(1-0.1))\} \\ &= \frac{1}{4} \{Q(2.6) + Q(2.2) + Q(1.4) + Q(1.8)\} \approx 0.033813 \end{aligned}$$

- From (6.6.6),

$$\begin{aligned} P_e &= E\left\{Q\left(\frac{1+X}{\sigma}\right)\right\} = E\{Q(2(1+X))\} \\ &\approx Q(2) + \frac{e^{-2}}{\sqrt{\pi}} \left\{ \mu_2 H_1(\sqrt{2}) + \frac{1}{6} \mu_4 H_3(\sqrt{2}) \right\} \end{aligned}$$

where $\mu_k = E\{X^k\}$, and H_i is the Hermite polynomial. Note that the odd moments of X are zero and hence the odd terms in the series expansion are zero. The even moments of the ISI

can be computed as

$$E\{X^2\} = \frac{1}{4}[0.3^2 + 0.1^2 + (-0.3)^2 + (-0.1)^2] = 0.05$$

$$E\{X^4\} = \frac{1}{4}[0.3^4 + 0.1^4 + (-0.3)^4 + (-0.1)^4] = 0.004$$

and the Hermite polynomials have the values (Appendix C)

$$H_1(\sqrt{2}) = 2\sqrt{2}$$

$$H_3(\sqrt{2}) = 8(\sqrt{2})^3 - 12(\sqrt{2})$$

Hence

$$P_e \approx Q(2) + \frac{e^{-2}}{\sqrt{\pi}}(0.1414 + 0.0038)$$

$$\approx 0.0228 + 0.01103 = 0.03383$$

This example illustrates that the moment method yields a good approximation. ■

■ **Example 6.6.3.** The random variable X has a Gaussian distribution with a mean of 2 and variance of 9. Using the quadrature formula, find

$$E\{X^4\} \approx \sum_{i=1}^5 w_i x_i^4$$

Solution: From Ref. 8, we have

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^5 w_i f(x'_i)$$

where (x'_i, w'_i) are

x'_i	w'_i
± 2.02018	0.001995
± 0.95857	0.039361
0.00000	0.094530

Using these abscissas and weights, we can compute $E\{X^4\}$ as

$$E\{g(X)\} \approx \frac{1}{\sqrt{\pi}} \sum_{i=1}^5 w'_i (\sqrt{2}x'_i\sigma + \mu)^4$$

Substituting w'_i and x'_i , $i = 1, \dots, 5$, we obtain

$$E\{X^4\} \approx 475$$

The reader can verify that the true value of $E\{X^4\}$ is 475, showing that the quadrature rule is an excellent approximation. ■

6.7. Random Processes

6.7.1. Basic Definitions and Notations

Random variables map the outcome of one or more random experiments to points in R_n . In a similar vein, a random process may be viewed as a mapping of the outcomes of a random experiment to functions of time. A simple example is shown in Figure 6.6. In this example the random experiment consists in tossing a coin twice and the random process maps the four outcomes to four functions of time.

A random process which is a collection or ensemble of waveforms can be denoted by $X(t, \Lambda)$, $t \in \Gamma$ and $\Lambda \in S$, where t represents time, Λ is a variable that represents an outcome in the sample space S of some underlying random experiment E , and Γ is the set of values of t for which the random process is defined. Associated with each specific outcome, say λ_i , we have a specific member function $x_i(t)$ of the ensemble. Each member function, also referred to as a sample function or a realization of the process, is a deterministic function of time even though we may not always be able to express it in a closed form.

For a specific value of time $t = t_0$, $X(t_0, \Lambda)$ represents a collection of numerical values of the various member functions at $t = t_0$ (see Figure 6.6). The actual value depends on the outcome of the random experiment and the member function associated with that outcome. Hence, $X(t_0, \Lambda)$ is a random variable and the probability distribution of the random variable $X(t_0, \Lambda)$ is derived from the probabilities of the various outcomes of the random experiment E .

When t and Λ are fixed at, say, $t = t_0$ and $\Lambda = \lambda_i$, then $X(t_0, \lambda_i)$ represents a single numerical value of the i th member function of the process at $t = t_0$. That is, $X(t_0, \lambda_i) = x_i(t_0)$. Thus, $X(t, \Lambda)$ can denote the following quantities:

1. $X(t, \Lambda) = \{X(t, \lambda_i) | \lambda_i \in S\} = \{x_1(t), x_2(t), \dots\}$, a collection of functions of time.
2. $X(t, \lambda_i) = x_i(t)$, a specific member function or deterministic function of time.
3. $X(t_0, \Lambda) = \{X(t_0, \lambda_i) | \lambda_i \in S\} = \{x_1(t_0), x_2(t_0), \dots\}$, a collection of the numerical values of the member functions at $t = t_0$, that is, a random variable.
4. $X(t_0, \lambda_i) = x_i(t_0)$, the numerical value of the i th member function at $t = t_0$.

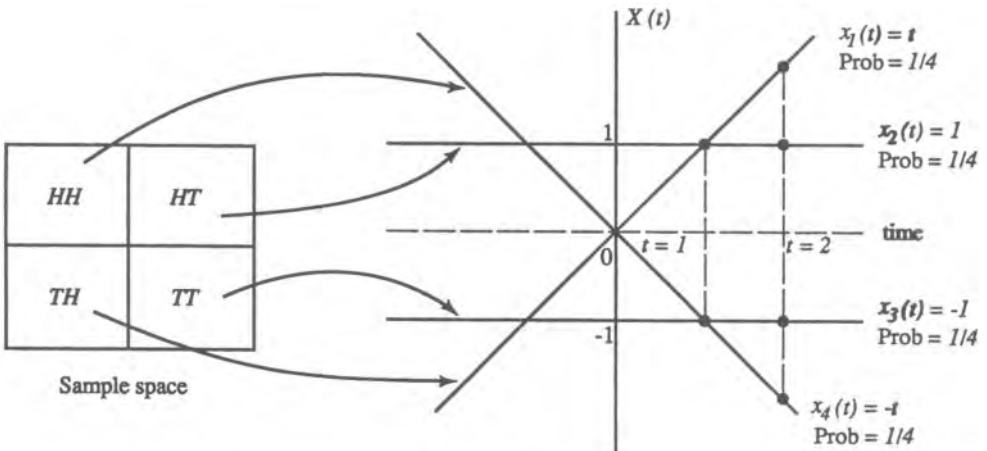


Figure 6.6. Example of a random process.

While the notation given in the preceding paragraphs is well defined, convention adds an element of confusion for the sake of conformity with the notation for deterministic signals by using $X(t)$ rather than $\mathbf{X}(t, \mathbf{\Lambda})$ to denote a random process. $X(t)$ may represent a family of time functions, a single time function, a random variable, or a single number. Fortunately, the specific interpretation of $X(t)$ usually can be understood from the context.

The probabilistic structure of a random process comes from the underlying random experiment E . Knowing the probability of each outcome of E and the time function it maps to, we can derive probability distribution functions for $P[X(t_1) < a_1]$, $P[X(t_1) \leq a_1]$ and $X(t_2) \leq a_2]$, and so on. If A_1 is a subset of the sample space S of E and it contains all the outcomes λ for which $X(t_1) \leq a_1$, then

$$P[X(t_1) \leq a_1] = P(A_1)$$

Note that A_1 is an event associated with E and its probability is derived from the probability structure of the random experiment E . In a similar fashion, we can define joint and conditional probabilities also by first identifying the event that is the inverse image of a given set of values of $X(t)$ and then calculating the probability of this event.

■ **Example 6.7.1.** For the random process shown in Figure 6.6, find (a) $P[X(1) = 1]$; (b) $P[X(1) = 1 \text{ and } X(2) = 2]$; (c) $P[X(2) = 2|X(1) = 1]$; (d) $E\{X(1)X(2)\}$.

Solution:

- $P[X(1) = 1] = P\{\text{set of all outcomes that map to } X(1) = 1\}$. From Figure 6.6, we see that this set consists of $\{TH, HT\}$ and hence $P[X(1) = 1] = P[TH, HT] = 1/2$.
- Similarly, $P[X(1) = 1 \text{ and } X(2) = 2] = P(TH) = 1/4$.
- $P[X(2) = 2|X(1) = 1] = \frac{P[X(1) = 1 \text{ and } X(2) = 2]}{P[X(1) = 1]} = \frac{1}{4} / \frac{1}{2} = \frac{1}{2}$

$$\begin{aligned} \text{d. } E\{X(1)X(2)\} &= \sum_i x_i(1)x_i(2)P[x_i(t)] \\ &= (1)(1)\frac{1}{4} + (1)(2)\frac{1}{4} + (-1)(-1)\frac{1}{4} + (-1)(-2)\frac{1}{4} \\ &= 6/4 \end{aligned} \quad \blacksquare$$

Random processes are classified according to the characteristics of the index set for time, Γ , and the random variable $X(t)$ at time t . If Γ contains a continuum of values in one or more intervals on the real line $\mathbf{R}_1$, then $X(t)$ is called a continuous-time random process, examples of which are shown in Figures 6.1 and 6.2. If Γ consists of a finite or countably infinite number of values, say $\{\dots, t_{-2}, t_{-1}, t_0, t_1, t_2, \dots\}$, then $X(t)$ is called a discrete-time random process or a random sequence, an example of which is the ensemble of random binary digits shown in Figure 6.1. We often denote a random sequence by $\{X(n)\}$, where n represents t_n .

$X(t)$ [or $X(n)$] is a discrete-state or discrete-valued process (or sequence) if its values are countable. Otherwise, it is a continuous-state or continuous-valued random process (or sequence). The ensemble of binary waveforms $X(t)$ shown in Figure 6.1 is a discrete-state, continuous-time random process.

A random process may be either real-valued or complex-valued. In many applications in communication systems, we deal with real-valued bandpass random processes of form.

$$Z(t) = A(t) \cos[2\pi f_c t + \theta(t)]$$

where f_c is the carrier or center frequency, and $A(t)$ and $\theta(t)$ are real-valued random processes. $Z(t)$ can be written as

$$\begin{aligned} Z(t) &= \text{Real part of } \{A(t) \exp[j\theta(t)] \exp(j2\pi f_c t)\} \\ &= \text{Real part of } \{\tilde{W}(t) \exp(j2\pi f_c t)\} \end{aligned}$$

where the complex envelope $\tilde{W}(t)$ is given by

$$\begin{aligned} \tilde{W}(t) &= A(t) \cos \theta(t) - jA(t) \sin \theta(t) \\ &= X(t) + jY(t) \end{aligned}$$

$\tilde{W}(t)$ is a complex-valued random process, whereas $X(t)$, $Y(t)$, and $Z(t)$ are real-valued random processes.

6.7.2. Methods of Description

A random process can be described in terms of a random experiment and the associated mapping. While such a description is a natural extension of the concept of random variables, there are alternate methods of characterizing random processes that are of use in simulation.

6.7.2.1. Joint Distribution

Since we defined a random process as a time indexed set of random variables, we can obviously use joint probability distribution functions to describe a random process. For a random process $X(t)$, we have many joint distribution functions. This leads to a formidable description of the process because at least one n -variate distribution function is required for each value of $t_1, t_2, \dots, t_n$. However, only the first-order distribution function(s) $P[X(t_1) \leq a_1]$ and the second-order distribution function(s) $P[X(t_1) \leq a_1, X(t_2) \leq a_2]$ are primarily used. The first-order distribution function describes the instantaneous amplitude distribution of the process and the second-order distribution function tells us something about the structure of the signal in the time domain and thus the spectral content of the signal. The higher order distribution functions describe the process in much finer detail and are not of interest in most simulation applications.

While the joint distribution functions of a process can be derived from a description of the random experiment and the mapping, there is no unique way to map joint distributions to member functions. Two different processes may have the same n th-order distribution, but the member functions need not have a one-to-one correspondence.

6.7.2.2. Analytical Description Using Random Variables

We are used to expressing deterministic signals in simple analytical forms such as $x(t) = 20 \sin(10t)$ or $y(t) = \exp(-t^2)$. It is sometimes possible to express a random process in an analytical form using one or more random variables. Consider, for example, an FM station that is broadcasting a “tone,” $x(t) = 100 \cos(10^8 t)$, to a large number of receivers distributed randomly in a metropolitan area (see Figure 6.7). The amplitude and phase of the waveform received by the i th receiver will depend on the distance between the transmitter and the receiver. If we assume a large number of receivers distributed randomly over an area, we can

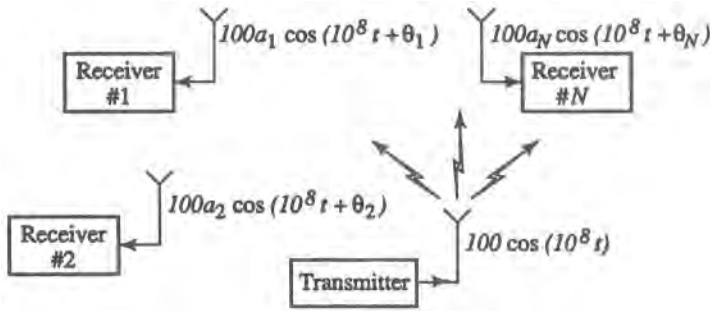


Figure 6.7. Modeling of a broadcasting system $Y(t) = A \cos(10^8 t + \theta)$.

model the distance as a continuous random variable. Since the attenuation and the phase are functions of distance, they are also random variables, and we can represent the ensemble of received waveforms by a random process $Y(t)$ of the form

$$Y(t) = A \cos(10^8 t + \theta)$$

where A and θ are random variables representing the amplitude and phase of the received waveforms. It might be reasonable to assume independent distributions for A and θ .

Representation of a random process in terms of one or more random variables whose probability distributions are known is used in a variety of applications in communication systems.

6.7.2.3. Average Values

As in the case of random variables, random processes can be described in terms of averages or expected values. In many applications, only certain averages derived from the first- and second-order distributions of $X(t)$ are of interest. For real- or complex-valued random processes, these averages are defined as follows:

Mean: The mean of $X(t)$ is the expected value of the random variable $X(t)$,

$$\mu_X(t) \triangleq E\{X(t)\} \quad (6.7.1)$$

Autocorrelation: The autocorrelation of $X(t)$, denoted by $R_{XX}(t_1, t_2)$, is the expected value of the product $X^*(t_1)X(t_2)$,

$$R_{XX}(t_1, t_2) \triangleq E\{X^*(t_1)X(t_2)\} \quad (6.7.2a)$$

where $*$ denotes conjugate.

Autocovariance: The autocovariance of $X(t)$ is defined as

$$C_{XX}(t_1, t_2) \triangleq R_{XX}(t_1, t_2) - \mu_X^*(t_1)\mu_X(t_2) \quad (6.7.2b)$$

Correlation Coefficient: The correlation coefficient of $X(t)$ is defined as

$$r_{XX}(t_1, t_2) \triangleq \frac{C_{XX}(t_1, t_2)}{[C_{XX}(t_1, t_1)C_{XX}(t_2, t_2)]^{1/2}} \quad (6.7.3)$$

The mean of the random process is the “ensemble” average of the values of all the member functions at time t , and the autocovariance function $C_{XX}(t_1, t_1)$ is the variance of the random variable $X(t_1)$. For $t_1 \neq t_2$, the second moments $R_{XX}(t_1, t_2)$, $C_{XX}(t_1, t_2)$, and $r_{XX}(t_1, t_2)$ partially describe the time-domain structure of the random process. We will see later that we can use these functions to derive the spectral properties of $X(t)$.

For random sequences the argument n is substituted for t , and n_1 and n_2 are substituted for t_1 and t_2 , respectively. In this case the four functions defined above are also discrete time functions.

6.7.2.4. Two or More Random Processes

When we deal with two or more random processes, we can use joint distribution functions, analytical descriptions, or averages to describe the relationship between the random processes. Consider two random processes $X(t)$ and $Y(t)$ whose joint distribution function is denoted by

$$P[X(t_1) \leq x_1, \dots, X(t_n) \leq x_n; Y(t'_1) \leq y_1, \dots, Y(t'_m) \leq y_m]$$

Three averages or expected values that are used to describe the relationship between $X(t)$ and $Y(t)$ are as follows:

Cross-Correlation Function:

$$R_{XY}(t_1, t_2) \triangleq E\{X^*(t_1)Y(t_2)\} \quad (6.7.4a)$$

Cross-Covariance Function:

$$C_{XY}(t_1, t_2) \triangleq R_{XY}(t_1, t_2) - \mu_X^*(t_1)\mu_Y(t_2) \quad (6.7.4b)$$

Correlation Coefficient:

$$r_{XY}(t_1, t_2) \triangleq \frac{C_{XY}(t_1, t_2)}{[C_{XX}(t_1, t_1)C_{YY}(t_2, t_2)]^{1/2}} \quad (6.7.4c)$$

Using the joint and marginal distribution functions as well as the expected values, we can determine the degree of dependence between two random processes.

The definitions given above can be used for random sequences with n replacing t and n_1 and n_2 replacing the arguments t_1 and t_2 .

Equality: Equality of two random processes will mean that their respective member functions are identical for each outcome $\lambda \in \mathcal{S}$. Note that equality also implies that the processes are defined on the same random experiment.

Uncorrelated: Two processes $X(t)$ and $Y(t)$ are uncorrelated when

$$C_{XY}(t_1, t_2) = 0, \quad t_1, t_2 \in \Gamma \quad (6.7.5)$$

Orthogonal: $X(t)$ and $Y(t)$ are said to be orthogonal if

$$R_{XY}(t_1, t_2) = 0, \quad t_1, t_2 \in \Gamma \quad (6.7.6)$$

Independent: Random processes $X(t)$ and $Y(t)$ are independent if

$$\begin{aligned} &P[X(t_1) \leq x_1, \dots, X(t_n) \leq x_n; Y(t'_1) \leq y_1, \dots, Y(t'_m) \leq y_m] \\ &= P[X(t_1) \leq x_1, \dots, X(t_n) \leq x_n]P[Y(t'_1) \leq y_1, \dots, Y(t'_m) \leq y_m] \\ &\quad \text{for all } n, m \text{ and } t_1, t_2, \dots, t_n, t'_1, t'_2, \dots, t'_m \in \Gamma \end{aligned} \quad (6.7.7)$$

As in the case of random variables, “independent” implies “uncorrelated,” but not conversely except in some special cases.

In communication systems it is customary to assume that signals from different users are independent; also, signals and noise are usually assumed to be independent processes.

6.7.3. Stationarity, Time Averaging, and Ergodicity

Both stationary and nonstationary models are used for simulation and analysis of communication systems. Stationarity or time invariance of a system implies that if the system produces a response $y(t)$ to an input $x(t)$, then the response to $x(t - t_0)$ is $y(t - t_0)$, where t_0 is an arbitrary translation of the time at which the input is applied to the system.

A random process $X(t)$, on the other hand, is said to be stationary in the strict sense if its statistical properties are not affected by a shift in the time origin. This means that $X(t)$ and $X(t + \epsilon)$, where ϵ is an arbitrary time shift, have the same statistical properties. Another idea of stationarity is that a time translation of a sample function results in another sample function of the random process having the same probability structure.

In the case when we are describing a random process by its mean and autocorrelation function, it seems reasonable to call the random process stationary if the mean and autocorrelation function do not vary with a shift in the time origin. We call a process *wide-sense stationary* (WSS) if

$$\begin{aligned} E\{X(t)\} &= \text{const}, \quad t \in (-\infty, \infty) \\ R_{XX}(t_1, t_2) &= R_{XX}(t_1 + t, t_2 + t), \quad t, t_1, t_2 \in (-\infty, \infty) \end{aligned} \quad (6.7.8)$$

Strict-sense Stationarity implies wide sense Stationarity, but not vice versa.

The auto correlation function of a stationary process is a function of the time difference, and hence it is often defined as

$$R_{XX}(\tau) \triangleq E\{X^*(t)X(t + \tau)\} \quad (6.7.9)$$

Whether a process is stationary or not can be verified easily if an analytical description of the process is available. Lacking such a description, one has to collect data and analyze them to

check whether the process is stationary. For details of procedures for collecting, analyzing, and testing data for stationarity the reader is directed to Ref. 2.

In modeling and simulation, we use stationary models most of the time. However, there are some occasions when we need nonstationary models, for example, to model the changes in the characteristics of a component due to aging.

6.7.3.1. Time Averages

In simulation-based analysis, we generate samples of the member functions of the input random processes, process them to obtain sampled values of the member functions of the output random processes, and then estimate various parameters such as power levels, error rates, etc. In many applications of simulations, we typically use only one member function of a process to estimate parameter values by averaging over time (i.e., we do the estimation from one simulation run).

Suppose we want to estimate the mean μ_X of a stationary random process $X(t)$. The mean is an ensemble average defined as

$$\mu_X = \int_{-\infty}^{\infty} x f_{X(t)}(x) dx$$

If we observe values of $X(t)$ over, say, m member functions $x_1(t), \dots, x_m(t)$, then $\mu_X(t)$ can be estimated as

$$\hat{\mu}_X = \frac{1}{m} \sum_{i=1}^m x_i(t)$$

On the other hand, if we have access to only one member function, say $x(t)$, then we can obtain an estimate of the mean by *time averaging* as

$$\langle \mu_X \rangle_T = \frac{1}{T} \int_{-T/2}^{T/2} x(t) dt \quad (6.7.10a)$$

or

$$\langle \mu_X \rangle_n = \frac{1}{n} \sum_{i=1}^n x(n) \quad (6.7.10b)$$

where $x(t_i)$, $i = 1, 2, \dots, n$, are the sampled values of $x(t)$.

Whereas the time average $\langle \mu_X \rangle_T$ or $\langle \mu_X \rangle_n$ is constant for a particular member function $x(t)$, the set of values taken over all the members function of $X(t)$ is a random variable (we will denote this random variable as $\langle \mu_X \rangle_T$ or $\langle \mu_X \rangle_n$). That is, $\langle \mu_X \rangle_T$ with the subscript X is a random variable and $\langle \mu_x \rangle_T$ with the subscript x is a particular value.

The quality of the time-averaged estimator will depend on its mean and variance. If

$$\lim_{T \rightarrow \infty} E\{\langle \mu_X \rangle_T\} = \mu_X \quad \text{or} \quad \lim_{n \rightarrow \infty} E\{\langle \mu_X \rangle_n\} = \mu_X$$

and

$$\lim_{T \rightarrow \infty} \text{Var}\{\langle \mu_X \rangle_T\} = 0 \quad \text{or} \quad \lim_{n \rightarrow \infty} \text{Var}\{\langle \mu_X \rangle_n\} = 0$$

then we conclude that the time-averaged estimate converges to the true value (in the mean square sense). In general, ensemble averages and time averages are not equal except for a very special class of random process called ergodic processes. The concept of ergodicity deals with the equality of time averages and ensemble averages.

6.7.3.2. Ergodicity

A stationary random process $X(t)$ is called ergodic if the time averages converge to the corresponding ensemble averages (i.e., expected values). This definition implies that any ensemble average of $X(t)$ can be determined from a single member function of $X(t)$. Since we might be interested in only a few ensemble averages, ergodicity can be defined with respect to these averages. Ergodicity is a stronger condition than stationarity and not all stationary processes are ergodic. Conditions for ergodicity of random processes are derived in Refs. 1 and 2. Except for certain simple cases, it is very difficult to establish whether a random process meets the conditions for the ergodicity of a particular parameter. In practice, we are usually forced to consider the physical origin of the random process to make an intuitive judgment on ergodicity.

For a process to be ergodic, each member function should “look” random even though we view each member function to be an ordinary time signal. For example, if we inspect the member functions of the random process $Y(t)$ shown in Figure 6.8b, randomness is evident in each member function, and it is reasonable to expect the random process to be at least “weakly” ergodic. On the other hand, each member function of $X(t)$ in Figure 6.8a is a constant, and by observing one member function, we learn nothing about the other member functions of the process. Hence, for this process, time averages will tell us nothing about ensemble averages. Thus, intuitive justification of ergodicity boils down to deciding whether a single member function “represents” a random signal whose variations along the time axis can be assumed to represent typical variations over the ensemble.

The comments given in the previous paragraphs may seem somewhat circular, and the reader may feel that the concept of ergodicity is on shaky ground. However, we would like to point out that in many practical situations we are forced to use models that might be hard to justify under rigorous examination.

Fortunately, for Gaussian random processes (described in Section 6.8.5), which are extensively used for modeling signals and noise in a variety of applications, the test for ergodicity is very simple.⁽²⁾ A stationary, zero-mean Gaussian random process is ergodic in the general (strict) sense if

$$\int_{-\infty}^{\infty} |R_{XX}(\tau)| d\tau < \infty$$

Note that we need not know $R_{XX}(\tau)$ exactly to verify this condition. For example, if we can assume that $R_{XX}(\tau)$ decays exponentially as $|\tau| \rightarrow \infty$, then the condition for ergodicity is satisfied.

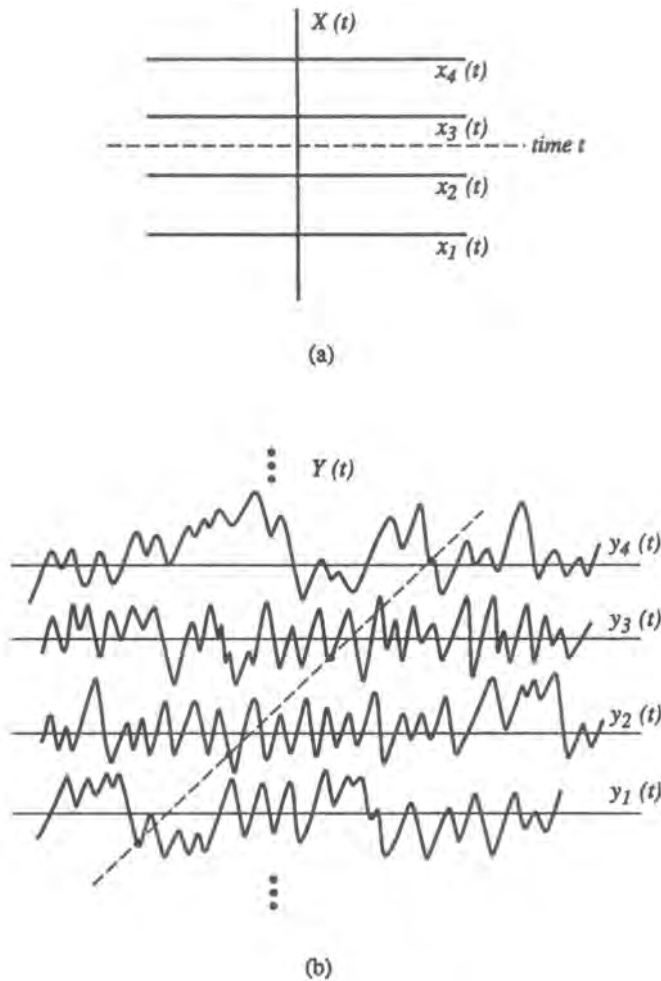


Figure 6.8. Examples of stationary and ergodic processes. (a) $X(t)$ is stationary, but not ergodic; (b) $Y(t)$ is stationary and ergodic.

It is common practice to assume ergodicity and simulate one sample function of all the random processes.

6.7.4. Correlation and Power Spectral Density Function of Stationary Random Processes

Frequency-domain descriptions of deterministic signals are obtained via their Fourier transforms, and this technique plays an important role in the characterization of random waveforms. However, direct transformation is not applicable for random waveforms since a transform of each member function of the ensemble may not exist. Thus, the spectral analysis of random signals differs from that of deterministic signals.

For stationary random processes, the autocorrelation function $R_{XX}(\tau)$ describes how rapidly the random signal changes as a function of time. If the autocorrelation function decays

rapidly to zero, it indicates that the process can be expected to change rapidly as a function of time. Also, if the autocorrelation function has periodic components, then the underlying process will have periodic components. Thus, we can conclude that the autocorrelation function contains information about the expected frequency content of the random process.

The relationship between the autocorrelation function and the frequency content of a stationary random process is the main topic of this section. Throughout this section we will assume the process to be real-valued. The concepts presented below can be easily extended to complex-valued random processes.

6.7.4.1. Autocorrelation Function and Its Properties

The autocorrelation function of a real-valued stationary random process defined by

$$R_{XX}(\tau) = E\{X(t)X(t + \tau)\}$$

has the following properties^(1,2):

1. If we assume $X(t)$ to be voltage waveform across a $1\text{-}\Omega$ resistor, then the ensemble average of $X^2(t)$ is the average power delivered to the $1\text{-}\Omega$ load, i.e.,

$$\begin{aligned} E\{X^2(t)\} &= \text{Average power} \\ &= R_{XX}(0) \geq 0 \end{aligned} \quad (6.7.11a)$$

2. $R_{XX}(\tau)$ is an even function,

$$R_{XX}(\tau) = R_{XX}(-\tau) \quad (6.7.11b)$$

3. The following condition holds:

$$|R_{XX}(\tau)| \leq R_{XX}(0) \quad (6.7.11c)$$

4. If $X(t)$ has a periodic component, then $R_{XX}(\tau)$ will also have a periodic component. Additional properties of $R_{XX}(\tau)$ and proofs are given in Refs. 1 and 2.

6.7.4.2. Cross-Correlation Function and Its Properties

The cross-correlation function of two real-valued random processes $X(t)$ and $Y(t)$ that are jointly WSS will be independent of t , and we can write it as

$$R_{XY}(\tau) = E\{X(t)Y(t + \tau)\}$$

The cross-correlation function has the following properties^(1,2):

$$1. \quad R_{XY}(\tau) = R_{YX}(-\tau) \quad (6.7.12a)$$

$$2. \quad |R_{XY}(\tau)| \leq [R_{XX}(0)R_{YY}(0)]^{1/2} \quad (6.7.12b)$$

$$3. \quad |R_{XY}(\tau)| \leq \frac{1}{2}[R_{XX}(0) + R_{YY}(0)] \quad (6.7.12c)$$

$$4. \quad \begin{aligned} R_{XY}(\tau) &= 0 && \text{if the processes are orthogonal} \\ R_{XY}(\tau) &= \mu_X \mu_Y && \text{if the processes are independent} \end{aligned}$$

6.7.4.3. Power Spectral Density

Suppose we define the function $S_{XX}(f)$ as the Fourier transform of $R_{XX}(\tau)$:

$$S_{XX}(f) = \int_{-\infty}^{\infty} R_{XX}(\tau) \exp(-j2\pi f\tau) d\tau \quad (6.7.13a)$$

Then we have

$$R_{XX}(\tau) = \int_{-\infty}^{\infty} S_{XX}(f) \exp(j2\pi f\tau) df \quad (6.7.13b)$$

and

$$R_{XX}(0) = \int_{-\infty}^{\infty} S_{XX}(f) df \quad (6.7.13c)$$

Since $R_{XX}(0)$ has the units of power, it follows from the preceding equation that $S_{XX}(f)$ has the units of power per hertz. That is, $S_{XX}(f)$ gives the distribution of power as a function of frequency and hence it is called the power spectral density (PSD) of $X(t)$.

The power spectral density function of a real-valued random process has the following properties:

1. $S_{XX}(f)$ is real and nonnegative.
2. $S_{XX}(f) = S_{XX}(-f)$.
3. If $X(t)$ has periodic components, then $S_{XX}(f)$ will have impulses at the frequencies of the periodic components.
4. The power in a finite bandwidth $[f_1, f_2]$ is given by

$$P_X[f_1, f_2] = 2 \int_{f_1}^{f_2} S_{XX}(f) df \quad (6.7.14)$$

Examples of power spectral densities are given in Figure 6.9. Notice that we are using positive and negative values of frequencies and the PSD is shown on both sides of $f = 0$. Such a spectral characterization is called a two-sided PSD.

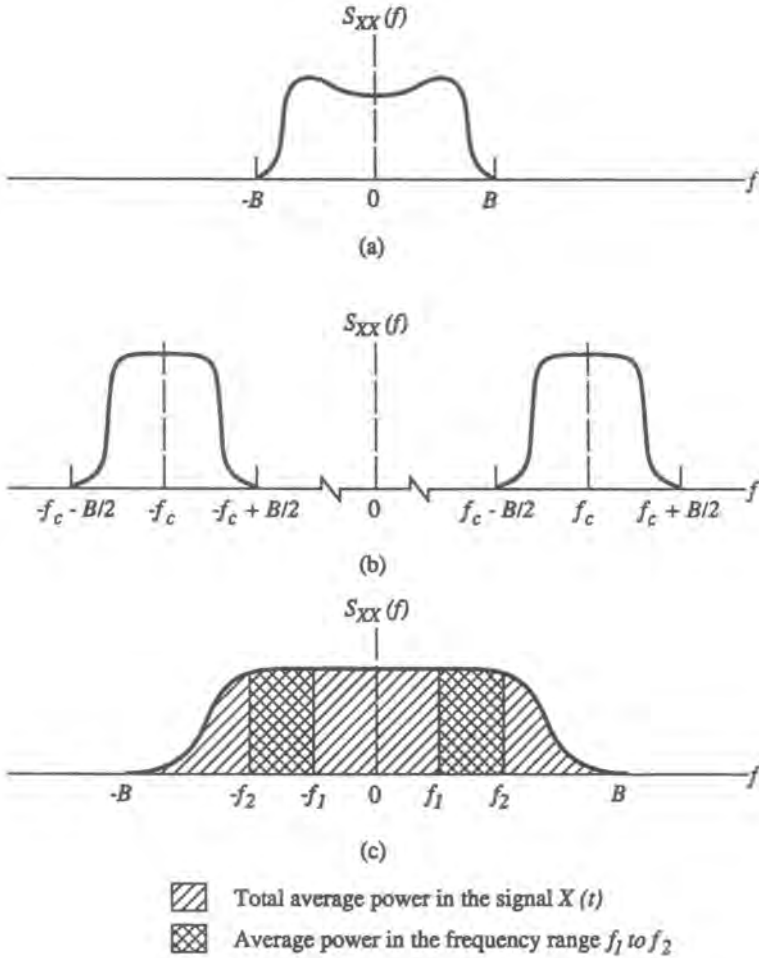


Figure 6.9. Examples of power spectral densities. (a) Lowpass spectrum; (b) bandpass spectrum; (c) power calculations.

6.7.4.4. Lowpass and Bandpass Processes

A random process is said to be lowpass if its PSD is zero for $|f| > B$, and B is called the bandwidth of the process. On the other hand, a process is said to be bandpass if its PSD is zero outside the band

$$f_c - \frac{B}{2} \leq |f| \leq f_c + \frac{B}{2}$$

f_c is usually referred to as the center frequency and B is the bandwidth of the process. Examples of lowpass and bandpass spectra are shown in Figure 6.9. A bandpass process can be represented using a complex lowpass equivalent (see Section 6.8.5).

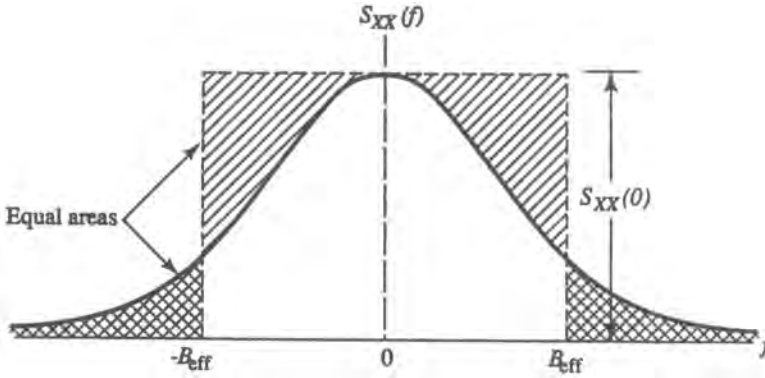


Figure 6.10. Definition of effective bandwidth for a lowpass signal.

6.7.4.5. Power and Bandwidth Calculations

As stated in (6.7.13c), the area under the PSD function gives the total power in a real-valued random process $X(t)$. The power in a finite band of frequencies f_1 to f_2 , $0 < f_1 < f_2$, is the area under the PSD function from $-f_2$ to $-f_1$ plus the area from f_1 to f_2 (Figure 6.9c).

Some processes may have PSD functions with nonzero values for all values of f , for example, $S_{XX}(f) = \exp(-f^2/2)$. For such processes, several indicators are used as measures of the spread of the PSD in the frequency domain. One popular measure is the effective (or equivalent) bandwidth B_{eff} . For zero mean random processes with continuous PSD, B_{eff} is defined as (see Figure 6.10)

$$B_{\text{eff}} = \frac{1}{2} \frac{\int_{-\infty}^{\infty} S_{XX}(f) df}{\max[S_{XX}(f)]} \quad (6.7.15)$$

The effective bandwidth is related to a measure of the spread of the autocorrelation function called the correlation time τ_c , where

$$\tau_c = \frac{1}{2} \frac{\int_{-\infty}^{\infty} R_{XX}(\tau) d\tau}{R_{XX}(0)} \quad (6.7.16)$$

If $S_{XX}(f)$ is continuous and has a maximum at $f = 0$, then it can be shown that

$$B_{\text{eff}} = \frac{1}{2\tau_c} \quad (6.7.17)$$

6.7.5. Cross-Power Spectral Density Function and Its Properties

The relationship between two real-valued random processes $X(t)$ and $Y(t)$ is expressed in the frequency domain via the cross-power spectral density (CPSD) function $S_{XY}(f)$, which is defined as the Fourier transform of the cross-correlation function $R_{XY}(\tau)$,

$$S_{XY}(f) = \int_{-\infty}^{\infty} R_{XY}(\tau) \exp(-j2\pi f\tau) d\tau \quad (6.7.18)$$

and

$$R_{XY}(\tau) = \int_{-\infty}^{\infty} S_{XY}(f) \exp(j2\pi f\tau) df \quad (6.7.19)$$

Unlike the PSD, which is a real-valued function of f , the CPSD will in general be a complex-valued function. Some of the properties of CPSD are as follows:

1. $S_{XY}(f) = S_{YX}^*(f)$.
2. The real part of $S_{XY}(f)$ is an even function of f , and the imaginary part of $S_{XY}(f)$ is an odd function of f .
3. $S_{XY}(f) = 0$ if $X(t)$ and $Y(t)$ are orthogonal and $S_{XY}(f) = \mu_X \mu_Y \delta(f)$ if $X(t)$ and $Y(t)$ are independent.

In many applications involving the CPSD, a real-valued function

$$\rho_{XY}^2(f) = \frac{|S_{XY}(f)|^2}{S_{XX}(f)S_{YY}(f)} \leq 1 \quad (6.7.20)$$

called the coherence function is used as an indicator of the dependence between two random processes $X(t)$ and $Y(t)$. When $\rho_{XY}^2(f_0) = 0$ at a particular frequency f_0 , then $X(t)$ and $Y(t)$ are said to be incoherent at that frequency. Two processes are said to be fully coherent at a particular frequency f_0 when $\rho_{XY}^2(f_0) = 1$. If $X(t)$ and $Y(t)$ are statistically independent, then $\rho_{XY}^2(f) = 0$ at all frequencies except at $f = 0$.

6.7.6. Power Spectral Density Functions of Random Sequences

The PSD of a random sequence $X(nT_s)$ with a uniform sampling time of 1 ($T_s = 1$) is defined by the Fourier transform of the autocorrelation sequence as

$$S_{XX}(f) = \sum_{n=-\infty}^{\infty} \exp(-j2\pi fn) R_{XX}(n), \quad -1/2 < f < 1/2 \quad (6.7.21a)$$

The definition implies that $S_{XX}(f)$ is periodic in f with period 1. We will only consider the principal part, $-1/2 < f < 1/2$. Then it follows that

$$R_{XX}(n) = \int_{-1/2}^{1/2} S_{XX}(f) \exp(j2\pi fn) df \quad (6.7.21b)$$

It is important to observe that if the uniform sampling time T_s is not 1 s (i.e., if nT_s is the time index instead of n), then the actual frequency range is not 1, but $1/T_s$. That is, frequencies are scaled by f_s .

If $X(n)$ is real, then $R_{XX}(n)$ will be even and

$$S_{XX}(f) = \sum_{n=-\infty}^{\infty} R_{XX}(n) \cos(2\pi fn), \quad |f| < 1/2 \quad (6.7.22)$$

which implies that $S_{xx}(f)$ is real and even. It is also nonnegative. In fact, $S_{xx}(f)$ of a sequence has the same properties as $S_{xx}(f)$ of a continuous-time process except, of course, as defined, $S_{xx}(f)$ of a sequence is periodic.

6.8. Random Process Models

Signals such as impulses and complex exponentials play a central role in the analysis and design of deterministic signals and systems. By operating on these simple signals, we can generate more complex signal structures which can be used to drive analysis and simulations. There are several classes of random processes which play a similar role in analysis and simulations. In the following sections, we present a number of these models and discuss their properties and applications.

6.8.1. Random Sequences

6.8.1.1. Independent Sequences

A stationary random sequence $\{X(n)\}$ in which $X(k)$ and $X(k+j)$ are independent for all k and $j \neq 0$ is often used to simulate information sources and sampled values of noise in communication systems. Such a sequence has the following properties:

$$E\{X(n)\} = \mu_x \quad (6.8.1a)$$

$$E\{X(n)X(n+j)\} = \begin{cases} \sigma_x^2, & j = 0 \\ 0, & j \neq 0 \end{cases} \quad (6.8.1b)$$

and

$$S_{xx}(f) = \sigma_x^2, \quad |f| < 1/2$$

The sequence $\{X(n)\}$ is also called a white noise sequence because of its flat power spectral density (analogy: white light has a flat spectrum).

6.8.1.2. Markov Sequences (First Order)

These sequences have the property that

$$P[X(n)|X(n-1), X(n-2), \dots, X(n-k)] = P[X(n)|X(n-1)]$$

that is, future values of the sequences depend only on the most recent value. Markov sequences are used to model the output of information sources that emit dependent sequences of symbols. Examples include the sequence of letters in an English language message and sampled values of speech and video signals.

Suppose that the random sequence $\{X(n)\}$ is discrete-valued, i.e., $X(n)$ has one of N possible values, say $a_1, a_2, \dots, a_N$. If we use the notation $\pi_i(n) = P(X_n = a_i)$ and $p_{ij}(s, n) = P(X_n = a_j | X_s = a_i), (n > s)$, then

$$\pi_j(n) = \sum_{i=1}^N \pi_i(s) p_{ij}(s, n) \quad (6.8.2)$$

If $p_{ij}(s, n) = p_{ij}(n - s)$, that is, if the transition probabilities depend only on the time difference, then the sequence $\{X(n)\}$ is called homogeneous. Such a sequence is completely characterized by a set of initial probabilities $\pi_i(1), (i = 1, 2, \dots, N)$ and a set of transition probabilities $p_{ij}(1), (i, j = 1, 2, \dots, N)$. If we let $\boldsymbol{\pi}(k)$ be a column vector whose i th entry is $\pi_i(k)$ and let $\mathbf{P}$ be an $N \times N$ matrix whose (i, j) th entry is $p_{ij}(1)$, then we have

$$\boldsymbol{\pi}(k+1) = \mathbf{P}^T \boldsymbol{\pi}(k) \quad (6.8.3a)$$

or

$$\boldsymbol{\pi}(k+1) = (\mathbf{P}^T)^k \boldsymbol{\pi}(1) \quad (6.8.3b)$$

The matrix $\mathbf{P}$ is called the probability transition matrix of the Markov sequence. The Markov sequence is stationary if $\boldsymbol{\pi}(k+1) = \boldsymbol{\pi}(k)$.

Markov sequences are also represented by state transition diagrams as shown in Figure 6.11, in which states are represented by nodes, transitions between nodes are shown as directed lines, and the transition probabilities p_{ij} are shown along the lines. By assigning a symbol to each transition path, we can construct models of information sources emitting dependent sequences of symbols. For example, Figure 6.11 can be used to model an information source that emits a sequence of symbols from the alphabet $\{A, B, C\}$. Details of models of Markov-type information sources may be found in Ref. 11. This model can also be

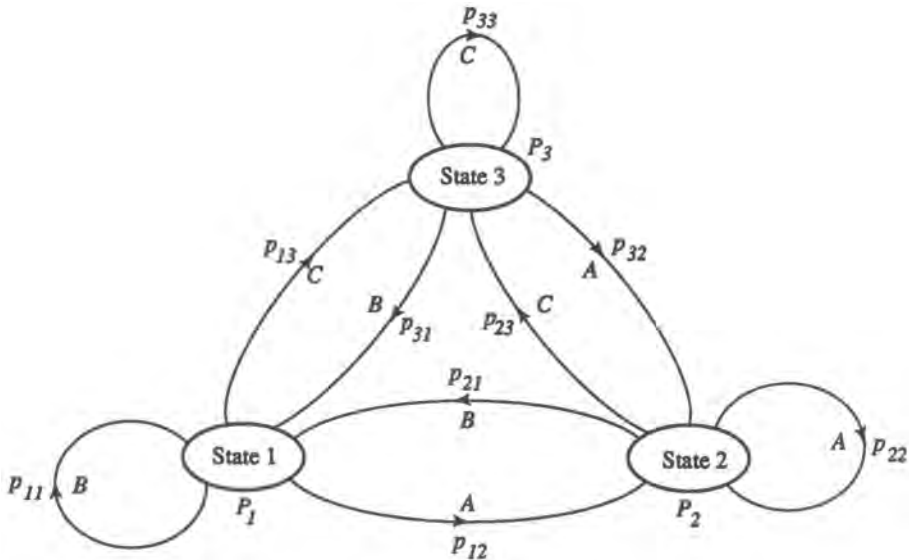


Figure 6.11. State transition diagram.

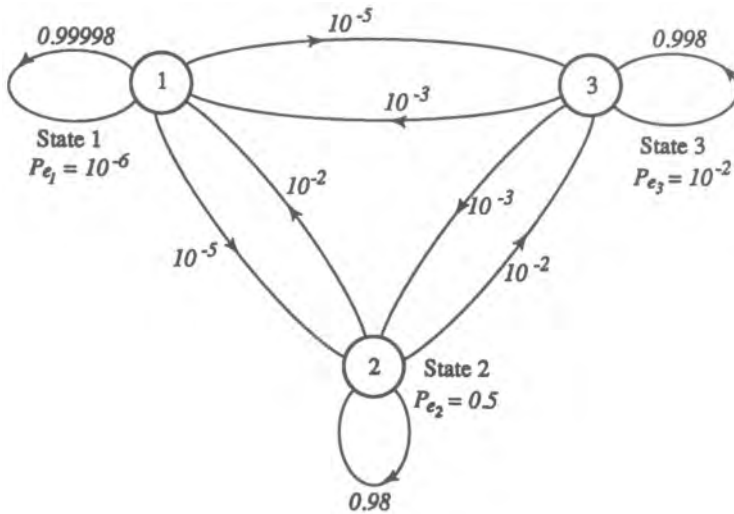


Figure 6.12. Markov channel model.

used to represent the amplitude sequence at the output of an A/D converter by assigning numerical values to the symbols of the alphabet.

Markov models are frequently used for characterizing error-generating mechanisms in digital communication channels. The Markov channel model considers the channel to be in one of several states (three states in the example shown in Figure 6.12). In State 1, transmission errors occur at a rate of $P_{e1} = 10^{-6}$, whereas in State 2, transmission errors occur at a rate of $P_{e2} = 10^{-2}$, and in State 3, at a rate $P_{e3} = 0.5$. Transitions between states occur with probability p_{ij} .

Typically, the probability of remaining in a given state is much higher than leaving the state. Since State 3 represents an error rate of 0.5, errors tend to occur in bursts when the channel is in State 3. On the other hand, State 1 represents a very low error rate and errors tend to occur in a random fashion. Ordinarily, three or four states are sufficient to define a model whose intererror interval distribution is similar to that of a real channel.

An excellent discussion of Markov channel models may be found in Ref. 12. The application of the modeling of channels as discrete Markov models to the simulation context is described in Chapters 9 and 12.

6.8.1.3. Autoregressive and Moving Average (ARMA) Sequences

This model plays an important role in estimating the PSD of random processes in parametric form and is also used to generate random sequences with a given autocorrelation or power spectral density function. An ARMA sequence is generated by the model

$$X(n) = \underbrace{\sum_{i=1}^p \phi_{p,i} X(n-i)}_{\text{Autoregressive part}} + \underbrace{\sum_{k=1}^q \theta_{q,k} e(n-k) + e(n)}_{\text{Moving average part}} \quad (6.8.4a)$$

where $e(n)$ is a stationary white Gaussian sequence with a pdf

$$f_{e(n)}(e) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(-\frac{e^2}{2\sigma^2}\right)$$

σ^2 , $\phi_{p,i}$, $i = 1, 2, \dots, p$, and $\theta_{q,k}$, $k = 1, 2, \dots, q$, are parameters of the ARMA (p, q) model. The sequence $X(n)$ has the following properties:

1. $X(n)$ has a Gaussian pdf with $\mu_X = 0$.
2. In the stationary case, the PSD of the sequence is given by

$$S_{XX}(f) = \frac{\sigma^2 |1 + \sum_{k=1}^q \theta_{q,k} \exp(-j2\pi f k)|^2}{|1 - \sum_{i=1}^p \phi_{p,i} \exp(-j2\pi f i)|^2}, \quad |f| < \frac{1}{2} \quad (6.8.4b)$$

When $q = 0$, the ARMA model reduces to an autoregressive AR(p) model of the form

$$X(n) = \sum_{i=1}^p \phi_{p,i} X(n-i) + e(n) \quad (6.8.5a)$$

The autocorrelation and power spectral density functions of an AR(p) process are given by

$$S_{XX}(f) = \frac{\sigma^2}{|1 - \sum_{i=1}^p \phi_{p,i} \exp(-j2\pi f i)|^2}, \quad |f| < \frac{1}{2} \quad (6.8.5b)$$

and

$$r_{XX}(k) = \sum_{i=1}^p \phi_{p,i} r_{XX}(k-i), \quad k \geq 1 \quad (6.8.5c)$$

where

$$r_{XX}(k) = R_{XX}(k)/R_{XX}(0)$$

Note that $r_{XX}(k)$ is the normalized autocorrelation function.

If we want to generate a Gaussian sequence with a given set of p correlation values, then we can use (6.8.5c), which is called the Yule–Walker equation, to solve for the coefficients of an AR(p) model and use them in (6.8.5a) to transform an independent Gaussian sequence $e(n)$ into a correlated Gaussian sequence $X(n)$.

Additional details of these models may be found in Ref. 2. ARMA models are quite useful for model-based estimation of spectral densities^(2,13) and for generating sampled values of a random process with a given power spectral density or autocorrelation values.

6.8.2. *M*-ary Digital Waveforms

6.8.2.1. Introduction

Information-bearing waveforms in digital communication systems can often be modeled as

$$X(t) = \sum_{k=-\infty}^{\infty} A_k p(t - kT - D) \quad (6.8.6)$$

where $\{A_k\}$ is an amplitude sequence, $p(t)$ is a pulse of duration T , $1/T$ is the symbol rate of the sequence, and D is a random delay with a uniform pdf in the interval $[0, T]$. A_k can have one of M possible numerical values, and the sequence $\{A_k\}$ is assumed to be stationary. A sample function of $X(t)$ is shown in Figure 6.13 with $M = 4$ and $T = 1$ ms. Here ℓ_1, ℓ_2, ℓ_3 , and ℓ_4 are the four values that A_k can have, and $p(t)$ is a unit-amplitude pulse with a duration of 1 ms.

$X(t)$ is called a pulse amplitude-modulated (PAM) waveform with the following auto-correlation and power spectral density functions⁽²⁾:

$$R_{XX}(\tau) = \frac{1}{T} \left[\sum_{k=-\infty}^{\infty} R_{AA}(k) \delta(\tau - kT) \right] * [p(\tau) * p(-\tau)] \quad (6.8.7a)$$

$$S_{XX}(f) = \frac{|P(f)|^2}{T} \left[R_{AA}(0) + 2 \sum_{k=1}^{\infty} R_{AA}(k) \cos(2\pi k f T) \right] \quad (6.8.7b)$$

where we assume that

$$E\{A(k)\} = 0$$

and

$$R_{AA}(k) = E\{A(n)A(n+k)\}$$

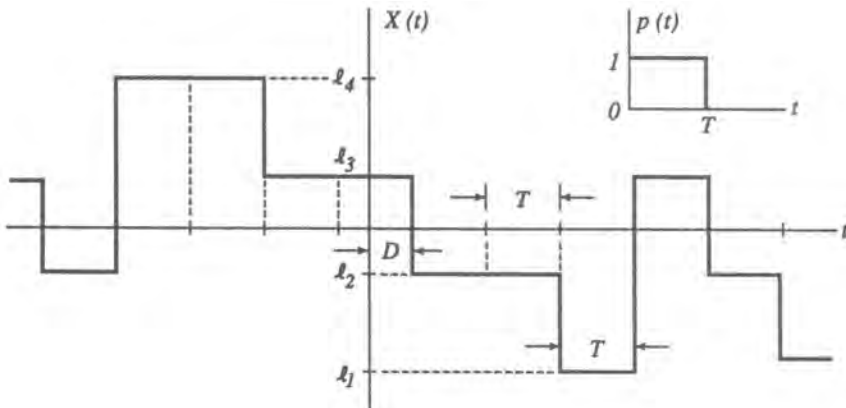


Figure 6.13. Example of a four-level digital waveform; $X(t) = \sum_k A_k p(t - kT - D)$.

6.8.2.2. Random Binary Waveform

A special case of $X(t)$ is the random binary waveform in which $A_k = \pm A$ with probability $1/2$, and $E\{A_k A_{k+j}\} = 0$ for $j \neq 0$. A sample function of the random binary waveform is shown in Figure 6.14. The autocorrelation function and PSD of the random binary waveform have the form

$$R_{XX}(\tau) = \begin{cases} A^2[1 - |\tau|/T_b], & |\tau| < T_b \\ 0, & |\tau| > T_b \end{cases} \quad (6.8.8a)$$

$$\begin{aligned} S_{XX}(f) &= A^2 T_b \left(\frac{\sin(\pi f T_b)}{\pi f T_b} \right)^2 \\ &= A^2 T_b \text{sinc}^2(f T_b) \end{aligned} \quad (6.8.8b)$$

Equation (6.8.7b) shows that the shape of the PSD depends on $P(f)$ as well as the correlation of the amplitude sequence. Thus, spectral density can be shaped by changing $p(t)$ (pulse shaping) or by controlling the correlation of the amplitude sequence (line coding).

The autocorrelation function and the power spectral density of digital waveforms in which the amplitude sequence is modeled by a Markov process is derived in Refs. 14 and 15.

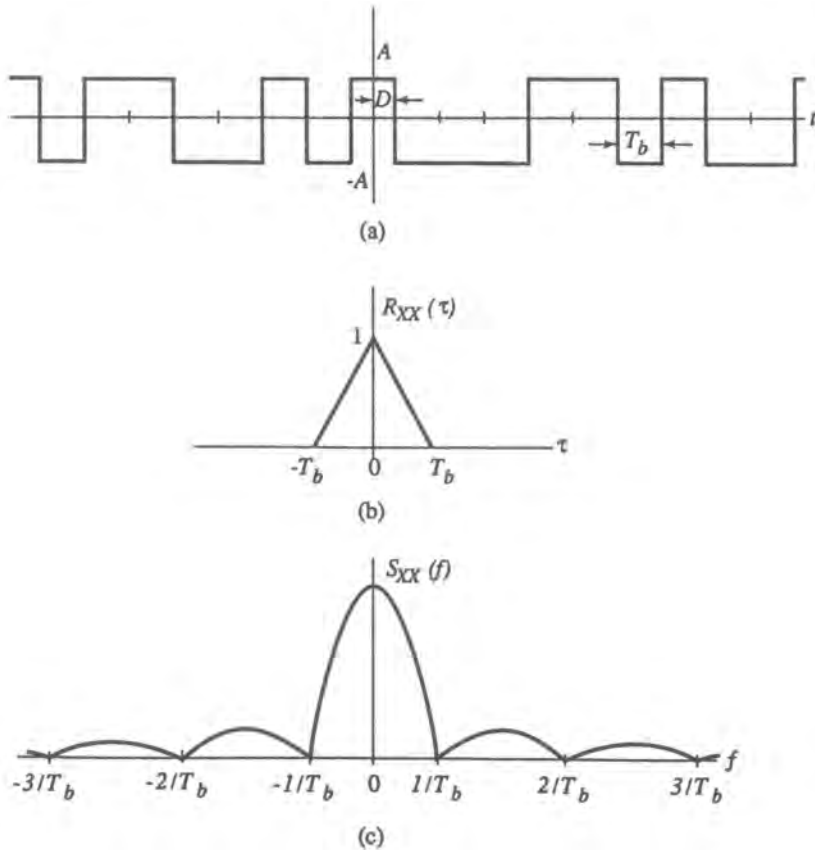


Figure 6.14. Random binary waveform, (a) $X(t)$; (b) $R_{XX}(\tau)$; $A = 1$ (c) $S_{XX}(f)$.

The expressions for these functions are rather complex and are omitted here since they provide only a limited insight into the relationships between the parameters of the model and the spectral properties of the signals.

6.8.3. Poisson Process

The Poisson process is a counting process that is used extensively in modeling traffic in communication networks (such as the number of calls received by a telephone exchange), number of passengers arriving at an airline service counter, and number of photoelectrons emitted by a p - i - n diode. The Poisson process $X(t)$ represents the number of occurrences of such events in the time interval $(0, t)$ subjected to the following assumptions:

1. Times of occurrences are distinct.
2. A finite interval contains only a finite number of occurrences.
3. An infinite interval contains an infinite number of occurrences.
4. Events do not occur at predetermined times.
5. Numbers of occurrences in nonoverlapping intervals are independent.

If the rate of occurrence $\lambda(t)$ is constant, then we have a homogeneous Poisson process with the following properties:

$$1. \quad P(X(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, \quad k = 0, 1, 2, \dots \quad (6.8.9)$$

$$2. \quad E\{X(t)\} = \lambda t \quad (6.8.10a)$$

$$\text{Var}\{X(t)\} = \lambda t \quad (6.8.10b)$$

$$3. \quad R_{XX}(t_1, t_2) = \lambda^2 t_1 t_2 + \lambda \min(t_1, t_2) \quad (6.8.11)$$

Note that the Poisson process is nonstationary because its mean, variance, and autocorrelation function are time-varying.

The time between occurrences in a Poisson process (waiting time or interarrival time) is of interest in analyzing queuing problems. It can be shown that the interarrival time W has an exponential density,

$$f_W(w) = \lambda e^{-\lambda w}, \quad w > 0 \quad (6.8.12)$$

and that nonoverlapping interarrival times are “independent.”

To simulate a Poisson process, we simulate the interarrival times using samples of independent exponentially distributed random variables.

6.8.4. Shot Noise and Impulsive Noise

6.8.4.1 Shot Noise

Emission of charged particles is a fundamental phenomenon that occurs in electronic systems. In many lightwave communication systems, for example, the transmitted information is represented by intensities of light and is detected by a photosensitive device in the

receiver that produces electrical current pulses in proportion to the incident light. The current pulses result from photoelectrons emitted at random times τ_k with an average rate λ proportional to the incident light energy. The resulting waveform can be represented by a random process of the form

$$X(t) = \sum_{k=-\infty}^{\infty} h(t - \tau_k) \quad (6.8.13)$$

where $h(t)$ is the electrical pulse produced by a single photoelectron emitted at $t = 0$ (see Figure 6.15). The emission times τ_k are assumed to form a Poisson point process with a fixed rate λ , and the Poisson process is defined for all values of $t \in (-\infty, \infty)$. Because λ is constant and there is no distinguishing origin of time, the process $X(t)$ is stationary.

$X(t)$ is called a shot-noise process and has been used extensively to model randomly fluctuating components of currents in electronic circuits containing devices such as photodiodes and vacuum tubes. This process can also be used to model phenomena such as the background return (clutter) seen by a radar. Consider, for example, an airplane flying over a terrain and sending out radar pulses. These pulses will be reflected by various objects in the terrain and the radar receiver will receive a multitude of echoes of random amplitudes and delays. The received signal can be modeled as a random process of the form

$$Y(t) = \sum_{-\infty}^{\infty} A_k h(t - \tau_k) \quad (6.8.14)$$

where $h(t)$ is the shape of the reflected pulse and τ_k is the time at which the pulse reaches the receiver. The amplitude A_k is independent of τ_k and A_j for $j \neq k$, and the amplitude sequence can be assumed to be independent and identically distributed with the probability density function $f_A(a)$. Equation (6.8.14) is also used as a model for “impulsive noise” in a variety of communication systems.

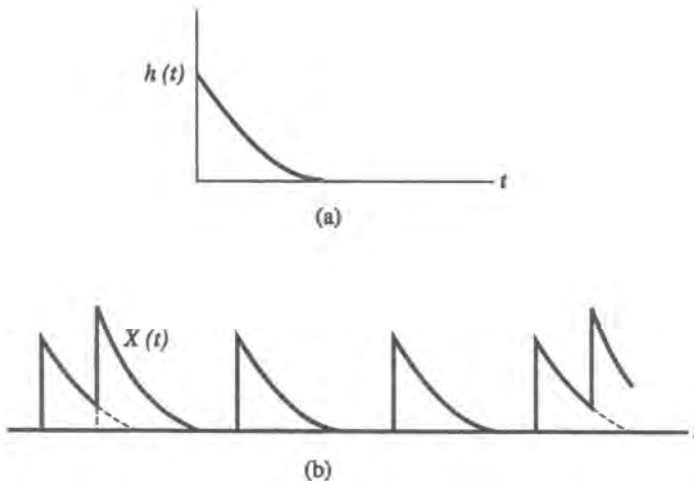


Figure 6.15. Shot-noise process. (a) Single pulse; (b) sample function of $X(t)$.

$X(t)$ and $Y(t)$ defined in (6.8.13) and (6.8.14) are stationary random processes with

$$E\{X(t)\} = \mu_X = \lambda \int_{-\infty}^{\infty} h(u) du \quad (6.8.15a)$$

$$R_{XX}(\tau) = \lambda \int_{-\infty}^{\infty} h(u)h(\tau + u) du + \mu_X^2 \quad (6.8.15b)$$

$$E\{Y(t)\} = \mu_Y = \lambda E\{A\} \int_{-\infty}^{\infty} h(u) du \quad (6.8.15c)$$

$$R_{YY}(\tau) = \lambda E\{A^2\} \int_{-\infty}^{\infty} h(u)h(u + \tau) d\tau + \mu_Y^2 \quad (6.8.15d)$$

The derivation of the probability density functions of $X(t)$ and $Y(t)$ is very difficult and the interested reader is referred to Ref. 3.

6.8.4.2 Impulsive Noise

A variety of naturally occurring and man-made phenomena exhibit “impulsive” noise behavior.⁽¹⁶⁻²¹⁾ Channels operating in the ELF and VLF bands experience atmospheric discharge; the radiofrequency range comprising HF, VHF, and UHF is plagued by man-made noise in metropolitan areas, and generally contaminated by galactic and solar noise. Wire communications are subject to lightning and switching noise. These various phenomena induce “impulsive” noise in communication systems. Impulsive noise also occurs naturally *within* some communication systems, for example, in optical receivers. While the term *impulsive* is suggestive, there is not a standard or unique definition. However, it is generally understood as a point process in time of fairly low density (interarrival times generally exceeding system time constants), with “spiky” pulses appearing at the occurrence times.

From the simulation point of view, our main concern is with the modeling of impulsive noise sources. As a practical matter, it is desirable to have a single general model which can represent the behavior of a variety of sources simply by changing a small number of parameters. Such a model is presented below. Let $\tilde{N}_t(t)$ be the complex lowpass-equivalent representation of the impulsive noise process. Then, we write

$$\tilde{N}_t(t) = \tilde{Z}(t) * \tilde{h}(t) \quad (6.8.16)$$

where we take $\tilde{Z}(t)$ to be given by

$$\tilde{Z}(t) = \sum_{i=0}^{N(t)} Z_i \delta(t - t_i) \quad (6.8.17)$$

where $Z_i = Z_{ic} - jZ_{is}$ is a sequence of complex, independent, identically distributed random variables, and $\{t_i\}$ are the “pulse” arrival times, which are dictated by some counting process $N(t)$. The pulse

$$\tilde{h}(t) = h_c(t) - jh_s(t) \quad (6.8.18)$$

can be used to control the shape of each impulse, and can be thought of as the impulse response of the actual receiver, or simply an abstract specification.

The impulse coefficient Z_i can be written as

$$Z_i = R_i(\cos \theta_i - j \sin \theta_i) \quad (6.8.19)$$

and part of the description process is to specify the probability laws of R_i and θ_i . In the model, θ_i is assumed to be uniformly distributed over $[-\pi, \pi]$, and R_i is given a choice of two distributions, a log-normal distribution,

$$f_R(r) = \frac{1}{(2\pi)^{1/2} \sigma r} \exp \left[-\frac{(\ln r - \mu)^2}{2\sigma^2} \right] \quad (6.8.20a)$$

whose mean and variance, respectively, are

$$m = e^{\mu + \sigma^2/2}, \quad s^2 = e^{2\mu} [e^{2\sigma^2} - e^{\sigma^2}] \quad (6.8.20b)$$

and a “power-Rayleigh” distribution,

$$f_R(r) = \frac{\alpha r^{\alpha-1}}{R_0^\alpha} \exp \left[-\left(\frac{r}{R_0} \right)^\alpha \right] \quad (6.8.21a)$$

whose mean and variance, respectively, are

$$m = R_0 \Gamma(1 + \alpha^{-1}), \quad s^2 = R_0^2 [\Gamma(1 + 2\alpha^{-1}) - \Gamma^2(1 + \alpha^{-1})] \quad (6.8.21b)$$

The counting process $N(t)$ in the same model is taken to have interarrival times described by a gamma distribution. That is, if X is the interarrival time random variable, then we have

$$f_X(x) = \frac{(x)^{\nu-1}}{\Gamma(\nu) \beta^\nu} e^{-x/\beta} \quad (6.8.22)$$

Here $x \geq 0$, $\nu \geq 0$, and we can set $\beta = (\nu\lambda)^{-1}$, where λ is the average number of pulses per unit time. This representation is quite flexible; for example, if $\nu = 1$, we have an exponential distribution

$$f_X(x) = \lambda e^{-\lambda x} \quad (6.8.23)$$

and if $\nu \rightarrow \infty$, (6.8.22) tends to a periodic function

$$f_X(x) = \delta(x - \lambda^{-1}) \quad (6.8.24)$$

Equation (6.8.23) describes the interarrival times for a Poisson process. Thus, (6.8.16) and (6.8.17) include as a special case the shot-noise process discussed in Section 6.8.4.1. Methods of generating the distributions above, (6.8.20)–(6.8.22), are discussed in Chapter 7.

It remains only to specify the pulse shape, which, in view of (6.8.16), and (6.8.17), is given by $\tilde{h}(t)$. Any reasonable shape is admissible, and of course any shape corresponding to observations. Four possibilities, the first three suggested in Ref. 18, are tabulated in Table 6.2.

Figure 6.16 illustrates a typical sample function of an impulsive noise (model) output.

**Table 6.2 Some Pulse Shapes for Impulsive Noise Model:
Real or Imaginary Part of $h(t)$**

Form of basic pulse shape	Coefficient A for unit peak height	Coefficient A for unit area
$Ae^{-at}u(t)$	1	a
$A(e^{-at}/\omega_c)\sin\omega_ctu(t)$	$\omega_c\exp(\pi a/2\omega_c)$	$(a^2 + \omega_c^2)/\omega_c$
$Ae^{-at}\cos\omega_0tu(t)$	1	$(a^2 + \omega_0^2)/a$
$Ap_T(t)$	1	T^{-1}

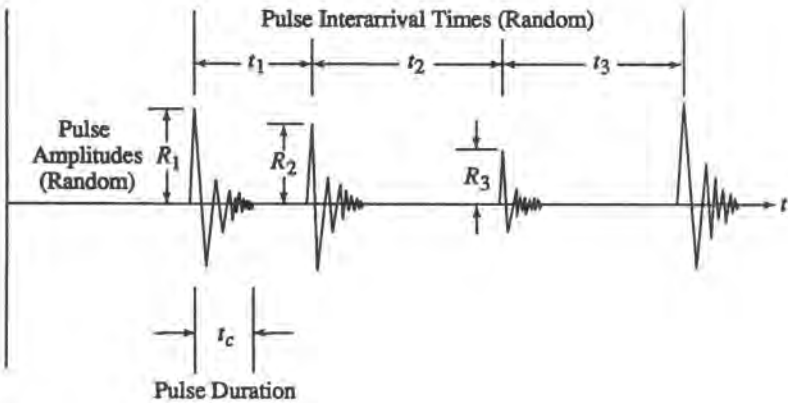


Figure 6.16. Typical impulsive noise behavior.

The model is sufficient for simulation purposes. However, it should be pointed out that the performance of standard receiver structures can be greatly affected by the presence of impulsive noise. This behavior is not simply characterized, as it depends on the ratio of thermal noise to impulsive noise and on the parameters of the impulsive noise. The reader is referred to the literature; see, e.g., Refs. 19 and 20 for sample results. It should also be noted that there are nonlinear receiver structures that are much more efficient for detecting signals accompanied by impulsive noise than are the standard (additive white Gaussian noise, AWGN, channel) receiver structures.⁽²¹⁾

6.8.5. Gaussian Process

In this section we introduce the Gaussian random process, which is the most commonly used model for noise, interference, and signals in communication systems. Many random phenomena in physical problems are well approximated by Gaussian random processes. By virtue of the central limit theorem, a number of processes such as a dense shot-noise process can be approximated by a Gaussian process. Furthermore, the output of a linear system made up of a weighted sum of a large number of independent samples of the input random process tends to approach a Gaussian process. Gaussian processes play a central role in the theory and analysis of random phenomena both because they are good approximations to the observations and because multivariate Gaussian distributions are analytically simple. Simulation of sampled values of a Gaussian process is also very easy.

One of the most important uses of the Gaussian process is to model and analyze the effects of “thermal” noise in electronic circuits used in communication systems. Individual circuits contain resistors, inductors, and capacitors as well as semiconductor devices. The resistors and semiconductor elements contain charged particles subjected to random motion due to thermal agitation. The random motion of charged particles causes fluctuations in the current waveforms (or information-bearing signals) that flow through these components. These fluctuations are called thermal noise and are often of sufficient strength to mask a weak signal and make the detection of signals a difficult task. Models of thermal noise are used to analyze and minimize the effects of noise on signal detection.

6.8.5.1. Definition of a Gaussian Process

A real-valued random process $X(t)$, $t \in \Gamma$, is called a Gaussian process if all its n th-order density functions are n -variate Gaussian. If we denote $X(t_i)$ by X_i , then for any n

$$f_{\mathbf{x}}(\mathbf{x}) = [(2\pi^{n/2}|\Sigma_{\mathbf{x}}|^{1/2})]^{-1} \exp[-\frac{1}{2}(\mathbf{x} - \mu_{\mathbf{x}})^T \Sigma_{\mathbf{x}}^{-1}(\mathbf{x} - \mu_{\mathbf{x}})] \quad (6.8.25)$$

where

$$\begin{aligned} \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} &= \begin{bmatrix} X(t_1) \\ X(t_2) \\ \vdots \\ X(t_n) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\ \mu_{\mathbf{X}} &= \begin{bmatrix} E\{X(t_1)\} \\ E\{X(t_2)\} \\ \vdots \\ E\{X(t_n)\} \end{bmatrix} = \begin{bmatrix} \mu_X(t_1) \\ \mu_X(t_2) \\ \vdots \\ \mu_X(t_n) \end{bmatrix} \\ \Sigma_{\mathbf{x}} &= \begin{bmatrix} C_{XX}(t_1, t_1) & C_{XX}(t_1, t_j) & \cdots & C_{XX}(t_1, t_n) \\ & \ddots & \vdots & \\ \vdots & & C_{XX}(t_i, t_j) & \cdots \\ & \cdots & & C_{XX}(t_n, t_1) & C_{XX}(t_n, t_j) & \cdots & C_{XX}(t_n, t_n) \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} C_{XX}(t_i, t_j) &= E\{X(t_i)X(t_j)\} - \mu_X(t_i)\mu_X(t_j) \\ &= R_{XX}(t_i, t_j) - \mu_X(t_i)\mu_X(t_j) \end{aligned}$$

The joint density function given in (6.8.25) is a multivariate Gaussian density function; its properties are given in Section 6.4.2.

If the process is WSS, then we have

$$E\{X(t_i)\} = \mu_X$$

and

$$C_{XX}(t_i, t_j) = R_{XX}(|t_i - t_j|) - \mu_X^2$$

The n th-order distributions of a Gaussian process depend on the two functions $\mu_X(t)$ and $C_{XX}(t, t + \tau)$. When the process is WSS, then $\mu_X(t)$ and $C_{XX}(t, t + \tau)$ do not depend on t , which implies that

$$f_{\mathbf{x}}[x(t_1), x(t_2), \dots, x(t_n)] = f_{\mathbf{x}}[x(t_1 + \tau), x(t_2 + \tau), \dots, x(t_n + \tau)]$$

that is, the process is strict-sense stationary. In this case $\mu_X(t) = \mu_X$, $C_{XX}(t, t + \tau) = C_{XX}(\tau)$, and $R_{XX}(t, t + \tau) = R_{XX}(\tau)$.

6.8.5.2. Models of White and Bandlimited White Noise

Thermal noise generated in resistors and semiconductors is modeled as a zero-mean, stationary, Gaussian random process $N(t)$ with a power spectral density that is flat over a very wide range of frequencies. Such a process is called white (Gaussian) noise in analogy to white light, whose spectral density is broad and uniform over a wide frequency range. The power spectral density of thermal noise has been shown to have the value $kT/2$ joules, where k is Boltzmann's constant (1.38×10^{-23} J/K) and T is the equivalent temperature in Kelvins of the noise source.

It is customary to denote the uniform spectral density of white noise by $\eta/2$ (or $N_0/2$), the factor of 2 signifying two-sided spectral density:

$$S_{NN}(f) = \frac{\eta}{2}, \quad -\infty < f < \infty \quad (6.8.26a)$$

Note that the spectral density given in (6.8.17a) yields

$$R_{NN}(\tau) = \frac{\eta}{2} \delta(\tau) \quad (6.8.26b)$$

which implies that $X(t)$ and $X(t + \tau)$ are independent for any value of $\tau \neq 0$.

The spectral density given in (6.8.26a) is not physically realizable since it implies infinite average power, that is,

$$\int_{-\infty}^{\infty} S_{NN}(f) df \rightarrow \infty$$

However, bandwidths of real systems are always finite, and since

$$\int_{-B}^B S_{NN}(f) df = \eta B < \infty$$

for any finite bandwidth B , the spectral density given in (6.8.26a) can be used over finite bandwidths.

If $N(t)$ has a power spectral density of the form (Figure 6.17)

$$S_{NN}(f) = \begin{cases} \eta/2, & |f| < B \\ 0, & \text{elsewhere} \end{cases}$$

then $N(t)$ is called bandlimited white noise.

The reader can verify that this process has the following properties:

1. $E\{N^2(t)\} = \eta B$.
2. $R_{NN}(\tau) = \eta B \frac{\sin(2\pi B\tau)}{2\pi B\tau} \rightarrow \frac{\eta}{2} \delta(\tau)$ as $B \rightarrow \infty$
3. $N(t)$ and $N(t + k\tau_0)$, where k is an integer (nonzero) and $\tau_0 = 1/2B$, are independent.

It should be pointed out that the terms “white” and “bandlimited white” refer to the spectral shape of the process. These terms by themselves do not imply that the distributions associated with $X(t)$ are Gaussian. A non-Gaussian process may also have a flat, that is, white, power spectral density, and Gaussian pdf does not imply white PSD.

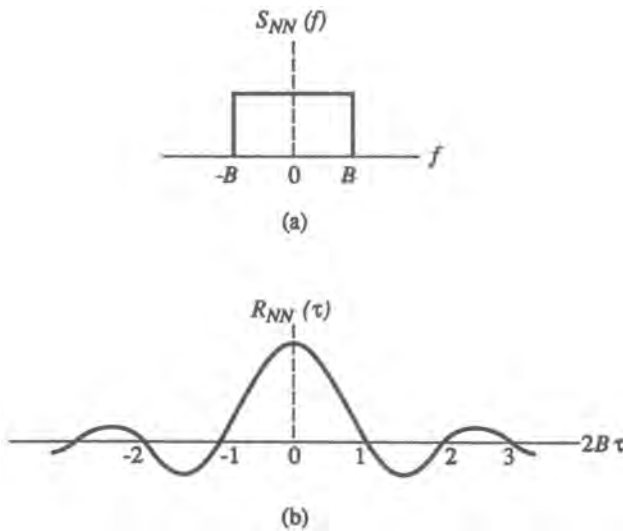


Figure 6.17. Bandlimited white noise. (a) Power spectral density; (b) autocorrelation function.

6.8.5.3. Quadrature Representation of Bandpass (Gaussian) Signals

In communication systems, information-bearing signals often have the form

$$X(t) = A_X(t) \cos[2\pi f_0 t + \theta_X(t)] \quad (6.8.27a)$$

where $X(t)$ is a bandpass signal, and $A_X(t)$ and $\theta_X(t)$ are lowpass signals. $A_X(t)$ is called the envelope of the bandpass signal $X(t)$, and $\theta_X(t)$ is the phase; f_0 is the carrier or center frequency. $X(t)$ can also be expressed as

$$\begin{aligned} X(t) &= A_X(t) \cos \theta_X(t) \cos 2\pi f_0 t - A_X(t) \sin \theta_X(t) \sin 2\pi f_0 t \\ &= X_c(t) \cos 2\pi f_0 t - X_s(t) \sin 2\pi f_0 t \end{aligned} \quad (6.8.27b)$$

$X_c(t) = A_X(t) \cos \theta_X(t)$ and $X_s(t) = A_X(t) \sin \theta_X(t)$ are called, respectively, the in-phase and quadrature components of $X(t)$.

If the noise in the communication system is additive, then the receiver observes and processes $X(t) + N(t)$ and attempts to extract $X(t)$. In order to analyze the performance of the receiver, it is useful to derive a time-domain representation of the noise in quadrature form.

It is shown in Refs. 1 and 2 that a zero-mean, stationary, Gaussian bandpass process $N(t)$ can be represented in quadrature form as

$$N(t) = N_c(t) \cos(2\pi f_0 t) - N_s(t) \sin(2\pi f_0 t)$$

where $N_c(t)$ and $N_s(t)$ are the quadrature components with the following properties^(1,2):

1. $N_c(t)$ and $N_s(t)$ are two jointly stationary Gaussian lowpass random processes.
- 2.

$$E\{N_c(t)\} = E\{N_s(t)\} = 0 \quad (6.8.28)$$

3. The power spectral densities of $N_c(t)$ and $N_s(t)$ are related to the PSD of $N(t)$ by

$$\begin{aligned} S_{N_c N_c}(f) &= S_{N_s N_s}(f) \\ S_{N_c N_c}(f) &= S_{NN}(f + f_0)U(f + f_0) \\ &\quad + S_{NN}(-f + f_0)U(-f + f_0) \end{aligned} \quad (6.8.29a)$$

and

$$\begin{aligned} jS_{N_s N_c}(f) &= S_{NN}(f + f_0)U(f + f_0) \\ &\quad - S_{NN}(-f + f_0)U(-f + f_0) \end{aligned} \quad (6.8.29b)$$

where $U(f)$ is the unit step function.

An example of the relationship between the spectra is shown in Figure 6.18. We note that if the positive part of $S_{NN}(f)$ is symmetrical about f_0 , then $S_{N_s N_c}(f) = 0$. In this case, $N_c(t)$ and $N_s(t)$ are orthogonal.

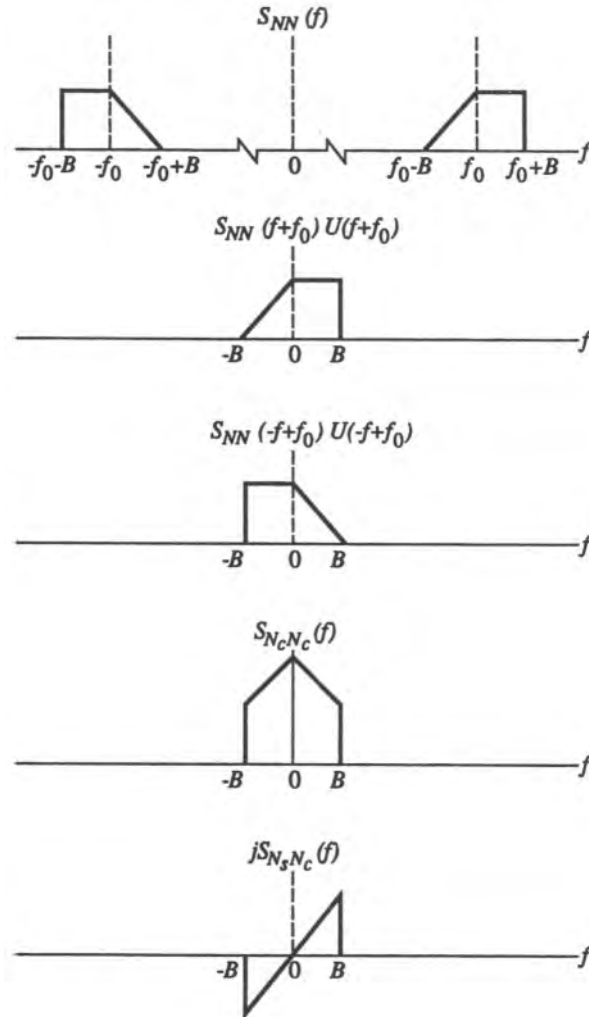


Figure 6.18. Quadrature representation of a bandpass process. (From K. Sam Shanmugan and A. M. Breipohl, *Random Data: Detection, Estimation and Data Analysis*, © John Wiley and Sons, 1988, reproduced with permission.)

The quadrature components are combined to form the complex lowpass representation of $N(t)$ as

$$\tilde{N}(t) = N_c(t) + jN_s(t)$$

This representation is similar to the representation for deterministic bandpass signals discussed in Chapter 3, and for simulation purposes we use sampled values of $\tilde{N}(t)$.

Given the representation in quadrature form

$$N(t) = N_c(t) \cos(2\pi f_0 t) - N_s(t) \sin(2\pi f_0 t)$$

we can convert it easily to envelope and phase form as

$$N(t) = A_N(t) \cos[2\pi f_0 t + \theta_N(t)]$$

where

$$A_N(t) = [N_c^2(t) + N_s^2(t)]^{1/2} \quad (6.8.30a)$$

and

$$\theta_N(t) = \tan^{-1} \left[\frac{N_s(t)}{N_c(t)} \right] \quad (6.8.30b)$$

The envelope A_N has a Rayleigh pdf and the phase θ_N has a uniform pdf.

The input to the receiver in many communication systems will consist of

$$Y(t) = S(t) + N(t)$$

where $S(t)$ is a signal component which is often modeled as an unmodulated carrier of the form

$$S(t) = A \cos(2\pi f_0 t)$$

and $N(t)$ is bandlimited Gaussian noise. Substituting the quadrature representation for $N(t)$, we can write $Y(t)$ as

$$Y(t) = A \cos(2\pi f_0 t) + N_c(t) \cos(2\pi f_0 t) - N_s(t) \sin[2\pi f_0 t]$$

The envelope R of $Y(t)$,

$$R = \{[A + N_c(t)]^2 + [N_s(t)]^2\}^{1/2}$$

can be shown⁽²⁾ to have a Rice pdf of the form

$$f_R(r) = \frac{r}{N_0} I_0 \left(\frac{Ar}{N_0} \right) \exp \left(-\frac{r^2 + A^2}{2N_0} \right), \quad r > 0 \quad (6.8.31)$$

where $I_0(a)$ is the modified Bessel function of the first kind defined by

$$\begin{aligned} I_0(a) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \exp[a \cos(u)] du \\ &= \frac{1}{2\pi} \int_0^{2\pi} \exp[a \cos(u)] du \end{aligned} \quad (6.8.32)$$

The Rice pdf given in (6.8.31) is also used for modeling fading in communication channels (see Chapter 9).

6.9. Transformation of Random Processes

In Section 6.5, we developed methods of obtaining the distribution of $Y = g(X)$, where X is an “input” random variable, Y is the “output” or the transformed variable, and $g(\cdot)$ represents the “transfer characteristic” of the “system.”

We now describe procedures for obtaining the properties of the output $Y(t)$ of a system whose input is a random process $X(t)$. The system may perform signal processing operations such as integration, differentiation, and filtering.

6.9.1. Response of Linear Time-Invariant Causal (LTIVC) System

Many physical systems can be modeled as (lumped), linear, time-invariant, and causal, and their dynamic behavior is described by the convolution integral

$$\begin{aligned} y(t) &= \int_{-\infty}^{\infty} x(\tau)h(t - \tau) d\tau \\ &= \int_{-\infty}^{\infty} h(\tau)x(t - \tau) d\tau \end{aligned} \quad (6.9.1)$$

where $x(t)$ is the input, $h(t)$ is the impulse response [$h(t) = 0$ for $t < 0$ for causal systems], and $y(t)$ is the output.

When the input is a random process $X(t)$ then each member function $x(t)$ of $X(t)$ produces a member function $y(t)$ of the output random process $Y(t)$. Given a description of the input process $X(t)$ and a description of the system, we want to obtain the properties of $Y(t)$ such as the mean, the autocorrelation function, and at least some of the lower order probability distribution functions of $Y(t)$. In most cases we might be interested in the mean and autocorrelation function. Only in some special cases will we want to (and be able to) determine the probability distribution functions.

6.9.1.1. Stationarity

If the input $X(t)$ is stationary, then the output $Y(t)$ of a time-invariant system can be expected to be stationary. This is indeed the case. If the input is wide-sense stationary (WSS), then the output is WSS, and if the input is strict-sense stationary (SSS), then the output is also SSS. Proof of this may be found in Refs. 1 and 2.

6.9.1.2. Probability Distribution

The probability distribution function of $Y(t)$ is, in general, difficult to obtain except for the Gaussian case. If the input is Gaussian, then by viewing the convolution integral as a linear operation, we can show that all output distributions will be Gaussian.

6.9.1.3. Mean, Autocorrelation, and Power Spectral Density Functions

The input–output relationship of the LTIC system can be derived as^(1,2)

$$\mu_Y = \mu_X H(0) \quad (6.9.2a)$$

$$R_{XY}(\tau) = R_{XX}(\tau) * h(\tau) \quad (6.9.2b)$$

$$R_{YY}(\tau) = R_{XY}(\tau) * h(-\tau) = R_{XX}(\tau) * h(\tau) * h(-\tau) \quad (6.9.2c)$$

$$S_{YY}(f) = S_{XX}(f)|H(f)|^2 \quad (6.9.2d)$$

where $h(t)$ is the impulse response of the system and $H(f)$ is the transfer function,

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{-j2\pi ft} dt$$

and $*$ indicates convolution.

Equation (6.9.2a) shows that the mean of the output is the mean of the input multiplied by the “dc response” $H(0)$ of the system, and (6.9.2d) shows that the spectral properties of the input signal are modified by the system according to $|H(f)|^2$, which is sometimes referred to as the “power transfer function,” whereas $H(f)$ is the voltage-to-voltage transfer function.

6.9.2. Filtering

Filtering is used in communication systems to select a desired signal and reject interference and noise. The filter transfer function is carefully selected to modify the spectral components according to (6.9.2d). An ideal filter has a transfer function whose magnitude is flat within its “passband” (bandwidth occupied by the desired signal) and zero outside of this band of frequencies; its midband gain is unity and its phase is a linear function of frequency.

The transfer function of practical filters will deviate from their corresponding ideal versions. The Butterworth lowpass filter, for example, as shown in Chapter 4, has a magnitude response of the form

$$|\tilde{H}(f)|^2 = \frac{1}{1 + (f/B)^{2n}}$$

where n is the order of the filter and B is a parameter that determines the bandwidth of the filter. To simplify analysis, it is often convenient to approximate the transfer function of a practical filter $\tilde{H}(f)$ by an ideal version $\tilde{H}(f)$ as shown in Figure 6.19. In replacing an actual system with an ideal one, the latter would be assigned “midband” gain and phase slope that approximate the actual values. The bandwidth B_N of the ideal approximation (in lowpass and bandpass cases) is chosen according to some convenient basis. For example, the bandwidth of the ideal filter can be set equal to the 3-dB (or half-power) bandwidth of the actual filter or it can be chosen to satisfy a specific requirement. An example of the latter case is to choose B_N such that the actual and ideal systems produce the same output power when each is excited by the same source.

Consider an ideal and actual lowpass filter whose input is white noise, that is, a noise process whose power spectral density has a constant value, say $\eta/2$, for all frequencies. The average output powers of the two filters are given by

$$E|\tilde{Y}(t)|^2 = \frac{\eta}{2} \int_{-\infty}^{\infty} |\tilde{H}(f)|^2 df$$

for the actual filter and

$$E|\tilde{\tilde{Y}}(t)|^2 = \left(\frac{\eta}{2}\right) |\tilde{H}(0)|^2 2B_N \quad (6.9.3)$$

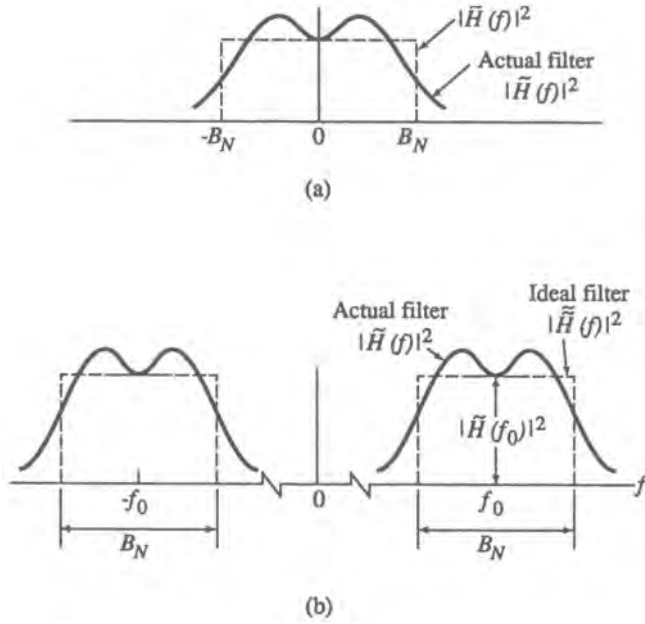


Figure 6.19. Noise bandwidth of filter; areas under $|\tilde{H}(f)|^2$ and $|\tilde{H}(f)|^2$ are equal. (a) Lowpass; (b) bandpass.

for the ideal version. By equating the output powers, we obtain

$$B_N = \frac{\int_{-\infty}^{\infty} |\tilde{H}(f)|^2 df}{2|\tilde{H}(0)|^2} \quad (6.9.4)$$

and by applying Parseval's theorem, we can show that

$$B_N = \frac{\int_{-\infty}^{\infty} |\tilde{h}(t)|^2 dt}{2|\int_{-\infty}^{\infty} \tilde{h}(t) dt|^2} \quad (6.9.5)$$

The time-domain expression given in (6.9.5) is used in simulations for computing B_N from the impulse response of the filter.

This value of B_N is called the noise-equivalent bandwidth of the actual filter. Extension of this definition to the bandpass case is obvious (see Figure 6.19). For an n th-order Butterworth lowpass filter, the noise bandwidth is

$$B_N = B(\pi/2n)/\sin(\pi/2n) \quad (6.9.6)$$

As $n \rightarrow \infty$, the Butterworth filter approaches the transfer function of an ideal lowpass filter with a bandwidth B . The noise bandwidths of other filters are given in Ref. 22.

6.9.3. Integration

Integration (or weighted averaging) is a signal-processing operation that is commonly used in communication receivers to minimize the effects of noise. The integration operation represented by

$$Y(t) = \frac{1}{T} \int_{t-T}^t X(\alpha) d\alpha$$

is equivalent to passing $X(t)$ through an LTIVC system with an impulse response

$$h(t) = \begin{cases} 1/T & \text{for } 0 < t < T \\ 0 & \text{otherwise} \end{cases} \quad (6.9.7a)$$

and the corresponding transfer function

$$H(f) = e^{-j\pi f T} \frac{\sin(\pi f T)}{\pi f T} \quad (6.9.7b)$$

Note that the integration attenuates high-frequency spectral components of the input by $1/f^2$ since $|H(f)|^2$ falls off as $1/f^2$.

■ *Example 6.9.1.* $X(t)$ is a zero-mean Gaussian noise with the PSD shown in Figure 6.20. Let

$$Y(t) = \frac{1}{T} \int_{t-T}^t X(\alpha) d\alpha$$

where $T \gg 1/B$. Calculate σ_Y^2 and compare it with σ_X^2 .

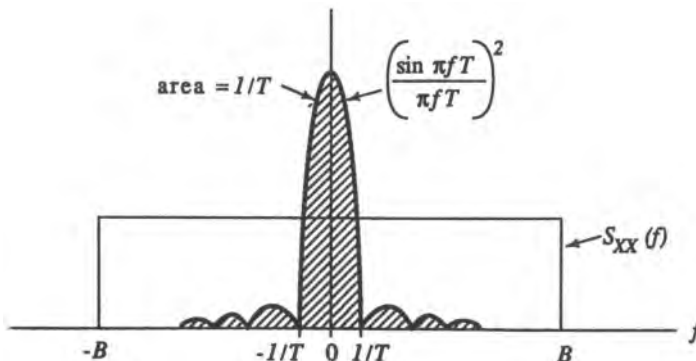


Figure 6.20. Variance calculations in the frequency domain.

Solution:

$$\begin{aligned}\sigma_X^2 &= \int_{-\infty}^{\infty} S_{XX}(f) df = 2B \\ \sigma_Y^2 &= \int_{-\infty}^{\infty} S_{YY}(f) df = \int_{-\infty}^{\infty} S_{XX}(f) |H(f)|^2 df \\ &= \int_{-B}^B \left(\frac{\sin(\pi f T)}{\pi f T} \right)^2 df \approx \int_{-\infty}^{\infty} \left(\frac{\sin(\pi f T)}{\pi f T} \right)^2 df = \frac{1}{T}\end{aligned}$$

as shown in Figure 6.20. Hence, we have

$$\frac{\sigma_Y^2}{\sigma_X^2} \approx 2BT \quad \text{when } BT \gg 1 \quad \blacksquare$$

■ **Example 6.9.2** The noise at the output of “matched filters” in communication systems will have the form

$$Y = \int_0^T N(t)s(t) dt$$

where $N(t)$ is zero-mean white Gaussian noise with a PSD of $\eta/2$, and $s(t)$ is a deterministic signal which is zero outside the time interval $[0, T]$. Find the mean and variance of Y .

Solution:

$$\begin{aligned}E\{Y\} &= E\left\{ \int_0^T N(t)s(t) dt \right\} = \int_0^T E\{N(t)\}s(t) dt = 0 \\ \text{var}\{Y\} &= E\{Y^2\} = E\left\{ \int_0^T N(t_1)s(t_1) dt_1 \int_0^T N(t_2)s(t_2) dt_2 \right\} \\ &= \int_0^T \int_0^T E\{N(t_1)N(t_2)\}s(t_1)s(t_2) dt_1 dt_2 \\ &= \int_0^T \int_0^T \frac{\eta}{2} \delta(t_1 - t_2)s(t_1)s(t_2) dt_1 dt_2 \\ &= \int_0^T \frac{\eta}{2} s(t_1)s(t_1) dt_1 \\ &= \frac{\eta}{2} \int_0^T s^2(t_1) dt_1 = \frac{\eta}{2} E_s\end{aligned}$$

where E_s is the energy in the signal $s(t)$. ■

6.9.4. Response of Nonlinear and Time-Varying Systems

6.9.4.1. Nonlinear Systems

Unlike the case of LTIV systems, it is difficult to compute the response of nonlinear and time-varying systems in closed analytical form. Simulations play an important role here.

In the case of nonlinear systems we can use the simulation approach to compute the response as follows:

1. Generate a set of sampled values of the input process.
2. Generate a set of sampled values of the output processes using the transfer characteristic of the nonlinearity, which is assumed to be given.
3. Estimate the statistical properties of the output process from the sampled values generated via simulation.

(Ergodicity is normally assumed so that the output process can be characterized from a single member function.)

Methods for generating sampled values of random processes are given in Section 6.10 of this chapter. Estimation techniques are discussed in Chapters 10 and 11.

6.9.4.2. Time-Varying Systems

While most of the functional blocks in communication systems can be modeled as stationary, there are some blocks that have time-varying transfer characteristics. An example of this is a fading (multipath) radio channel whose input–output relationship is modeled as

$$\tilde{Y}(t) = \sum_{n=1}^m \tilde{W}_n(t) \tilde{X}(t - \tau_n) \quad (6.9.8)$$

where $\tilde{X}(t)$ is the complex-valued input, τ_n is the delay of the n th multipath, and $\tilde{W}_n(t)$ is the gain of the n th path. $\tilde{W}_n(t)$ are assumed to be independent, zero-mean, complex-valued Gaussian processes.

If $\tilde{W}_n(t)$ is assumed to be slowly varying compared to the signal (slow fading) then (6.9.8) can be reduced to a transfer function and results of Section 6.9.1 can be used to derive the input–output relationships. If $\tilde{W}_n(t)$ changes at about the same rate as $\tilde{X}(t)$ (fast fading), then analytical results are difficult to derive and simulation techniques are used to obtain the statistical properties of the output signal.

6.10. Sampling of Stationary Random Processes

Analog signals are converted to digital signals using sampling and quantization operations. The sampling operation converts a continuous-time signal into a discrete-time sequence and quantizing converts a continuous-amplitude signal into a discrete-amplitude signal. Since digital simulation is a discrete-time operation and might also involve finite-precision arithmetic, it is important to understand the implications of sampling and quantization.

6.10.1 Sampling

6.10.1.1. Sampling of Lowpass Random Processes

The sampling theorem for deterministic signals discussed in Section 3.5 states that a deterministic lowpass signal $x(t)$ with a bandwidth B can be reconstructed from sampled

values $\{x(kT_s)\}$ if the sampling rate is greater than $2B$ (i.e., $T_s < 1/2B$). Similarly, a lowpass random process $X(t)$ can be represented using the sampled values $X(kT_s)$ as

$$X(t) \approx X_N(t) = 2BT_s \sum_{n=-N}^N X(nT_s) \text{sinc}[2B(t - nT_s)] \quad (6.10.1)$$

as long as $T_s < 1/2B$. It can be shown that⁽²⁾

$$\lim_{N \rightarrow \infty} E\{[X(t) - X_N(t)]^2\} = 0$$

That is, $X(t)$ can be reconstructed from $X(kT_s)$ with a mean-squared error of zero.

If the signal bandwidth is large and the sampling rate f_s is not high enough, then we have aliasing.

6.10.1.2. Aliasing Effect

To examine the aliasing effect, let us define the sampling operation as

$$X_s(t) = X(t) \cdot S(t) \quad (6.10.2)$$

where $X_s(t)$ is the sampled version of a lowpass process $X(t)$, and $S(t)$ is the sampling waveform. Assume that the sampling waveform $S(t)$ is an impulse sequence (see Figure 6.21b) of the form

$$S(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s - D) \quad (6.10.3)$$

where D is a random variable with a uniform distribution in the interval $[0, T_s]$, and D is independent of $X(t)$. The product $X_s(t) = X(t) \cdot S(t)$ is the sampled version of $X(t)$ as shown in Figure 6.21c.

The sampled waveform $X_s(t)$ can also be written as

$$X_s(t) = \sum_{k=-\infty}^{\infty} X(t - kT_s - D) \delta(t - kT_s - D) \quad (6.10.4)$$

and it can be shown⁽²⁾ that the power spectral density function $S_{X_s X_s}(f)$ of $X_s(t)$ is given by

$$\begin{aligned} S_{X_s X_s}(f) &= \frac{1}{T_s^2} \left[\sum_{k=-\infty}^{\infty} S_{XX}(f - kf_s) \right] \\ &= \frac{1}{T_s^2} [S_{XX}(f) + S_{XX}(f - f_s) + S_{XX}(f + f_s) \\ &\quad + S_{XX}(f - 2f_s) + S_{XX}(f + 2f_s) + \cdots] \end{aligned} \quad (6.10.5)$$

where $S_{XX}(f)$ is the PSD of $X(t)$.

The preceding equation shows that the PSD of the sampled version $X_s(t)$ of $X(t)$ consists of replicates of the original spectrum $S_{XX}(f)$ with a replication rate of f_s . For a lowpass

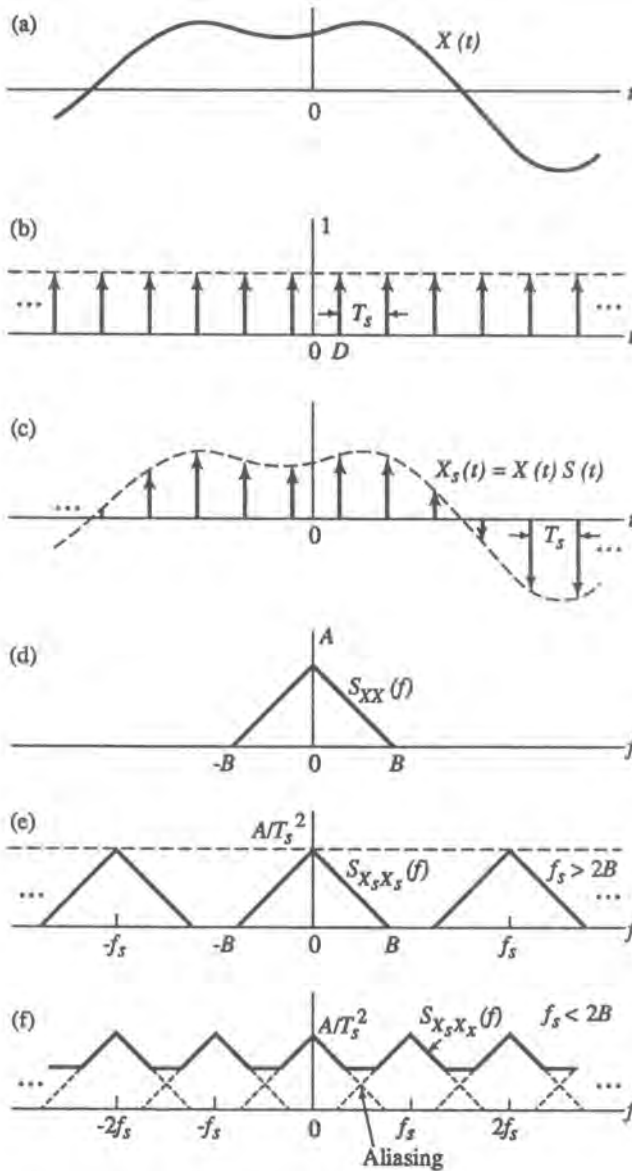


Figure 6.21. Sampling operation. (a) Continuous waveform $X(t)$; (b) sampling waveform $S(t)$; (c) sampled version of $X(t)$; (d) psd of $X(t)$; (e) psd of sampled waveform, $f_s > 2B$; (f) psd of sampled waveform, $f_s < 2B$. (From K. Sam Shanmugan and A. M. Breipohl, *Random Data: Detection, Estimation and Data Analysis*, © John Wiley and Sons, 1988, reproduced with permission.)

process $X(t)$, the PSD of $X(t)$ and $X_s(t)$ are shown in Figure 6.21d–f for two sampling rates, $f_s > 2B$ and $f_s < 2B$.

When $f_s > 2B$ or $T_s < 1/(2B)$, $S_{X_s X_s}(f)$ contains the original spectrum of $X(t)$ intact and recovery of $X(t)$ from $X_s(t)$ is possible. But when $f_s < 2B$, replicates of $S_{XX}(f)$ overlap and the PSD of $X_s(t)$ does not bear much resemblance to the PSD of $X(t)$. This is called the aliasing effect, and it prevents us from reconstructing $X(t)$ from $X_s(t)$ without error.

If $f_s > 2B$, then $X(t)$ can be reconstructed in the time domain from samples of $X(t)$ according to (6.11.2). Examination of Figure 6.20e shows that if we select only that portion of $S_{X_s X_s}(f)$ that lies in the interval $[-B, B]$, we can recover the PSD of $X(t)$. That is, $X(t)$ can be recovered from $X_s(t)$ by lowpass filtering.

6.10.1.3. Sampling Rate for Simulations

Since we use sampled values of signals for simulation, and since $f_s > 2B$ to avoid aliasing, the highest signal frequency content that can be simulated is $f_s/2$. This establishes a “simulation bandwidth” of $f_s/2$. While the Nyquist rate of $2B$ is the minimum sampling rate in an ideal system, we need to use a much higher sampling rate for simulations. The higher sampling rate is needed not only for faithful representation of time-domain waveforms, but also for accurately modeling filters (i.e., to minimize frequency warping, etc.), nonlinearities, and other devices for simulation purposes.

The sampling rate for simulations can vary from 4 to 16 times the bandwidth. To illustrate the deleterious effects of lower sampling rates, consider the sampling of a random binary waveform with a period of 1 s at a sampling rate of 5 samples/s (Figure 6.22). The aliasing effect is shown in Figure 6.22b as “aliasing noise.” We define the signal power S to aliasing noise power N_A ratio as

$$\frac{S}{N_A} = \frac{\text{Signal power in } [-f_s/2, f_s/2]}{\text{Aliased power in } [-f_s/2, f_s/2]}$$

The aliased power is the shaded area in Figure 6.22b. Approximate values of S/N_A are shown in Table 6.3 as a function of sampling rate.

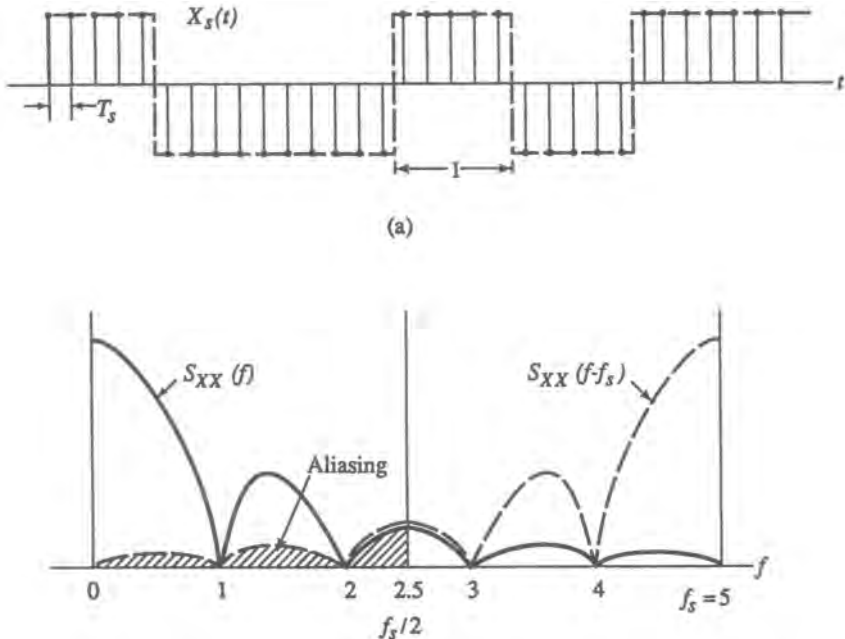


Figure 6.22. Sampling of random binary waveform, (a) Sampled waveform; (b) Aliasing effect.

Table 6.3 In-Band Signal Power/Aliased Power (S/N_A) for Random Binary Waveform and Gaussian Spectrum

Number of samples/bit	S/N_A (dB)
Random binary waveform	
4	15.8
8	18.7
10	19.8
16	21.9
20	22.9
Gaussian Spectrum	
Rms bandwidth = $\sigma_f = 1$	
4	13.4
8	42.0
10	62.4
12	87.0
14	116.0

Since baseband binary systems operate with a true S/N of the order of 10 dB, we need to choose a sampling rate such that S/N_A is $\gg 10$ dB so that aliasing error does not affect the simulated performance of such systems. This requires a sampling rate of 8–16 times the bit rate (or “bandwidth”). Note that the S/N values given in Table 6.3 are rather pessimistic. Typically, when signals are filtered prior to sampling, the aliasing effect will be much smaller. An example for Gaussian-shaped psd is also shown in Table 6.3. For this case it can be seen that the aliasing power drops significantly and faster as the sampling rate is increased.

6.10.1.4. Sampling of Bandpass Random Process

If the process to be sampled is bandpass, then the minimum sampling rate f_s will be a function of the center frequency f_0 of the PSD and the bandwidth B . The relationship between the minimum sampling rate f_s , the lowest frequency $f_0 - B/2$, and the bandwidth B is shown in Figure 6.23. Note that when $f_0 \gg B$, the minimum sampling rate is $2B$.

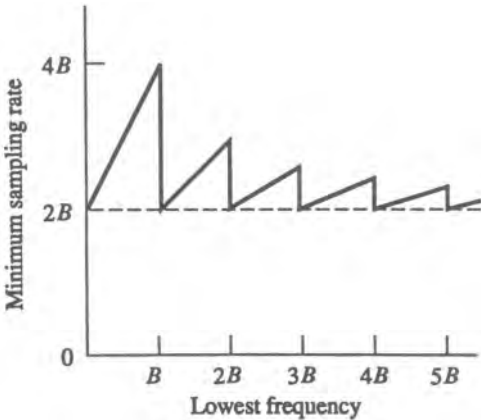


Figure 6.23. Sampling of bandpass signals; bandwidth is B , lowest frequency is $f_0 - B/2$.

If we use the complex envelope representation for a bandpass process of the form

$$X_{\text{BP}}(t) = \text{Real part of } \{\tilde{X}_{\text{LPF}}(t) \exp(2\pi f_0 t)\}$$

for simulation, then we are interested only in the complex lowpass envelope $\tilde{X}_{\text{LPF}}(t)$, which has a one-sided bandwidth of $B/2$, and it needs to be sampled at a minimum sampling rate of B , i.e., B complex-valued samples/s for the lowpass equivalent, which is the same as $2B$ real-valued samples/s.

6.10.2. Quantization

The instantaneous value of a continuous-amplitude (analog) random process $X(t)$ is a continuous random variable. If the instantaneous values are to be transmitted and processed digitally, then the continuous random variable X , which can have an uncountably infinite number of possible values, has to be represented by a discrete random variable with a finite number of values. For example, if the instantaneous value is sampled by a 4-bit analog-to-digital converter, then X is approximated at the output by a discrete random variable with one of 2^4 possible values.

The quantizer takes a continuous random variable X and converts it into a discrete random variable X_q according to some predetermined rule:

$$\begin{aligned} X_q &= m_i & \text{if } x_{i-1} < X \leq x_i \\ x_0 &= -\infty, & x_Q = +\infty \end{aligned}$$

$m_1, m_2, \dots, m_Q$ are the Q output levels, and $x_0, \dots, x_Q$ are the endpoints of the quantizing intervals. The x_i and m_i and Q are chosen to minimize a performance criterion such as the normalized mean square error

$$\frac{N_Q}{S_Q} = \frac{E\{[X - X_q]^2\}}{E\{[X_q]^2\}}$$

6.10.2.1. Uniform Quantization

The simplest quantizing algorithm is the uniform quantizing algorithm. In this method, the range of X is divided into Q intervals of length Δ . If the value of X falls in the i th interval, then the quantized value is taken to be the mean (or midpoint) of the interval (Figure 6.24).

The uniform quantizing rule is optimum (i.e., minimizes N_Q/S_Q for a given number of levels Q) when X has a uniform pdf. In this case it can be shown that⁽²⁾

$$\frac{N_Q}{S_Q} \approx \frac{1}{Q^2} \quad (6.10.6a)$$

and

$$\left(\frac{S_Q}{N_Q} \right)_{\text{db}} = 10 \log_{10} Q^2 \approx 6n \quad (6.10.6b)$$

where $n = \log_2 Q$. Thus, if we use an n -bit A/D, converter, $Q = 2^n$, then the signal to quantizing noise power ratio is $6n$. For faithful reproduction of analog signals, this ratio has to

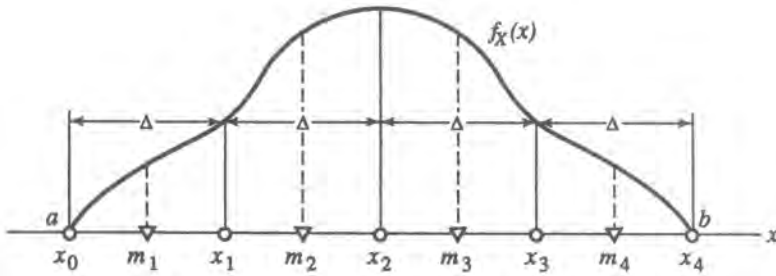


Figure 6.24. Example of uniform quantizing. Step size is Δ , $Q = 4$. (From K. Sam Shanmugan and A. M. Breipohl, *Random Data: Detection, Estimation and Data Analysis*, © John Wiley and Sons, 1988, reproduced with permission.)

be > 30 dB, which requires $n > 5$ bits. (Note: 6- to-10 bit A/D converters are used to digitize voice and video.)

When simulations are done on general-purpose computers with a word size > 32 bits, digital representation of analog (real-valued) variables introduces very little error. However, quantization may arise in modeling of real hardware, and the roundoff error can be significant in long simulations involving the cumulative processing of several million samples.

6.10.2.2. Nonuniform Quantizer

If X has a nonuniform pdf, then it is obvious that the quantizer step size should be a variable, with smaller step sizes near the mode of the pdf and larger step sizes near the tails in order to minimize the quantizing error. An example of nonuniform quantizing is shown in Figure 6.25.

The endpoints and output levels of a nonuniform quantizer are chosen to minimize

$$N_Q = \sum_{i=1}^Q \int_{x_{i-1}}^{x_i} (x - m_i)^2 f_X(x) dx, \quad x_0 = -\infty, \quad x_Q = \infty \quad (6.10.7)$$

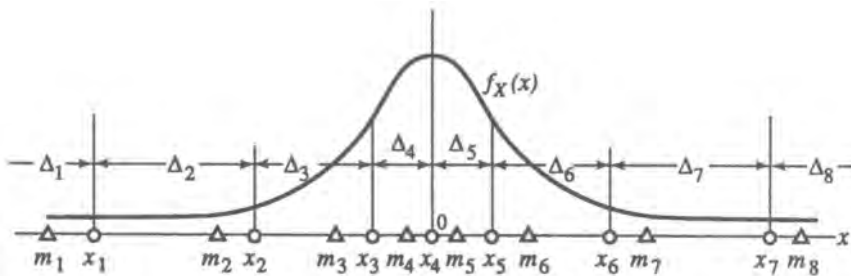


Figure 6.25. A nonuniform quantizer for a Gaussian variable. $X_0 = -\infty$, $X_Q = \infty$, $Q = 8$, and $\Delta_i = \Delta_{Q+1-i}$ ($i = 1, 2, 3, 4$). (From K. Sam Shanmugan and A. M. Breipohl, *Random Data: Detection, Estimation and Data Analysis*, © John Wiley and Sons, 1988, reproduced with permission.)

Since we wish to minimize N_Q for a fixed Q , we get the necessary conditions by differentiating N_Q with respect to the x_j and m_j and setting the derivatives equal to zero:

$$\frac{\partial N_Q}{\partial x_j} = (x_j - m_j)^2 f_X(x_j) - (x_j - m_{j+1})^2 f_X(x_j) = 0, \quad j = 1, 2, \dots, Q-1 \quad (6.10.8a)$$

$$\frac{\partial N_Q}{\partial m_j} = -2 \int_{x_{j-1}}^{x_j} (x - m_j) f_X(x) dx = 0, \quad j = 1, 2, \dots, Q \quad (6.10.8b)$$

These equations can be solved for the x_j and m_j .

The endpoints and the output levels for a Gaussian random variable with zero mean and unit variance can be found in Ref. 23. In this case, S_Q/N_Q can be approximated by

$$\frac{S_Q}{N_Q} \approx \frac{Q^{1.96}}{2.2} \quad \text{when } Q \gg 1 \quad (6.10.9a)$$

$$= (6n - 3.2) \text{ db} \quad (6.10.9b)$$

In practice, nonuniform quantizers are implemented using the combination of a uniform quantizer preceded by a compressing operator and followed by a complementary expanding operator (see Ref. 15 for details).

6.11. Summary

Random variables and random processes are used to model signals and noise encountered in communication systems and for modeling the randomly time-varying behavior of some types of communication channels and other components. A basic understanding of the concepts and the theory underlying random variables and random processes is necessary for developing simulation models and performance estimation techniques for communication systems.

The focus of this chapter was on the basic concepts and properties of random variables and random processes, and models which will be used for simulating communication systems. We also presented analytical and simulation techniques for computing the response of systems driven by random inputs. Bounds and approximations which can be used to simplify simulation procedures and validate simulation procedures were also discussed.

Material presented in this chapter will be used in the following chapters to:

1. Develop techniques for generating sequences of random numbers which will be used as stimuli to drive the simulation models.
2. Develop procedures for estimating performance measures.
3. Aid in simplifying, analyzing, and validating simulation results.

References

1. A. Papoulis, *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, New York (1984).
2. K. Sam Shanmugan and A. M. Breipohl, *Random Signals*, Wiley, New York (1988).
3. H. L. Larson and B. O. Shubert, *Probabilistic Models in Engineering Sciences*, Vols. I and II, Wiley, New York (1979).

4. R. K. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*, Macmillan, New York (1978).
5. A. Leon-Garcia, *Probability and Random Processes*, Addison-Wesley, Reading, Massachusetts (1988).
6. W. A. Gardner, *Introduction to Random Processes*, Macmillan, New York (1988).
7. M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, Vols. 1 and 2, Hafner, New York (1973).
8. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover, New York (1965).
9. P. O. Börjesson and C. E. W. Sundberg, Simple approximation of the error function $Q(X)$ for communication applications, *IEEE Trans. Commun.* **COM-27**, 639–643 (1979).
10. S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
11. K. S. Shanmugan, *Digital and Analog Communication Systems*, Wiley, New York (1979).
12. W. Turin, *Performance Analysis of Digital Communication Systems*, Computer Science Press, New York (1990).
13. N. Mohanty, *Random Signals, Estimation and Identification*, Van Nostrand and Reinhold, New York (1985).
14. J. K. Holmes, *Coherent Spread Spectrum Systems*, Wiley, New York (1982).
15. I. Korn, *Digital Communications*, Van Nostrand and Reinhold, New York (1985).
16. J. W. Modestino and K. Matis, Interactive simulation of digital communication systems, *IEEE J. Select. Areas Commun.* **SAC-2**, 51–62 (1984).
17. J. W. Modestino and B. Sankur, Modeling and simulation of ELF/VLF noise, in *Proc. 7th Annual Conference on Modeling & Simulation*, Pittsburgh, Pennsylvania (April 1986).
18. J. W. Modestino and B. Sankur, Analysis and modeling of impulsive noise, *Arch. Elek Übertragung*, **35**(12), 481–488 (1981).
19. R. E. Ziemer, Error probabilities due to additive combinations of Gaussian and impulsive noise, *IEEE Trans. Commun.* **COM-15**(3), 471–74 (1967).
20. S. Oshita and K. Feher, Performance of coherent PSK and DPSK systems in an impulsive and Gaussian noise environment, *IEEE Trans. Commun.* **COM-30**(12), 2540–2546 (1982).
21. B. Sankur and J. W. Modestino, Performance of receivers in impulsive noise, *Arch. Elek Übertragung*, **36**(3), 111–118 (1982).
22. R. D. Shelton and A. F. Adkins, Noise bandwidth of filters, *IEEE Trans. Commun. Technol.* **COM-18**, 827–830 (1970).
23. J. Max, Quantizing for minimum distortion, *IEEE Trans. Inf. Theory* **IT-6**, 7–12 (1960).
24. G. H. Golub and G. H. Welsh, Calculation of Gauss quadrature rules, *Math. Comput.* **23**, 221–230 (1969).

Monte Carlo Simulation and Generation of Random Numbers

Simulation of a communication system requires the generation of sampled values of all the input waveforms, processing these samples through models of functional blocks in the system, and estimating system performance from the output samples at various points in the model. Chapters 2–5 dealt with the principles of modeling functional blocks in communication systems and Chapter 6 covered random process models used to characterize signal waveforms, noise, and interference that are the inputs in communication systems. The focus of this chapter is on how to generate sampled values of these input waveforms to *drive* the simulation model. We begin this chapter with a discussion of the *Monte Carlo* technique, which is the basis for simulating systems that are driven by input signals that are modeled as random processes. The rest of the chapter is devoted to algorithms for generating random sequences which represent sampled values of the random input signals. Estimation of performance metrics using Monte Carlo (MC) simulation is discussed in Chapters 10 and 11.

7.1. Principle of Monte Carlo Simulation

7.1.1. Definition of Monte Carlo Simulation

Mechanical random number generators were first used to simulate games of chance, and simulation techniques using random number generators bear the name of the city of Monte Carlo, the home of the world-famous casino. While there are many variations of the Monte Carlo technique, it basically involves the simulation of a random experiment using artificial means, i.e., without fully repeating the underlying physical experiment. In the context of, say, estimating the bit error rate in a communication system, we can define Monte Carlo simulation as follows. With respect to Figure 7.1, we are given a model of a communication system and a description of the input signals $U(t)$, $V(t)$, and $W(t)$, which are assumed to be random processes. Our objective is to find the statistical properties of $Y(t)$ or the expected value of some function $g(Y(t))$ of $Y(t)$. If we solve this problem by emulating the system including the time evolution of *all* the waveforms in the system, then we have a *pure* Monte Carlo simulation. This implies generating sampled values of all the input processes, letting the models of functional blocks in the communication system operate on them, and observing the output waveforms. Ideally, the Monte Carlo simulation is a one-to-one correspondence

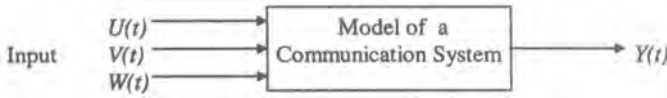


Figure 7.1. Definition of Monte Carlo simulation.

with the real system within the limits of modeling assumptions and approximations. The expected value $E\{g(Y(t))\}$ is estimated from MC simulations according to

$$E\{g(\hat{Y}(t))\} = \frac{1}{N} \sum_{i=1}^N g(Y(i))$$

where the caret indicates estimated value and N is the number of samples simulated.

In the context of the specific example shown in Figure 7.2, Monte Carlo simulation for estimating the bit error rate in a digital communication system involves the following steps.

1. Generate sampled values of the input bit sequence $\{A(k)\}$, $k = 1, 2, \dots, N$, and the noise samples $\{N(j)\}$, $j = 1, 2, \dots, mN$ (the sampling rate is m samples per bit).
2. Process these samples through the models of the functional blocks and generate an output sequence $Y(k)$.
3. Estimate $E(g(Y(k)))$ as

$$\hat{P}_e = \frac{1}{N} \sum_{k=1}^N g(Y(k))$$

where $g(Y(k)) = 1$ if $Y(k) \neq A(k)$ and $g(Y(k)) = 0$ if $Y(k) = A(k)$ (this step is equivalent to counting errors).

The accuracy of estimates obtained via MC simulation will depend on the estimation procedure used, sample size N , the ability to reproduce sampled values of the input processes accurately, and modeling assumptions and approximations. Estimation procedures and analysis of the properties of the estimators will be discussed in the following chapter. In general, the accuracy will be proportional to $1/\sqrt{N}$, which means a fairly large number of

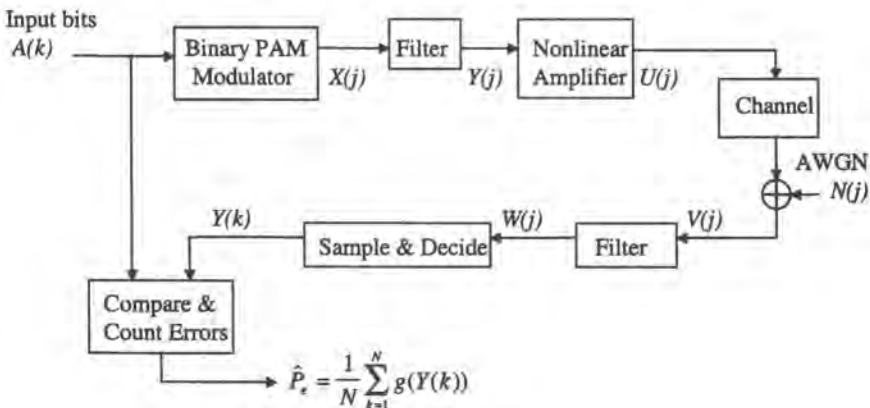


Figure 7.2. Simulation model of a communication system.

samples will have to be simulated in order to obtain accurate estimates via MC simulations. While the MC technique is general and can be applied to a broad range of problems, the large-sample-size requirements and hence the length of simulation are often limiting factors.

7.1.2. Variations of Monte Carlo Simulation—Quasianalytical Monte Carlo Simulation

Not all simulations are Monte Carlo or *pure* Monte Carlo simulations. Monte Carlo implies that at least *one* random process is emulated and *pure* Monte Carlo implies *all* the input processes are emulated. In many simulation applications it is not necessary to simulate all the random processes present in the system. In the example shown in Figure 7.2 the error rate performance of the system will depend on the noise and the cumulative distortion introduced by the nonlinearity and the filters. Since the noise is Gaussian and additive at the input, its net effect at the output of the filter and hence on the decision metric will also be additive and Gaussian. In the absence of distortion, the effect of additive Gaussian noise can be handled analytically without simulation. Even with distortion, the effects of AWGN can still be characterized analytically for a given value of distortion. However, the distribution of the distortion values introduced by the nonlinearity and filters may be difficult to characterize analytically, but easy to simulate. In such a case, we only need to simulate the cumulative effect of all the functional blocks on the *input* binary waveform and there is no need to explicitly simulate the noise waveform.

MC simulations in which only *some* (but not all) input processes into the system are simulated explicitly while the effects of other processes are handled using analytical techniques are called partial MC or quasianalytical (QA) simulations. The main advantage of a QA simulation is that it will require the simulation of fewer samples than a pure MC simulation to produce estimates with the same accuracy. QA simulations are particularly useful for linear systems and other cases discussed in Chapters 10 and 11.

7.2. Random Number Generation

Whether we use pure Monte Carlo or partial Monte Carlo, the key to implementing a Monte Carlo simulation is the generation of sequences of random numbers which represent the sampled values of the input random processes. From a modeling perspective, this requires an understanding of the principles of random variables and random processes discussed in the preceding chapter. Implementation of MC simulations, on the other hand, requires algorithms for generating sequences of random numbers. The overall accuracy of the simulation results will depend on how good these algorithms are in faithfully reproducing the statistical properties of the random processes they are supposed to mimic.

In modeling and simulation of random processes we often make a fundamental assumption: that the underlying random processes are ergodic. Ergodicity assumption is the key for applying simulation results obtained from one sample function of a process to the entire ensemble of the process. Ergodicity implies both wide-sense and strict-sense stationarity, which means that the important properties of the process are captured by the first-order and second-order distributions and the properties of the process are time-invariant. This not only simplifies the models for the random processes, but it also makes it easier to construct algorithms for generating random sequences that can accurately capture the essential prop-

erties of the underlying random processes such as first-order distributions, means, variances, and autocorrelation functions (and hence the power spectral densities).

From a simulation point of view all random processes must be represented by sequences of random variables and hence we require methods of generating random numbers from a wide variety of distributions and with arbitrary autocorrelation (power spectral density) functions. The remainder of this chapter is devoted to the development of algorithms for random number generation.

The starting point for random number generators is a recursive algorithm for generating an independent sequence of uniform random numbers. By applying an appropriate memoryless nonlinear transformation, we can transform this sequence to another independent sequence with an arbitrary first-order distribution. An independent sequence can also be transformed into a sequence with arbitrary correlation and power spectral density by applying a linear or nonlinear transformation with memory. In this section, we discuss algorithms for generating uniform random numbers and also derive the transformations that can be applied to convert the output of a uniform random number generator to a sequence with arbitrary distribution and correlation function. We also develop computationally efficient algorithms for generating binary and M -ary sequences which represent input symbol sequences in digital communication systems

Since Monte Carlo simulations involve the generation and processing of large numbers of samples, computational efficiency of random number generators is extremely important. On the other hand, since the overall accuracy of simulation results also depends on the “goodness” of the random number generators, we have to balance the efficiency considerations versus the accuracy requirements. Tests for verifying the goodness of random number generators are also presented in this chapter.

7.2.1. Generation of Uniform Random Numbers

Uniform random numbers are generated using recursive formulas that are computationally very efficient. One such method, called the congruential or the power residue method,⁽¹⁾ uses the recursive formula

$$X(k) = [aX(k-1) + c] \bmod M \quad (7.2.1)$$

where M is the modulus, $M > 0$, a large (prime) integer value; a is the multiplier, $0 < a < M$; c is the increment, usually = 1 or 0; and $X(0)$ is the starting value or the seed $0 < X(0) < M$. The algorithm in Equation (7.2.1) is executed in integer arithmetic. The seed value $X(0)$ is provided by the user and is the only thing that is random about the recursion given in Equation (7.2.1).

Equation (7.2.1) produces a random sequence of integer values in the range $0 \leq X(k) \leq M-1$ and this sequence can be converted into a random sequence of uniform random numbers in the interval $[0, 1]$ by executing the floating-point operation

$$U(k) = \text{Float}[X(k)/M] \quad (7.2.2)$$

The output of Equation (7.2.1) consists of integers that are a subset of $[0, M-1]$, and they correspond to the state sequence of a finite-state machine with a maximum number of M states. Hence, the maximum period of the output sequence is M .

The statistical properties of the output of the random number generator given in (7.2.1), and its period will depend on the values of a , c , and M , and a good random generator is designed by careful choice of these parameters. These properties may also be affected by the seed value which is chosen by the user. The statistical properties of a good random number generator should not be affected by the seed value; it simply specifies the starting point of the periodic output sequence. However, the seed will affect the properties of sub sequences whose length is much shorter than the period.

There are two aspects of the output of the random number generator that are very important: (1) algebraic (temporal) properties such as the structure and period, and (2) statistical properties such as the distribution of the output. Good values of a , c , and M should lead to good algebraic and statistical properties, and also maximize the computational efficiency and compatibility with the arithmetic used on a particular machine.

Among the structural properties, it is important that the output sequence be uncorrelated and have the maximum period. The period should be longer than the simulation length, otherwise there is the danger that part of the simulation will be replicated and the accuracy of the simulations will not improve beyond the point where replication begins. While the maximum period is M , an arbitrary choice of a , c , and M does not guarantee the maximum period. Correlation between the output values is not desirable since estimators based on positively correlated samples will have larger variances than estimators obtained using uncorrelated samples. There are some guidelines for choosing constants that lead to maximum period and uncorrelated output⁽¹⁻⁴⁾.

The most commonly used uniform random number generators are as follows:

- $X(k) = 16,807X(k-1)(\text{Mod}2^{31} - 1)$ for 32-bit machines with $(31+1)$ -sign bit; **period** = $2^{31} - 2$.
- $X(k) = [69,069X(k-1) + 1](\text{Mod}2^{32})$ for 32-bit machines; **period** = 2^{32} .

Random number generators with longer periods can be constructed using variations of the linear congruential algorithm. For example, it is possible to guarantee a maximum period by using a recursive algorithm of the form

$$X(k) = [a_1X(k-1) + \dots + a_mX(k-m)] \bmod p \quad (7.2.3)$$

where $(a_1, a_2, \dots, a_m)$ are the coefficients of a primitive polynomial

$$f(x) = x^m - a_1x^{m-1} - \dots - a_m$$

over the Galois field $\text{GF}(p)$. The period of the sequence produced by Equation (7.2.3) is $p^m - 1$. Comparing Equations (7.2.1) and (7.2.3), it is clear that the algorithm given in Equation (7.2.3) requires more computations per each output sample (with $p = 2$, this algorithm is more commonly used for generating binary sequences as described in the following section).

Two other methods for generating uniform random sequences with very long periods are described below. These algorithms are computationally efficient and produce sequences with very long periods and very good statistical properties.

7.2.1.1. Wichman–Hill Algorithm

This approach to generating sequences with a longer period involves linearly combining the outputs of different random number generators with shorter periods (sequences with shorter periods can be observed and tested well over an entire period). If we sum two periodic waveforms with periods N_1 and N_2 , the resulting waveform will have a period

$$N = \text{lcm}(N_1, N_2)$$

If N_1 and N_2 are relatively prime with respect to each other, then

$$N = N_1 \times N_2$$

Thus, by combining the outputs of several random number generators, we can produce a sequence with a much longer period.

The Wichman–Hill algorithm⁽⁵⁾ is based on this principle and it combines the outputs of three random number generators according to

$$\begin{aligned} X(n) &= 171X(n-1) \bmod 30,269 \\ Y(n) &= 172X(n-1) \bmod 30,307 \\ Z(n) &= 170X(n-1) \bmod 30,323 \end{aligned} \quad (7.2.4a)$$

and

$$U(n) = \left\{ \frac{X(n)}{30,269} + \frac{Y(n)}{30,307} + \frac{Z(n)}{30,323} \right\} \bmod 1 \quad (7.2.4b)$$

The first three steps in the algorithm are implemented using integer arithmetic and the last step is done in floating point. The period of the sequence produced by this algorithm is $p = 30,268 \times 30,306 \times 30,322 \approx 7 \times 10^{12}$. This algorithm is machine independent as long as the integer arithmetic can handle integers up to 5,212,632 without roundoff and the output has a long period and good randomness properties. It is of course possible to extend the period by combining the output of more random number generators, but this leads to an increase in the computational load.

Note that the Wichman–Hill algorithm is computationally more complex than the algorithm given in Equation (7.2.1). Its main advantage is that it is portable (machine-independent), and it guarantees maximum period by construction and there is no need to verify its period empirically, which is very difficult to do when period is very long.

7.2.1.2. Marsaglia–Zaman Algorithm

This algorithm, introduced by Marsaglia and Zaman,⁽⁶⁾ is a linear recursive algorithm and it employs longer memory and a *carry* or *borrow* operation in the recursion. There are

two similar versions of this algorithm, the *subtract-with-borrow* and the *add-with-carry* algorithm. We describe below the subtract-with-borrow algorithm, which has the form

$$\begin{aligned} Z_i &= X_{i-r} - X_{i-s} - C_{i-1} \\ X_i &= \begin{cases} Z_i & \text{if } Z_i \geq 0 \\ Z_i + b & \text{if } Z_i < 0 \end{cases} \\ C_i &= \begin{cases} 0 & \text{if } Z_i \geq 0 \\ 1 & \text{if } Z_i < 0 \end{cases} \end{aligned}$$

In the preceding algorithm all entries are positive integers and $i > r$. The borrow bit C_i is initially set to zero and is toggled between 0 and 1 depending on whether the recursion produces a positive or negative integer Z_i .

The parameters that define this generator include a base b and lags r and s . It is shown by Marsaglia and Zaman that, in order to guarantee maximum period $M - 1$, the constants b , r , and s must be chosen such that $M = b^r - b^s + 1$ is a prime with b a primitive root mod M . For 32-bit machines, $b = 2^{32} - 5$, $r = 43$, and $s = 22$ will produce a period

$$M - 1 \approx 1.65 \times 10^{414}$$

Another required element of this generator is the seed vector, whose length is r . In order to produce a sequence of uniform random numbers in the interval $[0, 1]$, it is necessary to transform X_i to U_i according to

$$U_i = \text{Float}(X_i/b)$$

7.2.2. Methods of Generating Random Numbers from an Arbitrary pdf

7.2.2.1. Transform Method (Analytical)

By applying a simple transformation to a uniform random variable U , we can generate a random variable Z with an arbitrary pdf $f_z(z)$ as follows: Consider the transformation

$$Y = F_z(Z) \tag{7.2.5}$$

where $F_z(z)$ is the cumulative distribution function (CDF) of the random variable Z . Following the results presented in Section 6.5, it can be shown that Y has a uniform distribution in the interval $[0, 1]$.

We can use this fact to generate Z using $Z = F_z^{-1}(Y)$, where Y has a uniform pdf:

1. Generate U uniformly distributed in $[0, 1]$.
2. Output $Z = F_z^{-1}(U)$; Z has the pdf $f_z(z)$ (see Figure 7.3).

If the cdf and the inverse of the cdf of Z can be expressed in closed form, then step 2 in the transform method is easy to implement. Otherwise both $F_z(\cdot)$ and the inverse $F_z^{-1}(\cdot)$ have to be computed using numerical methods. $F_z(z)$ is constructed by numerically integrating $f_z(z)$

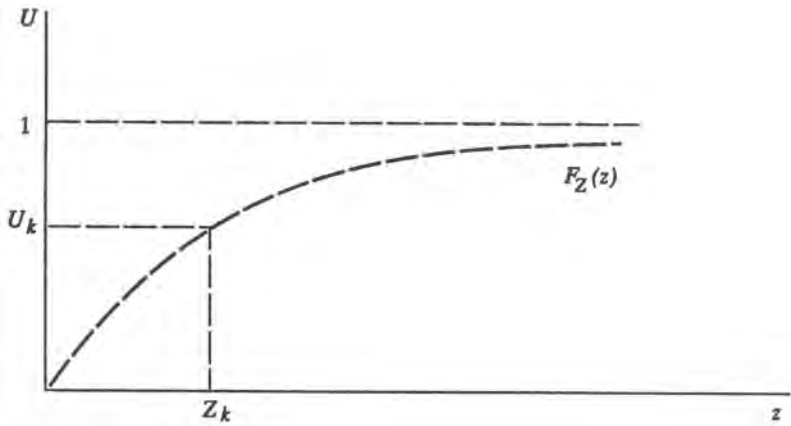


Figure 7.3. Transform method of generating random numbers.

and storing the values in a table, and the inverse $F_Z^{-1}(U)$ is found by searching and interpolating over the tabulated values of $F_Z(z)$.

■ *Example 7.2.1:*

- (a) Find an algorithm for generating an exponentially distributed random variable using the transform method.
- (b) Find an algorithm for generating a geometric distributed random variable using the transform method.

(a) *Solution:*

$$f_Z(z) = \lambda e^{-\lambda z}$$

and

$$F_Z(z) = 1 - e^{-\lambda z}$$

Hence

$$U = F_Z(Z) \quad \text{yields} \quad U = 1 - e^{-\lambda Z}$$

or

$$Z = (-1/\lambda) \ln(1 - U)$$

If U is uniform in $[0, 1]$, then it is easy to see that $1 - U$ is also uniform in $[0, 1]$. Hence, one subtraction can be saved by using

$$Z = (-1/\lambda) \ln U$$

The sequence of random numbers will not be identical, but will be statistically equivalent.

Algorithm for Generating Exponentially Distributed Random Variable:

1. Generate U .
2. $Z = -(1/\lambda) \ln(1 - U)$ or $Z = -(1/\lambda) \ln U$.

(b) *Solution:* If X is geometric, then

$$P(X = k) = \theta(1 - \theta)^{k-1}, \quad k = 1, 2, \dots$$

X is a discrete variable, and represents, e.g., the waiting time in a Bernoulli process having success probability θ .

X can be generated by a slight modification of the solution in (a). If Z is exponential, then we have

$$\begin{aligned} P[n < Z \leq n + 1] &= [1 - e^{-\lambda(n+1)}] - [1 - e^{-\lambda n}] \\ &= e^{-\lambda n}(1 - e^{-\lambda}) \\ &= P(X = n + 1), \quad \text{with } 1 - \theta = e^{-\lambda} \end{aligned}$$

Thus, we obtain a geometric variable X by generating an exponential variable Z [with $\lambda = -\ln(1 - \theta)$] and rounding to $[Z + 1]$, where $[\cdot]$ represents integer part. ■

Algorithm for Generating Geometrically Distributed Random Variable:

1. Generate U .
2. Compute $Z = \ln U / \ln(1 - \theta)$.
3. Return $X = [1 + Z]$.

Algorithms for Generating Gamma and Poisson Variables. Based on the transformation method, we can derive the following algorithms for generating gamma and Poisson random variables:

Gamma random variable X (for integer α , x is Erlang):

1. Set $X = 0$.
2. Generate V from exponential with $\lambda = 1$.
3. Set $X = X + V$.
4. If $\alpha = 1$, set $X = \beta X$, deliver X , and return to step 1. Otherwise:
5. Set $\alpha = \alpha - 1$ and return to step 2.

Poisson random variable X :

1. Set $A = 1$ and $k = 0$.
2. Generate U_k from uniform $[0, 1]$.
3. Set $A = U_k A$.
4. If $A < e^{-\lambda}$ deliver $X = k$ and return to step 1. Otherwise:
5. Set $k = k + 1$ and return to step 3.

7.2.2.2. Transform Method (Empirical)

When the inverse transform cannot be expressed in closed form, then the following empirical search algorithms can be used to implement the transform method.

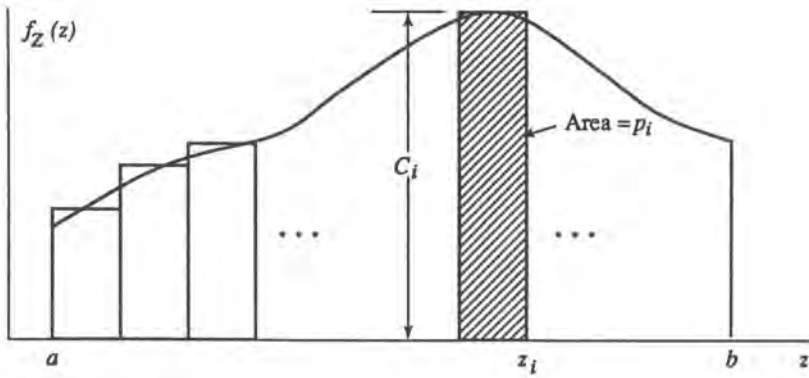


Figure 7.4. Generation of random numbers from an arbitrary distribution.

If Z is a continuous random variable, then the distribution is first quantized (see Figure 7.4). If $p_1, p_2, \dots, p_N$ are probabilities of the N cells of the uniformly quantized distribution, then the following algorithm can be used to generate samples of the random variable Z :

1. Generate U , uniform in $[0, 1]$.
2. Let $F_i = \sum_{j=1}^i p_j$, $i = 0, 1, 2, \dots, N$, with $F_0 = 0$.
3. Find the smallest value of i that satisfies

$$F_{i-1} < U \leq F_i, \quad i = 1, 2, \dots, N$$

4. Output $Z = z_{i-1} + (U - F_{i-1})/C_i$ and return to step 1.

The last step is used to obtain an interpolated value of Z in the interval $[z_{i-1}, z_i]$ (see Figure 7.4).

7.2.2.3. Transform Method for Discrete Random Variables

When Z is a discrete random variable with values $z_1, z_2, \dots$, $P(Z = z_i) = p_i$ and $F_i = \sum_{j=1}^i p_j$, and then the following algorithms can be used to generate samples of Z .

Algorithm 1 (Finite number of values of Z):

- Step 0: Set $k = 1$.
- Step 2: Generate U , uniform on $[0, 1]$.
- Step 2: If $U \leq F_k$, output $Z = z_k$ and return to step 0. Otherwise:
- Step 3: Increment $k = k + 1$ and return to step 2.

Algorithm 2 (Uncountable number of values for Z):

- Step 0: Let $C = p_1$, $B = C$, and $k = 1$.
- Step 1: Generate U , uniform on $[0, 1]$.
- Step 2: If $U \leq B(U \leq F_k)$ output $Z = z_k$ and return to step 0. Otherwise:
- Step 3: Set $k = k + 1$.
- Step 4: Set $C = A_{k+1}C(A_{k+1} = p_{k+1}/p_k)$ and $B = B + C$ and return to step 2.

As shown in Figure 7.5, we simply find the inverse of the distribution function empirically and output the right end of the interval into which U falls. There is no interpolation within the interval as in the continuous case.

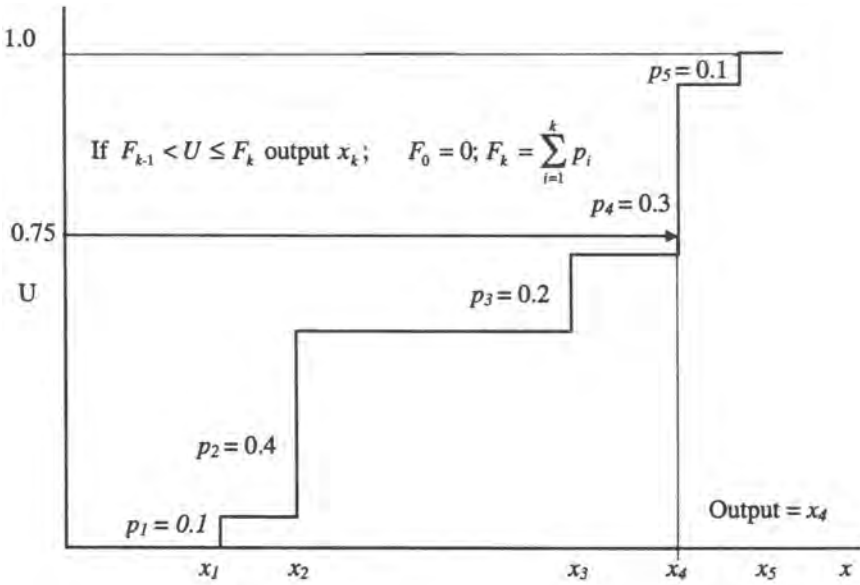


Figure 7.5. Inverse transform method for discrete distributions.

In algorithm 1, the p_i and F_i , $i = 1, 2, \dots, N$, are computed *a priori* and stored in a table and the search is confined to the N elements of the table starting from one end of the table. The search procedure can be speeded up by starting the search from the middle of the table and proceeding up or down based on the first comparison.

In algorithm 2, cumulative probabilities are computed as part of the search (it is not possible to compute p_i and F_i for all values of z_i and store them when Z has an uncountable number of values). For distributions such as Poisson, $A_{k+1}/A_k = \lambda/(k+1)$, and step 4 is easy to compute, and the whole procedure is computationally very efficient.

7.2.2.4. Acceptance/Rejection Method of Generating Random Numbers

Suppose we want to generate X according to the pdf $f_X(x)$. Consider the following algorithm for generating X (Figure 7.6):

1. Generate U_1 , uniform in $[0, a]$.
2. Generate U_2 , uniform in $[0, b]$.
3. If $U_2 < f_X(U_1)$, then output $X = U_1$; otherwise reject U_1 and return to step 1.

We can show that this algorithm outputs a variable with the pdf $f_X(x)$ as follows:

$$\begin{aligned}
 &P[x \leq U_1 < x + dx | U_1 \text{ is accepted}] \\
 &= \frac{P[(x \leq U_1 < x + dx) \cap (U_1 \text{ is accepted})]}{P[U_1 \text{ is accepted}]} \\
 &= \frac{f_X(x) dx/ab}{1/ab} = f_X(x) dx
 \end{aligned}$$

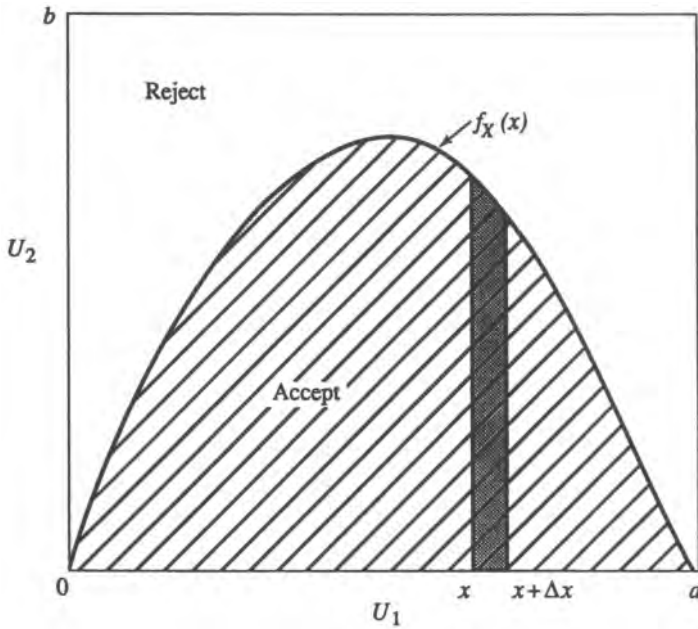


Figure 7.6. Acceptance/rejection method of generating random numbers.

The last step follows from the fact that since U_1 and U_2 are uniform, the probability of $(U_1, U_2) \in$ any region in Figure 7.6 is proportional to the area of the region divided by ab . Note that the probability that U_1 is accepted is $1/ab$ since the area under the pdf is 1.

The acceptance/rejection method can have two problems. First, if the rectangle in Figure 7.6 does not fit snugly around $f_X(x)$ (that is, $ab \gg 1$), then the number of U 's that have to be generated before we accept a value may be excessive; second, if the range of X is not finite, the method cannot be used without some modifications.

The following modification to the algorithm overcomes both problems. Suppose we want to generate a random variable with X with a pdf $f_X(x)$. Let W be a random variable with a pdf $f_W(w)$ that is “easy” to generate and such that for some $k > 1$,

$$f_X(x) \leq kf_W(x) \quad \text{for all } x$$

that is, $kf_W(x)$ is a tight upper bound on $f_X(x)$ as shown in Figure 7.7. Then the following method can be used to generate samples of X :

Modified Acceptance/Rejection Method (Figure 7.7):

1. Choose $kf_W(x)$, and find $F_W^{-1}(\cdot)$.
2. Generate U_1 and find $X = F_W^{-1}(U_1)$.
3. Generate U_2 uniform in $[0, b]$, $b = kf_W(X)$.
4. If $U_2 < a$, $a = f_X(X)$, output X , otherwise reject X and return to step 2.

The computational efficiency of this algorithm will depend on how tightly $kf_W(x)$ bounds $f_X(x)$. An example of the application of this method for generating random numbers from distributions encountered in simulating lightwave communication systems may be found in Refs. 7,8.

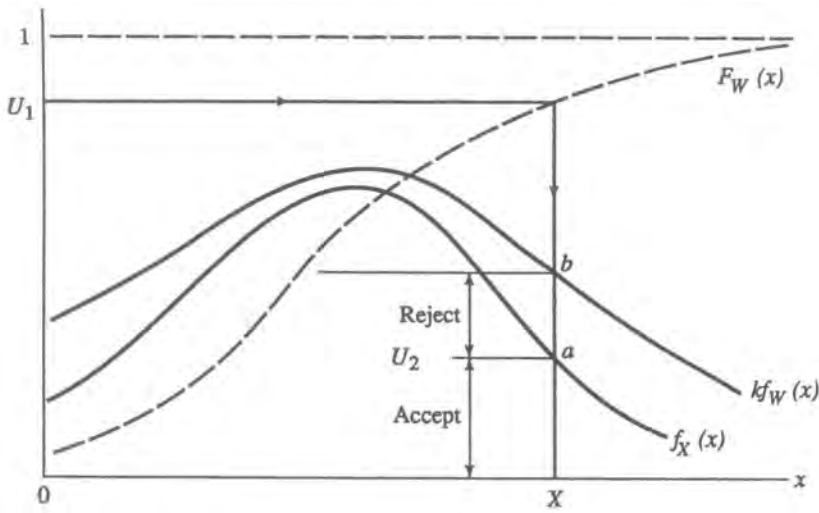


Figure 7.7. Modified form of acceptance/rejection method; U_1 uniform in $[0, 1]$; U_2 uniform in $[0, b]$.

7.2.3. Generating Gaussian Random Variables

7.2.3.1. Sum-of-12 Method

The simplest method of generating random numbers with a standard Gaussian pdf ($\mu_Y = 0$ and $\sigma_Y^2 = 1$) is via the use of the central limit theorem according to the formula

$$Y = \sum_{k=1}^{12} U(k) - 6.0 \quad (7.2.6)$$

where $U(k)$, $k = 1, 2, \dots, 12$, is a set of independent, identically distributed random variables with uniform distribution in the interval $[0, 1]$. Since $U(k)$ has a mean of 0.5 and variance of $1/12$, it is easy to verify that the right-hand side of (7.2.6) yields a mean of 0 and variance of 1.

It should be noted that whereas a Gaussian random variable has values ranging from $-\infty$ to ∞ , Equation (7.2.6) produces values of Y in the interval $[-6.0, 6.0]$. The number 12 is “traditional” and represents some compromise between speed and “accuracy,” but there is no reason to limit k to 12.

7.2.3.2. Box-Muller Method

It is well known that if X and Y are independent zero-mean Gaussian variables, then $R = (X^2 + Y^2)^{1/2}$ and $\theta = \tan^{-1}\{Y/X\}$ have Rayleigh and uniform pdfs, respectively. This fact can be used to generate two samples of a Gaussian variable by transforming a pair of Rayleigh and uniform variables as follows: If U_1 and U_2 are two independent variables uniformly distributed in the unit interval, then

$$X = [-2 \ln(U_1)]^{1/2} \cos(2\pi U_2) \quad (7.2.7a)$$

and

$$Y = [-2 \ln(U_1)]^{1/2} \sin(2\pi U_2) \quad (7.2.7b)$$

are independent Gaussian random variables with mean zero and variance 1.⁽¹⁾ This algorithm is known as the Box–Müller method.

Note that (7.2.7) produces numbers that are distributed in the interval $(-\infty, \infty)$. However, this algorithm is much slower than the one given in (7.2.6). Also, if $R_{\min}$ is the smallest positive integer that can be represented in a particular computer, then the algorithm will produce numbers in the interval $(-a, a)$, $a = [-2 \ln(R_{\min})]^{1/2}$.

It is recommended that Equation (7.7) be implemented in double-precision arithmetic in order to extend the range of the output values.

A summary of the methods used for generating independent sequences of random variables is shown in Figure 7.8 and Table 7.1.

7.3. Generating Independent Random Sequence

In the preceding sections we discussed methods for generating sequences of random numbers with a given pdf. Members of the sequence $\{X(k)\}$ are independent. In this section we use these methods to generate independent sequences of random variables that represent sampled values of random processes of interest.

7.3.1. White Gaussian Noise

White noise has a constant power spectral density (PSD) for all frequencies

$$S_{NN}(f) = \frac{N_0}{2} \quad \text{for } -\infty < f < \infty$$

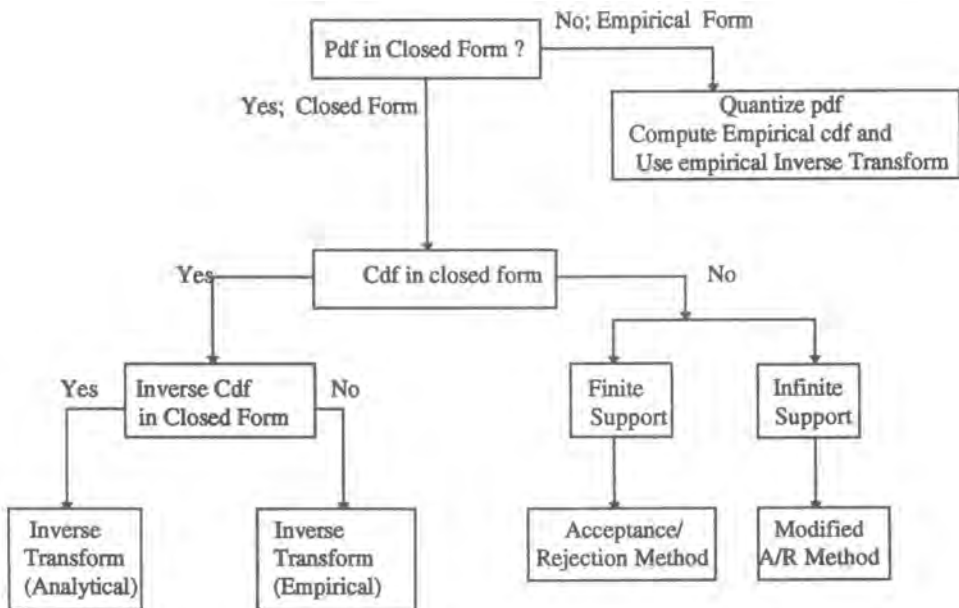


Figure 7.8. Summary of procedures for random number generation.

Table 7.1 Summary of Random Number Generators

Distribution	Method of generation
Discrete	
Uniform	Inverse transform method
Binomial (n, p)	Inverse transform method; or sum of n independent $B(1, p)$ values
Geometric (θ)	$Y = [Z + 1]_{\text{integer}}$; $Z = \ln(U)/\ln(1 - \theta)$
Poisson (λ)	Sum k exponential variables until sum > 1 ; output k ; or, multiply k uniform variables until $U_1 U_2 \dots U_k < \exp(-\lambda)$; output k
Gamma (α, β)	α integer; output = sum of α exponentials ($\lambda = 1$) and scaled by β
Arbitrary	Finite number of values: inverse transform, tabular search Infinite number of values: inverse transform, compute cumulative on the fly
Continuous	
Uniform [a, b]	Inverse transform method: $X = a + U(b - a)$
Exponential	$Z = -[\ln U]/\lambda$
Gaussian $N(0, 1)$	Sum-of-12 method: $X = [U_1 + U_2 + \dots + U_{12}] - 6.0$ Box-Müller method: $X = \sqrt{-2 \ln(U_1)} \cos 2\pi U_2$; $Y = \sqrt{-2 \ln(U_1)} \sin 2\pi U_2$
Rayleigh	$R = \sqrt{X^2 + Y^2}$
Rice	$R = \sqrt{(A + X)^2 + Y^2}$; A is a constant
Chi-square (r)	$\sum_{k=1}^r X_k^2$; X_k are iid $N(0, 1)$
Log normal	$Y = \exp(X)$; X is normal
Others	See Figure 7.6

However, practical systems have finite bandwidths, say B (Figure 7.9b), and the sampling rate for simulation of f_s is chosen to be greater than $2B$. If we use bandlimited white Gaussian noise with a constant PSD over the simulation bandwidth $-f_s/2$ to $f_s/2$,

$$S_{N_s N_s}(f) = N_0/2 \quad \text{for} \quad -f_s/2 < f < f_s/2 \quad (7.3.1)$$

then the PSD at the output of the system is the same whether the input has the PSD $S_{NN}(f)$ or $S_{N_s N_s}(f)$.

For simulation purposes, sampled values of Gaussian white noise are generated using the PSD shown in Figure 7.9c. Note that this PSD leads to an autocorrelation function with zero crossings at $\tau = kT_s$, $T_s = 1/f_s$; i.e., the samples in the time domain are uncorrelated, and independent in the Gaussian case. The variance of the samples is $N_0 f_s/2$. Hence, sampled values of bandlimited white Gaussian noise can be simulated using an independent sequence of Gaussian variables with zero mean and variance

$$\sigma_{N_s}^2 = N_0 f_s/2 \quad (7.3.2)$$

Such a sequence can be generated by multiplying the output of a zero-mean, unit-variance Gaussian random number generator by σ_{N_s} .

7.3.2. Random Binary Sequences and Random Binary Waveforms

A random binary sequence $\{b_k\}$, $b_k = 0$ or 1 , can be generated from a uniform sequence $\{U_k\}$ by

$$b_k = \begin{cases} 1 & \text{if } U_k > p_1 \\ 0 & \text{if } U_k \leq p_0 \end{cases}$$

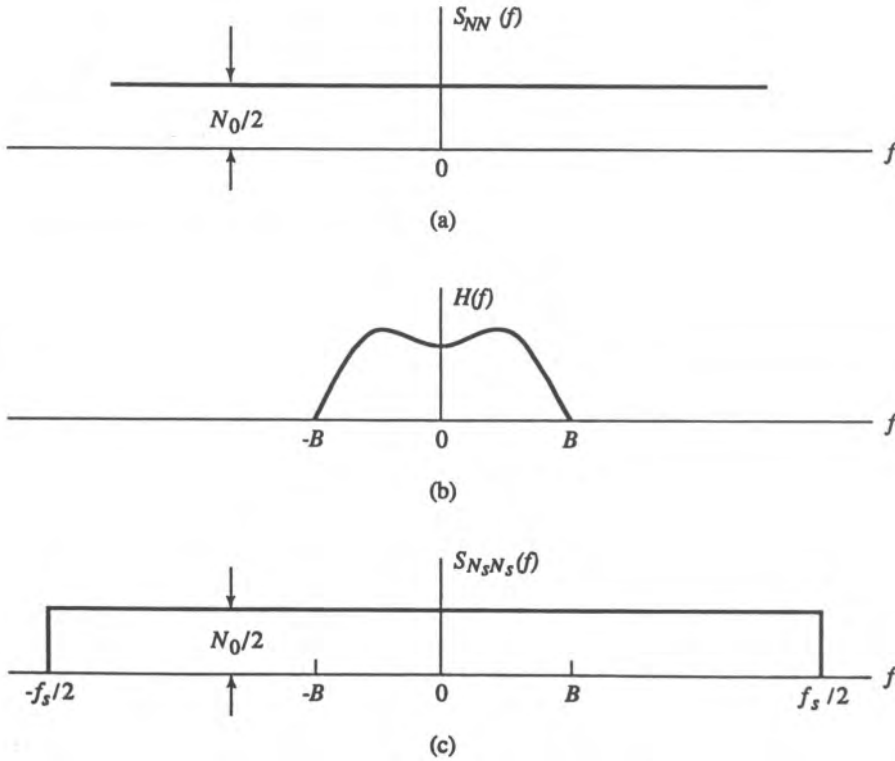


Figure 7.9. Simulation model for white noise, (a) Noise PSD; (b) filter transfer function; (c) simulated PSD.

where $p_0 = P[b_k = 0]$. Sampled values of a random binary waveform can be generated by

$$X(kN + m) = (2b_k - 1)p(mT_s),$$

$$k = \dots, -3, -2, -1, 0, 1, 2, 3, \dots, \quad m = 1, 2, \dots, N$$

where $p(mT_s)$ are the sampled values of a unit-amplitude pulse with a duration equal to the bit interval, and the sampling rate is N times the bit rate.

Sampled values of M -ary digital waveforms with independent amplitude sequences can be generated similarly.

7.3.3. Pseudorandom Binary Sequences

A random binary sequence consists of a statistically independent sequence of 0's and 1's each occurring with probability of 1/2. A pseudonoise (PN) or pseudorandom sequence is a *periodic* binary sequence with an autocorrelation function that resembles the autocorrelation function of a random binary sequence. The PN sequence is generated using a feedback shift register arrangement (Figure 7.10). Although deterministic, the PN sequence has many characteristics similar to those of a random binary sequence.⁽⁹⁾

The feedback shift register arrangement consists of binary storage elements and feedback logic. Binary sequences are shifted through the shift register in response to clock pulses. The contents of the shift register are logically combined to produce the input to the first stage. The initial contents of the register and the feedback logic determine the successive contents of the shift register. A feedback shift register is called linear if the feedback logic consists entirely of modulo 2 adders. An m -stage linear feedback shift register generates PN sequences according to

$$S_n = c_{m-1}S_{n-1} \oplus c_{m-2}S_{n-2} \oplus \cdots \oplus c_1S_{n-m+1} \oplus c_0S_{n-m} \quad (7.3.3)$$

where S_n is the value of the sequence at time n , and the coefficients c_i are binary-valued (0 or 1), $C_0 = 1$, and $\oplus$ indicates modulo 2 addition.

Because the number of distinct states of an m -state shift register is 2^m , the sequence of states and the output sequences must eventually become periodic with a period of at most 2^m . If a linear feedback shift register contains all zeros at any time, it will be “stuck” in that state forever. Since there are exactly $2^m - 1$ nonzero states, the maximum period cannot exceed $2^m - 1$.

Let us associate with the coefficients in (7.3.3) a polynomial of degree m ,

$$P_m(X) = X^m + c_{m-1}X_{m-1} + \cdots + c_1X + c_0 \quad (7.3.4)$$

Then it can be shown that the sequence $\{S_n\}$ will be a maximal-length sequence if and only if $P_m(\cdot)$ is a primitive polynomial.⁽⁹⁾ A polynomial of degree m with coefficients in the binary field is primitive if it divides $X^r + 1$ for r not less than $2^m - 1$. Table 7.2 lists primitive polynomials of degree up to 16, with binary coefficients, and Figure 7.11 shows an implementation for $m = 10$.

Maximal-length PN sequences have the following properties:

1. In each period, the number of 1's is always one more than the number of 0's (i.e., the sequence is nearly balanced)
2. The autocorrelation function of a PN sequence with $Y(n) = (-1)^{S_n}$ is defined as

$$R_p(k) = \frac{1}{N} \sum_{n=1}^N Y(n)Y(n-k), \quad N = 2^m - 1 \quad (7.3.5)$$

$R_p(k)$ is periodic and is similar to the autocorrelation function of a random binary sequence over one period (see Figure 7.12).

3. Among the runs of 1's and 0's in each period of a maximal-length sequence, one half of the runs of each kind are of length 1, one fourth are of length 2, one eighth are of length 3, and so on.
4. A maximal-length sequence has patterns of all possible m -bit combinations each of which appears only once within each period, except for a pattern of m zeros; the longest sequence of zeros is $m - 1$.

The last property is especially useful for simulating intersymbol interference⁽¹⁰⁾ introduced by filters in digital communication systems. Intersymbol interference (ISI) is a major source of performance degradation in digital communication systems. To simulate the effects of ISI, we need to drive the system with a binary sequence that has all possible combinations of m bits, where m is the memory length of the system measured in bit intervals. While a

Table 7.2 Table of Some PN Generators for Binary Sequences (Primitive Polynomials)

Example: $m = 5, [2, 5] \rightarrow 1 + x^2 + x^5$

Register length m	
2	[1, 2]
3	[1, 3]
4	[1, 4]
5	[2, 5][2, 3, 4, 5][1, 2, 4, 5]
6	[1, 6][1, 2, 5, 6][2, 3, 5, 6]
7	[3, 7][1, 2, 3, 7][1, 2, 4, 5, 6, 7][2, 3, 4, 7] [1, 2, 3, 4, 5, 7][2, 4, 6, 7][1, 7][1, 3, 6, 7] [2, 5, 6, 7]
8	[2, 3, 4, 8][3, 5, 6, 8][1, 2, 5, 6, 7, 8] [1, 3, 5, 8][2, 5, 6, 8][1, 5, 6, 8] [1, 2, 3, 4, 6, 8][1, 6, 7, 8]
9	[4, 9][3, 4, 6, 9][4, 5, 8, 9] [1, 4, 8, 9][2, 3, 5, 9][1, 2, 4, 5, 6, 9] [5, 6, 8, 9][1, 3, 4, 6, 7, 9][2, 7, 8, 9]
10	[3, 10][2, 3, 8, 10][3, 4, 5, 6, 7, 8, 9, 10] [1, 2, 3, 5, 6, 10][2, 3, 6, 8, 9, 10][1, 3, 4, 5, 6, 7, 8, 10]
11	[2, 11][2, 5, 8, 11][2, 3, 7, 11] [2, 3, 5, 11][1, 3, 8, 9, 10, 11]
12	[1, 4, 6, 12][1, 2, 5, 7, 8, 9, 11, 12] [1, 3, 4, 6, 8, 10, 11, 12][1, 2, 5, 10, 11, 12] [2, 3, 9, 12][1, 2, 4, 6, 11, 12]
13	[1, 3, 4, 13][4, 5, 7, 9, 10, 13][1, 4, 7, 8, 11, 13] [1, 2, 3, 6, 8, 9, 10, 13][5, 6, 7, 8, 12, 13][1, 5, 7, 8, 9, 13]
14	[1, 6, 10, 14][3, 4, 6, 7, 9, 10, 14] [4, 5, 6, 7, 8, 9, 12, 14][1, 6, 8, 14] [5, 6, 9, 10, 11, 12, 13, 14][1, 2, 3, 4, 5, 7, 8, 10, 13, 14] [1, 15][1, 5, 10, 15][1, 3, 12, 15]
15	[1, 2, 4, 5, 10, 15][1, 2, 6, 7, 11, 15][1, 2, 3, 6, 7, 15] [1, 3, 12, 16][1, 3, 6, 7, 11, 12, 13, 16]
16	[1, 2, 4, 6, 8, 9, 10, 11, 15, 16][1, 6, 8, 14]

random binary sequence can be used as the input, there is no guarantee that a random binary sequence of any length will contain, with probability 1, a particular bit pattern. However, a maximal-length sequence generated using an m -stage feedback shift register will contain all but one m -bit combinations within a length of $2^m - 1$ bits. The memory length of communication systems will typically be less than 10 bits and hence a PN sequence of less than 1024

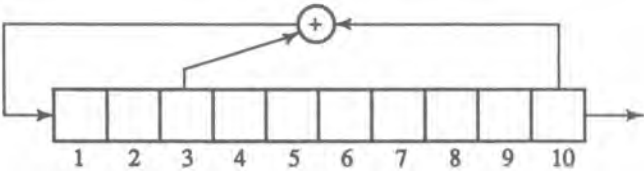


Figure 7.11. Linear shift register generator of length 10 associated with [3, 10].

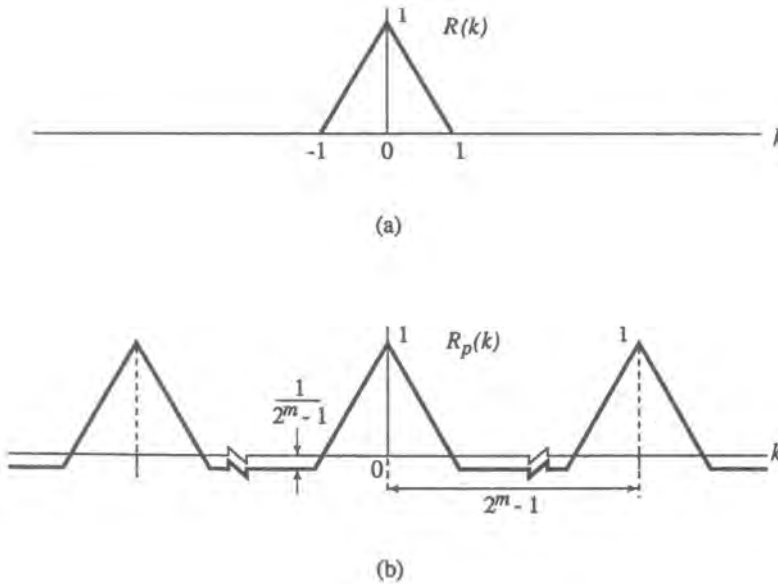


Figure 7.12. Autocorrelation function, (a) Random binary sequence (± 1); (b) PN sequence, **period** = $2^m - 1$.

bits may be sufficient to simulate ISI for a binary system. If we use a random binary sequence, a much longer sequence will be required.

Note that the PN sequences can be generated much faster than random binary sequences. To start the PN sequence generation, the shift register is loaded with a seed, which can be any binary number between 1 and $2^m - 1$. Two PN sequences that are shifted by approximately $[2^m]/2$ are uncorrelated (or we can use different primitive polynomials).

A maximal-length sequence with period 2^m is one that has all possible patterns of m bits. As noted above, only the all-zero m -bit pattern does not naturally occur. To produce such a sequence, we only need to add a 0 digit to the output at a place where there are $(m - 1)$ zeros. This sequence is called the deBruijn⁽⁹⁾ sequence. To simplify the generation of such a sequence, we enforce the $m - 1$ all-zero sequence at the right place. If the initial fill of the shift register is $(10 \dots 00)$ with 1 in the leftmost register, the first $m - 1$ outputs will be zero, which we need only precede with a zero to produce a deBruijn sequence.

7.3.4. M -ary Pseudonoise Sequences

Many digital communication systems use M -ary waveforms (M -ary phase, frequency, and amplitude). To simulate the performance of these systems, we may use M -ary PN sequences which can also be generated using feedback shift registers. The generation of such sequences can be done by a generalization of the binary case. Consider the degree- m monic ($a_m = 1$) polynomial

$$P_m(X) = X^m + a_{m-1}X^{m-1} + \dots + a_1X + a_0 \quad (7.3.6)$$

with coefficients a_i in a finite field of q elements. Finite fields are called Galois fields^(9,11) and a finite field of q elements is denoted $\text{GF}(q)$. A polynomial of degree m , over $\text{GF}(q)$ is

primitive if it divides $X^r + 1$ for r not smaller than $q^m - 1$. Analogous to the binary case, we can state the following result: A primitive polynomial of degree m over $\text{GF}(q)$ generates a linear recursive q -ary sequence with maximal length $q^m - 1$. The recursion is given by

$$S_n = \sum_{i=1}^m a_{m-i} S_{n-i} \quad (7.3.7)$$

where S_n is the output at time n . The sequence $\{S_n\}$ can be generated by a linear shift register exactly like that of Figure 7.10 with the c_i replaced by the corresponding a_i ; the major difference is that instead of modulo 2 arithmetic, we need to use appropriate arithmetic in $\text{GF}(q)$, which we will illustrate shortly (see Ref. 11 for additional details).

In M -ary communication we normally use values of $M = 2^k$. Hence, we want to set $q = M = 2^k$ and develop the arithmetic in $\text{GF}(2^k)$. The elements of $\text{GF}(2^k)$ can be identified with all the polynomials of degree less than k with coefficients in $\text{GF}(2)$. Addition is then ordinary addition of polynomials, modulo 2. In the field $\text{GF}(2^k)$, subtraction is identical to addition. Multiplication is defined as ordinary polynomial multiplication modulo a certain polynomial $g(\cdot)$ of degree k irreducible over $\text{GF}(2)$; that is, the result of multiplying two polynomials $a(X)$ and $b(X)$ is the remainder of $a(X)b(X)$ after dividing by $g(X)$. To illustrate the preceding, consider the irreducible polynomial over $\text{GF}(2)$

$$g(X) = X^3 + X + 1$$

which can be used to generate the field $\text{GF}(2^3)$. Let $a(X) = X$ and $b(X) = X^2 + X$; then in $\text{GF}(2^3)$ the product $a(X)b(X)$ is

$$\begin{aligned} c(X) &= a(X)b(X) \bmod g(X) \\ &= X^2 + X + 1 \end{aligned}$$

There are several alternative representations for $\text{GF}(2^k)$, the polynomial representation already mentioned, and the corresponding binary k -tuple. A third representation is based on the fact that in every finite field there is an element α , called *primitive*, such that every nonzero element of the field can be expressed as some power of α . We illustrate in Table 7.3 the elements of $\text{GF}(2^3)$ generated by the polynomial $g(X) = X^3 + X + 1$, which happens to be primitive; the leftmost column is just short-hand notation which we shall use below. In practice, addition is performed by “exclusive-or” logic, and multiplication is performed using a lookup table.

In order to generate a maximum-length PN sequence of octal numbers, we need a primitive polynomial whose coefficients are elements of $\text{GF}(8)$. If we use the representation of $\text{GF}(8)$ defined above, then one such polynomial^(11,25) is

$$f_1(X) = X^3 + X + \alpha \sim (101A) \quad (7.3.8)$$

This polynomial generates the recursive sequence

$$S_n = S_{n-2} + \alpha S_{n-3} \quad (7.3.9)$$

Table 7.3 Representation of Elements in GF(2³)

	Power of primitive root	Polynomial representation	Binary <i>k</i> -tuple representation
A	α	X	(0, 1, 0)
B	α^2	X^2	(1, 0, 0)
C	α^3	$1 + X$	(0, 1, 1)
D	α^4	$X + X^2$	(1, 1, 0)
E	α^5	$1 + X + X^2$	(1, 1, 1)
F	α^6	$1 + X^2$	(1, 0, 1)
1	$\alpha^7 = \alpha^0$	1	(0, 0, 1)
0	0	0	(0, 0, 0)

An octal shift register circuit for this sequence and a sample octal sequence are shown in Fig. 7.13. Since $f_1(X)$ is a degree 8 polynomial, it is suitable for a channel with a 3-symbol memory.

If two octal sequences are needed to simulate the in-phase and quadrature components, then a second primitive polynomial is needed. Such a (degree 3) polynomial is

$$f_2(X) = X^3 + X + \alpha^2 \sim (101B)$$

with the corresponding recursive sequence

$$S_n = S_{n-2} + \alpha^2 S_{n-3}$$

For applications involving 16-QAM transmission⁽¹³⁾, the primitive generator polynomials are selected with coefficients in GF(4), to represent the I and Q channels. A primitive polynomial over this field⁽¹³⁾ is

$$g(X) = X^2 + X + 1 \tag{7.3.10}$$

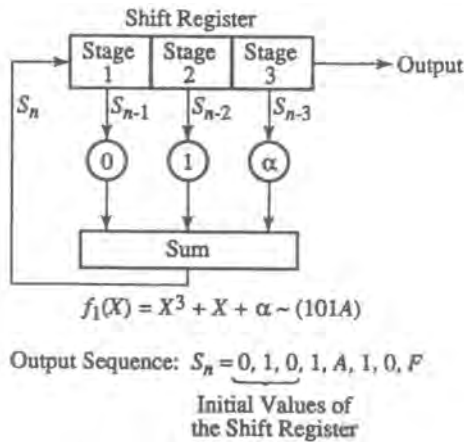


Figure 7.13. Octal PN sequence generator.

Table 7.4 Generating Polynomials

Degree of polynomial	Quaternary generating polynomials	Octal generating polynomials
1	1G	1A
1	1H	1B
2	11G	11C
2	11H	11E
3	111G	101A
3	111H	101B
4	101HG	1001C
4	10GGG	1001E
5	10001G	10011C
5	10001H	10011E

If α is a root of this polynomial, the field elements are defined by $(0, 1, G, H)$, where the nonzero elements can be identified as

$$\begin{aligned} 0 &\sim (00), & \alpha^0 &= 1 \sim (01), & \alpha^1 &= G \sim (10), \\ & & \alpha^2 &= \alpha + 1 = H \sim (11) \end{aligned} \quad (7.3.11)$$

Table 7.4 lists generating polynomials of degree m for quaternary and octal sequences^(11–13) These are suitable for use on channels with memory of m symbols, $m = 1, 2, 3, 4, 5$.

As in the case of binary PN sequences, an additional 0 can be inserted into a nonbinary PN sequence with a period of $M^m - 1$ to produce a deBruijn sequence with a period equal to M^m .

7.4. Generation of Correlated Random Sequences

In many applications there is often the need to generate a random sequence which has a specified autocorrelation function $R(k)$ or $\mathbf{R}(\tau)$ or power spectral density $S(f)$. For example, in simulating a randomly time-varying mobile communication channel, the time variations are modeled by a random process with the so-called Jakes power spectral density

$$S(f) = \frac{1}{\sqrt{1 - (f/f_D)^2}}, |f| \leq f_D \quad (7.4.1)$$

where f_D is the maximum Doppler frequency.

Another example of this situation occurs when we seek an equivalent representation for phase noise in a communication system where the phase noise might be characterized as a Gaussian process with an arbitrary power spectral density, which in many cases might be specified empirically based on measurements.

When $\mathbf{R}(\tau) \neq 0$ for $\tau = kT_s, k = \pm 1, \pm 2, \dots$, the process is correlated at multiples of the sampling interval and hence samples of the random process will be correlated. In order to

simulate the sampled values of such a process, we need algorithms for generating correlated sequences of random numbers with a given autocorrelation function $R(kT_s)$.

There are also many occasions when we have to generate sampled values of multiple random processes which are correlated. For example, when we represent a bandpass Gaussian random process with a power spectral density that is nonsymmetric around the center frequency, the two random processes that represent the real and imaginary components of the complex lowpass-equivalent representation will be correlated. Another example occurs in modeling of multipath propagation in radio channels where the time-varying nature of each path is modeled as a random process. When this model is implemented in a tapped delay line form, the tap gains become a set of correlated random processes, and in order to simulate this model, we need to generate sampled values of a set of correlated processes with a given power spectral density.

Multiple random processes can be represented as a vector-valued random process. Each component of the vector-valued random process is a scalar-valued process with an arbitrary autocorrelation function. The components of the vector-valued random process may be correlated as discussed in the preceding examples. Whereas the correlation along the time axis is temporal in nature and it leads to the notion of power spectral density, the correlation between the components of a vector-valued random process represents a correlation in a different dimension (often referred to as spatial correlation) as shown in Figure 7.14

In the first part of this section we will focus our attention on algorithms for generating scalar-valued random sequences with a given autocorrelation or power spectral density. We will then turn our attention to the generation of vector sequences with a given temporal correlation and spatial correlation. For the Gaussian case we will see that these algorithms will consist in finding and applying a linear transformation either temporally (“horizontally” along the time axis in Figure 7.14) or spatially (applying the transformation “vertically” in Figure 7.14).

The non-Gaussian case will involve nonlinear transformations and it is in general difficult to find these transformations.

7.4.1. Correlated Gaussian Sequences: Scalar Case

An uncorrelated Gaussian sequence can be transformed into a correlated Gaussian sequence through a linear transformation or filtering which preserves the Gaussian distribution, but alters the correlation properties. The coefficients of the linear transformation can be obtained from the specified correlation function of the output. We present below two approaches, a time-domain approach based on the correlation function, and a frequency-domain approach based on the power spectral density. The time-domain approach uses a class of discrete time models called autoregressive and moving average (ARMA) processes.

7.4.1.1. Autoregressive and Moving Average (ARMA) Models

A correlated Gaussian sequence $Y(n)$ can be generated from an uncorrelated Gaussian sequence $X(n)$ using an autoregressive moving average [ARMA (p, q)] model of the form

$$Y(n) = \underbrace{\sum_{i=1}^p \phi_{pi} Y(n-i)}_{\text{Autoregressive part}} + \underbrace{\sum_{j=1}^q \theta_{qj} X(n-j)}_{\text{Moving average part}} + X(n) \quad (7.4.2)$$

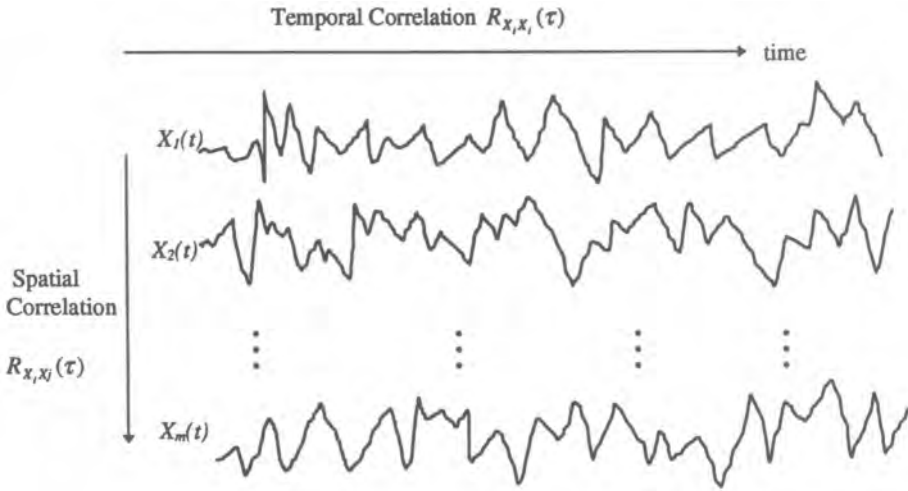


Figure 7.14. Example of a vector-valued random process.

where $X(n)$ is the “input” sequence that is an uncorrelated Gaussian sequence with zero mean and variance σ_X^2 , $Y(n)$ is the “output” sequence, and $\phi_{pi}, i = 1, 2, \dots, p$, and $\theta_{qj}, j = 1, 2, \dots, q$, are the parameters of the autoregressive and the moving average parts of the model, respectively. The sampling time is normalized to 1 for convenience. The model given in Equation (7.4.2) is a linear time-invariant discrete-time filter with a transfer function

$$H(f) = \frac{1 + \sum_{i=1}^p \phi_{pi} \exp(-j2\pi fi)}{1 - \sum_{i=1}^q \theta_{qi} \exp(-j2\pi fi)} \quad (7.4.3)$$

and produces an output power spectral density of the form

$$S_{YY}(f) = S_{XX}(f) |H(f)|^2 = \sigma_X^2 \left| \frac{1 + \sum_{i=1}^p \phi_{pi} \exp(-j2\pi fi)}{1 - \sum_{i=1}^q \theta_{qi} \exp(-j2\pi fi)} \right|^2 \quad \text{for } |f| < \frac{1}{2} \quad (7.4.4)$$

A simpler version of the model, called the autoregressive AR(p) model, results if the coefficients of the moving average part of the model are zero:

$$Y(n) = \sum_{i=1}^p \phi_{pi} Y(n-i) + X(n) \quad (7.4.5)$$

For the AR(p) model, it can be shown that the autocorrelation function $Y(n)$ is given by⁽¹⁴⁾

$$R_{YY}(0) = \sum_{i=1}^p \phi_{pi} R_{YY}(i) + \sigma_X^2 \quad (7.4.6)$$

and

$$R_{YY}(k) = \sum_{i=1}^p \phi_{pi} R_{XX}(k-i), \quad k \geq 1 \quad (7.4.7)$$

We can express Equation (7.4.7) for $k = 1, 2, \dots, p$ in a matrix form,

$$\begin{bmatrix} R_{XX}(1) \\ R_{XX}(2) \\ \vdots \\ R_{XX}(p) \end{bmatrix} = \begin{bmatrix} R_{XX}(0) & R_{XX}(1) & \cdots & R_{XX}(p-1) \\ R_{XX}(1) & R_{XX}(0) & \cdots & R_{XX}(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_{XX}(p-1) & R_{XX}(p-2) & \cdots & R_{XX}(0) \end{bmatrix} \begin{bmatrix} \phi_{p1} \\ \phi_{p2} \\ \vdots \\ \phi_{pp} \end{bmatrix} \quad (7.4.8)$$

Equation (7.4.8) is called the Yule–Walker equation and it relates the values of the autocorrelation function of the output and the parameters of the model. Given $R_{XX}(k)$, $k = 0, 1, 2, \dots, p$, we can obtain the coefficients ϕ_{pi} , $i = 1, 2, \dots, p$, by inverting Equation (7.4.8) and the value of σ_X^2 from Equation (7.4.7). Recursive procedures for determining the model order p and for solving the Yule–Walker equation can be found in Ref. 15. Once the model parameters have been obtained, they are substituted in Equation (7.4.5) to transform an uncorrelated Gaussian sequence into a correlated Gaussian sequence.

While it is easy to obtain the parameters of an AR(p) model, the relationship between the coefficients of the ARMA(p, q) model and the autocorrelation function of the output are more complex and nonlinear in nature and hence it is very difficult to solve for the coefficients of the model, given the autocorrelation values. While a large body of literature exists on this topic, the most commonly used procedure involves converting an ARMA(p, q) model into an equivalent AR model of order $\gg p + q$ and solving for the parameters of the AR model using the Yule–Walker equation. Another approach solves for the AR part of the ARMA model first and then obtains the coefficients of the MA part. The ARMA model is used in situations where the power spectral density for the output sequence has a complex structure with many peaks and valleys. From Equation (7.4.3) it can be seen that the AR parts influence the poles of the transfer function (and hence the peaks in the output power spectral density) and the MA part corresponds to the zeros of the transfer function (and hence affects the valleys in the psd).

When we implement an ARMA model using Equation (7.4.2) we need p initial values of $Y(k)$ to get the recursion started. If these initial conditions are chosen arbitrarily, then the output will have a transient in the beginning and the first few samples (of the order of $10p$) should be ignored.

An example of the use of ARMA model as an approximation of a phase noise process is presented in Chapter 12.

7.4.1.2. Spectral Factorization Method

We can also design a filter in the frequency domain to transform an uncorrelated Gaussian sequence into a correlated sequence. This approach is particularly useful if the power spectral density of the output process is given in closed form as a ratio of polynomials in f^2 . When we pass an independent Gaussian sequence through a filter, the output Gaussian process will have a power spectral density given by

$$S_{YY}(f) = S_{XX}(f) |H(f)|^2 \quad (7.4.9)$$

where

$$S_{XX}(f) = \sigma_X^2 \quad \text{for } |f| < 1/2 \quad (7.4.10)$$

and hence if we choose $\sigma_X^2 = 1$, we have

$$S_{YY}(f) = |H(f)|^2 \quad (7.4.11)$$

Equation (7.4.9) shows that by choosing the filter transfer function carefully, we can produce the desired output psd (autocorrelation function). The transfer function of a realizable and stable filter can be synthesized by applying the transformation $s = 2\pi jf$ to Equation (7.4.10), *factoring* the given $S_{YY}(f)$ and hence $|H(f)|^2$ into a product of the form $H(s)H(-s)$. The transfer function of the filter is chosen as $H(s)$, which has all of its poles in the left half of the s plane. The example given below illustrates this.

■ **Example 7.4.1.** Find the transfer function of a filter to generate zero-mean Gaussian noise with the PSD

$$S_{YY}(f) = \frac{(2\pi f)^2}{a^2 + (2\pi f)^2}$$

Assume that the input is a zero-mean Gaussian process with $S_{XX}(f) = 1$.

Solution:

$$|H(f)|^2 = \frac{(2\pi f)^2}{a^2 + (2\pi f)^2}$$

Substituting $s = j2\pi f$, we obtain

$$|H(s)|^2 = \frac{-s^2}{a^2 - s^2} = \left(\frac{s}{a+s} \right) \left(\frac{-s}{a-s} \right)$$

The first factor, $s/(a+s)$, is the transfer function of the realizable filter. That is,

$$H(s) = \frac{s}{a+s}, \quad H(f) = \frac{2\pi jf}{a + 2\pi jf}$$

which yields

$$|H(f)|^2 = \frac{(2\pi f)^2}{a^2 + (2\pi f)^2} \quad \blacksquare$$

Once the transfer function of the filter is obtained via spectral factorization, it can be implemented as an IIR or FIR filter.

In many cases the output spectral density may be given in empirical form or it may be in an analytical form that is not conducive to factoring (for example, the Jakes spectrum). In these cases, one of two approaches can be taken.

1. Empirically fit an analytical form (as a ratio of polynomials in f^2) to the given psd

and then apply the spectral factorization.

2. Directly synthesize an FIR filter by setting the filter transfer function to be equal to the square root of the given power spectral density (the phase of the filter is not important since it does not affect the power spectral density).

The second method can be used to produce a Gaussian sequence with the Jakes spectrum given in Equation (7.4.1) by setting

$$H(f) = \sqrt{S_{YY}(f)} = [1 - (f/f_d)^2]^{-1/4} \quad (7.4.12a)$$

This filter has an impulse response

$$h(t) = 1.457f_d^{-1/4} J_{1/4}(x), \quad x = 2\pi f_d t \quad (7.4.12b)$$

where $J_{1/4}(x)$ is the fractional Bessel function. The filter given in Equation (7.4.12) can be implemented as an IIR or FIR filter.

7.4.2. Correlated Gaussian Vector Sequences

7.4.2.1. Special Case

We now consider the problem of generating sampled values of m zero-mean Gaussian random processes $Y_1(t), Y_2(t), \dots, Y_m(t)$ with arbitrary power spectral densities and arbitrary correlation between them. We begin with the simpler case where the autocorrelation function of each of these processes is the same except for a scale factor and the cross-correlation function has the same functional form as the autocorrelation function. (This assumption is justified for the modeling multipath channels discussed in Chapter 9). The sampled values of these m scalar-valued processes can be treated as components of a vector-valued process $\tilde{\mathbf{Y}}(k)$,

$$\tilde{\mathbf{Y}}(k) = \begin{bmatrix} Y_1(k) \\ Y_2(k) \\ \vdots \\ Y_m(k) \end{bmatrix} \quad (7.4.13)$$

with

$$R_{Y_i Y_j}(n) = E\{Y(k)Y(k+n)\} = \sigma_{ij}R(n) \quad (7.4.14)$$

where σ_{ij} is the covariance *between the components* of the process at a given time, and $R(n)$ describes the *temporal correlation*. In this model the spatial and temporal correlations are separable.

We can generate a sequence of Gaussian vectors $\tilde{\mathbf{Y}}(k)$ with a given set of covariances *between its components* by transforming a sequence of Gaussian vectors $\tilde{\mathbf{X}}(k)$ with uncorrelated (and hence independent) Gaussian components where each component has the temporal correlation specified by $R(n)$. We first generate m independent sequences (components of the vector $\tilde{\mathbf{X}}$) with a given temporal correlation $R(n)$ using say the ARMA method, and then transform the uncorrelated components of the vector $\tilde{\mathbf{X}}$ into correlated components using a memoryless linear transformation. This problem is an easy one and is restated as

follows (we can drop the time index k here since the memoryless transformation is applied to the components of $\tilde{\mathbf{X}}$ for each value of k).

Given a random vector $\tilde{\mathbf{X}}$ with a multivariate Gaussian distribution with mean zero and covariance matrix

$$\Sigma_{\tilde{\mathbf{X}}} = \tilde{\mathbf{I}} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \quad (7.4.15)$$

we want to transform it into another Gaussian vector $\tilde{\mathbf{Y}}$ with zero mean and covariance matrix

$$\Sigma_{\tilde{\mathbf{Y}}} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \cdots & \sigma_{mm} \end{bmatrix} \quad (7.4.16)$$

by applying a linear transformation of the form

$$\tilde{\mathbf{Y}} = \mathbf{A}\tilde{\mathbf{X}} \quad (7.4.17)$$

The linear transformation given above transforms the covariance matrix of $\tilde{\mathbf{X}}$, which is the identity matrix given in Equation (7.4.15), into

$$\Sigma_{\tilde{\mathbf{Y}}} = \mathbf{A}\Sigma_{\tilde{\mathbf{X}}}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T \quad (7.4.18)$$

For a given covariance of matrix $\tilde{\mathbf{Y}}$, Equation (7.4.18) can be used to find the transformation matrix $\mathbf{A}$ by *factoring* the covariance matrix. If $\Sigma_{\tilde{\mathbf{Y}}}$ is positive-definite, then there exists a unique lower diagonal matrix $\mathbf{A}$ of the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix} \quad (7.4.19)$$

such that

$$\Sigma_{\tilde{\mathbf{Y}}} = \mathbf{A}\mathbf{A}^T \quad (7.4.20)$$

This decomposition is called the Cholesky decomposition, and the elements of $\mathbf{A}$ can be obtained from the elements of $\Sigma_{\tilde{\mathbf{Y}}}$ according to

$$a_{ij} = \frac{\sigma_{ij} - \sum_{k=1}^{j-1} a_{ik}a_{jk}}{\left[\sigma_{ij} - \sum_{k=1}^{i-1} a_{jk}^2\right]^{1/2}}, \quad \text{where} \quad \sum_{k=1}^0 a_{ik}a_{jk} = 0, \quad 1 \leq j \leq i \leq m \quad (7.4.21)$$

It is also possible to use an eigenvector decomposition to find the linear transformation. In any case, after the transformation is found, the algorithm for generating a sequence of correlated Gaussian vectors is shown in Figure 7.15

7.4.2.2. General Case

In the general case, the spatial and temporal correlation could be arbitrary and each component of $\tilde{\mathbf{Y}}$ could have a different temporal autocorrelation function and the cross-

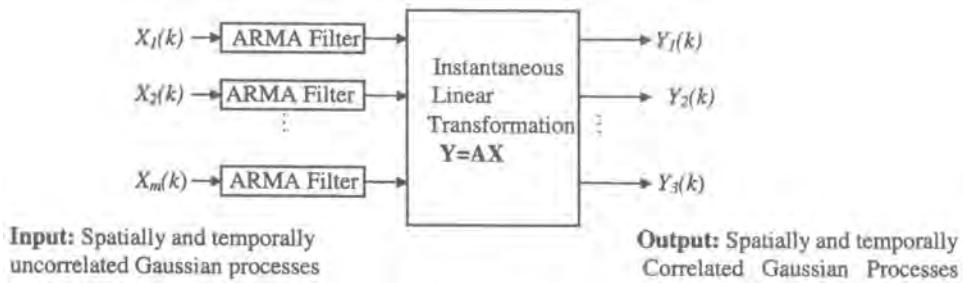


Figure 7.15. Algorithm for generating correlated Gaussian vectors.

correlation functions can also have different functional forms. The process $\tilde{\mathbf{Y}}$ is now specified by a sequence of covariance matrices $\tilde{\Sigma}_{\tilde{\mathbf{Y}}}(j)$ for each time lag j and it can be generated by transforming $\tilde{\mathbf{X}}$ using a vector-valued ARMA process similar to the scalar version discussed in Section 7.4.1.1,

$$\tilde{\mathbf{Y}}(n) = \sum_{i=1}^p \tilde{\Phi}_{pi} \tilde{\mathbf{Y}}(n-i) + \sum_{j=1}^q \tilde{\Theta}_{qj} \tilde{\mathbf{X}}(n-j) + \tilde{\mathbf{X}}(n) \quad (7.4.22)$$

Note that the coefficients of the model are themselves matrices. The coefficients of the AR version of the vector model can be determined, using a vector version of the Yule–Walker equations. For detail, the reader is referred to Reference 15.

7.4.3. Correlated Non-Gaussian Sequences

Generation of random sequences with an arbitrary probability density function and autocorrelation is more difficult than generating correlated Gaussian sequences. In the Gaussian case a linear transformation when applied to an uncorrelated Gaussian sequence *preserves* the Gaussian pdf while producing the desired correlation. This is not the case in the non-Gaussian case. For example, if $Y(n) = X(n-1) + X(n)$, where $X(n)$ is an independent sequence of uniform random variables, the sequence $Y(n)$ will be correlated, but its pdf will be triangular. Thus a linear transformation alters both the correlation function and the pdf (except in the Gaussian case, where the form of the pdf is preserved by a linear transformation).

It is possible, however, to control both the pdf and the correlation function using a combination of linear and nonlinear transformation as shown in Figure 7.16.

The nonlinearities $f(\cdot)$ and $g(\cdot)$ and the filter response $h(\cdot)$ are chosen to produce the desired output pdf and PSD. The procedure used to f , g , and h is rather complex and is detailed in Ref. 16. Briefly, $f(\cdot)$ maps the uniform uncorrelated sequence X to an uncorrelated Gaussian sequence V . The filter h maps the uncorrelated Gaussian sequence V to a correlated Gaussian sequence W , which is then mapped to the desired pdf by the nonlinear transformation $g(\cdot)$, which also modifies the predistorted PSD to the desired PSD. These transformations cannot be chosen independently since they affect both the pdf and the PSD. They have to be chosen together subject to the requirements of the output pdf and PSD.

The multidimensional version of this problem is much more difficult to solve and no general solutions are available.

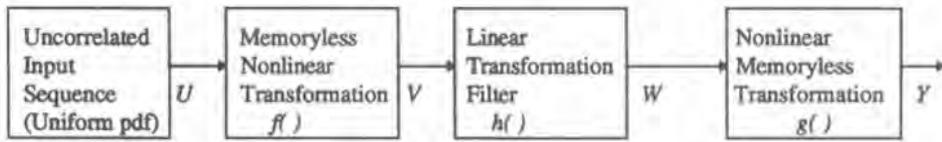


Figure 7.16. Generation of a random sequence with an arbitrary pdf and autocorrelation (PSD).

7.5. Testing of Random Number Generators

Outputs of random number generators (RNGs) used to drive long Monte Carlo simulations must be tested to ensure that they have the “right” statistical properties (temporal and distributional). While it is not necessary to perform exhaustive tests on all RNGs that are part of a simulation package, it is necessary to test at least the uniform and Gaussian RNGs since other RNGs are usually derived from these.

There are two general sets of tests that are normally used to verify the statistical properties of the outputs of RNGs. The first set checks the temporal properties of the outputs of RNGs, the most important temporal properties being stationarity and independence. The second set verifies the distributional properties. This set of tests includes simple checks on the values of parameters such as means and variances of the output sequences to, goodness-of-fit tests, which provide indications of how closely the pdfs of the actual outputs of the RNGs fit the distributions they are supposed to produce.

Tests might range from producing various plots derived from the output sequences and inspecting them visually to see if they “look right”, to more sophisticated (statistical) hypothesis tests at a desired level of confidence. We describe below a few typical tests that are useful and easy to implement. Additional tests may be found in Refs. 1, 3, 13, and 16–20.

7.5.1. Stationarity and Uncorrelatedness

7.5.1.1. Introduction

The output of RNGs should be stationary and uncorrelated (except when we want to generate correlated sequences; even in this case, the input sequence used to generate a correlated sequence is usually white, i.e., uncorrelated). Simple tests for stationarity involve the following steps:

1. Generate a long sequence of N samples.
2. Divide it into m nonoverlapping segments and compute the means and variances of each segment.
3. Test for the equality of the means and variances (and other parameters).

For a stationary process, the mean, variances, and other parameters computed from different segments of the data should be equal, within the limits of statistical variations. Hypothesis tests for equality of means, variances, etc., can be found in Ref. 14.

One of the more serious problems with RNGs is that the output sequence might be correlated and in the worst case be periodic with a period shorter than the simulation length. Estimated values of the normalized autocovariance function can be used to test the uncor-

relatedness of the output sequence of an RNG (stationarity is assumed). If the normalized autocovariance function is defined as

$$\rho_{xx}(k) = \frac{E\{X(j)X(j+k)\} - \mu_x^2}{E\{X^2(j)\} - \mu_x^2} = \frac{C_{xx}(k)}{C_{xx}(0)}$$

then it is desirable that

$$\rho_{xx}(k) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{for } k \neq 0 \end{cases}$$

If $|\rho_{xx}(k)| \geq \varepsilon$ for some values of k , it indicates that the output sequence is correlated. Furthermore, if $\rho_{xx}(k)$ has peaks at $k = mN_0$, $m = 1, 2, \dots$, this indicates periodicity. Since we have to use estimated values $\hat{\rho}_{xx}(k)$ instead of $\rho_{xx}(k)$ to test for uncorrelatedness and periodicity, we need to take into account the statistical variability of $\hat{\rho}_{xx}(k)$.

The following procedure can be used to test for uncorrelatedness and periodicity:

1. Generate a very long sequence.
2. Divide the data into overlapping segments of N points with an overlap of $N/2$ (see Figure 7.17). Let $\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2$, etc., represent the data vectors.
3. Compute the normalized cross-correlation between $\mathbf{X}_0$ and $\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2, \dots$ (FFT techniques can be used to compute the cross-correlation.)
4. Plot $|\hat{\rho}_{xx}(k)|$ and check for peaks.

Figure 7.18 shows some typical results. Note that periodic components are easier to detect in the time domain than in the frequency domain because of the inherent “noisy” nature of periodogram-type spectral estimators.

Testing the RNGs for periodicity is often a computationally intensive task since the period may be very long and $\hat{\rho}_{xx}(k)$ has to be computed for a large number of lags. Furthermore, the tests have to be repeated for different values of the seed used to prime the RNGs. Nevertheless, it is necessary to run these tests once for a set of values for the seed. “Good” values of the seed should be saved and used to start long simulations.

7.5.1.2. Durbin–Watson Test for Correlation^(14,21)

This is a more rigorous procedure for testing the hypothesis that adjacent members of the sequence $X(n)$ are uncorrelated. The test statistic used is

$$D = \frac{\sum_{n=2}^N [X(n) - X(n-1)]^2}{\sum_{n=1}^N [X(n)]^2} \quad (7.5.1)$$

Note that if $X(n)$ and $X(n-1)$ are uncorrelated (correlation = 0), then D would have an expected value of 2. The value of D would be much smaller than 2 if there is strong positive correlation and would approach 4 if there is strong negative correlation. The test is based on two thresholds and is applied as shown in Figure 7.19 (the thresholds d_L and d_U shown in Figure 7.19 are for large values of N). It is better to construct random number generators in such a way that their period can be established analytically. One such example is the Wichman–Hill algorithm, which constructs a random number generator with a long period by

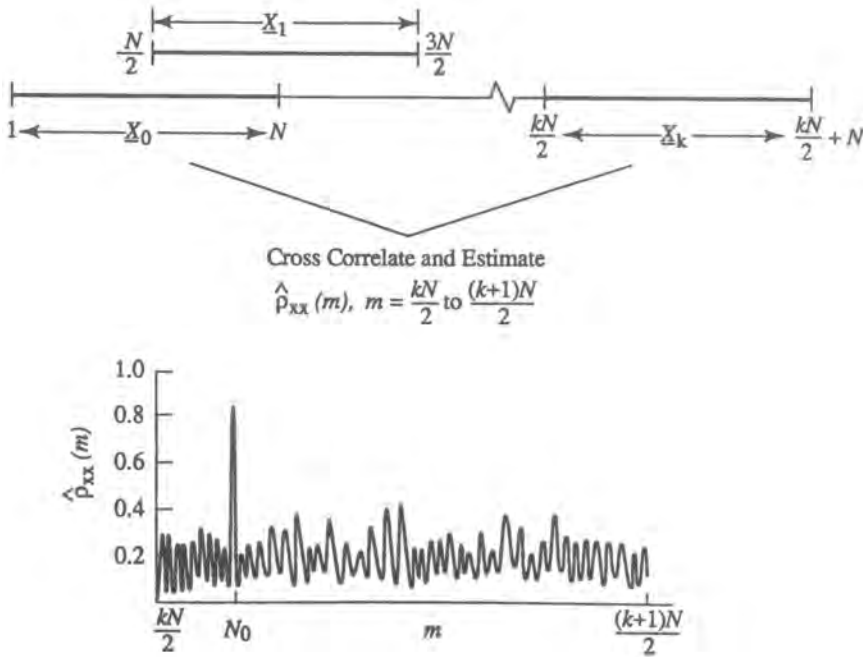


Figure 7.17. Details of test for periodicity; estimated period = N_0 .

combining the output of several generators with shorter periods. The properties of the individual generators used in the Wichman–Hill algorithm can be tested exhaustively since their periods are short.

7.5.2. Goodness-of-Fit Tests

Random number generators are designed to produce output sequences that represent samples from given distributions such as uniform and Gaussian distributions. Whether they do so can be tested using a variety of goodness-of-fit tests. The simplest tests consist of at least making sure that RNGs produce numbers with the desired mean and variance. This is done by estimating the mean and variance of the output of the RNG and comparing it to the desired values. Estimators for the mean and variance are given in Chapter 10 and tests on the mean and variance can be found in Ref. 13.

In addition to testing parameter values, it is often necessary to compare the “shape” of the distribution of the numbers generated by the RNG to the distribution that the RNG is supposed to produce. A popular test used for this purpose is the chi-square test, which is administered as follows.

Suppose we generate N samples and construct a histogram of the N samples. Let $N_i, i = 1, 2, \dots, m$, be the actual cell counts of the m cells of the histogram. Let $p_i, i = 1, 2, \dots, m$, be the true probability of a sample falling into the i th cell under the

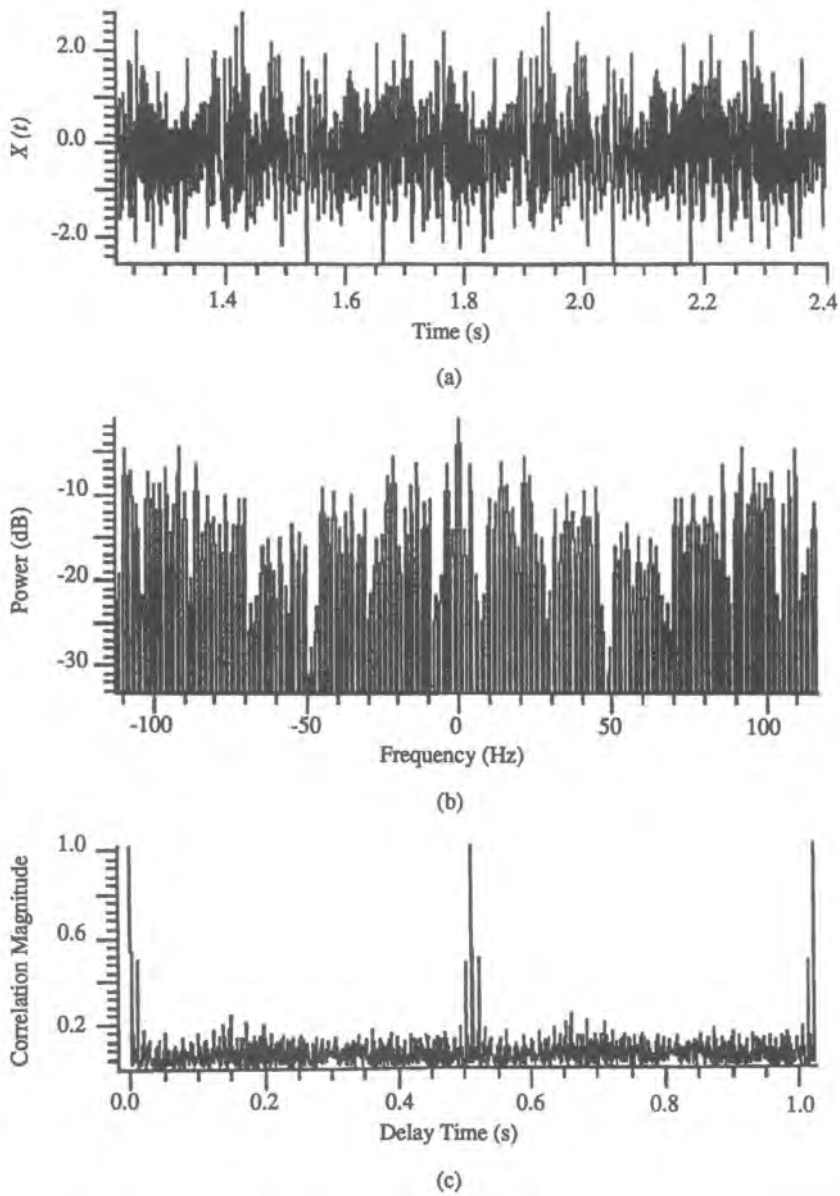


Figure 7.18. Example of a correlated sequence, (a) Sample function $x(t)$; (b) periodogram of $x(t)$ (c) autocorrelation of $x(t)$.

assumed pdf, the one the RNG is supposed to reproduce. Then, under some mild assumptions,⁽¹⁹⁾ the variable

$$Z = \sum_{i=1}^m \frac{[N_i - Np_i]^2}{Np_i} \tag{7.5.2}$$

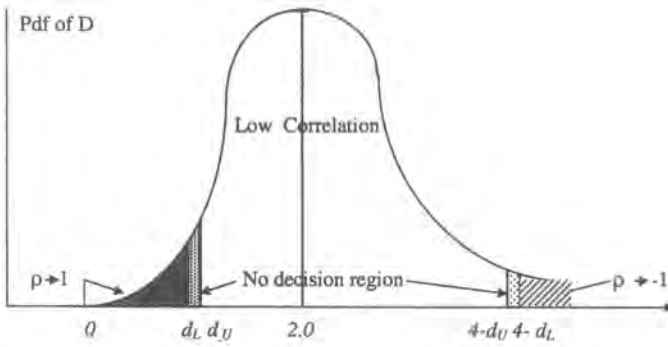


Figure 7.19. Durbin-Watson test; $d_U = 1.70$, $d_L = 1.65$, $N = 100$; probability in the lower tail (high positive correlation region) = 0.05.

has approximately a chi-square distribution with $(m - 1)$ degrees of freedom when N is large and $m > 5$. A small value of Z indicates a good fit and a large value is an indication of bad fit. The value of Z that will be acceptable is determined from the tail probability of the chi-square distribution with $m - 1$ degrees of freedom for given confidence level (see Refs. 14 and 19 for details). The Gaussian approximation might also be used to select an appropriate threshold value of Z .

One difficult aspect of implementing a chi-square test is choosing the right intervals for constructing the histogram. While there is no general rule that can guarantee good results for all sample sizes and distributions, there are a few guidelines that can be used. It is recommended that the intervals be chosen so that $p_1 \approx p_2 \approx \dots \approx p_m$ (i.e., equal cell counts) and that m be kept small, say < 50 . If the sample size N is large, then m can be chosen to be equal to $\sqrt{N}$.

Another goodness of fit test, called the Kolmogorov-Smirnov (KS) test, is based on the KS statistic

$$D_N = \sup_x \{|\hat{F}_N(x) - F(x)|\} \quad (7.5.3)$$

where $\hat{F}_N(x)$ is the empirical distribution function,

$$\hat{F}_N(x) = \frac{\text{number of samples } \leq x}{N} \quad (7.5.4)$$

and $F(x)$ is the distribution the RNG is supposed to produce. Small values of D_N indicate a good fit and large values of D_N indicate a poor fit. It has been shown⁽¹⁹⁾ that, when $F(x)$ is completely known, the fit can be judged poor if

$$\left(\sqrt{N} + 0.12 + \frac{0.11}{\sqrt{N}} \right) D_N > C_{1-\alpha} \quad (7.5.5)$$

where values of $C_{1-\alpha}$ (which does not depend on N !) are shown in Table 7.5 for various values of α , the significance level of the test.

Table 7.5 Critical Values of the Modified KS Statistic

$1 - \alpha$	0.80	0.90	0.95	0.975	0.99
$C_{1-\alpha}$	1.138	1.224	1.358	1.480	1.628

The number, variety, and complexity of tests for random number generators is truly overwhelming. It is possible to limit the scope of the test by considering only those tests that are consistent with the intended use of the generated numbers. Nevertheless, one has to be careful in choosing and testing random number generators if the simulation in which they are used is costly (i.e., time-consuming, requires high-precision results, or is a critical component of a large design). The results of a simulation are only as good as the models and the driving functions, i.e., the random number generators.

7.6. Summary

Random variables and random processes are used to model signals, noise, and interference in communication systems and also for modeling the randomly time-varying behavior of components such as certain types of radio communication channels. This chapter developed procedures for generating sampled values of random processes to drive Monte Carlo simulations in which the time evolution of one or more random processes are emulated during the simulation.

The starting point for random number generation is a computationally efficient algorithm for generating an uncorrelated sequence of uniformly distributed random numbers. The output of such a generator can be used to produce both correlated sequences and arbitrary distributions by applying linear and nonlinear transformations. The Gaussian case is the easier one, where a nonlinear linear transformation can be applied to the output of a uniform random number generator to convert the uniformly distributed output into a Gaussian distribution. A linear transform can then be applied to the uncorrected output sequence and convert it into a correlated sequence. The non-Gaussian case is more difficult.

Random sequences that represent the output of digital sources can be generated using a linear feedback shift register arrangement which is computationally very efficient. Linear shift register sequences have also many other nice properties that are useful for simulating the performance of digital communication systems.

Since the accuracy of simulations depends to a large extent on the “goodness” of the random number generators, it is important to test the output of random number generators for their period, correlation, and distribution. A representative subset of the large number of available tests was presented. Once we have a good simulation model for the system and the appropriate random number generators, then we can drive the simulation model with the outputs of the random number generators as the stimuli and generate sampled values of the outputs at various points of interest in the system. From the simulation output we can extract various measures of performance. Techniques for estimating performance measures are the main focus of the following chapter.

References

1. R. Y. Rubenstein, *Simulation and the Monte-Carlo Method*, Wiley, New York (1981).
2. A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York (1982).
3. D. E. Knuth, *The Art of Computer Programming*, Vol. 2, Addison-Wesley, Reading, Massachusetts (1981).
4. W. H. Press *et al.*, *Numerical Recipes in C*, Cambridge University Press, Cambridge (1988).
5. B. A. Wichman and I. D. Hill, An efficient and portable pseudo random number generator, *Appl. Stat.* **AS-183**, 188–190 (1982).
6. G. Marsaglia and A. Zaman, A new class of random number generators, *Ann. Appl. Prob.* **1**(3), 462–480 (1991).
7. R. F. Coates, G. J. Janacek, and K. V. Lever, Monte Carlo simulation and random number generation, *IEEE J. Select. Areas Commun.* **6**, 58–66 (1988).
8. J. K. Townsend and K. Sam Shanmugan, On improving the computational efficiency of digital lightwave link simulation, *IEEE Trans. Commun.* **38**, 2040–2049 (1990).
9. S. Golomb, *Shift Register Sequences*, Aegean Press, Laguna Hills, California (1982).
10. J. Proakis, *Digital Communication Systems*, 3rd ed., McGraw-Hill, New York (1995).
11. D. H. Green and I. S. Taylor, Irreducible polynomials over composite Galois fields and their applications in coding techniques, *Proc. IEEE*, **121**(9), 935–939 (1974).
12. L. Lin and D. J. Costello, *Error Control Coding*, Prentice-Hall, Englewood Cliffs, New Jersey, (1983).
13. M. Kavehrad and P. Balaban, Digital radio transmission modeling and simulation, in *Conference Record IEEE Globecom Conference* (1989), pp. 1289–1294.
14. K. Sam Shanmugan and A. M. Breipohl, *Random Signals*, Wiley, New York (1988).
15. S. M. Kay and S. L. Marple, *Digital Spectrum Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey (1986).
16. M. M. Sondhi, Random processes with specified spectral density and first-order probability density, *Bell Syst. Tech. J.* **62**, 679–700 (1983).
17. E. J. Dudewicz and Z. A. Karian (eds.), *Modern Design and Analysis of Discrete-Event Simulations*, IEEE Computer Society Press, New York (1985).
18. E. J. Dudewicz and T. G. Ralley, *The Handbook of Random Number Generation and Testing with TESTRAND Computer Code*, American Sciences Press, Columbus, Ohio (1981).
19. M. A. Stephens, EDF statistics for goodness of fit and some comparisons, *J. Am. Statist. Assoc.* **69**, 730–737 (1974).
20. R. K. Hogg and A. T. Craig, *Introduction to Mathematical Analysis*, Macmillan, New York (1978).
21. J. Durbin, Testing for serial correlation, *Econometrica*, **38**, 410–419 (1970).

Modeling of Communication Systems

Transmitter and Receiver Subsystems

8.1. Introduction

The process of modeling communication systems entails a number of considerations that, for discussion purposes, can be separated into four categories:

1. Hierarchical level
2. Modeling level
3. Description level
4. Developmental level.

Conceptually, the *hierarchical* level sets the tone in the approach to system modeling. In Chapter 2 we introduced the idea of a modeling hierarchy and indicated that whenever we simulate a “system” we want the model entities to be at the next lower level of the hierarchy as much as possible, namely the “subsystem” level. Although what constitutes a “system” depends in general on the interest of the modeler, in this book we are interested in *communication* systems, which implies certain basic subsystem functions. Figure 8.1 (adapted from Ref. 1) shows a subsystem-level block diagram of a “generic” one-hop communication system, generic in the sense that virtually any communication system can be specified as a particular case of this block diagram or, for multihop systems, a concatenation of such (or similar) block diagrams. While there are many different types of communication systems, using a wide range of technologies, information transmission in all communication systems takes place through a series of basic signal processing functions, as indicated in Figure 8.1. Although this block diagram is intended for digital communications, some of the blocks apply for analog transmission as well, and for that purpose certain other obvious modifications have to be made. As implied above, in any given system, not all blocks need appear. In a good simulation package, however, we will have models for all blocks that are appropriate. We should also point out that what constitutes a subsystem is itself somewhat arbitrary. Basically, a subsystem is any entity that the system designer chooses to identify as such when drawing the block diagram.

What we are calling the *modeling* level refers to the general nature of the representation of the subsystems. In particular, we prefer a *functional* (also called *behavioral*) description, as

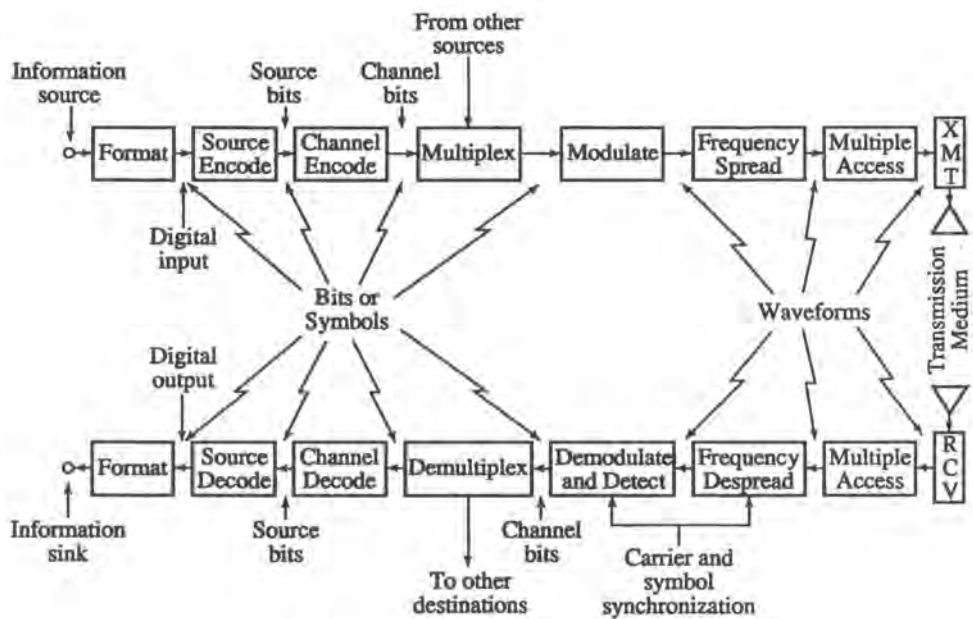


Figure 8.1. Block diagram of generic communication system (from Ref. 1, @IEEE, 1983).

opposed to a physical model, because this will generally be computationally more efficient. What this implies is an input–output or “black-box” characterization which provides a transfer function for the input signal. For carrier-modulated systems, in particular, we wish to have a transfer function for the input complex envelope. By and large, modeling at the subsystem level of the hierarchy does imply that the modeling level must be of the transfer function type. Depending upon exactly what the box is, this may not be always achievable, but in any case we always strive for the simplest modeling methodology possible.

By the *description* level, we mean the actual embodiment of the model, which can be an equation, a set of equations, an experimentally derived lookup table, or some kind of algorithm. Evidently, whatever the original description, it must be convertible to a software form.

Finally, the *developmental* level refers to the stage of a subsystem’s life. That is, the nature of the description will depend on how much is known. In the preliminary stages, little if anything will be known except specifications, or some form of general descriptor. As design advances, more detail will become available, up to an actual hardware design and perhaps some measurements on a prototype. A modeling methodology must be able to accommodate this evolution.

Returning to Figure 8.1, even this block diagram is not entirely complete with respect to what might appear in a subsystem block diagram. For example, filtering is not explicitly shown, nor are any frequency sources. It is therefore useful to set down the various types of subsystem blocks that we may have to deal with:

- Signal sources: radiofrequency and optical
- Information sources: analog and digital
- Source encoding/decoding
- Baseband modulation; line coding
- Radiofrequency and optical modulation
- Oscillators, mixers, frequency translators
- Power amplifiers
- Demodulators
- Filters/adaptive equalizers
- Channels
- Multiplexing/multiple access
- Noise and interference sources
- Error control and waveform coding
- Synchronization

Of course, not every item here is a subsystem in the usual sense, but at least in the sense that it must be accounted for in the system representation. As indicated in Figure 8.1, an important distinction arises with respect to the nature of the blocks, namely, that what is processed is either in the form of logical elements (bits or symbols) or a waveform. From the modeling standpoint, digital processing poses no significant issues at the subsystem level.[†] This does not mean that the processing algorithms are not complex or computationally undemanding. But the algorithms are *exact*. Waveform processing, on the other hand, is where the modeling challenge exists and where approximation must be managed.

The question naturally arises whether we need to simulate all the subsystem blocks that are actually in the system, or to simulate them *as-is*. Although it will usually be necessary to simulate many, if not most, of the subsystems, generally, the answer to the above question is

[†]This, of course, need not be the case at the circuit level.

“no.” Evidently, the seemingly trivial case when a subsystem is assumed to perform ideally does not require its explicit simulation, although accounting for its presence in some way may be compulsory. We say “seemingly trivial” because, as discussed in Chapter 2, it is in fact a common methodology to simplify an investigation by restricting its parameter space, which can be done by idealizing one or more of the subsystem functions. For example, ideal frequency conversion needs no explicit or implicit representation in a simulation which processes lowpass equivalents. Digital processing blocks at baseband which do not change the statistical properties of the symbol stream also do not need to be explicitly or implicitly present. However, idealizing bit synchronization, say, while not requiring an explicit model, does require the function to be explicitly represented. Finally, as also discussed in Chapter 2, it may be possible to use an *equivalent-process* model to replace a chain or group of subsystems by the process which is produced at the output. Some of these points will be taken up again later.

The preceding discussion implicitly raises a point, namely, what characterizes a good subsystem model? A general discussion of modeling issues is given in chapter 2. Translated to the present context, a good subsystem model should be as realistic as possible,[†] given the stage of development. Until the stage where at least the structure of a subsystem is known, the model will be an abstraction. A good abstract model should have changeable parameters (“knobs”) that can reflect departure from ideal in meaningful and physically realizable ways, and should be able to reach specification limits by appropriate setting of these parameters. Once a design is known as to structure, the model should become design-specific. The idealized form of that structure should be the starting point for determining the ways in which real impairments can make themselves felt. As we noted in Chapter 2, one of the useful things about simulation is that it is generally no more difficult to handle models of imperfect structures than ideal ones. The methodology just described will more or less form the basis for a number of the models in this chapter. It should be clear that an extensive discussion of these subsystems is beyond our intended scope. There are many excellent texts (e.g., Refs. 2–21) and a vast literature on the subjects of this chapter. The literature, by and large, deals with analytical descriptions and derivation of performance equations, usually in a somewhat idealized environment. While this viewpoint is fundamentally important, in simulation our inputs are the models of sources and devices, and the simulation itself determines the “performance,” which it is presumed would otherwise be too difficult to derive. Nevertheless, idealized performance equations are useful as a benchmark and as a “sanity check” for simulation results. For that purpose, we include in Appendix A a set of performance results for digital communication over more or less idealized channels.

Returning to the list of subsystems above, it is clear that we have already discussed a number of the items listed. In Chapters 3 and 4, we discussed techniques for simulating linear time-invariant subsystems (filters) in general, and later in this chapter we will describe some particular filter types. In Chapter 5 we discussed models for nonlinear subsystems such as amplifiers and nonlinear subsystems representable as nonlinear differential equations; in this chapter the latter will be reexamined in connection with synchronization circuits. In Chapters 6 and 7 we looked at models for noise and interference sources. In Chapter 9 we will develop models for channels of various types. Subsystems not discussed elsewhere will be addressed to varying degrees of detail in this chapter, more or less following the signal path around the block diagram of Figure 8.1.

[†]This statement, of course, does not apply to the situation where we deliberately idealize some subsystem for a good reason.

8.2. Information Sources

The first “subsystem” in Figure 8.1 is an information source. Depending upon whether this point is actually the origination of the information or is being relayed from a previous hop, this source may contain noise and interference as well as the desired component, “signal.” While there is a clear dichotomy between the *contents* of signals (information) and noise (useless), the *models* of signals and noise used in simulation are not necessarily dichotomous. For example, a tone might be used to represent a signal or interference, while we might use a bandlimited Gaussian process to model both signals and noise. Also, depending on whether this is the origin of the signal, some of the subsequent blocks may or may not be present. Since we have already discussed models of noise and interference, we will assume here that the information source is uncontaminated. In order to discuss models of information sources, and some of the subsequent models as well, it will be useful to redraw the first few blocks of Figure 8.1 into the more operationally oriented diagram of Figure 8.2.[†]

8.2.1. Analog Sources

Our emphasis in this book is on *digital transmission systems*; these last three words have implications with respect to the entities that are modeled and to the level of modeling detail. The first, *digital*, implies that an analog source would first need to be transformed into an “equivalent” digital source. This, of course, is a common procedure in audio, video, and other applications. This transformation may take place in several steps that we call formatting/source coding, which will be discussed in the next section. The question which now arises, from the *transmission-system-level* viewpoint, is whether the details of this process need to be simulated at all. The relevance of this question in the present context, of course, is that *if* the answer is “no,” then we also do not need to represent explicitly the analog source. All that is needed is simply a model for the digital source that would have been produced at the output of the formatting/source coding subsystem, although it is not necessarily the case that such a model can be readily inferred.

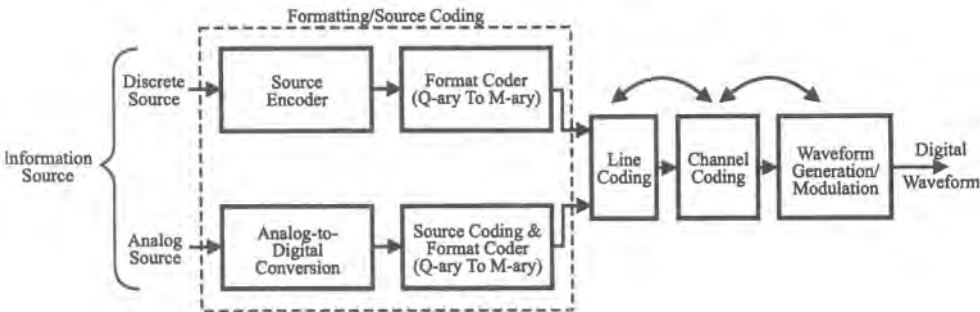


Figure 8.2. Expanded version of the first few blocks of Figure 8.1.

[†]This figure is itself somewhat simplified, but is adequate for present purposes. There are many possible variations to the implementation options. For example, depending on the application, channel coding and line coding may be interchanged: this is the implication of the double-headed arrow straddling those two blocks. The second double-headed arrow is intended to indicate the fact that in some cases there is not a clear separation between channel coding and modulation, as in TCM, for example.

Conversely, if it is desired to simulate explicitly the formatting/source coding block, the analog source of interest must also be explicitly modeled. From the transmission-system-level standpoint, however, this would generally represent a greater level of modeling detail than necessary. Rather than trying to emulate a particular type of analog source, what is perhaps more consistent with system-level simulation is to use analog test signals. These test signals could be used as inputs to the formatting/source coding subsystem if we insisted on modeling that level of detail, or, more often, they might be input directly into the modulator or injected into the system after the modulator, the purpose being to extract the system's analog transmission characteristics.

8.2.1.1. Single Test Tone

Single tones (usually called test tones) are perhaps the most common test signal used in communication systems. For simulation purposes we must of course use sampled values of the test tone. Thus, the discrete-time signal

$$X(t_k) = A \cos(2\pi f_0 t_k + \theta) \quad (8.2.1)$$

is used to represent a test tone within a simulation. The sequence of sampling instants $\{t_k\}$ is always equidistant in simulation and separated by the sampling interval T_s , i.e., $t_k = kT_s$. The frequency f_0 and the amplitude A are varied to obtain the so-called swept-frequency response and the swept-power response of the system. In theoretical models the angle θ is usually assumed to be uniformly distributed over $[0, 2\pi]$. However, the response of a system is normally not sensitive to the value of θ , hence, for simulation purposes the angle θ is usually fixed at some arbitrary value. In bandpass systems, the tone frequency f_0 is measured from the center frequency f_c . That is, the continuous-time signal is

$$X(t) = A \cos[2\pi(f_c + f_0)t + \theta]$$

and its sampled complex envelope counterpart, used in simulation, is

$$\tilde{X}(k) = A \exp(2\pi j k f_0 / f_s) \exp(j\theta) \quad (8.2.2)$$

where $f_s = T_s^{-1}$ is the sampling frequency. Note that f_s is typically set at 8–16 times f_0 . As we noted in Chapter 3, it is necessary that f_s be an integer multiple of f_0 in order that the sampled sinusoid be periodic. Put another way, if this condition is not satisfied, the corresponding simulated spectrum will contain some error.

8.2.1.2. Multiple Test Tones

A set of multiple tones is typically used to evaluate intermodulation distortion in nonlinear systems. The complex envelope representation for a set of tones has the form

$$\tilde{X}(k) = \sum_{n=1}^M A_n \exp\left(2\pi j k \frac{f_n}{f_s} + j\theta_n\right) \quad (8.2.3)$$

where A_n , θ_n , and f_n represent the amplitude, phase, and frequency of the n th tone

$$X_n(t) = A_n \cos[2\pi(f_0 + f_n)t + \theta_n] \quad (8.2.4)$$

Unless there is known to be a particular relationship among the θ_n , they would be assumed independent and uniformly distributed over $[0, 2\pi]$. Thus, a uniform random number generator would be called to generate the θ_n .

We pointed out above that f_s should be a multiple of the tone frequency. In the case of (8.2.3), therefore, this should also be kept in mind when choosing f_n and f_s . That is, unless the tone frequencies are harmonically related, f_s may not be an integer multiple of all frequencies.

8.2.1.3. Filtered Random Processes

A natural generalization of the multiple-tone test signal is a process that contains an infinite number of frequency components. The natural power spectral density (PSD) of the process can be further shaped to some purpose by appropriate filtering. As discussed in Chapter 7, it is very difficult to devise pseudo-random number generators that will imitate arbitrary random processes. Even the less ambitious aim of reproducing a process' PSD and first-order amplitude distribution is generally difficult. Only in the case of a Gaussian random process (GRP) is it relatively straightforward to implement procedures that control both of these attributes. Consequently, it is convenient to use a GRP as an analog test signal.

With the Gaussian assumption, sampled values of spectrally shaped signals can be generated by applying a linear transformation to sequences of independent Gaussian variables. The linear transformation preserves the Gaussian amplitude distribution and alters only the power spectral density.

By choosing the coefficients of the transformation appropriately, one can shape the spectral density of the transformed sequence to match some desired shape as closely as possible. The coefficients of the transformation could be determined, for example, by fitting an autoregressive (AR) or autoregressive moving-average (ARMA) model to the given spectral density. Alternatively, we could use spectral factorization, that is, factor the given spectral density (assuming it is a ratio of polynomials) and construct a filter that contains the poles and zeros of the factored spectral density that lie in the left half of the s plane (refer to Chapters 3 and 7 for more discussion). The transformation can also be implemented, but generally only approximatively, by an FIR filter.

8.2.1.4. Stored Experimental Data

Because of the fact that a time series for a more or less arbitrary process is difficult to reproduce by an RNG, when it is imperative to imitate a given type of signal, one may need to resort to the simple expedient of recording a sample of an actual waveform of interest and use its (time-sampled) values as inputs to a simulation. In the current computing environments, this should pose no memory or run-time difficulties for simulations of reasonable length; in fact, it may be faster than other means of generating a signal. Of course, one has to incur the initial expense and inconvenience of the data gathering.

8.2.2. Digital Sources

A digital source means simply a source which is discrete and has (in practical applications) a finite alphabet. This, of course, applies to sources with a naturally finite alphabet, such as natural languages, or to analog sources that have been quantized. We distinguish a

digital source from a digital signal or digital waveform. The digital source's elements are considered to be logical or abstract entities. The digital signal is a waveform that carries digital information. This distinction is necessary because there are in fact different ways to generate waveform realizations for the same digital source. Digital waveforms will be considered later. Assuming a finite alphabet, a digital source can always be represented by a finite-state machine (FSM). The only information that is needed is the FSM's transition probability matrix, specifying the allowable state transitions and their associated probabilities. Given this matrix, it is straightforward[†] to generate a sequence having the requisite statistics. Transitions are simulated by using a uniform RNG partitioned into the number of allowable “go-to” states in accordance with the corresponding probabilities.

Although simulating digital sources is fairly easy, the question does arise again whether one is well advised to do so. And again, the answer hinges on the tradeoff between verisimilitude and computing burden. If there are features to the typical time series that will have a significant effect on the transmission performance, there may be no choice. On the other hand, there may also be subsequent processing (formatting/source coding) that will change the statistical nature of the digital sequence entering the transmission system. If so, this of course has to be accounted for. We will discuss this point further in the next section.

8.3. Formatting/Source Coding

As we noted above, there can be a sequence of operations that takes an information source into an “equivalent” discrete source,[‡] and each such operation could be treated as a separate subsystem. However, as already hinted, we shall not consider the modeling of these particular subsystems in detail because that is generally beyond the objective of system-level simulation. A short discussion is worthwhile, however, to place these operations in context. For discussion purposes we treat all of these operations as a single subsystem that we are calling the formatting/source coding (FSC) subsystem. In this terminology, we are following Sklar,⁽¹⁾ though the distinction between formatting and source coding is not necessarily standard. In any case, what is generally involved is to convert the source into a form suitable for digital transmission over the particular channel at hand and for the application of interest.

If the source is discrete (say text), FSC converts single characters or groups of characters into corresponding groups of logical (say Q -ary) characters. Source coding often implies that the mapping is done in such a way as to produce “data compression,” that is, reduce redundancy.^(22–24) The Q -ary characters may then be converted (losslessly) into M -ary characters, for example, for purposes of transmission. Because source coding algorithms are discrete operations, no significant modeling issues are involved, hence we will not further discuss any such algorithms. On the other hand, if the source is continuous, there are potentially significant modeling issues involved in the analog-to-digital (A/D) conversion that must be performed to transform it into an “equivalent” discrete source, if indeed the A/D subsystem is simulated at all.

8.3.1. Analog-to-Digital (A/D) Conversion

Analog-to-digital conversion (also often abbreviated ADC) consists of two major steps: sampling and quantization. The first produces a discrete-time sampled sequence $\{X(kT_s)\}$

[†]It is straightforward in principle. For digital sources representing natural languages, for example, a finite-state machine that generates meaningful text would not be practical, though one can visualize it

[‡]We put “equivalent” in quotes because, in general, the equivalence is not quite complete.

from the given continuous process $X(t)$. The sampling operation was discussed in detail in Chapters 3 and 7 and hence will not be repeated here. We note, however, that (ideal) sampling is inherent in simulation. That is, we need not model the sampling equipment per se unless we want to model the A/D subsystem itself at that level of detail (see the Remark below).

Quantization is the heart of the A/D process and, in the simplest case, converts the continuous variable $X(kT_s)$ into a discrete variable X_q having Q values: this is called scalar quantization. More generally, in vector quantization,^(25–27) blocks of values of $X(kT_s)$ are converted into blocks of quantized values.

In the classical scalar quantization scheme, the input voltage range is divided into equal intervals of size Δ ; the i th interval corresponds to $i\Delta \pm (\Delta/2)$. If the input voltage falls in the i th interval, it is assigned the integer i ; if i is written in binary form, the output of the process is a bit sequence and the result termed *pulse code modulation* or PCM. Because the step sizes are equal, this scheme is called *uniform* quantization. Quantization introduces an error $X_q - X$, called quantization noise, some aspects of which have been discussed in Chapter 7.

Nonuniform quantization divides the input range into unequal intervals. In practice, this is accomplished in two steps (Figure 8.3a). First, the signal is processed through a nonlinear device called a *compressor*, and the result is fed to a uniform quantizer. This two-step process is clearly equivalent to nonuniform quantization of the original input. In practice, there are two standard compression schemes, referred to as μ -law and A-law compression. Let v_c be the compressor output, v the input, and V the maximum (or “overload”) input voltage, and define $x_c = v_c/V$ and $x = v/V$. Then, the compressed output for μ -law compression is given by

$$x_c = (\text{sgn } x) \frac{\log(1 + \mu|x|)}{\log(1 + \mu)} \quad (8.3.1)$$

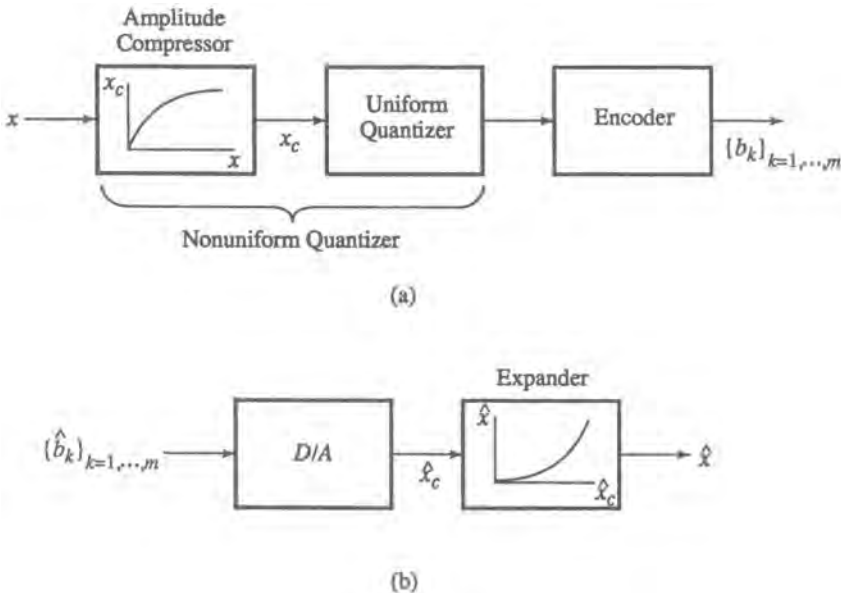


Figure 8.3. Nonuniform quantization process and its inverse, (a) Nonuniform analog-to-digital conversion; (b) nonuniform digital-to-analog conversion.

The expression (8.3.1) is also called *logarithmic* compression; a commonly used value for μ is 255.⁽²⁸⁾

The A-law characteristic is described by

$$x_c = (\text{sgn } x) \frac{A|x|}{1 + \ln A}, \quad 0 \leq |x| \leq A^{-1} \quad (8.3.2a)$$

$$= (\text{sgn } x) \frac{1 + \ln A|x|}{1 + \ln A}, \quad A^{-1} \leq |x| \leq 1 \quad (8.3.2b)$$

The source decoder is shown in Figure 8.3b. The estimate of x_c is simply the inverse mapping from bits to the corresponding integer, i.e.,

$$\hat{x}_c = (\text{sgn } x)(\Delta/2 + i\Delta), \quad \Delta = 2^{-m}, \quad i = 0, 1, \dots, 2^m - 1$$

where m is the number of bits per sample (excluding the sign bit). An estimate of the original input is then provided by the expander, which performs the inverse of compression. Thus, if we express (8.3.1) or (8.3.2) as $(\text{sgn } x) g(x)$, then the expander output is

$$\hat{x}_c = \text{sgn}(\hat{x}_c) g^{-1}(\hat{x}_c) \quad (8.3.3)$$

8.3.2. On Simulating the FSC Subsystem

With regard to simulating digital-to-digital (D/D) conversion, as we have already noted, this is a discrete operation which carries no modeling difficulties. If implemented, D/D conversion is generally fast since it involves only operations on symbols or bits.

With regard to ADC, we note that *ideal* ADC is easily simulated. A physically idealized A/D operation, whether it involves uniform or nonuniform quantization, is basically a memoryless nonlinear transformation. Such a transformation is straightforward to implement, namely read the value of the input signal at the sampling instant and compute or “look up” the corresponding quantizer output value. Real A/D converters depart from this idealization in a number of ways depending upon a number of hardware design considerations and the processing speed.^(29,30) But there would be little point in simulating an ADC, idealized or otherwise, without simulating its counterpart at the receiver end of the link, namely the digital-to-analog converter (DAC). This aspect of the system, while of course important to the eventual application, lies outside the digital transmission system’s purview. Thus, while at least an idealized version of the FSC subsystem can be simulated without difficulty, the question arises whether it is necessary or useful to do so.

Since it is generally the aim of an FSC subsystem to produce an equivalent source with the highest entropy, leaving the redundancy to be put back in a more focused way, we are justified in assuming that the source output has been encoded into an independent binary sequence with the digits “0” and “1” occurring with equal probability, or an independent M -ary sequence with each symbol occurring equally probably. With this assumption, we can replace the source and the FSC subsystem with a random binary (or M -ary) sequence or a PN sequence for simulation purposes: in other words, this is the “equivalent” source that we have been speaking of, and this way of proceeding is of course computationally economical. A perhaps more realistic and possibly not much more computationally demanding model for an equivalent discrete source is one based on a Markov chain, if a specific such model were known to be representative for a particular application.

■ *Remark.* If the primary focus of a design is on the ADC–DAC operations, one may have to simulate the source and FSC subsystem in detail. But one would then be speaking of a subsystem-level simulation, rather than a system-level simulation. In such a simulation, we would want to simulate the subsystem with as much verisimilitude as possible, since the simulation’s objective would generally be to obtain such measures as output signal-to-noise ratio as a function of certain design parameters, where the “noise” here refers to the difference between the source waveform and the reconstructed waveform.

Thus, we might want to consider using here a record of experimental data (e.g., test sentences) as the analog information source. The details of the A/D circuitry, or at least some of the possible imperfections of the process that one might encounter, should be explicitly simulated, such as sampling jitter, aperture error, tracking error, quantizer dc offset, and threshold level errors, among others. Similarly, the D/A operation should be emulated as realistically as possible, though this is not as demanding as the ADC. The effects of some of the impairments mentioned have been investigated analytically (see, e.g., Ref. 31 and references therein), but usually one at a time. Generally, their combined effect can be realistically appraised only through simulation.

In the simulation of this particular subsystem, the transmission system plays a role through the errors that may be produced on the link. Such errors will cause occasional misinterpretation of the transmitted level. This is where a discrete channel model (DCM) for the link (discussed in detail in the next chapter) becomes very useful. As indicated in Figure 8.4, the transmission link is replaced by a DCM, which is essentially an error generator producing errors having temporal occurrences that are statistically consistent with the actual channel. Methods of synthesizing such an error generator are discussed in the next chapter, where we will see that such error sources are typically modeled as hidden Markov models (HMM), which are much more efficient to simulate than the analog link. ■

8.4. Digital Waveforms: Baseband Modulation (I)

At some point in the system, the logical entities must be propagated as waveforms. This may occur at the output of the FSC subsystem, or at the output of some subsequent logical processing, if present, such as a channel encoder. There is generally not a subsystem as such that converts logical entities into waveforms. This function is usually embedded in one of the identified blocks. In any case, we pause in this section to describe the nature of this conversion. We call a waveform associated with a logical sequence a digital waveform. A relatively general representation for such a waveform is⁽³⁾

$$X(t) = \sum_{k=-\infty}^{\infty} q(t - kT; \alpha_k, \sigma_k) \quad (8.4.1)$$

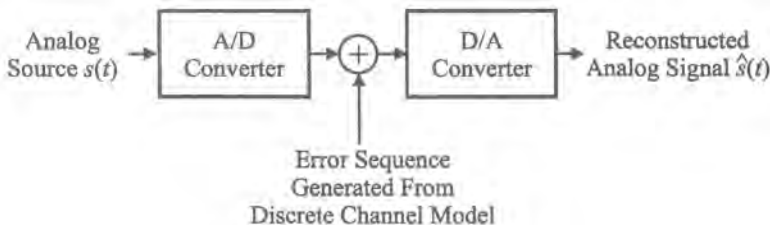


Figure 8.4. Structure of the subsystem-level simulation for studying analog-to-digital and digital-to-analog conversion.

where the symbols have the following meaning. The random sequence $\{\alpha_k\}$, which can be complex, is a stationary sequence of random variables, referred to as the *source symbols*. Here, these symbols would be the some mapping of the output of the FSC subsystem. For example, these symbols can be thought of as amplitudes, complex numbers, or a vector of symbols. The sequence $\{\sigma_k\}$ is a stationary sequence of discrete RVs referred to as the *states* of the waveform. The random waveforms $q(t - kT; \alpha_k, \sigma_k)$ take values in a set $\{q_i(t)\}_{i=1}^M$ of deterministic finite-energy signals. These waveforms may also be complex. They are output sequentially once every T seconds in accordance with a rule that assigns one of these waveforms to every pair (α_k, σ_k) . Here, T can correspond to a single symbol, in the traditional use of the term, or to a block of such.

The expression (8.4.1) can also be thought of as representing a modulator, since input symbols are “mapped” to waveforms, which is one way of conceptualizing modulation. However, this representation may not apply when the input to a waveform modulator is itself a waveform. The representation (8.4.1) can represent a baseband signal or the complex envelope of a modulated carrier, depending upon where we place $X(t)$. For the moment, we shall assume $X(t)$ is a real waveform, hence we can call the process of generating it baseband modulation. We will return to the more general interpretation in Section 8.7.

In its general form, as shown, (8.4.1) is applicable to waveform generators that use finite-state machines (FSM), such as certain types of line encoders (see the next subsection) and to trellis-coded modulation (Section 8.7.4). When there is no FSM structure, $X(t)$ is said to be memoryless and reduces to

$$X(t) = \sum_{k=-\infty}^{\infty} q(t - kT; \alpha_k) \quad (8.4.2)$$

which says that the waveform launched at time kT depends only on the source symbol at that time, α_k . A familiar special case of (8.4.1) occurs when $X(t)$ is not only memoryless, but the waveforms in successive time intervals are scalar multiples of a basis signal $p(t)$,

$$q(t, \alpha_k) = \alpha_k p(t) \quad (8.4.3)$$

so that

$$X(t) = \sum_{k=-\infty}^{\infty} \alpha_k p(t - kT) \quad (8.4.4)$$

Sometimes we represent $p(t)$ as $p(t - D)$, where D is a random delay, to remind ourselves that symbol synchronization must be recovered at the receiver. This subject will be addressed later. Equation (8.4.4) can be used to represent, for example, M -ary PAM if α_k takes on one of M values and $p(t)$ is the standard rectangular pulse.

With respect to simulation, the waveforms $q_i(t)$ or $p(t)$ dictate, through their frequency content, the sampling interval. This is the main implication for the simulator of the nature of these pulses. These waveforms are otherwise constrained only to be of finite energy. They can in principle be of arbitrary shape and arbitrary duration. They can be of limited duration, like the standard rectangular pulse, or of infinite duration, such as a Nyquist pulse (see Section 8.9). Of course, in the latter instance, the pulse has to be truncated at some point.

The models (8.4.1)–(8.4.4), wherever they apply, can be used to represent impairments in the waveform generation, that is, nonideal realizations of the intended pulse shape. For example, suppose one had in mind an ideal rectangular binary waveform. In real equipment, the transitions cannot be instantaneous, hence a relatively simple step toward a more realistic model might bring us to the trapezoidal model in Figure 8.5, which assumes nonzero and not

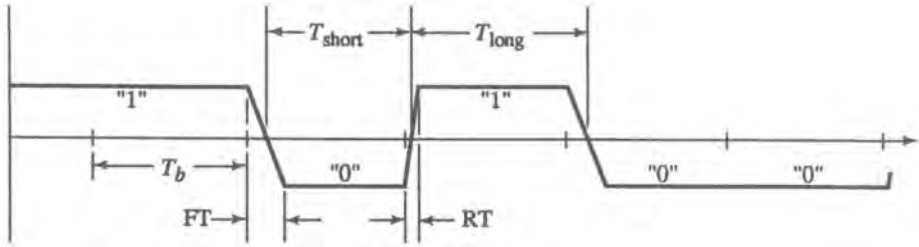


Figure 8.5. One possible model for simulating a digital source waveform. RT , risetime; FT , falltime; nominal bit duration $T_b = (T_{short} + T_{long})/2$.

necessarily equal rise times (RT) and fall times (FT). Note that more realistic models such as this one are not so easy to bring into analysis, but are no more difficult to incorporate in simulation. The model assumes that the signal remains in a given state until “commanded” to switch at the end of its sojourn in that state, when it transitions to the other state. It should be observed, however, that in order to distinguish any difference between this model and the idealized waveform, the sampling must be frequent enough to “see” the rise and fall times.

As relatively simple as it is, this model requires the format of (8.4.1) to describe it. The state diagram in Figure 8.6a shows that q can actually take on one of four different shapes, $p_i(t)$, $i = 1,2,3,4$ (Figure 8.6b), depending upon the “value” of the existing symbol (“1” or

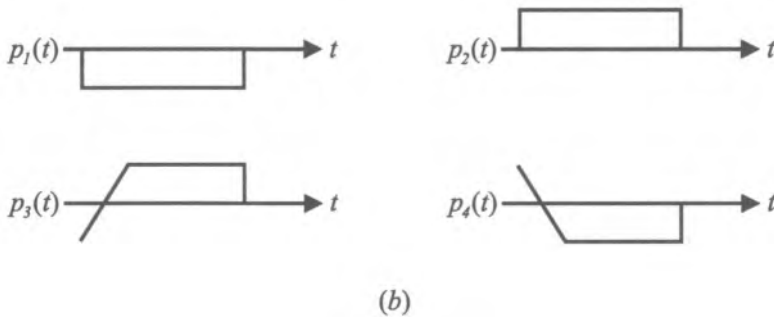
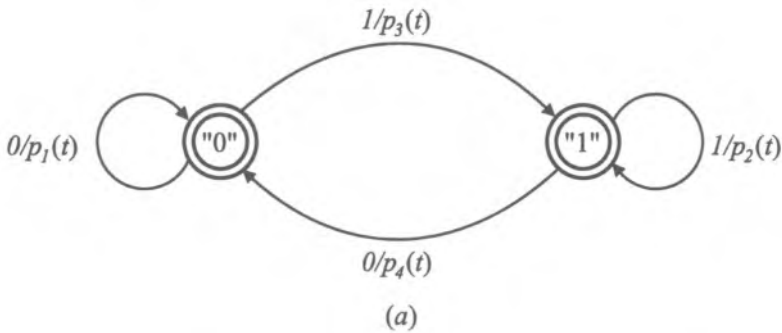


Figure 8.6. Extended representation of the waveform model of Figure 8.5. (a) State diagram; (b) waveforms corresponding to each state transition.

“0”) and the value of the next symbol. The labeling of the branches “ a ”/ $p_i(t)$ indicates the value of the next bit (“ a ”) and the corresponding waveform. In the next section, we will look at some additional mappings from symbols to waveforms.

Note that a waveform at a particular point, such as the one in Figure 8.5, can be used to represent the cumulative effect of successive impairments, instead of simulating each impairment where it arises. If we can find a way properly to describe this cumulative effect, we will generally reduce our computational load since the description (the number of instructions) of a single model will very likely be more compact than for several boxes. In Chapter 2, we labeled this modeling notion an “equivalent random process” methodology. It will be revisited in Section 8.12.3.

8.5. Line Coding: Baseband Modulation (II)

According to Figure 8.2, a “line coding” subsystem may follow the FSC subsystem. Line coding is a type of coding that is done primarily to shape the signal spectrum or to embed certain statistical properties in the signal sequence, such as ensuring a certain transition density in order to aid synchronization. These desirable properties are put to use in baseband transmission as well as in radiofrequency and optical carrier-modulated systems. It is useful to consider line coding in two steps. The first is a *logical-to-logical* mapping which converts a binary or M -ary sequence into another sequence with desirable properties. If the subsequent subsystem also processes logical symbols, the line coding function can be considered to be purely a logical-to-logical conversion. If, as is often the case, what follows is conversion of the symbols to a waveform, then this *logical-to-waveform* mapping can also be considered modulation. There is no reason why this mapping cannot be represented by the general form in (8.4.1). But here we limit ourselves to illustrating the situation with a few standard baseband schemes.

From the simulation standpoint, what in the line coding/baseband modulation needs to be explicitly simulated? The logical-to-logical mapping has to be simulated only if its use (after conversion to a waveform) will alter the spectrum relative to what it would have been otherwise. For, the distortion experienced by a waveform will generally depend on its frequency content as it passes through various kinds of filtering on its way to the receiver.

8.5.1. Logical-to-Logical Mapping I: Binary Differential Encoding

To avoid confusion, we shall label logical sequences by lower case letters. This mapping converts an input (0, 1) binary sequence $\{a_m\}$ into a new (0, 1) sequence $\{b_m\}$ which conveys the information through changes between the current output bit and the next source bits. Symbolically, the relationship between the sequences is given by

$$b_m = a_m \oplus b_{m-1} \quad (8.5.1)$$

where $\oplus$ represents modulo 2 addition.

Many line coding schemes can be represented in terms of Markov chains, the state diagram for which is often more enlightening than the analytic representation. The mapping above can be so represented, as shown in Figure 8.7, where states S_0 and S_1 correspond, respectively, to “0” or “1” for the previous output, and the branch labels indicate the new input symbol followed by the new output symbol.

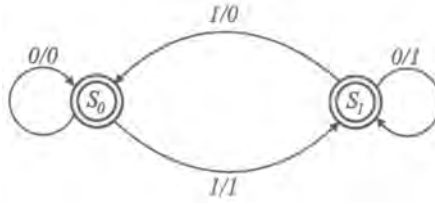


Figure 8.7. State diagram for “precoding,” or binary differential encoding.

8.5.2. Logical-to-Logical Mapping II: Correlative Coding

This mapping converts an M -ary sequence $\{b_m\}$ into a Q -ary sequence $\{c_m\}$ with properties considered to be desirable, particularly in the frequency domain (see, e.g., Refs. 32 and 33 for a good discussion). Typically, the M -ary source sequence $\{a_m\}$ is first “precoded,” to avoid error propagation, into sequence $\{b_m\}$. Precoding with $M = 2$ is equivalent to differential encoding. Pulse waveforms associated with the encoded sequence are referred to as *partial response signaling*, and will be considered in Section 8.5.9. The precoded sequence is obtained through the operation (see, e.g., Ref. 3)

$$b_m = \frac{1}{g_0} \left\{ a_m - \sum_{i=1}^{N-1} g_i b_{m-i} \right\} \bmod M \quad (8.5.2)$$

and the coded sequence is given by

$$c_m = \sum_{i=0}^{N-1} g_i b_{m-i} \quad (8.5.3)$$

where Q , the alphabet size of $\{c_m\}$, is given by

$$Q = (M - 1) \sum_{i=0}^{N-1} |g_i| + 1 \quad (8.5.4)$$

The coefficients $\{g_i\}$ in (8.5.3) are called *partial response* coefficients, and determine the properties of the resulting waveform. The parameter N is referred to as the memory of the encoding process.

■ **Remark.** A variety of partial response signals have been devised. These differ from one another by the set of coefficients $\{g_i\}$. The most common classes of sets are capsuled in Table 8.1. A concise way of describing these coefficients is through a polynomial in the delay variable D , i.e., $F(D) = \sum g_i D^i$. ■

8.5.3. Logical-to-Logical Mapping III: Miller Code

This code arises as the concatenation of a (d, k) code with $d = 1$ and $k = 3$ and the binary differential encoding above (see, e.g., Ref. 12). The resulting code is called a Miller code, as is the (d, k) code itself. The code has rate $1/2$, and is best understood from the state diagram in Figure 8.8. The previous output pair of binary digits represent the state, and the branch labels indicate the new input bit followed by the new output pair.

Table 8.1 List of Partial Response (PR) Polynomials

Nomenclature	$F(D)$
Duobinary, or class I PR	$1 + D$
Dicode, bipolar, or alternate-mark-inversion	$1 - D$
Class II PR	$1 + 2D + D^2$
Class III PR	$2 + D - D^2$
Modified duobinary, or class IV PR	$1 - D^2$
Class V PR	$1 - 2D^2 + D^3$

8.5.4. Logical-to-Real Mapping I: Non-Return-to-Zero (NRZ) Binary Signaling

Non-return-to-zero (NRZ) signaling is the “traditional” logical-to-real mapping and is commonly used for carrier modulation schemes such as PSK. The logical sequence can be directly the source output or a coded sequence. The mapping is given by

$$1 \rightarrow Ap(t) \tag{8.5.5a}$$

$$0 \rightarrow -Ap(t) \tag{8.5.5b}$$

This case corresponds to (8.4.3) with $\alpha_k \in (\pm A)$, where A is a real number. Generally, any pulse shape $p(t)$ with desirable properties can be used. The choice of $p(t)$ will generally depend on the nature of the channel following the pulse modulator. The most commonly assumed pulse is the unit rectangular pulse,

$$p_T(t) = \begin{cases} 1, & 0 \leq t \leq T \\ 0, & \text{elsewhere} \end{cases} \tag{8.5.6}$$

The notation $p_T(t)$, as in (8.5.6), will be used consistently in this subsection. (Note that this is a slight variation from the way it is used in Chapter 3.)

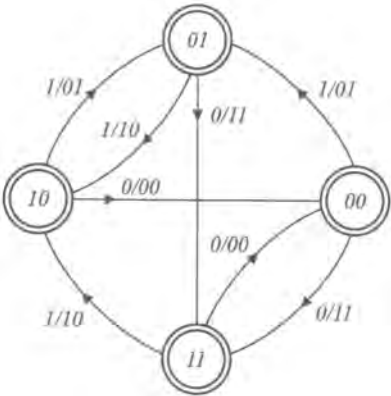


Figure 8.8. State diagram for the “Miller” code.

8.5.5. Logical-to-Real Mapping II: NRZ M -ary Signaling (PAM)

This mapping is an extension of binary NRZ to M -ary alphabets. It produces what is commonly called pulse amplitude modulation (PAM), namely,

$$s_m(t) = A[2a_m - (M - 1)]p_T(t) \quad (8.5.7)$$

This mapping yields $M/2$ pairs of antipodal signals, that is $\alpha_k \in \{\pm A, \pm 3A, \dots, \pm (M - 1)A\}$.

8.5.6. Logical-to-Real Mapping III: Return-to-Zero (RZ) Binary Signaling

RZ signaling is another method of associating a pulse waveform with a sequence. The signal always contains a discrete spectral component. The mapping takes the form

$$0 \rightarrow 0 \quad (8.5.8a)$$

$$1 \rightarrow \begin{cases} Ap_{T/2}(t), & 0 \leq t \leq T/2 \\ 0, & T/2 \leq t \leq T \end{cases} \quad (8.5.8b)$$

Thus, $\alpha_k \in \{0, A\}$.

8.5.7. Logical-to-Real Mapping IV: Biphase Signaling or Manchester Code

This method of generating real waveforms corresponding to a logical sequence has three useful properties. It has precisely zero average value, i.e., no dc wander; it has a spectral null near dc, which is desirable for channels with poor dc response, or allows insertion of a discrete carrier component in carrier transmission; and it is self-synchronizing in the sense that there is a zero crossing in the middle of every bit. The mapping is given by

$$0 \rightarrow A[p_{T/2}(t) - p_{T/2}(t - T/2)] = Ap_b(t) \quad (8.5.9a)$$

$$1 \rightarrow -Ap_b(t) \quad (8.5.9b)$$

8.5.8. Logical-to-Real Mapping V: Miller Code or Delay Modulation

The association of waveforms to the Miller code above sometimes retains that name, but is also called delay modulation. If the output digits are converted to waveforms using the NRZ mapping, and each of the coded digits has duration $T/2$, then the four pairs of bits are mapped out as follows:

$$11 \rightarrow p_T(t) \quad 00 \rightarrow -p_T(t)$$

$$10 \rightarrow p_b(t) \quad 01 \rightarrow -p_b(t)$$

The spectrum of this process is plotted in Figure 8.9b.

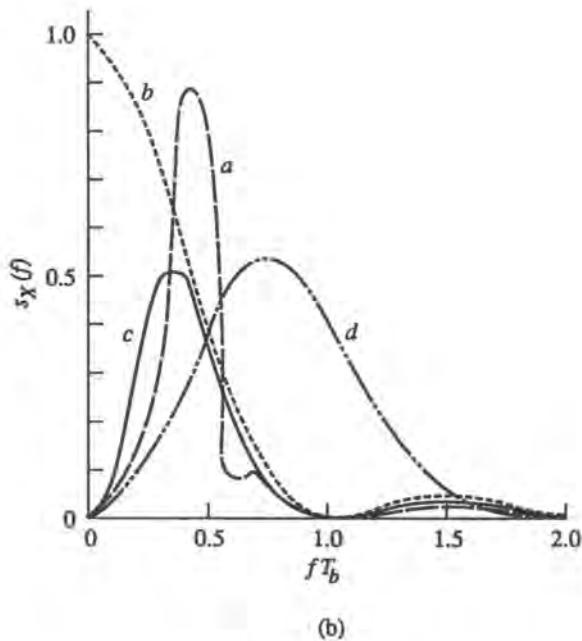
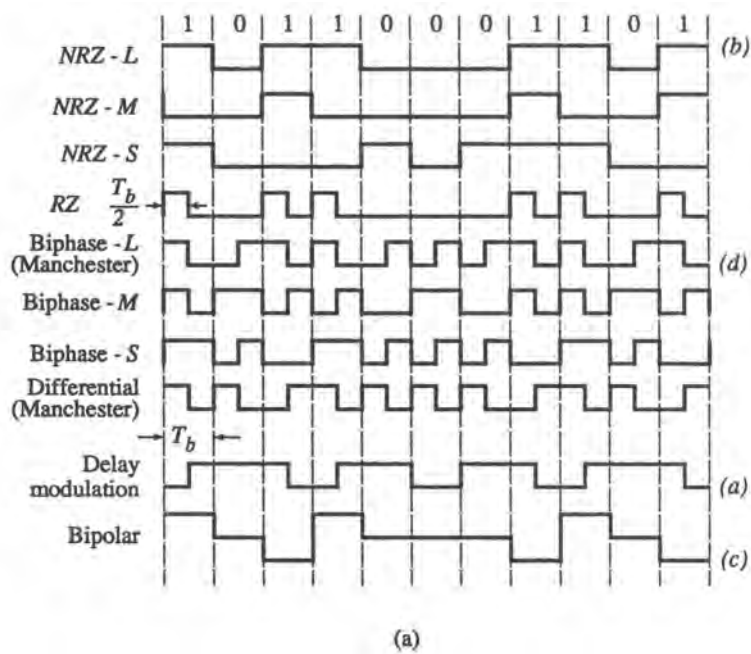


Figure 8.9. (a) Some digital signaling line-coding formats; (b) power spectra of several binary data line-coding formats, a, Delay; b, NRZ-L; c, NRZ bipolar; d, Manchester.

8.5.9. Logical-to-Real Mapping VI: Partial Response Signaling

This mapping may be thought of as the method of associating real waveforms with correlative encoding. The typical pulse is given by

$$s_m(t) = A[(Q - 1) - 2c_m]p(t) \quad (8.5.10a)$$

Assuming $c_m \in (0, 1, 2, \dots, Q - 1)$, we find that the above equation creates $Q/2$ pairs of antipodal signals. The range of c_m will vary depending upon the coefficients g_i (Section 8.5.2). If they are naturally balanced around zero, then we have simply

$$s_m(t) = Ac_m p(t) \quad (8.5.10b)$$

The pulse $p(t)$ is often assumed to be a *Nyquist pulse*, which has desirable zero-crossing properties (Section 8.9.2). This pulse describes the wanted overall system response, which is a composite of the responses of the individual elements of the system. When we simulate the system, we do not directly produce the system response, because we simulate on a block-by-block basis. Thus, an equation like (8.5.10b) describes the transmitter output, where the pulse $p(t)$ is appropriately chosen, but is not the overall response.

The above equation can be represented in an alternative fashion. Let

$$q(t) = \sum_{i=0}^{N-1} g_i p(t - iT) \quad (8.5.11)$$

represent a *partial response* pulse. Then, we have

$$s_m(t) = A[2b_m - (M - 1)]q(t) \quad (8.5.12)$$

where $\{b_m\}$ is the precoded sequence (Section 8.5.2), assumed to be in the set $(0, 1, 2, \dots, M - 1)$.

A summary of the definitions of the line-coding formats discussed earlier, as well as some others, is given in Table 8.2. A pictorial representation of the effect of each of these formats on a particular logical sequence is shown in Figure 8.9a. As was mentioned earlier, one of the objectives of line coding is spectral shaping. As an illustration of the way in which line coding can have such an effect, Figure 8.9b shows the power spectral density of several of these formats, assuming the basic pulse shape is rectangular.

8.6. Channel Coding

Shannon^(24,34) proved that it is possible to transmit information with arbitrarily high reliability provided only that the rate of transmission R does not exceed a certain rate C called the channel capacity. It is well known, for example, that for an AWGN channel with bandwidth B and noise PSD N_0 W/Hz, the channel capacity is

$$C = B \log_2(1 + S/N) \quad [\text{bits/s}] \quad (8.6.1)$$

where S is the average signal power and $N = N_0 B$ is the noise power. Another well-known example is the M -ary symmetric channel, which has M inputs and M outputs with the

Table 8.2 Definition of Some Digital Line-Coding Formats

Nonreturn-to-zero-level (NRZ-L)
1 = high level
0 = low level
Nonreturn-to-zero-mark (NRZ-M)
1 = transition at beginning of interval
0 = no transition
Nonreturn-to-zero-space (NRZ-S)
1 = no transition
0 = transition at beginning of interval
Return-to-zero (RZ)
1 = pulse in first half of bit interval
0 = no pulse
Biphase-level (Manchester)
1 = transition from high to low in middle of interval
0 = transition from low to high in middle of interval
Biphase-mark
Always a transition at beginning of interval
1 = transition in middle of interval
0 = no transition in middle of interval
Differential Manchester
1 = no transition at beginning of interval
0 = transition at beginning of interval
Always a transition in middle of interval
Delay modulation (Miller)
1 = transition in middle of interval
0 = no transition if followed by 1
Transition at end of interval if followed by 0
Bipolar
1 = pulse in first half of bit interval, alternating polarity from pulse to pulse
0 = no pulse

probability q of correct reception and the probability of a transmitted symbol being received as any one of the other $M - 1$ symbols equal to $p/(M - 1)$. For this channel we have

$$C = \log_2 M + q \log_2 [p/(M - 1)] \quad [\text{bits/symbol}] \quad (8.6.2)$$

In order to attain the theoretical predictions, research has proceeded historically along three main lines. One has been to seek appropriate modulation schemes, another has been to devise appropriate coding[†] schemes, and the third has been to find good combinations of modulation and coding. In some coding schemes, the receiver can request a retransmission from the transmitter. Those coding arrangements in which the receiver attempts by itself to detect or correct errors are also called *forward* error-correction (FEC) schemes. Until relatively recently, the only practical solution has been in the coding arena, and this will be the subject of this section. Combined modulation and coding techniques, which are being increasingly used, are addressed in Section 8.7.

Traditionally, coding methods have been viewed as belonging to one of two categories, block coding and convolutional coding. In either case, the use of redundant bits (without

[†]Various terminology exists for the type of coding in question. The terms channel coding, error-correction coding, and error-control coding are in common usage.

alphabet change) necessitates greater bandwidth. Thus, bandwidth is traded for performance, the latter meaning a reduced average bit error probability (BER), for a given signal power, or conversely, a reduced power to achieve a given error probability. The design and operation of error control coding is complex and is dealt with in many excellent textbooks.^(19,35–40) Even there, the typical environment assumed is either the AWGN channel or a distortionless one, so that the advantage of coding (usually capsuled in the term *coding gain*) will usually require the assistance of simulation to evaluate. Our objective in this section, therefore, is mainly to outline the issues involved with simulating systems that incorporate FEC, and to outline the approaches available to the simulation practitioner to deal with channel coding. Here we shall take a detour from the sequential consideration of the blocks in Figure 8.1 and discuss both *encoding* (on the transmit side) and *decoding* (on the receive side) since they are so intertwined.

There are in effect two main approaches to evaluating the effect of coding. One of these does attempt to emulate directly the encoding and decoding algorithms, and we will briefly discuss the implications of this approach below. But first we will speak about the alternative approach, which is generally the more practical one.

This alternative is based on partitioning the link into two independent entities, the encoder/decoder pair (the *codec*) and the equivalent *discrete channel* that represents the input/output interface to the codec. (A detailed discussion of the discrete channel representation is given in the next chapter.) Each entity can be characterized separately, and if it can be justified, we are free to do this characterization in an expedient manner, such as using good and easy-to-compute approximations. Why might we want to approach the problem in two parts? Basically, the reasons are computational efficiency and flexibility.

One good reason to separate the channel from the codec is that, in order to characterize the (discrete) channel, we generally have to conduct a waveform-level simulation, which requires modeling every block between the encoder and decoder at many samples per symbol, while a codec by its very nature deals only with a single number per symbol, that being the logical value of the input symbol at the encoder input, or the (generally quantized) value of the sampled waveform at the decoder input. If we wanted to try different codecs with the same discrete channel, we would not have to simulate again the analog channel underlying the discrete channel. We would only need to use the developed discrete channel model, which runs at the symbol rate, not the sampling rate. The computational savings may or may not be greatly significant depending upon the complexity of the codec relative to the rest of the system. In any case, the separation between the two is useful in providing insight to the system engineer.

The explicit simulation of a codec is generally not necessary for purposes of system performance evaluation because one can usually apply adequate approximations or bounds. Avoiding explicit simulation of the codec is desirable for two reasons. First, the computation involved with duplicating the encoding/decoding algorithm is eliminated. (Below, we will estimate the associated computational burden.) Perhaps more importantly, when the purpose of the code is to lower the error probability, the BER target is often quite low, say 10^{-6} or less. Here we have the typical difficulty that in Monte Carlo simulation the corresponding run length becomes burdensome to prohibitive. This difficulty is explained in detail in Chapter 11. When the purpose of the code is to lower the BER, the channel BER (in other words, the performance of the discrete channel) into the decoder is often high enough to be computationally manageable. The projection of the channel BER into that of the decoder output is amenable to the analytical treatment alluded to above. This method is also discussed in Chapter 11. As explained there, the encoder does not have to be explicitly emulated; the only

the figure that are nonzero correspond to multiplication in $\text{GF}(q)$. Thus, in this case there are $k \cdot wt(\mathbf{g})$ field additions and $k \cdot wt(\mathbf{g})$ field multiplications per codeword.

A block diagram of a BCH decoder is shown in Figure 8.11. (The class of BCH codes also includes RS codes.) The decoding algorithm takes place in several steps. The first step is the syndrome computation. Based on Horner's method⁽⁴¹⁾ for evaluating polynomials, one can deduce on average that this step requires $t(n - 2)$ field multiplications and $t(n - 2)$ field additions, where the field is $\text{GF}(q^m)$, m being defined as $\min\{m | q^m \equiv 1 \pmod{n}\}$, with the understanding that q and n are relatively prime.

The second step, if the syndrome is not zero, is to compute the error locator polynomial $\omega(x)$, which we assume is done using the Berlekamp–Massey algorithm.^(35,36) The complexity of the Berlekamp–Massey algorithm is bounded above by

$$\sum_{j=0}^{t-1} [(j+1) \text{ FM} + j \text{ FA}] + t \text{ IA} \quad (8.6.4)$$

where FM and FA stand for field multiplications and field additions, respectively, and IA stands for integer additions. This computation is for one code word.

If the degree of ω is less than or equal to 2, a table lookup can be used to find the error locations. For moderate block lengths, the required memory is of practical size. If the degree of ω is ≥ 3 , we have to find roots of the error locator polynomial: this gives the location of the errors. The standard procedure is the Chien search.⁽³⁶⁾ An upper bound to the computational work for finding the roots can be established, assuming again the use of Horner's method for evaluating polynomials. Since the error locator polynomial is a polynomial of degree t with coefficients in $\text{GF}(q^m)$, we conclude that finding its roots requires at most $(t \text{ FM} + t \text{ FA})q^m$ field operations.

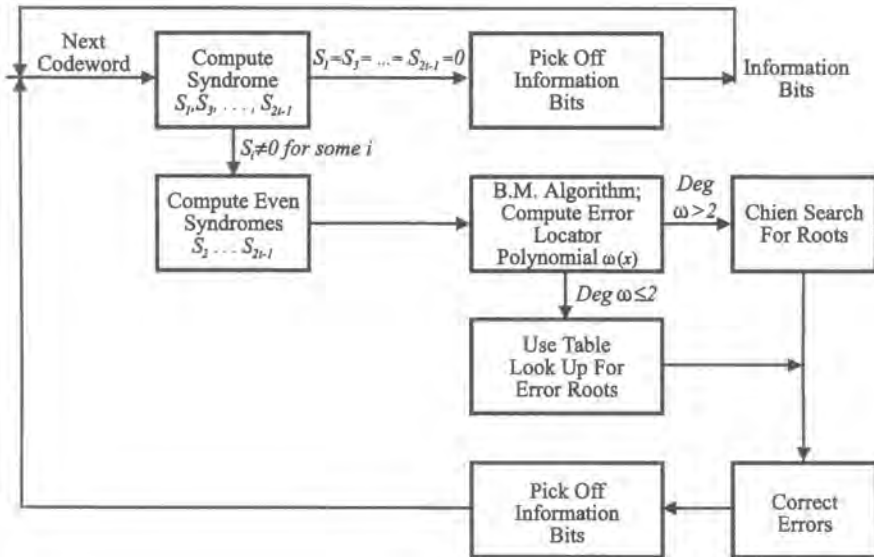


Figure 8.11. Procedure for decoding a BCH (also RS) block code.

Finally, if the code is nonbinary, we have to determine not merely the location of an error, but its “value.” To do this, we can use Forney’s algorithm (see, e.g., Ref. 36). This evaluation requires approximately the following operations:

$$t[(2t - 1) \text{ FM} + (2t - 1) \text{ FA} + (t - 1) \text{ FM} + (t - 1) \text{ FA} + \text{FD}]$$

where FD denotes field division.

We can now summarize the computational complexity per “chip” (a coded symbol). In the current context there are n chips per codeword:

- *Encoding:* $R_c wt(g) \text{ FA}$ (binary); $R_c[wt(g) \text{ FA} + wt(g) \text{ FM}]$ (nonbinary)
- *Decoding:*

Syndrome computation	$t(n - 2)/n \text{ FM} + t(n - 2)/n \text{ FA}$
Berlekamp–Massey algorithm	$t(t + 1)/2n \text{ FM} + t(t - 1)/2n \text{ FA} + t/n \text{ IA}$
Error locator polynomial	$q^m/n(t \text{ FM} + t \text{ FA})$
Error evaluation	$t(3t - 2)/n \text{ FM} + t(3t - 2)/n \text{ FA} + \text{FD}/n$

Because of the wide range of parameters (n, k, t) and different possible values of q , the computational requirements are not immediately obvious from this listing. In order to get a sense of the complexity of decoding, let us take the following simple example. Let us assume the binary BCH (255, 231, 3) code, so $n = 255, k = 231, t = 3, q = 2$, and $m = 8$. Since this is a binary code, the error evaluation step is not done. Using the above table, we find the number of arithmetic operations (making no distinction here between the different types of operations) to be about 34 per chip.

In order to assess the significance of this computational load in the system context, it will be interesting to compare it to the computation required for perhaps the most common operation in simulation, namely FIR filtering (see Chapter 4). FIR filtering is the discrete convolution $y_j = \sum_{i=0}^{N-1} h_i x_{j-i}$, where y is the output, x is the input, and h is the impulse response, which is assumed truncated to N samples. Letting M be the filter memory in chips and f_s the simulation sampling rate (samples/chip), we obtain $N = Mf_s$. Hence, for each output sample of a FIR filter we have Mf_s floating-point multiplications (FPM) and $Mf_s - 1$ floating-point additions (FPA). To process the waveform for one chip duration we then have Mf_s^2 FPM and $Mf_s^2 - 1$ FPA. For a more or less typical values of $M = 5$ and $f_s = 16$ we have 80 FPM and 79 FPA for the computation of one output sample, or, for simplicity, about 160 arithmetic operations (if again we do not distinguish between types); and for one chip duration we have about $160 \times 16 = 2560$ operations. Thus, for this code, explicit encoding and decoding would clearly not add a great deal of computing time to a typical simulation. (It is true that the comparison between FIR filtering and decoding is not quite equivalent, in the sense that one is a sample-spaced waveform processing floating-point operation, while the other is a symbol-spaced field-arithmetic operation, but our only objective here is indeed to obtain a sense of the relative computational load.) From the table above we can see that the decoding complexity increases with t or t^2 (depending on the decoding step), so for more powerful codes the computation will increase. For the majority of practical applications, however, t is relatively small. Hence, we can say that for many, if not most, cases of interest, simulating the encoding and decoding algorithms per se will increase the run time by only a

small factor. Rather, for small BER, it is the statistical aspect mentioned above that will burden the run time.

8.6.2. Computational Load for Convolutional Coding/Decoding

As with block codes, there are different possible implementations for convolutional coding and decoding. For the latter, in particular, we will assume decoding is done by using the Viterbi algorithm (VA).⁽⁴²⁾

The parameters for a convolutional code can be defined with reference to the general block diagram for a binary convolutional encoder shown in Figure 8.12. At the input, the encoder accepts k bits at a time and reads out n bits, hence the code rate is k/n . The *constraint length* is defined as L (bytes). As will be seen, these parameters dictate the complexity of the decoder. As before, we can associate a polynomial with the connections to the modulo-2 adders, actually n polynomials, one for each modulo-2 adder. Each polynomial has degree Lk and the coefficient is 1 if there is a connection from a given shift register stage to the modulo-2 adder in question, and zero otherwise. Thus, if we call w_i the weight of the polynomial representing the connections to the i th modulo-2 adder, it is clear that there are $\sum_{i=1}^n (w_i - 1)$ modulo-2 additions for every k bits. This defines the computational complexity of the encoder.

The Viterbi algorithm (VA) is well known, and described in detail in numerous publications (e.g., Refs. 19, 42, 43, among others). In the following, we will assume familiarity with the VA and describe its operation only in the briefest terms to set terminology. Our aim is to describe its computational complexity for the purpose of assessing the burden of explicitly emulating it in simulation. However, any such finding must be regarded as a rough calculation (which is all we need here) because there are different possibilities for implementing the VA both in software and hardware (see, e.g., Ref. 44 for a recent implementation proposal). We will consider only a “straightforward” implementation.

The VA finds the most likely “path,” that is, some sequence of symbols, through a trellis with N_s nodes, called the states of the code, where $N_s = 2^{k(L-1)}$. The decoder keeps score via a “metric” of the relative probability of different paths terminating on each of the nodes. Evidently, there can be only one most likely path terminating on each of the nodes (assuming

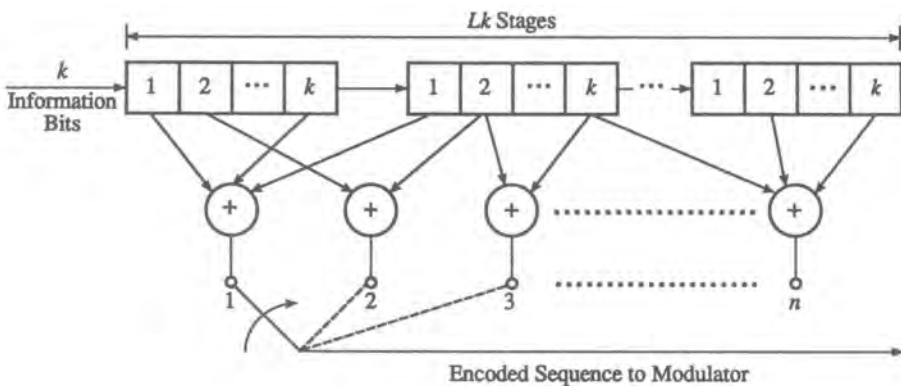


Figure 8.12. General convolutional encoder (from Ref. 11, © McGraw-Hill, 1989, reproduced with permission).

no “ties”) at each discrete time $l^\dagger$. Each such path is called a *survivor* path and we will call the associated scores the survivor metrics. Given the survivor metrics at discrete time $l - 1$, we proceed to find the survivors at time l in the following fashion. Let $\mathcal{I}(j) = \{j_1, j_2, \dots, j_{2^k}\}$ be an index set representing the possible 2^k transitions to state j from other states. We now need to calculate for each state another metric representing the relative probability of each of the 2^k transitions into it. This metric is called a *branch* metric, and we will denote by $Z_{j_i, j}^l$ the branch metric associated with the transition from state j_i at time $l - 1$ to state j at time l . Thus, there are 2^k branch metrics to be calculated for each of the N_s states, that is, 2^{Lk} metrics in total.

Each branch metric calculation consists of the operation

$$Z_{j_i, j}^l = \sum_{m=1}^n c_m(l; j_i, j) y_m(l) \quad (8.6.5)$$

where $c_m(l; j_i, j)$ represents the value (± 1) of the m th code bit (out of n) on the transition from state j_i to state j at time l , and $y_m(l)$ is the corresponding demodulator output. Equation (8.6.5) represents a correlation between the actual demodulator output and the decoder output that would have been produced by the transition in question. The demodulator output is initially a continuous variable, usually quantized to a small number of bits, typically three. Thus, the operation in (8.6.5) involves n integer multiplications (IM) and $n - 1$ integer additions.

In order to obtain the survivor path into state j , say, we add each of the metrics in (8.6.5) to the metric of the survivor at time $n - 1$ for $j_i \in \mathcal{I}(j)$. We can represent this process as follows. Calling $C_j^{(l)}$ the path metric survivor at time l into state j , we have

$$C_j^{(l)} = \max_{j_i} \left\{ C_{j_i}^{(l-1)} + Z_{j_i, j}^l \right\}, \quad j_i \in \mathcal{I}(j) \quad (8.6.6)$$

Equation (8.6.6) is often called the “add–compare–select” (ACS) operation. Evaluating (8.6.6) implies performing 2^k integer additions and a “max of” comparison: this computation is done for each state, that is, N_s times.

All of the computations indicated above are performed on a “per symbol” basis, that is, for every n coded bits. We can therefore summarize the computational complexity *per output “chip”* as follows:

- Encoding: $\leq (Lk - 1)$ modulo-2 addition (M2A)
- Decoding:

Branch metrics: 2^{Lk} IM + $2^{Lk}(n - 1)/n$ IA

ACS: $2^{Lk}/n$ IA + $2^{Lk}(L-1)/n$ COMP

Here IM is integer multiplication, IA integer addition, and COMP a comparison (and retention).

As before, to get a sense of the relative computational burden, we compare this estimate to the amount of work performed for FIR filtering, per chip. For each filter output sample this was shown to be Mf_s FPM and $Mf_s - 1$ FPA, and for processing one chip duration of waveform these become Mf_s^2 FPM and $Mf_s^2 - 1$ FPA. Using the same values as above, $M = 5$ and $f_s = 16$, we have for each output sample 80 FPM and 79 FPA, or roughly 2560 operations per chip time. For the “classic” rate-1/2, constraint-length-7 binary convolutional code, the preceding estimates translate to 6 M2A for encoding and 128 IM, 192 IA, and 16 COMP per code chip. Thus, the computational work for explicitly replicating the encoding/decoding of a

[†]Here, we assume that l takes on integer values $0, \pm 1, \pm 2, \dots$, representing symbol duration increments.

convolutional code (via the VA) is not trivial, but not overwhelming, and still small compared to FIR filtering one chip's worth of waveform. Depending on the processor used, which determines the relative duration of each operation, the computational effort can be likened roughly as adding a filter at most. Thus, we can conclude, as before, that emulating the encoding and decoding algorithms for convolutional codes adds relatively little to the run time if the system is moderately complex to begin with.

8.7. Radiofrequency and Optical Modulation

Modulation is one of the most important signal processing operations that takes place in a communication system. It can be effectively used to match a signal with the channel characteristics, to minimize the effects of channel noise, and to provide the capability to multiplex many signals. Perhaps the most important function is to transmit a lowpass signal $X(t)$ over a bandpass channel centered at f_c . For example, we can translate the spectrum of $X(t)$ by multiplying it with a carrier $C(t)$ of the form

$$C(t) = A \cos(2\pi f_c t + \theta) \quad (8.7.1)$$

where A is the amplitude of the carrier, f_c is the carrier frequency, and θ is an arbitrary phase constant assumed to be a random variable uniformly distributed in the interval $[-\pi, \pi]$. The product (or modulated) signal $Y(t)$,

$$Y(t) = C(t)X(t) = AX(t) \cos(2\pi f_c t + \theta) \quad (8.7.2)$$

has a power spectral density

$$S_{YY}(f) = \frac{A^2}{4} [S_{XX}(f - f_c) + S_{XX}(f + f_c)] \quad (8.7.3)$$

Equation (8.7.3) shows that multiplying $X(t)$ by a carrier $C(t)$ translates the spectrum of $X(t)$ by the carrier frequency. Thus, the lowpass signal $X(t)$ is transformed to a bandpass signal suitable for transmission over a bandpass channel. All methods of modulating a carrier can be shown to have a similar effect in the sense of centering the modulated spectrum around the carrier frequency. However, not all modulation methods have the same bandwidth properties, which are important to know in order properly to select the sampling rate.

A modulation method can be thought of as an operator on the baseband signal. In particular, the *amplitude modulation* in (8.7.2) can be viewed as an operator $\mathcal{L}$,

$$\mathcal{L}(X) \equiv AX(t) \cos(2\pi f_c t + \theta) \quad (8.7.4)$$

which is time-varying but linear, for

$$\mathcal{L}(X_1 + X_2) = \mathcal{L}(X_1) + \mathcal{L}(X_2) \quad (8.7.5)$$

which implies no bandwidth expansion relative to the carrier frequency. Viewed another way, we can see that (8.7.2) can be written as

$$Y(t) = \text{Re}\{AX(t)e^{j\theta}e^{j2\pi f_c t}\}$$

Hence, the complex envelope of $Y(t)$ has the same bandwidth properties as $X(t)$. This means the sampling rate adequate for representing $X(t)$ is also adequate for $Y(t)$. If (8.7.5) does not hold, the modulation method is nonlinear, and the modulated spectrum is not merely a translation of the baseband spectrum. Then, as discussed in Chapter 5, bandwidth expansion will (generally) take place. This implies that the baseband waveform fed to such a modulator will generally have to be sampled at a higher rate, to anticipate this bandwidth expansion.

Much more detail about various aspects of modulation methods can be found in more specialized references, e.g., Refs. 3, 5, 8, 11 and 13, among many others.

8.7.1. Analog Modulation

In analog modulation, the modulating signal is one whose amplitude can take a continuum of values, as in speech or images. The signal

$$Y(t) = AX(t) \cos(2\pi f_c t + \theta) \quad (8.7.6)$$

is said to be amplitude-modulated since $X(t)$ modifies the amplitude of $C(t)$.

In a similar fashion, we can modulate the phase or frequency of $C(t)$ according to

$$Z(t) = A \cos[2\pi f_c t + \theta + k_p X(t)] \quad (8.7.7)$$

or

$$W(t) = A \cos\left[2\pi f_c t + k_f 2\pi \int X(t) dt + \theta\right] \quad (8.7.8)$$

where the instantaneous frequency deviation is given by

$$f_i(t) = k_f X(t) \quad (8.7.9)$$

and k_p and k_f are constants. [When $X(t)$ is normalized so that $|X(t)| \leq 1$, k_p and k_f are referred to as the phase deviation and frequency deviation, respectively; and the ratio of the peak frequency deviation to highest frequency in the modulating signal is called the modulation index.] $Z(t)$ is called a phase-modulated carrier and $W(t)$ is called a frequency-modulated carrier. The simulator should bear in mind that when the modulation index is large, the bandwidth of $W(t)$ can be many times larger than that of $X(t)$. Therefore, the simulation sample rate should be chosen accordingly.

In general, a modulated carrier can be represented in quadrature form as

$$Y(t) = X_1(t) \cos(2\pi f_c t + \theta) - X_2(t) \sin(2\pi f_c t + \theta) \quad (8.7.10)$$

where $X_1(t)$ and $X_2(t)$ are lowpass processes with bandwidth B , and the carrier frequency f_c is typically $\gg B$. The $X_1(t)$ and $X_2(t)$ could be two independent analog signals or they may be

Table 8.3 Quadrature Representation of Analog Modulation Schemes

Modulation scheme	$X_1(t)$	$X_2(t)$	Comments
Amplitude modulation (AM)	$a[1 + k_a X(t)]$	0	k_a -modulation index
Quadrature AM	$X_1(t)$	$X_2(t)$	
Double sideband (DSB)	$X(t)$	0	
Single sideband $X(t)$	$X(t)$	$\hat{X}(t)$ -Hilbert transform of $X(t)$	
Phase modulation	$\cos[k_p X(t)]$	$\sin[k_p X(t)]$	k_p -modulation sensitivity
Frequency modulation	$\cos[k_f \int X(\alpha) d\alpha]$	$\sin[k_f \int X(\alpha) d\alpha]$	k_f -modulation sensitivity

uniquely related to a common signal $X(t)$ as in the case of some modulation schemes such as vestigial sideband modulation or single-sideband modulation. In the case of (8.7.7) for example, $X_1(t) = a \cos[k_p X(t)]$ and $X_2(t) = a \sin[k_p X(t)]$. Table 8.3 lists $X_1(t)$ and $X_2(t)$ for a number of analog modulation schemes.

For simulation purposes, we represent modulated waveforms in the complex envelope form (see Section 3.4)

$$\tilde{Y}(t) = [X_1(t) + jX_2(t)]e^{j\theta} \quad (8.7.11)$$

The random phase offset of the carrier, θ , is often assumed to be zero. The complex envelope $\tilde{Y}(t)$ can also be expressed as

$$\tilde{Y}(t) = R(t)e^{j\psi(t)} \quad (8.7.12a)$$

where the real envelope $R(t)$ is

$$R(t) = [X_1^2(t) + X_2^2(t)]^{1/2} \quad (8.7.12b)$$

and

$$\psi(t) = \theta + \tan^{-1} \frac{X_2(t)}{X_1(t)} \quad (8.7.12c)$$

Sampled values of $\tilde{Y}(t)$ are used in simulations. Although we do not sample the bandpass-modulated signal, effects such as frequency offsets and frequency-selective behavior can be modeled using a lowpass equivalent.

8.7.2. Digital Quadrature Modulation

Equation (8.4.1) presents a general framework for digital modulation schemes. An important subclass can be naturally represented in the form (8.7.10) or (8.7.11) (Figure 8.13), where $X_1(t)$ and $X_2(t)$ might be two different waveforms of the form

$$X_k(t) = \sum_k A_k p_1(t - kT_1 - D_1) \quad (8.7.13a)$$

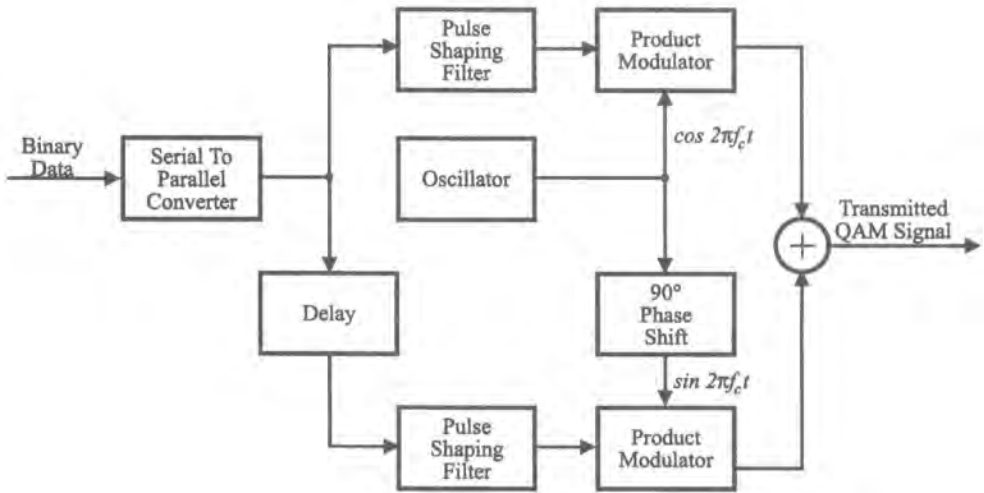


Figure 8.13. Generic quadrature amplitude modulator.

and

$$X_2(t) = \sum_k B_k p_2(t - kT_2 - D_2) \quad (8.7.13b)$$

where $p_1(t)$ and $p_2(t)$ are finite-energy pulses (rectangular, filtered rectangular, or sinc, for example). A_k and B_k are sequences of usually independent discrete random variables with symbol rates $1/T_1$ and $1/T_2$, respectively, and D_1 and D_2 are possible (relative) delays. [If in fact T_1 and T_2 were different, the waveforms $X_1(t)$ and $X_2(t)$ would typically arrive on different input lines in the model of Figure 8.13.] If $X_1(t)$ and $X_2(t)$ are unsynchronized, a proper model for D_1 and D_2 is to assume they are random and uniformly distributed on the intervals $[0, T_1]$ and $[0, T_2]$, respectively. For simulation purposes, an equivalent result is obtained if we set $D_1 = 0$, say, and let D_2 be random. In many important cases, $T_1 = T_2$ and D_1 and D_2 have a fixed, nonrandom relationship. Many digital schemes of practical interest are subsumed under the representation (8.7.13), in particular, quaternary phase-shift-keying (QPSK), offset quaternary phase-shift-keying (O-QPSK), unbalanced quaternary phase-shift-keying (U-QPSK), M -ary quadrature amplitude modulation (M-QAM), and minimum-shift-keying (MSK), among others (see Table 8.4). Three simple examples are sketched in Figure 8.14.

Not all quadrature modulation schemes, however, fall neatly into the format of (8.7.13). In particular, two important practical cases that do not are the $\pi/4$ QPSK scheme and multitone modulation (orthogonal frequency division multiplexing, OFDM). We outline these methods below.

■ *Remark on Simulating Delays.* In Figure 8.13 a “delay” is indicated in one of the quadrature channels to represent the offset that might be used intentionally. Usually, this offset is either zero or one-half symbol duration. Whatever the intended delay, in actuality it will be realized differently. Perhaps $0.5T$ might be built as $0.49T$. This brings up the question of how to simulate small delays, like 0.01 of a symbol duration. Taking this number as an example, it might seem that sampling at a rate of 100 samples/symbol would be required. Certainly, such a sampling rate would accommodate this situation, but it is not required. This is easily seen from the following simple argument. If we had analytical forms for two

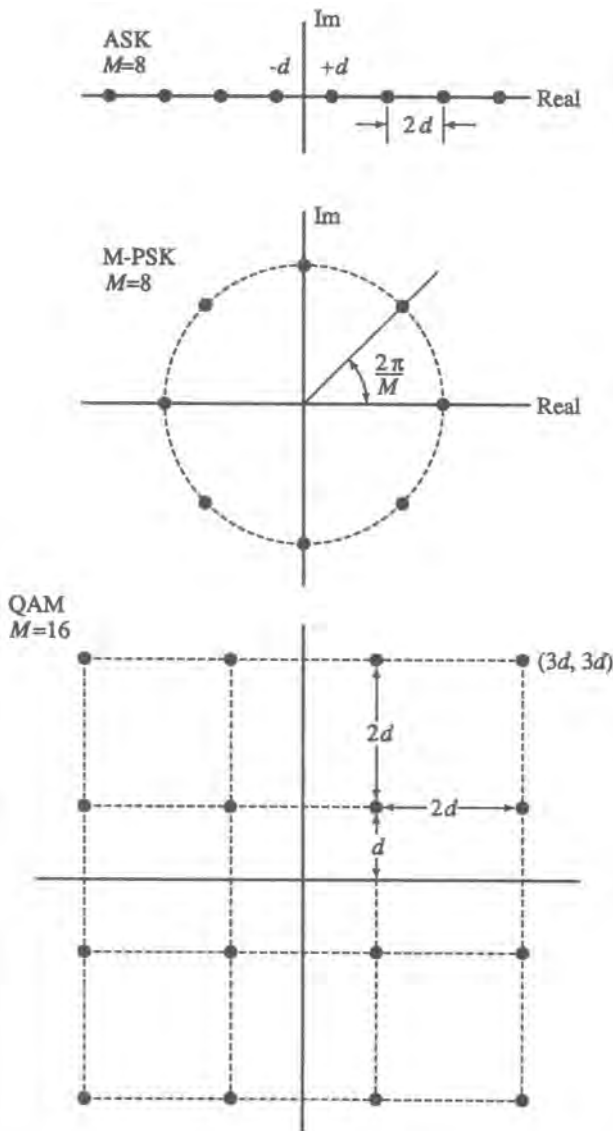


Figure 8.14. Some examples of signal “constellations” for digital modulation.

waveforms, say $s_1(t)$ and $s_2(t)$, both of equal bandwidth, then if we wanted sampled versions of $s_1(t)$ and $s_2(t - \tau)$ at the same sequence of instants, we need only replace t by kT_s since the sampling theorem works equally well on a shifted version of a bandlimited waveform.

Often, however, we do not have analytical forms and are presented with sampled versions of $s_1(t)$ and $s_2(t)$. If we wanted to shift the latter to $s_2(t - \tau)$ and if τ were an integer multiple of the sampling interval T_s , we would shift the sequence $s_2(kT_s)$ by the corresponding number of samples. If τ were not an integer multiple of T_s , it would present a difficulty in the sense that, ultimately, the sampling epoch of all processes have to be coincident. The solution is, simply, to interpolate $s_2(kT_s)$ and resample at the same epoch as that of $s_1(t)$. ■

Table 8.4 Some Digital Modulation Schemes Defined by Equation (8.7.13) for the Case $T_1 = T_2 = T$ and $\theta = 0$

Modulation scheme	(A_k, B_k)	$p_1(t), p_2(t)$
Amplitude shift keying (M -ary ASK)	$A_k = \pm nd, n = 1, 2, \dots, M/2;$ $B_k = 0$	$p_1(t) = 1, 0 \leq t \leq T;$ $p_2(t) = 0$
Phase shift keying (M -PSK)	$A_k + jB_k = \exp(j\phi_k),$ $\phi_k = 2\pi n/M, n = 0, 1, \dots, M-1$	$p_1(t) = 1, 0 \leq t \leq T;$ $p_2(t) = p_1(t)$
QPSK (M -PSK, $M = 4$)	$(A_k, B_k) = (\pm 1, \pm 1)$ or $\phi_k = 45^\circ, 135^\circ, 225^\circ, 315^\circ$	$p_1(t), p_2(t)$ same as M -PSK
O-QPSK	$(A_k, B_k) = (\pm 1, \pm 1)$	$p_1(t) = 1, 0 \leq t \leq T;$ $p_2(t) = 1, \frac{1}{2}T \leq t \leq \frac{3}{2}T$ $p_1(t) = \sin(\pi t/2T),$ $0 \leq t \leq T;$
Minimum shift keying (MSK)	$(A_k, B_k) = (\pm 1, \pm 1)$	$p_2(t) = \sin(\pi t/2T - T/2),$ $\frac{1}{2}T \leq t \leq \frac{3}{2}T$
M -ary quadrature amplitude modulation (M -QAM)	$(A_k, B_k) \in (\pm 1, \pm 3, \dots, \pm\sqrt{M}-1)$	$p_1(t), p_2(t)$ same as M -PSK

8.7.2.1. $\pi/4$ QPSK: Differential Quaternary Phase-Shift-Keying (DQPSK)

The modulation scheme called $\pi/4$ QPSK has been adopted as the method of choice in certain cellular standards. A block diagram of the transmitter is shown in Figure 8.15 (see, e.g., Refs. 45–47). The transmitted signal is given by

$$S(t) = Au_k \cos(2\pi f_c t) - Av_k \sin(2\pi f_c t), \quad kT \leq t \leq (k+1)T \quad (8.7.14)$$

where u_k, v_k are defined by

$$u_k = u_{k-1} \cos \theta_k - v_{k-1} \sin \theta_k \quad (8.7.15a)$$

$$v_k = u_{k-1} \sin \theta_k + v_{k-1} \cos \theta_k \quad (8.7.15b)$$

and take values in $(0, \pm 1, \pm \sqrt{2}/2)$, and θ is related to the input bit pair (d_1, d_Q) as follows

$$\begin{array}{ll} 11 \rightarrow \pi/4; & 01 \rightarrow 3\pi/4 \\ 00 \rightarrow -3\pi/4; & 10 \rightarrow -\pi/4 \end{array}$$

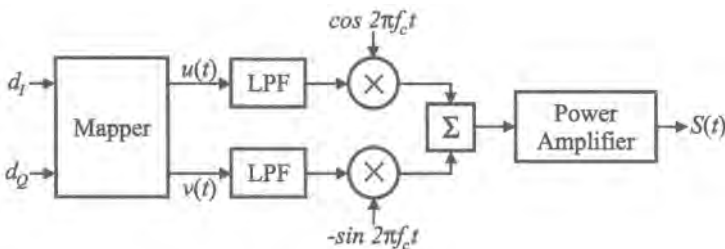


Figure 8.1S. Block diagram for a $\pi/4$ QPSK modulator.

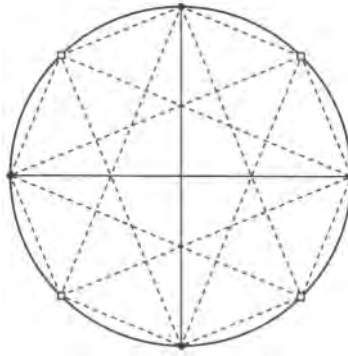


Figure 8.16. Signal points and trajectories for $\pi/4$ QPSK.

An examination of this scheme shows that during successive symbol periods T the transmitted signal is chosen alternatively from one of two QPSK constellations rotated one from another by $\pi/4$. Transitions from one constellation to another can only have values in the set $\pm\pi/4$ and $\pm3\pi/4$. This is illustrated in Figure 8.16. The main appeal of this scheme as revealed by the figure is that transitions do not pass through the origin. Put another way, envelope variations are decreased relative to conventional QPSK. It also turns out that the symbol-to-phase mapping is inherently differential. Hence, demodulation can be coherent, differentially coherent, or even noncoherent.

8.7.2.2. Multitone Modulation/OFDM

A technique which has gained increasing importance in a number of environments is multitone modulation,^(48–50) perhaps the best-known variant of which is OFDM (orthogonal frequency division multiplexed). OFDM has found a number of applications in fixed-wireless and mobile personal communication systems (PCS),⁽⁵¹⁾ direct audio broadcasting (DAB), and asymmetric digital subscriber line (ADSL) and very high rate digital subscriber line (VDSL) systems, which operate over twisted pair channels.⁽⁵²⁾ The basic idea behind multitone modulation is to replace one (potentially) wideband signal with many simultaneously transmitted narrowband signals (or subchannels) with the same overall bandwidth as the original signal. In principle, the two schemes are equivalent in an AWGN (flat) channel. However, there are advantages to the multiple narrowband scheme in a fading environment. If the individual channels are made narrower than the coherence bandwidth of the channel, which means that individual symbol times are typically much wider than the delay spread of the channel, the effect of intersymbol interference is greatly mitigated. This reduces, and possibly eliminates, the need for equalization in individual channels. Viewed another way, the use of many narrowband subchannels is one way of introducing frequency diversity. If there is frequency-selective fading, by using an appropriate subchannel bandwidth we can expect a large percentage of subchannels not to be in the faded part of the overall channel bandwidth. If the channel is sufficiently slowly varying, as in PCS or local fixed wireless loop systems, then it may be possible to learn the channel characteristics and relay this information back to the transmitter. This knowledge may then be used at the transmitter to design the transmitted signal on a dynamic basis, utilizing the “water-pouring” concept,⁽⁴⁹⁾ to match the transmission channel. This implies an adaptive transmitter which can maximize the transmitted bit rate by adjusting the power levels and number of bits per symbol for each individual

subchannel, in accordance with the channel information. The main drawback of OFDM is the high peak-to-average power ratio (PAR), which is a consequence of summing many “tones,” the result of which is well known to approximate a Gaussian random process. The high PAR forces the use of a power amplifier with a large linear range.

OFDM can be regarded both as a modulation method and a multiple-access technique. The basic scheme is illustrated in Figure 8.17.⁽⁵⁰⁾ Assume we have a (complex) data sequence $\{d(k)\} = \{a(k) + jb(k)\}$, $k = 0, 1, \dots$, arriving at a rate of R symbols/s. The symbol components (a, b) are in general drawn from the set $\{\pm 1, \pm 3, \dots, \pm \sqrt{M}\}$. This “high-rate” sequence is demultiplexed (or decimated) into N “low-rate” subsequences each having rate R/N . Hence, the output of the serial-to-parallel converter produces a sequence of data blocks (or OFDM symbols) $\{A(l)\} = \{d(lN), \dots, d(lN + N - 1)\}$, $l = 0, 1, \dots$. Each subsequence modulates a carrier of frequency $f_n = f_0 + n\Delta f$, $n = 0, 1, \dots, N - 1$. Thus, in the OFDM symbol interval $0 \leq t \leq T$, $T = N/R$, the transmitted signal is of the form

$$D(t) = \sum_{n=0}^{N-1} a(n) \cos \omega_n t + b(n) \sin \omega_n t \quad (8.7.16)$$

As presented so far, this is no different than a parallel sum of M -ary QAM signals, i.e., “ordinary” frequency-division multiplexed signals. (This type of FDM will be considered in Section 8.10.) If, however, we stipulate that $\Delta f = 1/T$, the carriers ω_n are orthogonal over intervals of T -second duration. Hence, a receiver of the type shown in Figure 8.17b will ideally eliminate any interchannel interference.

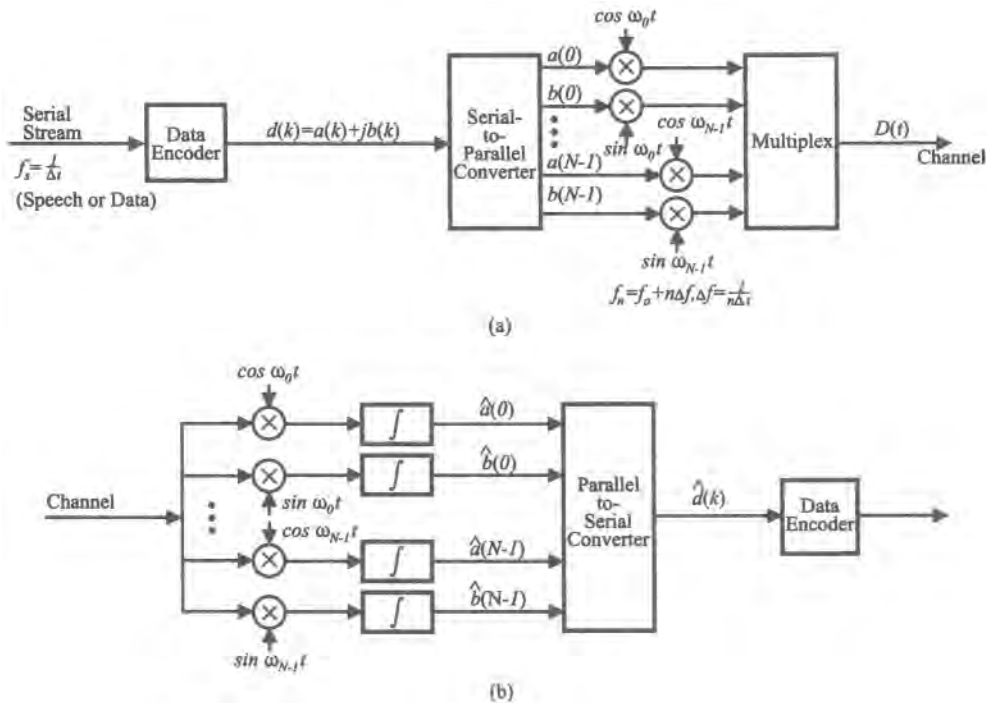


Figure 8.17. Basic arrangement for a multitone-OFDM system. (a) Transmitter, (b) receiver.

If N is moderately large, the implementation in Figure 8.17 can be costly since it requires N modulators, N oscillators, N demodulators, etc. An alternative implementation that avoids this factor-of- N penalty (but has its own difficulties) is based on the use of the discrete Fourier transform⁽⁵³⁾ and is sometimes referred to as discrete multitone.⁽⁵²⁾ The basic idea is as follows. By looking at Figure 8.17a or (8.7.16) we see that a block of data can be viewed as a set of ordered pairs $\{d(n), f_n\}$, $n = 0, 1, \dots, N-1$. So we can think of $d(n)$ as the complex amplitude associated with frequency f_n . Since the frequencies are discretely spaced, we can therefore choose to interpret $\{d(n), f_n\}$ as the discrete Fourier transform of some sequence, as indicated in Figure 8.18. In other words, $\{d(n), f_n\}$ is the (discrete) “frequency-domain” representation of the data. Here, the actual frequencies are not important; n can simply be thought of as an index. The corresponding “time-domain” sequence $D(m)$ is therefore the inverse DFT,

$$D(m) = \sum_{n=0}^{N-1} d(n) e^{j(2\pi/N)nm} \quad (8.7.17)$$

so that the original sequence is given by

$$d(n) = \frac{1}{N} \sum_{m=0}^{N-1} D(m) e^{-j(2\pi/N)nm} \quad (8.7.18)$$

Equations (8.7.17)–(8.7.18) form the basis of the transmission scheme (Figure 8.19). The (inverse) discrete Fourier transform is applied to the frequency-domain block $\{d(n)\}_{n=0}^{N-1}$, normally using the FFT. The resulting discrete Fourier coefficients $\{D(m)\}$ are then sent over the channel sequentially, over N time slots each equal to R^{-1} . In a baseband scheme, $f_0 = 0$,

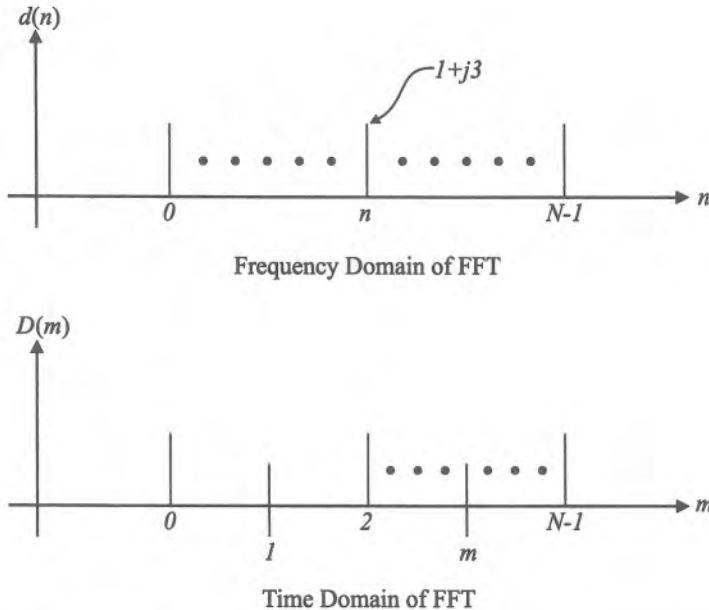


Figure 8.18. Association of OFDM data block with the discrete Fourier transform.

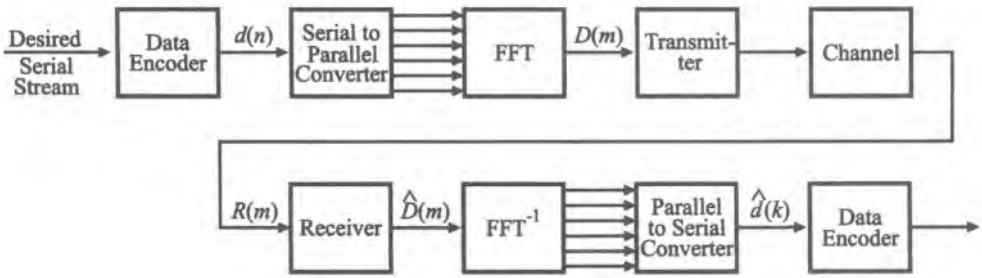


Figure 8.19. Basic arrangement for FFT-based OFDM system.

and we cannot literally send the imaginary component, but it can be recovered by doubling the sampling rate at the receiver.⁽⁵³⁾ In a bandpass scheme, the real and imaginary components can be sent on quadrature carriers. Thus, labeling $D_c(m)$ and $D_s(m)$ the real and imaginary components, respectively, of $D(m)$, we can write the transmitted OFDM symbol as

$$D(t) = \sum_{m=0}^{N-1} D_c(m)p(t - mR^{-1}) \cos \omega_c t + D_s(m)p(t - mR^{-1}) \sin \omega_c t \quad (8.7.19)$$

where $p(t)$ is some suitably chosen pulse shape. It should be realized that there are many possible values of $D(m)$ (in fact there are M^N) that can vary over a wide range. Hence we may view (8.7.19) as almost an analog transmission. In practice, it is also customary to send a slightly longer sequence than indicated by (8.7.19) to avoid interference between adjacent OFDM symbols. This guard time is usually filled with the last few samples of the transmitted OFDM symbol, which naturally wrap around because of the circular nature of the DFT (see Chapter 3). The actual guard time depends, of course, on the actual physical channel.

At the receiver, the signal (8.7.19) is demodulated, match-filtered, and sampled to estimate $D(m)$. The DFT (8.7.18) is taken, and, without noise or distortion, we would obviously recover $d(n)$. Of course, there *will* be noise and distortion, and as might be appreciated from the “analog” reference above, recovery of $d(n)$ would be impeded by amplitude and phase distortion. Different options exist with respect to ameliorating the effect of such distortions. One approach is to identify the channel, and simply apply the inverse transfer function (assuming, of course, a linear channel). This can be done, for example, by using a pilot tone,⁽⁵⁰⁾ or by transmitting a training sequence to fit a pole-zero model of the channel.⁽⁵²⁾ An alternative to learning the channel is to use differential encoding in the subchannels, though this has a significant power penalty.

Because the channel acts on the DFT of the desired sequence, it is generally very hard to analyze the total effect on the quality of data recovery due to frequency-selective channels which are encountered by real mobile systems, as well as the degree to which mitigation schemes do their job. Therefore, one often resorts to simulation to determine or estimate the effects of fading, adjacent cell interference, coding and interleaving, power amplifier (residual) nonlinearity, and other equipment imperfections. Simulation is also called for to investigate realistic evaluation of the “water-pouring” optimization, in particular, the speed and quality of channel estimation. The reader is referred to the cited references, among others, for detailed discussion of multitone and OFDM, as well as different implementation ideas.

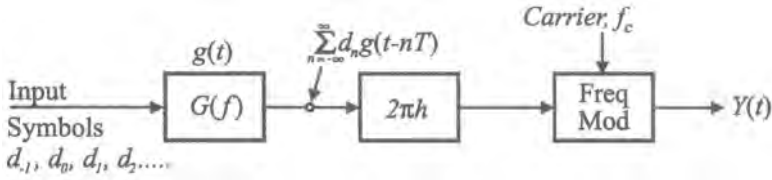


Figure 8.20. Block diagram of CPM transmitter.

8.7.3. Continuous Phase Modulation: CPFSK, MSK, GMSK

Another important class of digital modulation schemes, which is more readily interpretable in the form (8.7.7) or (8.7.8), is continuous phase modulation (CPM) (Figure 8.20). Both superior performance and efficient spectrum utilization are possible in principle by proper choice of the modulator parameters. An important subclass of CPM is continuous-phase frequency-shift-keying (CPFSK) and a popular special case of the latter is minimum-shift-keying (MSK). An important variant of MSK widely used in the GSM cellular architecture is Gaussian MSK, or GMSK.

8.7.3.1. Continuous Phase Modulation

CPM is a form of digital phase modulation where the phase is constrained to remain continuous; that is, the phase cannot jump discontinuously between symbols, as it can (in principle) in QPSK (see, for example, Refs. 2, 3, and 11). A general CPM signal is given by

$$Y(t) = A \cos[\omega_c t + \phi(t) + \phi_0] \quad (8.7.20a)$$

$$\tilde{Y}(t) = A \exp[j\phi(t) + j\phi_0] \quad (8.7.20b)$$

$$\phi(t) = 2\pi \int_{-\infty}^t \sum_{k=-\infty}^n d_k h_k g(\tau - kT) d\tau \quad (8.7.21a)$$

$$= 2\pi \sum_{k=-\infty}^n d_k h_k q(t - kT), \quad nT \leq t \leq (n+1)T \quad (8.7.21b)$$

where $g(t)$ is a “frequency pulse,” and

$$q(t) = \int_0^t g(t) dt$$

The constraint imposed in (8.7.21) establishes the continuity of the phase. The parameter T is the symbol duration; $\{d_k\}$ is the data sequence, where $d_k \in \{\pm 1, \pm 3, \dots, \pm(M-1)\}$; and h_k is called the modulation index. Commonly, we set $h_k = h$, a fixed value. In some instances, h_k varies with k in a cyclic manner; this situation is referred to as *multi-h* CPM. In practice, $g(t)$ is finite in extent,

$$g(t) = 0, \quad t < 0, \quad t > LT$$

and the normalization

$$\int_0^{LT} g(\tau) d\tau = \frac{1}{2}$$

is used. The following terminology is common: when $L = 1$, we have *full response CPM*; and when $L \geq 2$, we have *partial response CPM*. Some options for $g(t)$ are listed in Table 8.5, and two of these are sketched in Figure 8.21 along with the corresponding $q(t)$.⁽⁵⁴⁾

For L finite, we have

$$\begin{aligned} \phi(t) = 2\pi \sum_{k=n-L+1}^n d_k h_k q(t - kT) \\ + \pi \sum_{k=-\infty}^{n-L} d_k h_k, \quad nT \leq t \leq (n+1)T \end{aligned} \quad (8.7.22)$$

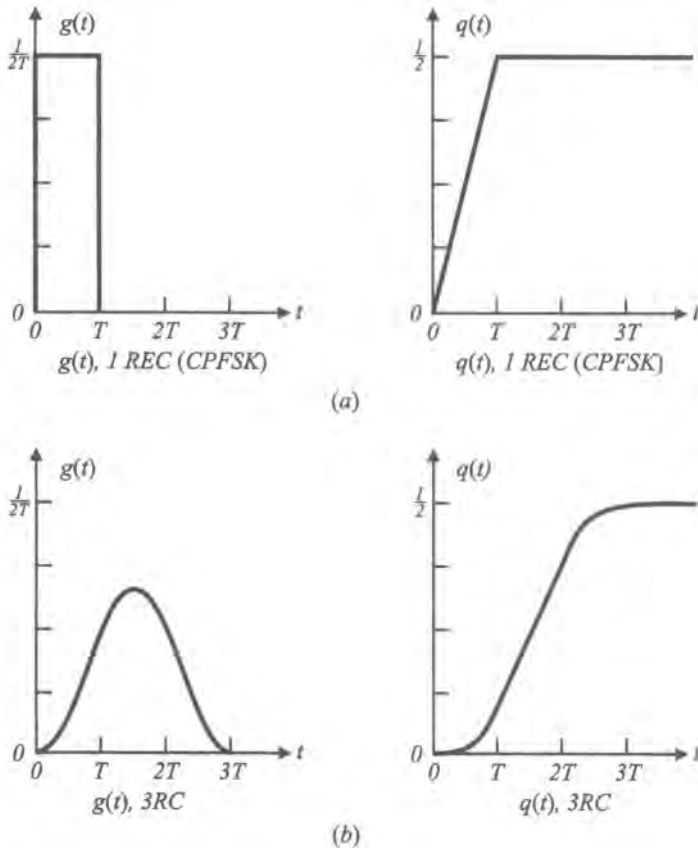


Figure 8.21. Some examples of frequency pulses and corresponding phase functions for CPM (from Ref. 54, @IEEE, 1986). (a) Rectangular frequency pulse, (b) raised-cosine frequency pulse.

Table 8.5 The Frequency Pulse Defined for a Some CPM Schemes^a

LRC	$g(t) = \begin{cases} \frac{1}{2LT} \left[1 - \cos\left(\frac{2\pi t}{LT}\right) \right], & 0 \leq t \leq LT \\ 0, & \text{otherwise} \end{cases}$ L is the pulse length, e.g., 3RC has $L = 3$
TFM	$g(t) = \frac{1}{8} [ag_0(t - T) + bg_0(t) + ag_0(t + T)]; \quad a = 1, \quad b = 2$ $g_0(t) \approx \sin\left(\frac{\pi t}{T}\right) \left[\frac{1}{\pi t} - \frac{2 - (2\pi t/T) \cot(\pi t/T) - \pi^2 t^2/T^2}{24\pi t^3/T^2} \right]$
LSRC	$g(t) = \frac{1}{LT} \frac{\sin(2\pi t/LT) \cos(\beta \cdot 2\pi t/LT)}{2\pi t/LT \cdot 1 - [(4\beta/LT)t]^2}; \quad 0 \leq \beta \leq 1$
GMSK	$g(t) = \frac{1}{2T} \left[Q\left(2\pi B_b \frac{t - T/2}{\sqrt{\ln 2}}\right) - Q\left(2\pi B_b \frac{t + T/2}{\sqrt{\ln 2}}\right) \right]; \quad 0 \leq B_b T < \infty$ $Q(t) = \int_t^\infty \frac{1}{\sqrt{2\pi}} \exp(-\tau^2/2) d\tau$
LREC	$g(t) = \begin{cases} 1/(2LT), & 0 \leq t \leq LT \\ 0 & \text{otherwise} \end{cases}$ $L = 1$ yields 1REC, which is most often referred to as CPFSK

^aThe letter L in LRC, LSRC, and LREC is an integer that indicates the duration of the frequency pulse $g(t)$, namely LT , where T is the length of an input symbol. RC stands for raised cosine, SRC for spectrally raised cosine, and REC for rectangular. TFM means tamed frequency modulation and GMSK is Gaussian minimum-shift-keying (from Ref. 54 © IEEE, 1986).

Of course, by using trigonometric identities we can always put (8.7.20) in the form (8.7.10), but the interpretation of the components $X_1(t)$ and $X_2(t)$ in terms of the applied modulation is generally not straightforward or useful except in one case discussed shortly.

8.7.3.2. Continuous-Phase Frequency-Shift-Keying

When the instantaneous frequency in each signaling interval is fixed and chosen from a set of M values, we have the subclass called continuous-phase frequency-shift-keying,⁽⁵⁵⁻⁵⁷⁾ or CPFSK. To obtain the desired condition, we set

$$g(t) = \frac{1}{2T} p_T(t) \quad (8.7.23)$$

so that

$$\phi(t) = \frac{1}{T} \pi h d_n(t - nT) + \pi h \sum_{k=-\infty}^{n-1} d_k, \quad nT \leq t \leq (n+1)T \quad (8.7.24)$$

It is customary to set $h_k = h$, a fixed value for all k , although that is not strictly necessary to create an FSK signal. From (8.7.24) we see that the instantaneous frequency is given by

$$f_i(t) = \frac{1}{2\pi} \frac{d\phi(t)}{dt} = \frac{h d_n}{2T}, \quad nT \leq t \leq (n+1)T$$

Notice that the increment of frequency shift is $f_d = h/2T$. Since h represents the ratio of the minimum separation $2f_d$ to the symbol rate, it is also referred to as the deviation ratio. It may also be noted that the second term in (8.7.24),

$$\psi_n = \pi h \sum_{k=-\infty}^{n-1} d_k$$

can be easily implemented recursively since $\psi_{n+1} = \psi_n + \pi h d_n$.

8.7.3.3. Minimum-Shift-Keying

A form of CPFSK known as minimum-shift-keying for which $M = 2$ and $h = 0.5$ has recently found wide application. Under this specialization, the phase function becomes

$$\phi(t) = \frac{\pi}{2T} d_n(t - nT) + \frac{\pi}{2} \sum_{k=-\infty}^{n-1} d_k, \quad nT \leq t \leq (n+1)T \quad (8.7.25)$$

It turns out for this case that expanding (8.7.20a) in quadrature form does lead to a useful expression, which in turn gives rise to a practical alternative implementation. Specifically, substituting (8.7.25) into (8.7.20b) yields

$$\begin{aligned} \tilde{Y}(t) = [A \cos u_n \cos(\pi t/2T) \\ + j d_n \cos u_n \sin(\pi t/2T)] \exp(j\phi_0), \quad nT \leq t \leq (n+1)T \end{aligned} \quad (8.7.26)$$

where

$$u_n = \frac{\pi}{2} \left\{ \sum_{k=-\infty}^{n-1} d_k - n d_n \right\}$$

Upon closer examination, it can be shown that the “equivalent” I and Q data streams $\cos u_n$ and $d_n \cos u_n$ are such that each (I or Q) data symbol has duration $2T$, and the data streams are offset from one another by T seconds.

Thus, (8.7.26) can be meaningfully implemented in quadrature form. However, this particular form has some drawbacks, as it stands. First, the equivalent data streams have to be computed, and second the “subcarriers” $\cos(\pi t/2T)$ and $\sin(\pi t/2T)$ reverse the sign of every other of these bits. Further reflection shows that an instantaneously different, but statistically equivalent, form of (8.7.26) results in a more conveniently implemented version. This form is obtained simply by replacing $\cos u_n$ and $d_n \cos u_n$ by d_{2n} and d_{2n-1} , respectively. In other words, demultiplex the input bit stream into “even” and “odd” bits, stretch each bit to length $2T$, and multiply by the subcarriers. The latter are aligned with their respective bit streams so that each bit pulse is a half-sinusoid. An illustration of this interpretation of MSK as “shaped offset QPSK” is given in Figure 8.22. The alternate bit inversion caused by the subcarriers makes detection somewhat awkward in simulation, but this problem can be avoided by rectifying the subcarrier. This would leave the bit signs unchanged. A block diagram of a possible implementation would thus appear as in Figure 8.23. In this block diagram, another variation of (8.7.26) is implicit. We have previously conceived of the I and Q data streams as demultiplexed versions of a single high-rate bit stream. It is equally valid to suppose that we

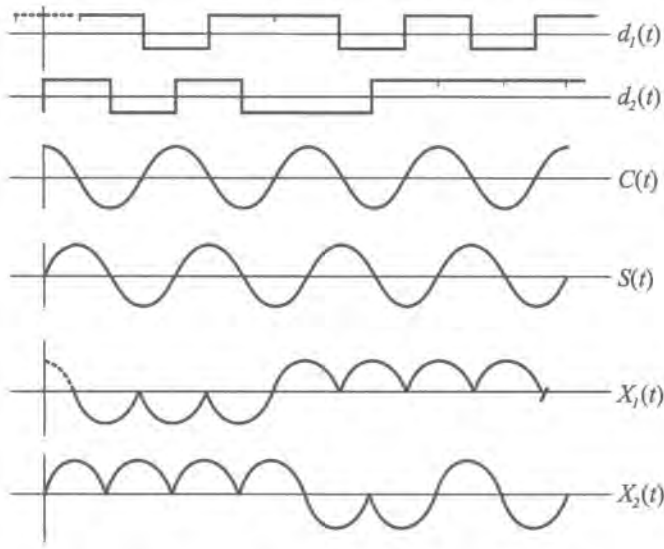


Figure 8.22. Interpretation of MSK as offset-shaped QPSK.

initially have two independent bit streams, as is done in Figure 8.23. The resulting MSK signal would not be pointwise identical to one in which the subcarrier is not rectified, but again it would be *statistically* identical. Note further that, in the rectified form, the MSK signal can be represented precisely in the form (8.7.10) with the quadrature components given by (8.7.13), where $p_1(t) = p_2(t) = \sin(\pi t/2T)$, $T_1 = T_2 = 2T$, $D_1 = 0$, and $D_2 = T$.

8.7.3.4. Gaussian Minimum-Shift-Keying

Figure 8.24 shows a simplified block diagram of a GMSK modulator, which gets its name from the filter $\mathcal{G}(f)$, which has a Gaussian-shaped amplitude transfer function

$$\mathcal{G}(f) = \frac{1}{\sqrt{2\pi}T\sigma} e^{-(f/T)^2/2\sigma^2} \quad (8.7.27)$$

with

$$\sigma = \frac{1}{K_1 B_3 T}; \quad K_1 = 2\pi(2 \ln \sqrt{2})^{-1/2} \quad (8.7.28)$$

where T is the symbol rate. The phase characteristic is ideally linear. Equations (8.7.27) and (8.7.28) are such that B_3 corresponds to the 3-dB bandwidth (one-sided) in Hz. Observe that the input to $\mathcal{G}(f)$ is a random binary waveform, not the δ -function sequence implicit in Figure 8.20. The normalized parameter $B_3 T$ affords a tradeoff between bandwidth and complexity. The spectrum of GMSK as a function of $B_3 T$ can be found, for example, in Ref. 58. The value standardly used in GSM is $B_3 T = 0.3$. The basic frequency pulse $g(t)$ is as shown in Table 8.5 and is displayed for $B_3 T = 0.3$ in Figure 8.25, which also shows the corresponding phase function $q(t)$.

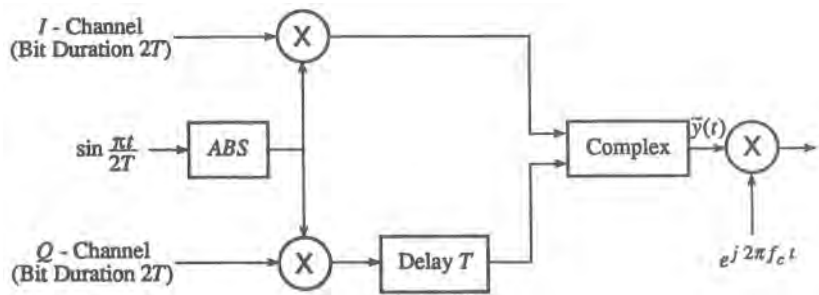


Figure 8.23. One implementation of MSK convenient for simulation purposes. The I and Q channels are initially synchronized with one another and with the zero crossings of $\sin(\pi t/2T)$.

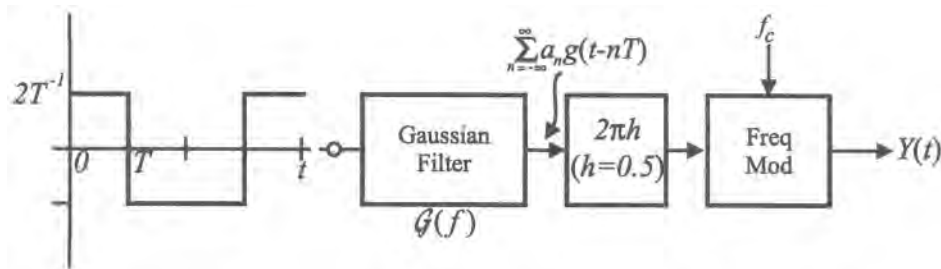


Figure 8.24. Block diagram of GMSK transmitter.

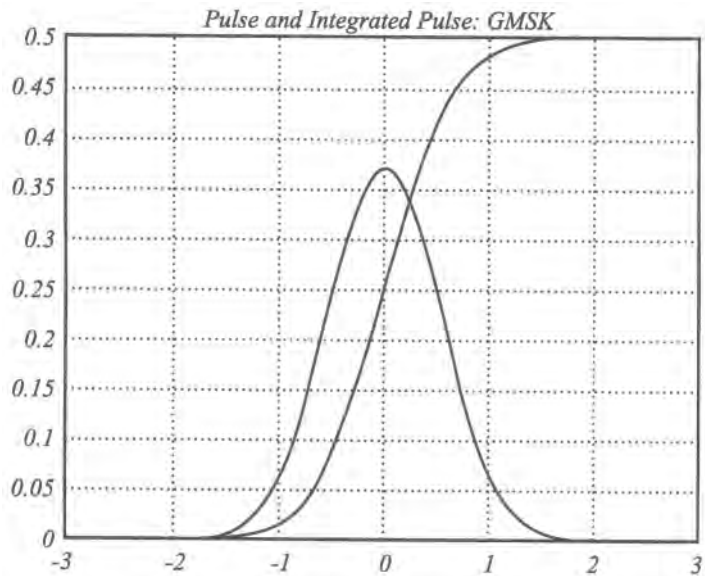


Figure 8.25. Frequency pulse and corresponding phase function for GMSK with $B_3T = 0.3$. The abscissa is graduated in units of T .

It can be seen that GMSK is a partial response signal, since its duration is greater than one symbol interval. In fact, theoretically, $g(t)$ has infinite duration. However, for practical purposes $g(t)$ can be truncated at some reasonable duration; the value $L = 3$ is often used.⁽⁵⁹⁾ For simulation purposes, we do not need explicitly to simulate the filtering action that produces $g(t)$, since it is already known. Its values sampled at T_s intervals can be stored and repeatedly sent every T seconds into the rest of the system, but first multiplied by ± 1 according to the logical value of the source symbol.

8.7.4. Coded Modulation

The application of “conventional” channel coding of the type discussed in Section 8.6 has been traditionally treated independent of the type of modulation method selected. From a system design standpoint, these selections were certainly made compatible. But from an operational standpoint, the modulation scheme, once selected, is not influenced by the presence or absence of channel coding, other than the need to operate at the code rate rather than at the information rate. Similarly, the modeling of a modulation method preceded by conventional channel coding is unaffected by that fact. Put in other words, the symbol-to-waveform mapping is independent of the presence of coding. The latter is not the case for the set of techniques called *coded modulation*, in which the coding and modulation are intertwined to some degree. There are two classes of coded modulation, one based on block codes, block-coded modulation^(60–64) (BCM), and one based on convolutional codes, called trellis-coded modulation^(65–71) (TCM).

A simple example of the application of each can be given for the following scenario. Typically, one starts with a “reference,” or baseline, condition, i.e., a system implementation one *might* have used, but which we will now consider modifying to obtain better performance (in some sense of that word) through the use of coded modulation. The tradeoff, as we will see, is different in BCM than in TCM, but basically what is gained is traded for complexity in the implementation of the scheme. To illustrate the distinctions between BCM and TCM, we will assume the same baseline condition, which is the transmission of R bits/s using QPSK (so we have $R/2$ bits/s per quadrature channel). For both BCM and TCM we will assume the transmitter alphabet is 8-ary and specifically the modulation is 8-PSK.

The transmitter block diagram for BCM is shown in Figure 8.26. There are three input binary data lines. We shift n bits into the first two, say stored in registers. For the third line we shift k bits into an (n, k) block encoder. We can thus visualize the input to the mapper as a table of three rows and n columns. The digits in the lowest row are referred to as the least significant bits (LSB).^(61,62) This simple scheme is a special case of a more general idea called unequal error protection (UEP) in which we could have different levels of coding in the three (or perhaps more) input lines. (For a recent application of UEP, see, e.g., Ref. 64 and references therein.) The block diagram is a particular case of the representation (8.4.2).

At each tick of the transmission clock a column or triplet (c_1, c_2, c_3) of the table is mapped into one of eight phases, as indicated in Figure 8.26. Since the code rate is $(2n + k)/3n$, there is some bandwidth expansion by the factor $3n/(2n + k)$. However, this can be a number fairly close to 1, hence the bandwidth expansion can be very modest. An examination of the mapping shows that the 8-ary constellation can be viewed as two QPSK constellations for each of which the LSB is different. If the code were sufficiently powerful, we might imagine the LSBs to be perfectly recovered, hence the BER would tend to that for QPSK, not 8-PSK. But the throughput is closer to 3 bits per transmission than 2 bits. This is

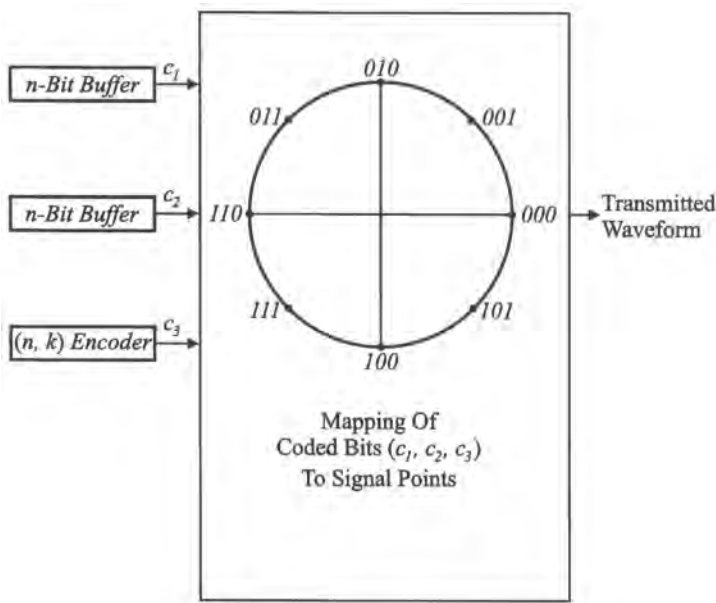


Figure 8.26. Transmitter block diagram for a particular case of 8-PSK block-coded modulation.

roughly the situation that we try to bring about in this case. The tradeoff here is higher throughput for the same, or perhaps slightly diminished, performance at a cost of slightly increased bandwidth and complexity (due to the coding).

The block diagram for TCM is shown in Figure 8.27, along with its trellis diagram. The abstract representation for the function of this (and other TCM) subsystem(s) is (8.4.1). In the case at hand, there are two input lines, one of which goes directly to the mapper, and the other of which enters a rate-1/2 convolutional encoder. Observe that the symbol-to-waveform mapping is different here than above. Because of the memory embedded in the convolutional code, we can expect better performance than that of QPSK, but clearly the throughput is unaltered. Hence, the trade here is better BER performance at the cost of complexity. It is possible to increase the code rate while still improving performance, but at the cost of higher complexity. The reader is referred to the cited references for further discussion of TCM schemes and their performance.

As with coding itself, the question arises whether we need explicitly to duplicate the actual operations involved with BCM or TCM, especially the receiver/decoder. Basically the same answers apply as were discussed in conjunction with conventional error-correction coding. That is, we do not need explicitly to simulate the encoding or decoding because, generally, good enough analytical tools can be used in conjunction with a waveform-level simulation up to the decoder input. The presence of the encoding has to be accounted for in the simulation only to the extent that there is bandwidth expansion. For the BCM scheme, the same type of approach as discussed in Section 11.2.7.6 is applicable (see also Ref. 62 for an application). As we saw in Section 8.6, emulation of the decoding algorithms does not pose an intolerable burden. Hence, for moderate channel error rates, say in the region of 10^{-3} , explicit simulation of the codec should be feasible.

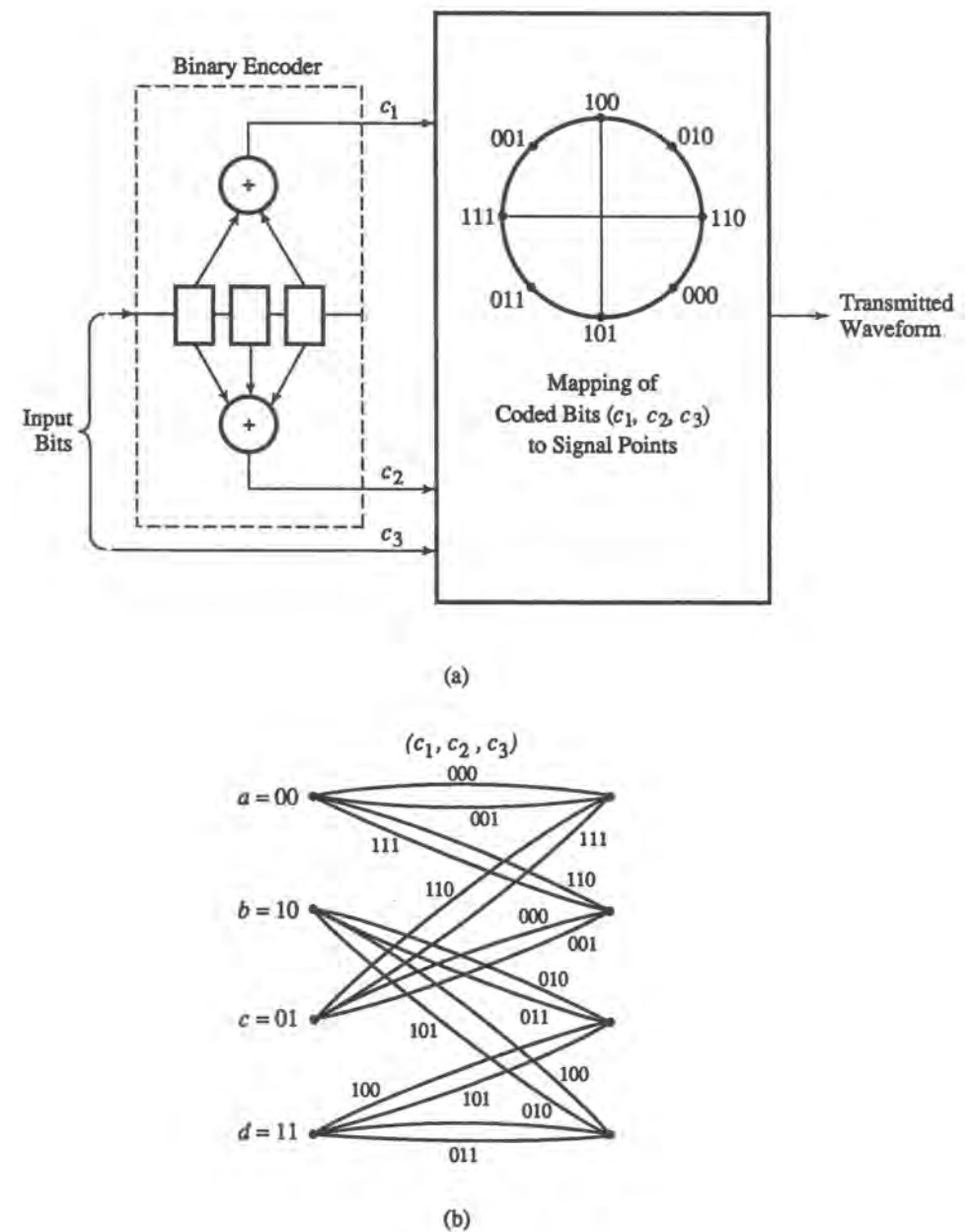


Figure 8.27. Example of four-state trellis-coded 8-PSK modulation (from Ref. 11, © McGraw-Hill, 1989 reproduced with permission), (a) Transmitter, (b) trellis.

8.7.5. Modeling Considerations

The modeling of ideally implemented modulators is evident on the surface since the complex envelope is inherent in the very definition of a modulation method. As hinted earlier, the main consideration is whether a method is linear or not, which dictates the necessary

sampling rate. Modulators, like any other equipment, cannot be built ideally. The fidelity of a demodulated signal is very much dependent on the quality of the modulator (and of course on the characteristics of every other subsystem). But the quality of the modulator becomes more central in the transmission of digital signals using higher order alphabets, for the decision regions there become increasingly small. In this subsection we will consider some of the ways to model modulators more realistically. The basic methodology follows the prescription of Chapter 2, which is in essence to examine the block diagram, and presume that every operation indicated or implicit can depart in some fashion from the ideal.

We have seen that the complex envelope of a modulated signal can be represented either in the *quadrature* (or *parallel*) form (8.7.11) or the *polar* (or *serial*) form (8.7.12). Depending upon the particular modulation method, one or the other of these forms will be more natural in the sense that it mimics more closely the physical realization. In this more natural representation one can modify more realistically the idealized form to take account of real-world impairments. As an example, let us look again at the phase-modulated signal

$$Z(t) = a \cos[2\pi f_c t + \theta + k_p X(t)]$$

previously given in (8.7.7). Its natural representation is the polar form, hence the complex envelope representation that we want to use is

$$\tilde{Z}(t) = a \exp[jk_p X(t)] \quad (8.7.29)$$

and its simulation implementation might look like that of the block diagram in Figure 8.28a. (For phase modulation the integrator is bypassed, but is switched in for frequency modulation.) It is simple to generalize (8.7.29) to represent nonideal equipment. Equation (8.7.29) is idealized in the sense that the voltage-to-phase transfer characteristic is perfectly linear, with

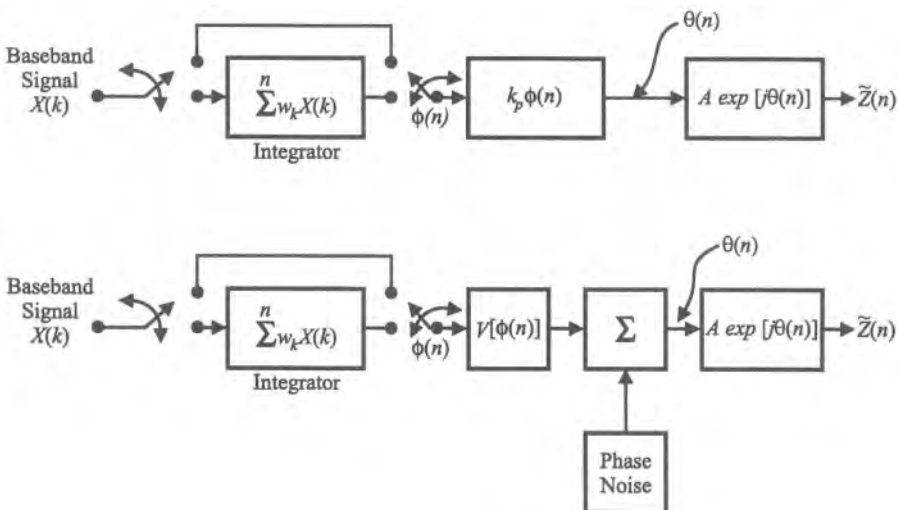


Figure 8.28. Representation of phase or frequency modulation, (a) block diagram of idealized modulator (discrete-time complex envelope); (b) block diagram of nonideal modulator with phase noise.

gain k_p rad/V. A simple generalization is to assume that this transfer characteristic is given by some function of $X(t)$, say $V[X(t)]$, for example, a third-order polynomial,

$$V[X(t)] = k_{p1}X(t) + k_{p2}X^2(t) + k_{p3}X^3(t)$$

We can add still another refinement to the modeling by including some phase noise, say $\delta(t)$, into the phase of the modulated signal, so that the complex envelope can now be represented as

$$\tilde{Z}(t) = a \exp\{jV[X(t)] + j\delta(t)\} \quad (8.7.29a)$$

The model of Figure 8.28a is then generalized to that in Figure 8.28b.

If we substitute $V[X(t)]$ for $k_p X(t)$ in (8.7.29) and use Euler's identity, we have

$$\tilde{Z}(t) = a \exp\{jV[X(t)]\} \quad (8.7.30a)$$

$$= a\{\cos V[X(t)] + j \sin V[X(t)]\} \quad (8.7.30b)$$

This quadrature form is mathematically equivalent, but much less natural, for the block diagram of Figure 8.28b would clearly not be a “mapping” of (8.7.24b).

On the other hand, the quadrature representation is obviously the natural form for modulation schemes that are actually implemented in hardware as modulated quadrature carriers, as in Figure 8.13. In this case, the hardware impairments can be reflected through straightforward modifications of (8.7.10). As a first step, the simplest model extension might be of the form

$$Y(t) = a_1 X_1(t) \cos(2\pi f_c t + \theta_1) + a_2 X_2(t) \sin(2\pi f_c t + \theta_2) \quad (8.7.31)$$

where a_1 and a_2 embody possible (unwanted) differences in the gain of the signal paths (the ratio a_1/a_2 , or its inverse, is often called amplitude or gain imbalance), and $\theta_2 - \theta_1$ represents a phase error (imbalance) that prevents the desired 90° phase difference from being realized. These parameters, a_1/a_2 and $\theta_2 - \theta_1$, are often specified in the design of modulators. The complex envelope corresponding to (8.7.31) is

$$\tilde{Y}(t) = a_1 X_1(t) e^{j\theta_1} + a_2 X_2(t) e^{j\theta_2} \quad (8.7.32)$$

which can be implemented in simulation according to the block diagram of Figure 8.29, with the lowpass filters (LPF) absent. This figure can serve as a generic simulation block diagram for a quadrature modulator. What has to be specified for any modulation scheme is the appropriate $X_1(t)$ and $X_2(t)$. As previously mentioned, some of these functions are given in Table 8.3 for analog signals and in Table 8.4 for some common digital signals. Detailed descriptions and analyses of these (usually idealized) modulation schemes and others are given in the textbooks cited earlier.

As the next step in model generalization, one might want to consider the amplitude and phase (with respect to some common reference) associated with each symbol in each of the quadrature channels to be dependent on the “state” of the modulator, defined as the sequence of previous symbols. Such a dependence could arise, for example, from some bandlimiting at the modulator input, or inherently embedded within the modulator structure. A simple way to

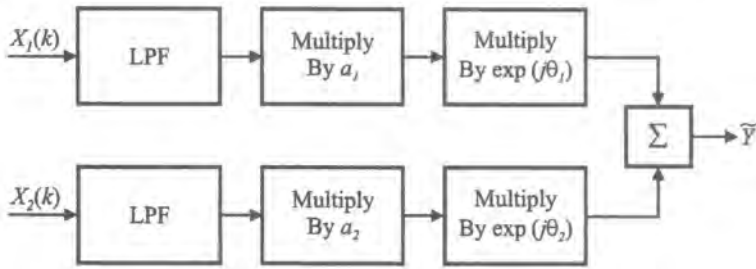


Figure 8.29. Simulation block diagram for an ideal or nonideal quadrature modulator.

account for such an effect is through the in-line filters shown in Figure 8.29. Such filtering would manifest itself in the “clouds” typically observed in scatter diagrams, as illustrated in Figure 8.30 (see also Chapter 10). The filters could be implemented as IIR or FIR structures, or could simply be an appropriate analytical rule to determine the parameters a_1 , a_2 , θ_1 , and θ_2 at each instant.

The MSK format is an interesting example to further highlight some of the preceding points. Here, the issue is not what is the most “natural” representation, for, as we have seen, *both* the serial and quadrature forms are natural in the sense that both lead to meaningful implementations. In fact, the unique relationship between the frequency separation and the data rate has led to *several* distinct implementations (see, e.g., Refs. 72–76). The issue for the simulator is, rather, *which* implementation is being contemplated because different implementations are, in general, subject to different types of impairments, and in fact may be specified in quite different terms. Lacking a choice of a specific implementation, the simulator must of course choose one. To illustrate the modeling issue further, let us consider the block

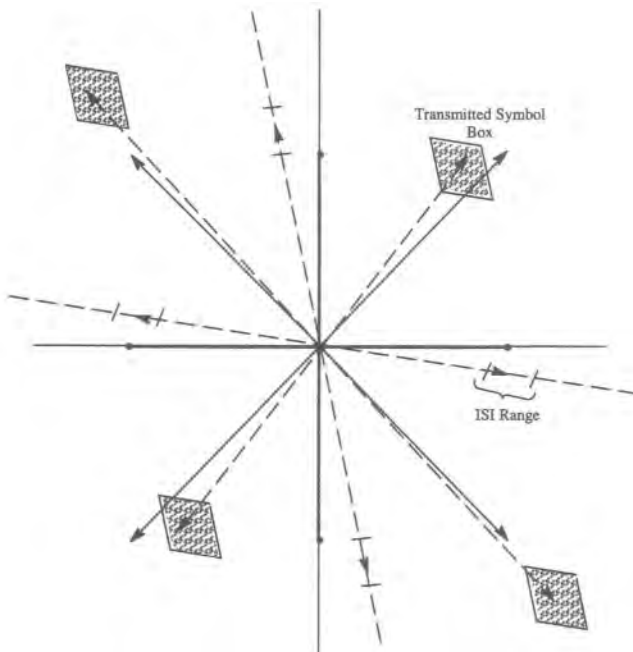


Figure 8.30. Illustration in signal space of some forms of imperfection for a quadrature modulator: amplitude and phase unbalance and ISI. Solid lines indicate ideal phasors.

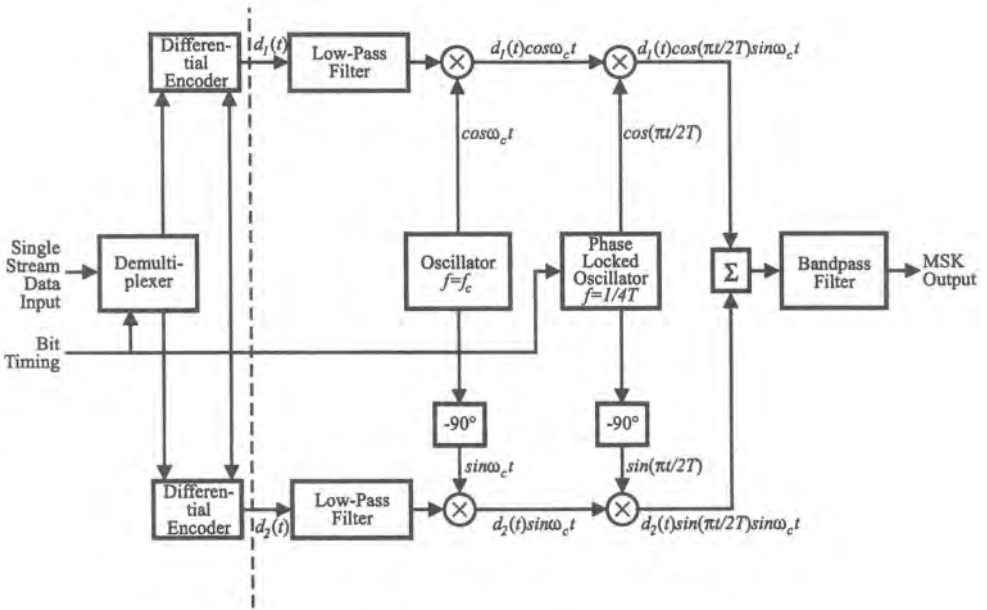


Figure 8.31. Block diagram for ideal implementation of quadrature-type MSK modulator.

diagram of Figure 8.31, which represents a shaped-pulse quadrature implementation. Ideally implemented, this modulator produces the signal

$$X(t) = X_1(t) \cos(2\pi f_c t) + X_2(t) \sin(2\pi f_c t)$$

where $X_1(t) = d_1(t)\mathcal{C}(t)$ and $X_2(t) = d_2(t)\mathcal{S}(t)$, where $d_1(t)$ and $d_2(t)$ are the I- and Q-channel data signals, respectively, and $\mathcal{C}(t) = \cos[(\pi/2T)t]$ and $\mathcal{S}(t) = \sin[(\pi/2T)t]$. These waveforms were sketched in Figure 8.22. To develop a realistic simulation model, we again essentially ask ourselves, “what can go wrong?” Figure 8.32 shows a modified version of the previous diagram, incorporating impairments that suggest themselves from the original block diagram. First, since the lowpass filter cannot be perfect, the nominally rectangular data streams $d_1(t)$ and $d_2(t)$ are transformed into filtered versions. Next, we can assume that no timing is perfect, hence the half-symbol delay is not exactly that. Similarly, phase shifts will not be exactly what they are intended to be, and this is reflected in the phase-shifter blocks. All of these impairments will generally be controlled by specifications. To provide a little appreciation for the impact of these impairments, we assume that the filtered data streams $g_1(t)$ and $g_2(t)$ have the trapezoidal waveform previously shown in Figure 8.5, but the timing relative to the sinusoidal pulse shaping is correct. Figure 8.33 shows the distortion in the resulting waveform. While any single impairment may not be major in itself, a succession of them can produce enough distortion in the end to induce significant degradation. It is the incorporation of these multiple real-life effects that simulation is especially well suited for.

8.8. Demodulation and Detection

Here again we will depart from our sequential path around the block diagram of Figure 8.1, because it is natural now to consider the counterpart of modulation at the receiver.

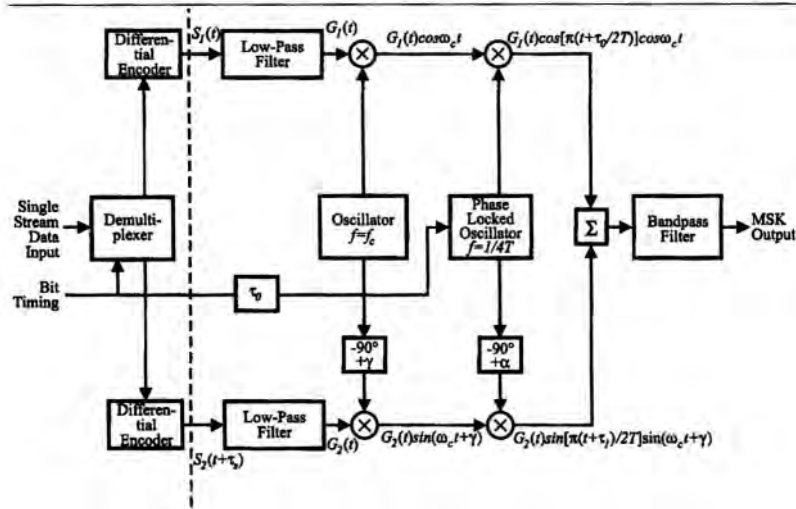


Figure 8.32. Block diagram for nonideal implementation of quadrature-type MSK modulator.

Actually, the word “counterpart” is something of a simplification because, although *demodulation* can certainly be thought of as the complementary operation or inverse of modulation, the demodulation method is not always unique. That is, it may be possible to demodulate a particular type of modulation using more than one distinct implementation. ($\pi/4$ QPSK was noted earlier as such an example.) Nevertheless, nearly all demodulators may be put into one of two classes: coherent demodulation, for which a locally generated carrier signal is used for demodulation, and noncoherent demodulation, which does not require a local carrier.

Demodulation is the inverse of modulation in the sense that, as modulation takes a lowpass signal into a passband about a carrier, demodulation takes a bandpass signal and

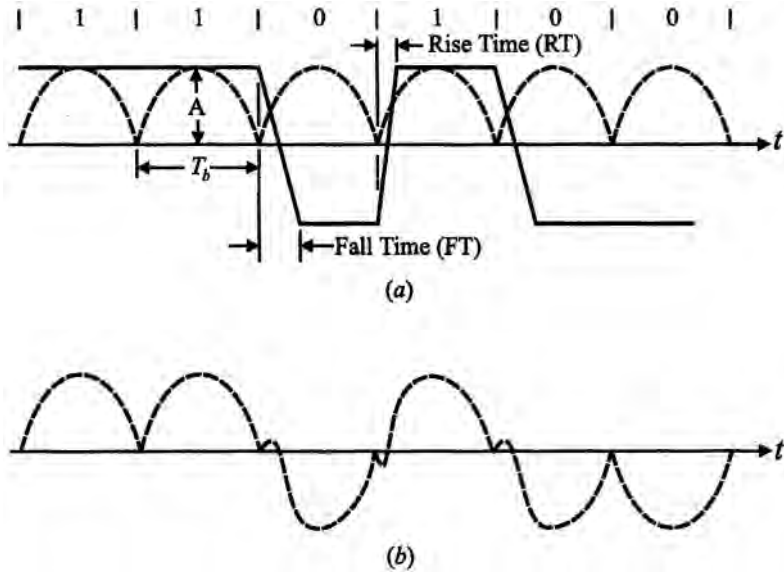


Figure 8.33. The waveform-distorting effects of modulator imperfections.

brings it back into a lowpass signal. By itself, however, this operation is generally not sufficient to recover the best estimate of the transmitted signal. Following the demodulation per se, some sort of processing or signal conditioning will be performed to extract the most faithful replica in the analog case, or the most reliable estimate of the symbol stream in the digital case. This processing will often be done by some type of filter designed expressly for the purpose at hand. We shall allude to this filtering here, but defer a detailed examination until the next section. We shall refer to this filtering as predetection filtering because in the digital case it is followed by *detection*, the process of forming a decision. This process can be as simple as sampling the predetection filter output and comparing to a threshold, or passing the result of this sampling to more complex algorithms such as equalizers or decoders when applicable. We consider these more complicated operations elsewhere.

Both demodulation (for coherent demodulators) and detection require synchronization signals. The synchronization circuitry is often the most complex part of the receiver, and again in this section we shall point to its presence, but defer a detailed discussion of it until later. For simplicity, we shall organize the remainder of the discussion under coherent or noncoherent demodulation.

The simulation of the receiver operations just mentioned involves distinct methodologies for each. Given a carrier reference signal, demodulation as such is simple, as will be seen. Predetection filtering using LTI filters is simulated using the same techniques as any other such filters, in the manner discussed in Chapter 4. In the next section we will show the form of several filter types and also discuss adaptive filters. Simulation techniques for the detection of coded digital signals have already been discussed in part in Section 8.6, and will be revisited in Chapter 11. Finally, a variety of techniques for simulating the synchronization functions will be reviewed in Section 8.12.

8.8.1. Coherent Demodulation

Suppose that we can generate a “local carrier” (or local oscillator) signal in the receiver of the form

$$\hat{C}(t) = 2 \cos(2\pi\hat{f}_c t + \hat{\theta})$$

where $\hat{f}_c$ and $\hat{\theta}$ are estimates, respectively, of the carrier frequency and phase generated by the “carrier recovery” mechanism in the receiver. Then, for the general quadrature representation (8.7.10), we can form estimates of $X_1(t)$ and $X_2(t)$ using the “generic” quadrature demodulator arrangement shown in Figure 8.34. The predemodulation or front-end filter in the figure limits the amount of noise, but ideally does not distort the signal. As mentioned, the predetection (LPF) filters shown condition the signal (such as matched filtering) for further processing. It is easy to show that

$$\begin{aligned} \hat{X}_1(t) &= X_1(t) \cos 2\pi[(f_c - \hat{f}_c)t + (\theta - \hat{\theta})] \\ &\quad - X_2(t) \sin 2\pi[(f_c - \hat{f}_c)t + (\theta - \hat{\theta})] \end{aligned} \quad (8.8.1a)$$

$$\begin{aligned} \hat{X}_2(t) &= X_2(t) \cos[2\pi(f_c - \hat{f}_c)t + (\theta - \hat{\theta})] \\ &\quad + X_1(t) \sin[2\pi(f_c - \hat{f}_c)t + (\theta - \hat{\theta})] \end{aligned} \quad (8.8.1b)$$

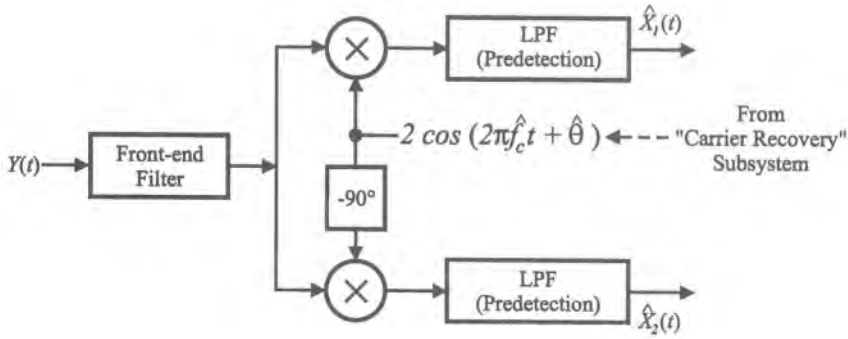


Figure 8.34. “Generic” quadrature demodulator.

If $\hat{f}_c = f_c$ and $\hat{\theta} = \theta$, i.e., if the carrier recovery mechanism provides error-free estimates of f_c and θ , then $\hat{X}_1(t) = X_1(t)$ and $\hat{X}_2(t) = X_2(t)$ (assuming no noise). As was the case with modulators, we can incorporate possible imperfections in the hardware. For example, we can visualize that $\hat{X}_1(t)$ (the *I*-channel) is recovered by using the local oscillator (LO) $2 \cos(2\pi\hat{f}_c t + \hat{\theta})$, and $\hat{X}_2(t)$ (the *Q*-channel) is obtained by using the LO $2a \cos(2\pi\hat{f}_c t - \pi/2 - \alpha)$, where a and α represent departures from ideal. The associated modifications to (8.8.1) are straightforward.

In practical systems, the input to the receiver is a distorted and noisy version of $Y(t)$. An additive model of the form

$$\tilde{Z}(t) = \tilde{W}(t) + \tilde{N}(t) \quad (8.8.2)$$

where $\tilde{W}(t)$ is the complex envelope of the distorted signal component and $\tilde{N}(t)$ is the complex envelope of the noise, is used to represent the input to the demodulator.

In terms of the complex envelope representation, the demodulator operation can be represented as

$$\begin{aligned} \hat{X}_1(t) + j\hat{X}_2(t) &= \{W_R(t) + n_c(t) + j[W_I(t) + n_s(t)]\} \\ &\times \exp[j2\pi(f_c - \hat{f}_c)t + j(\theta - \hat{\theta})] \end{aligned} \quad (8.8.3)$$

where $\tilde{W}(t) = W_R(t) + jW_I(t)$ and $\tilde{N}(t) = n_c(t) + jn_s(t)$. Sampled values of the preceding equation are what is implemented in simulating a coherent demodulator. The demodulated signals $\hat{X}_1(t)$ and $\hat{X}_2(t)$ are often referred to as *baseband* signals.

If $X_1(t)$ and $X_2(t)$ are lowpass analog signals, then $\hat{X}_1(t)$ and $\hat{X}_2(t)$ are usually filtered in order to minimize noise. In Section 8.9 the specific form of such a filter will be described.

If the demodulated signals $\hat{X}_1(t)$ and $\hat{X}_2(t)$ are digital signals, they must now be processed in some fashion so as to recover the original sequences $\{A_k\}$ and $\{B_k\}$ with as few errors as possible. We assume the original digital modulating signals $X_1(t)$ and $X_2(t)$ are given by (8.7.13), but for simplicity in the present context we assume $T_1 = T_2 = T$ and $D_1 = D_2 = 0$. Typically, two operations are performed on the demodulated signal, a filtering operation usually referred to as “matched filtering,” followed by sampling (observing) the signal. (We are using the term “matched filter” loosely here; strictly speaking, a filter that is “matched” to a pulse has a transfer function that is the complex conjugate of the pulse

spectrum. However, this usage is also widespread for a filter that *attempts* to do the matching.) These operations are sometimes collectively referred to as detection. The sampled signal values form the basis for making decisions as to what symbol was sent. In order to illustrate the points we wish to make, let us assume first that $\hat{f}_c = f_c$ and $\hat{\theta} = \theta$. Let us assume also that the distortion arises from a channel with impulse response $h_c(t)$, and also denote by $h_m(t)$ the matched filter impulse response. Then the waveforms at the input to the sampler can be written as

$$S_1(t) = \sum_{k=-\infty}^{\infty} A_k p(t - kT - D) * [h_c(t) * h_m(t)] + n_1(t) \quad (8.8.4a)$$

$$S_2(t) = \sum_{k=-\infty}^{\infty} B_k p(t - kT - D) * [h_c(t) * h_m(t)] + n_2(t) \quad (8.8.4b)$$

where D is a fixed, but unknown transmission delay, $*$ represents convolution, and $n_1(t) = n_c(t) * h_m(t)$, $n_2(t) = n_s(t) * h_m(t)$. In order to recover $\{A_k\}$ and $\{B_k\}$, the waveform (8.8.4) must be sampled at the appropriate instants. Thus, the sampling operation takes place at the instants

$$t_k = kT_b + \hat{D} \quad (8.8.5)$$

where $\hat{D}$ is an estimate of the “system” delay. The estimate $\hat{D}$ is provided by the “timing recovery” system in the receiver (Figure 8.35). In the simulation context there is normally no propagation delay, and the effective delay due to filtering can be estimated fairly accurately beforehand. Thus, an adequate “guess” at $\hat{D}$ can often be made without explicitly simulating timing recovery circuitry. This subject is discussed at greater length in Section 8.12. Proper selection of $h_m(t)$ is important for good detection performance. We will speak again about matched filtering in Section 8.9.5. Note that sampling for symbol detection takes place at the symbol rate, whereas simulation progresses at the sampling rate.

Both the carrier recovery and timing recovery operations are quite complex and very similar in their basic implementations. Both operations are normally mechanized with nonlinear feedback loops, the central element of which is a phase-locked loop (PLL) or a related structure. We will discuss carrier and timing recovery at greater length in Section 8.12.

In Section 8.7 we pointed out that the generic quadrature modulator might be topologically altered in particular instances to reflect properties specific to one or another modulation scheme belonging to the general class. This idea applies even more so in the case of

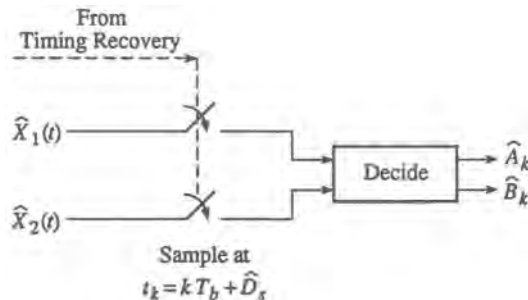


Figure 8.35. Sample-and-decide arrangement.

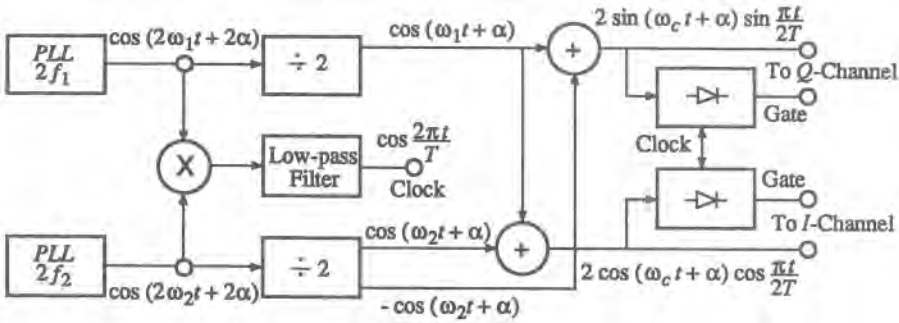


Figure 8.36. Integrated structure for the generation of carrier and timing references for MSK. (from Ref. 77, © IEEE, 1972).

demodulators, since there can be many variations of demodulator architecture, even for the same modulation scheme. As an example of this point, Figure 8.36 shows a modification of the topology of the generic quadrature receiver that applies specifically to the synchronization functions for an MSK signal. This architecture takes advantage of the properties peculiar to an MSK signal,^(77,78) in particular the fact that such a signal can be viewed either as a QAM signal of the form (8.7.10), or as a binary FSK signal with $h = 0.5$. Viewed in the latter form, it can be shown that squaring the MSK signal generates two spectral lines, at $2f_c \pm (1/2T)$, where $2T$ is the bit duration in the I or Q channel. Because of this particular relationship, the manipulations shown in Figure 8.36 result directly in separate I- and Q-channel carrier or “subcarrier” references as well as clock (timing) recovery.

The main implication for simulation is that different structures are potentially subject to different kinds of impairments. Hence, it is important to study the intended implementation, if known, or perhaps study several to develop a sense of their sensitivity to imperfections. In early stages of system design, however, a generic receiver may be quite adequate to study the tradeoffs between different types of modulation schemes.

8.8.2. Noncoherent Demodulation

The term *noncoherent demodulation* implies that the demodulator does not require a coherent local carrier to perform its function. Coherence in this context refers to knowledge of the carrier phase. Although phase is not needed, approximate knowledge of carrier *frequency* is needed in order properly to place the receiver passband around the signal spectrum. Note that some modulation schemes, such as FSK, may be demodulated by coherent or non-coherent structures.

8.8.2.1. Amplitude Demodulation

Consider the case when $Y(t)$ is an amplitude-modulated signal of the form

$$Y(t) = [1 + kX(t)] \cos(2\pi f_c t + \theta) \quad (8.8.6)$$

If $|kX(t)| \leq 1$, then by observing the real envelope of $Y(t)$, which is $[1 + kX(t)]$, we can extract the modulating signal $X(t)$. That is, since

$$|1 + kX(t)| = 1 + kX(t) \quad \text{when} \quad |kX(t)| < 1$$

except for the dc offset of 1 and scale factor k , the envelope contains $X(t)$ intact. A demodulation operation that retrieves the envelope is called envelope detection. Because of noise or distortion, $\tilde{Y}(t)$ is generally complex, hence the envelope detection operation simply involves taking the absolute value of the complex envelope.

Another noncoherent amplitude demodulation method involves taking the square of the magnitude of the complex envelope. This method is called *square law* demodulation. If the input signal is given by (8.8.6), square law demodulation takes the form

$$[1 + kX(t)]^2 = 1 + k^2 X^2(t) + 2kX(t) \approx 1 + 2kX(t) \quad \text{for} \quad |X(t)| \ll 1$$

Thus, this method requires a small modulation index. In terms of the complex envelope, both of the above noncoherent methods can be expressed as

$$\hat{X}(t) = \begin{cases} |\tilde{Z}(t)| & \text{envelope demodulation} \\ |\tilde{Z}(t)|^2 & \text{square law demodulation} \end{cases} \quad \begin{matrix} (8.8.7a) \\ (8.8.7b) \end{matrix}$$

where $\tilde{Z}(t)$ is the complex envelope at the input to the demodulator and $\hat{X}(t)$ is the real output of the demodulator.

8.8.2.2. Discriminator Detection of PM/FM Signals

Frequency-modulated and phase-modulated signals can be demodulated using a variety of schemes. The classical scheme for FM signals is termed *discriminator detection*. (Since phase is the integral of frequency, a PM demodulator is a discriminator followed by an integrating filter.) In principle, the operation is intended to be precisely the inverse of FM modulation. Looking at the FM signal (8.7.8), we see that the modulating signal is recovered by differentiating the phase. Thus, as in (8.7.12a), if

$$\tilde{W}(t) = R(t)e^{j\Psi(t)}$$

is the complex envelope of the modulating signal, we see that

$$X(t) = \frac{d}{dt} \Psi(t) \quad (8.8.8)$$

The actual received signal is a distorted and noisy version, say,

$$\tilde{W}_0(t) = R_0(t)e^{j\Psi_0(t)} \quad (8.8.9)$$

so that, in the simulation context, ideal discriminator detection can be implemented by differentiating the argument of the complex envelope, namely

$$\hat{X}(t) = \frac{d}{dt} \psi_0(t) \quad (8.8.10)$$

Although (8.8.10) is simple in appearance, care must be taken in simulating differentiation especially if one wishes to study operation below threshold, where FM clicks are significant contributors to noise.^(79,80)

Of course, in simulation we must implement this operation in discrete-time, hence we will refer to imitating the RHS of (8.8.10) as discrete-time differentiation (DTD), which can be considered a special case of numerical differentiation.⁽⁴⁰⁾ Numerical differentiation is known to be prone to inaccuracy in the presence of roundoff errors.⁽⁴¹⁾ We will briefly review some options for implementing DTD and their implications.

Probably the most natural approach to DTD is to approximate the derivative by a difference quotient using adjacent samples. Perhaps the most common version is the backward difference

$$\nabla^{(b)}(\psi_o(n)) = \frac{\psi_o(nT_s) - \psi_o((n-1)T_s)}{T_s} \approx \frac{d}{dt} \psi_o(t)|_{t=nT_s} \quad (8.8.11)$$

We would be tempted to suppose that the approximation would improve as T_s decreases, but this is true only up to a point, beyond which errors can increase rather than decrease⁽⁴¹⁾ in the presence of roundoff error. The discrete transfer function for the operator $\nabla^{(b)}$ is $(1 - z^{-1})/T_s$, so that its frequency response is given by

$$H_{\nabla^{(b)}}(f) = \frac{1 - e^{-j2\pi f T_s}}{T_s}, \quad |f| \leq \frac{f_s}{2} \quad (8.8.12)$$

which has the magnitude and phase responses

$$|H_{\nabla^{(b)}}(f)| = \sqrt{2f_s} \sqrt{1 - \cos 2\pi f T_s}, \quad |f| \leq f_s/2 \quad (8.8.13a)$$

and

$$\angle H_{\nabla^{(b)}}(f) = \tan^{-1} \frac{\sin(2\pi f T_s)}{1 - \cos(2\pi f T_s)} \quad (8.8.13b)$$

which are plotted as dashed lines in Figures 8.37 and 8.38, respectively.

Let us compare the preceding result to that for an ideal differentiator applied to a signal bandlimited to $f_s/2$. The frequency response for such a device is

$$H_d(f) = j2\pi f, \quad |f| \leq f_s/2 \quad (8.8.14)$$

with the corresponding magnitude and phase functions

$$|H_d(f)| = 2\pi |f|, \quad |f| \leq f_s/2 \quad (8.8.15a)$$

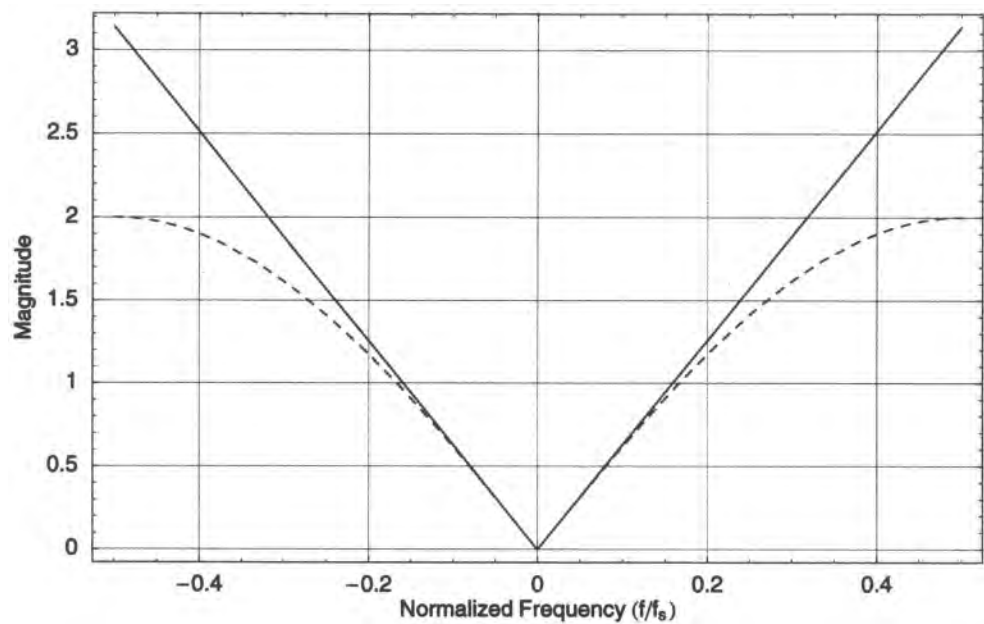


Figure 8.37. Transfer function magnitude for ideal differentiation (solid line) and discrete-time approximation (dashed line).

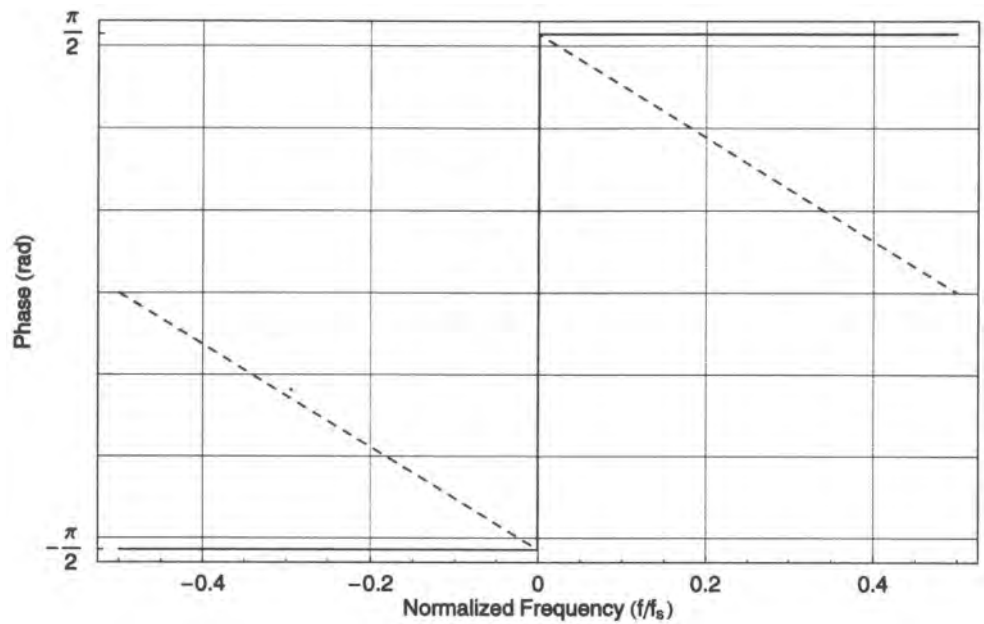


Figure 8.38. Transfer function phase for ideal differentiation (solid line) and discrete-time approximation (dashed line).

and

$$\angle H_d(f) = (\pi/2) \operatorname{sgn} f \quad (8.8.15b)$$

which are shown as solid lines in Figures 8.37 and 8.38, respectively. The approximate nature of the simple difference formula (8.8.11) is at least qualitatively evident from these figures. It can also be shown (Problem 8.12) that the discrete-time impulse response corresponding to (8.8.14) is given by

$$h_d(nT_s) = \frac{(-1)^n}{nT_s}, \quad n \neq 0 \quad (8.8.16)$$

Applying (8.8.16) to $\psi_o(t)$ yields

$$\hat{X}(t) = \sum_{k=-\infty, \neq 0}^{\infty} \frac{1}{kT_s} (-1)^k \psi_o((n-k)T_s) \quad (8.8.17)$$

which can be seen to be an alternating sequence of weighted divided differences. Of course, if we wanted to implement (8.8.17), it would be necessary to truncate h_d , or find an IIR approximation, but in any case we would also have error from the fact that the real signal is not bandlimited to begin with. (See, e.g., Ref. 81 for some discussion of FIR approximations to a discrete-time differentiator.) However, an alternative which comes at essentially no cost,⁽⁴¹⁾ but provides an order-of-magnitude improvement over (8.8.11) is to use a “symmetric difference” $\nabla^{(s)}$ rather than the “one-sided” difference in (8.8.11):

$$\nabla^{(s)}(\psi_o(nT_s)) = \frac{\psi_o((n+1)T_s) - \psi_o((n-1)T_s)}{2T_s} \quad (8.8.18)$$

Perhaps the best compromise between computational burden and accuracy is to differentiate an interpolating polynomial.⁽⁴¹⁾ Let us write $\psi_o(t)$ as

$$\psi_o(t) = P_{2m}(t) + r_m(t) \quad (8.8.19)$$

where $P_{2m}(t)$ is a degree- $2m$ polynomial passing through $\psi_o((n-m)T_s)$, $\psi_o((n-m+1)T_s)$, $\dots$, $\psi_o(nT_s)$, $\psi_o((n+1)T_s)$, $\dots$, $\psi_o((n+m)T_s)$ and $r_m(t)$ is a remainder. For notational convenience, we relabel the preceding sequence $\psi_o(k)$, $k = 0, 1, \dots, 2m$, so that $k = j$ corresponds to $t = t_j = (n-m+j)T_s$, etc. The polynomial then has the form

$$P_{2m}(t) = a_0 + \sum_{k=1}^{2m+1} a_k \prod_{j=0}^{k-1} (t - t_j)$$

where the coefficients $\{a_i\}$ are extracted from a table of divided differences. The derivative, evaluated at $t = nT_s$, is given by (Problem 8.13)

$$P'_{2m}(nT_s) = \sum_{k=1}^{2m+1} a_k b_k T_s^{k-1} \quad (8.8.20)$$

which can readily be calculated on the fly. The factors b_k can be precomputed for any value of m ; for example, for $m = 2$, $b_1 = 1$, $b_2 = 3$, $b_3 = 2$, $b_4 = -2$, $b_5 = 4$. The error is given by

$$r'_m(nT_s) = (T_s)^{2m} \frac{\psi_o^{(2m+1)}(\xi)}{(2m+1)!}$$

for some ξ in the interval $nT_s \leq \xi \leq (n+1)T_s$. A practical procedure might use $m = 2$ or 3 to form $P'_{2m}(nT_s)$ on a sliding basis.

Aside from these implementation issues, it should also be kept in mind that differentiation emphasizes the spectrum for larger frequencies. Hence, the possibility exists to increase aliasing noise (Problem 8.14). This should be kept in mind when choosing the initial sampling rate.

8.8.2.3. PLL Demodulation of PM/FM Signals

Probably the most widely used demodulator for PM and FM signals is the phase-locked loop (PLL), which, when used for this purpose, is referred to as a PLL demodulator; see Figure 8.39. Strictly speaking, PLL demodulation is a coherent process, or at least partially coherent. But in this case the demodulation method is such that no advantage can be taken of coherence, in terms of the SNR above threshold, although the threshold is lowered. The PLL demodulator relies on a device called a voltage-controlled oscillator (VCO) whose output is

$$V(t) = \sqrt{2}K_1 \cos \left[2\pi f_c t + K_2 \int Z(\alpha) d\alpha \right] \quad (8.8.21)$$

K_1 and K_2 are gain constants, f_c is the quiescent operating frequency (assumed equal to the unmodulated input carrier frequency), and $Z(t)$ is the “control” voltage. It can be easily verified that the feedback loop generates an error signal

$$\epsilon(t) = AK_1 K_m \sin[\theta(t) - \hat{\theta}(t)]$$

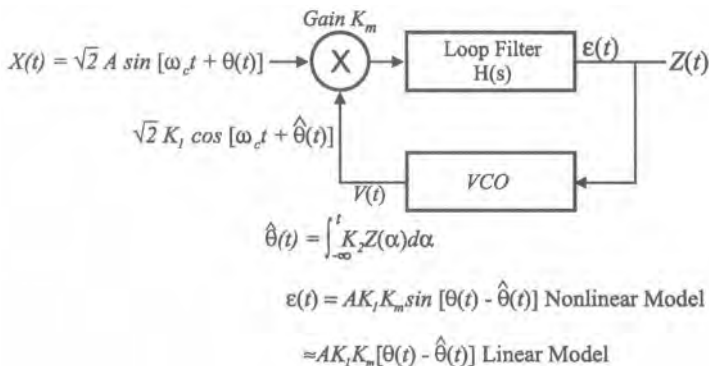


Figure 8.39. Phase-locked loop.

If the loop filter and the constants are chosen carefully, then the loop can be made to “lock” onto, or “track,” the input phase, i.e., the loop establishes the conditions

$$\hat{\theta}(t) \approx \theta(t), \quad \epsilon(t) \approx 0$$

Since the error signal $\epsilon(t)$ is nonlinear in $\theta(t) - \hat{\theta}(t)$, the loop is nonlinear and the analysis of the nonlinear loop is difficult; simulation, although more straightforward, can still be tricky (see Section 8.12). A linear assumption of the form

$$\sin[\theta(t) - \hat{\theta}(t)] \approx \theta(t) - \hat{\theta}(t) \quad (8.8.22)$$

is customarily made to simplify the analysis. With this assumption the PLL is an ordinary feedback loop whose response can be analyzed using the closed-loop transfer function. (See, e.g., Ref. 82). The nonlinear operation of the PLL can be simulated using the lowpass-equivalent model shown in Figure 8.40.

The PLL is a device of fundamental importance in communication systems. We already mentioned that the PLL is the basic structure for carrier recovery as well as for timing recovery in digital systems, and we see here its application in angle demodulation. In order properly to simulate a PLL using the sampled (discrete-time) version of the complex lowpass-equivalent model shown in Figure 8.40, it is necessary to understand the relationship between this model and the actual (continuous structure), which is described by a nonlinear differential equation. Some of the considerations relating to the solution of such equations by simulation have been discussed in Chapter 5, and in Section 8.12 we shall examine in more detail the nature of the discrete-time approximation to the PLL. For now it suffices to point out that when a PLL is simulated using the complex lowpass-equivalent model shown in Figure 8.40, special attention must be paid to the following:

1. The sampling rate must be much larger than the loop bandwidth in order to simulate accurately the nonlinear behavior.
2. FFT-type block processing operations cannot be used in the loop since such processing introduces a delay which may make the loop unstable.

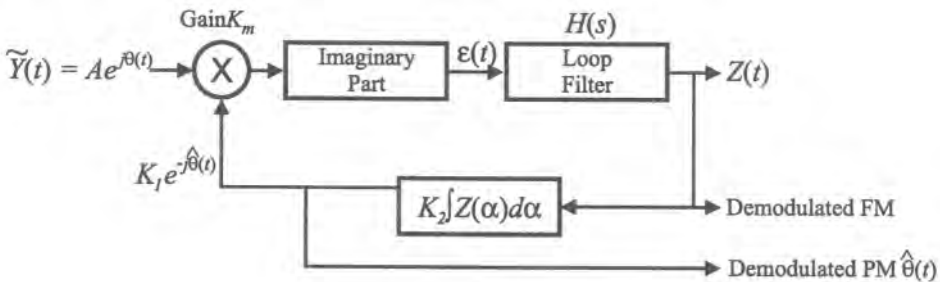


Figure 8.40. Complex lowpass equivalent of PLL.

8.9. Filtering

Here again, we cannot strictly follow the flow of Figure 8.1, for filtering is generally distributed throughout a system for different purposes, not to mention filtering which is inadvertent. The primary applications of filtering in communication systems are to select desired signals, minimize the effects of noise and interference, modify the spectra of signals, and shape the time-domain properties (zero crossings and pulse shapes) of digital waveforms to improve their detectability.

An example of the first application, namely, the selection of a desired signal, occurs in radio and TV receivers when we “tune” to pick up one of many stations that are broadcasting simultaneously. Receivers also use filters to reject “out-of-band” noise. Transmitters, on the other hand, have to meet regulatory constraints on the shape of the transmitted spectra or on “out-of-band” power, and filtering is used for controlling these.

Time-domain properties such as zero crossings and pulse shapes are important in digital communication systems. These properties can also be controlled via appropriate pulse-shaping filters.

In the following sections, we discuss the form of filter transfer functions for various types of applications in communication systems. Once the transfer functions are explicitly known, we can simulate these filters using the techniques described in Chapter 4.

Filters used in communications systems are sometimes required to be adaptive. These filters are required to change their response when the properties of the input signals change. For example, a filter designed to remove signal distortion introduced by a channel should change its response as the channel characteristics change. The most commonly used adaptive filter structure (also commonly referred to as an equalizer) is the adaptive tapped delay line (TDL). Details of the adaptive TDL are also presented in this section.

All the filters described in this section are linear and time-invariant except for the adaptive TDL filter, which is linear and time-varying.

8.9.1. Filters for Spectral Shaping

It was shown in Chapter 6 that the power spectral density $S_{YY}(f)$ of the output $Y(t)$ of a linear time-invariant system is

$$S_{YY}(f) = S_{XX}(f)|H(f)|^2 \quad (8.9.1)$$

where $S_{XX}(f)$ is the PSD of the input signal $X(t)$ and $H(f)$ is the transfer function of the system. By carefully selecting $H(f)$, we can emphasize or deemphasize selected spectral components of the input signals. This frequency-selective filtering operation is used extensively in communications systems. If $S_{XX}(f)$ and $H(f)$ are given, calculation of $S_{YY}(f)$ is straightforward. On the other hand, suppose that $S_{YY}(f)$ is a desired output PSD given that $S_{XX}(f)$ is an input PSD. Equation (8.9.1) can be rewritten as

$$|H(f)|^2 = \frac{S_{YY}(f)}{S_{XX}(f)} \quad (8.9.2)$$

and the problem here is to synthesize a realizable filter $H(f)$ that satisfies (8.9.2). This can be done by factoring the right-hand side of Equation (8.9.2) and including only poles and zeros

in the left half of the s plane for specifying $H(s)$. (The transformation $s = j2\pi f$ is applied before factoring.) This is called spectral factorization. Of course, it is implied that the right side of (8.9.2) is a ratio of polynomials in s . If this is not so, which is frequently the case, then one must first develop an approximation that is a ratio of polynomials.^(83,84) Examples of spectral factorization were presented in Chapters 3 and 7.

8.9.2. Filters for Pulse Shaping

The individual pulses in the waveforms used in digital transmission systems are typically required to satisfy two important conditions: bandwidth and zero crossings. If the embedded digital sequence has a symbol rate of R_b , then the bandwidth of the digital waveform should be of the order of R_b and it might be required to have zero crossings in the time domain once every T_b seconds, where $T_b = 1/R_b$. Nyquist showed that a time-domain pulse $p(t)$ will have zero crossings once every T_b seconds if its transform $P(f)$ meets the following constraint:

$$\sum_{k=-\infty}^{\infty} P(f + kR_b) = T_b \quad \text{for } |f| < R_b/2 \quad (8.9.3)$$

If $P(f)$ satisfies Equation (8.9.3), then $p(t)$ has the following properties:

$$\begin{aligned} p(0) &= 1 \\ p(kT_b) &= 0, \quad k = \pm 1, \pm 2, \dots \end{aligned} \quad (8.9.4)$$

Zero-crossing requirements are imposed to provide zero intersymbol interference (ISI) and assist in the timing recovery process.

A family of $P(f)$ which meets the Nyquist criterion given in Equation (8.9.3) is the *raised cosine* (or *cosine rolloff*) family with

$$P(f) = \begin{cases} T_b, & |f| \leq R_b/2 - \beta \\ T_b \cos^2 \frac{\pi}{4\beta} \left(|f| - \frac{R_b}{2} + \beta \right), & \frac{R_b}{2} - \beta < |f| \leq \frac{R_b}{2} + \beta \\ 0, & |f| > \frac{R_b}{2} + \beta \end{cases} \quad (8.9.5)$$

where β is the “excess bandwidth” parameter. Note that the raised cosine $P(f)$ is bandlimited to $\beta + (R_b/2)$ and hence it can meet selected bandwidth constraints. (See Figure 8.41a.)

The raised cosine family has an impulse response of the form

$$p(t) = \frac{\cos 2\pi\beta t}{1 - (4\beta t)^2} \left(\frac{\sin \pi R_b t}{\pi R_b t} \right) \quad (8.9.6)$$

(see Figure 8.41b) and it has zero crossings once every T_b seconds.

The raised cosine pulse can be produced by passing an impulse through a filter with $H(f) = P(f)$. If the input to the filter is $g(t)$ rather than an impulse, then, in order to produce an output in the raised cosine family, the filter transfer function should be $H(f) = P(f)/G(f)$, where $G(f)$ is the Fourier transform of $g(t)$.

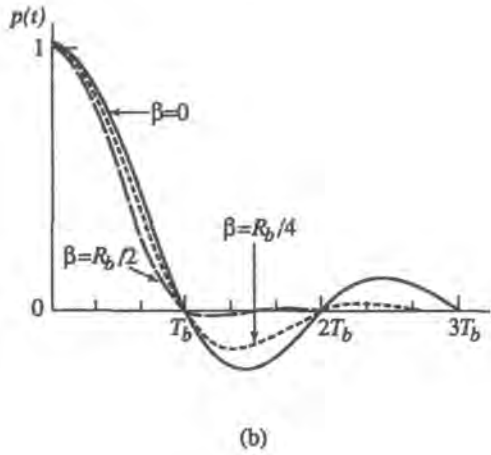
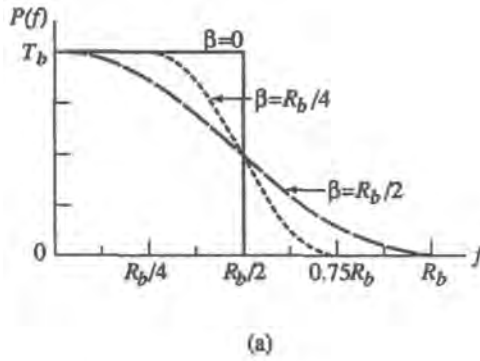


Figure 8.41. Pulses with raised cosine frequency characteristics. (a) $P(f)$ for three different values of β [note that $P(f) = P(-f)$]; (b) $p(t)$ for three different values of β .

In most applications the filtering operation to produce $P(f)$ is split between two filters, one at the transmitter, with transfer function denoted $H_T(f)$, and one at the receiver, with transfer function denoted $H_R(f)$ (see also Section 8.9.4).

With an impulse at the input to $H_T(f)$ and an ideal channel, we still must have $H_T(f)H_R(f) = P(f)$. In the relatively simple case where we have an additive noise source at the receiver input, it can be shown⁽¹⁰⁾ that the optimum partition, in the sense of optimizing the signal-to-noise ratio at the receiver output, is to split $P(f)$ equally between the transmitter and the receiver, i.e.,

$$H_T(f) = H_R(f) = [P(f)]^{1/2} \quad (8.9.7)$$

In Section 8.9.4, Equation (8.9.7) will be seen to be a special case of a more general optimization problem. This so-called “square root raised cosine” filter has the following impulse response⁽⁸⁾:

$$h_R(t) = h_T(t) = 8\beta \frac{\cos[(R_b + 2\beta)\pi t] + \sin[(R_b - 2\beta)\pi t](8\beta t)^{-1}}{(\pi R_b^{1/2})[1 - (8\beta t)^2]} \quad (8.9.8)$$

When implementing a filter such as (8.9.8) in the time domain, care should be taken to circumvent numerical difficulties that can arise when, for example, $x = 0$ in $(\sin x)/x$.

8.9.3. Linear Minimum MSE Filters

In many analog communication systems we use filters to estimate a signal $X(t)$ from a noisy version $Y(t)$. The filter response is chosen to minimize the mean squared error (MSE),

$$\text{MSE} = E\{[X(t) - \hat{X}(t)]^2\}$$

where $\hat{X}(t)$ is the estimate of $X(t)$ produced by the linear filter

$$\hat{X}(t) = \int_{-\infty}^{\infty} h(t, \tau) Y(\tau) d\tau$$

If we assume stationarity of $X(t)$ and $Y(t)$, then it can be shown that the filter will be time-invariant, and the transfer function that minimizes the MSE is

$$H(f) = \frac{S_{XY}(f)}{S_{YY}(f)} \quad (8.9.9)$$

where $S_{XY}(f)$ is the cross PSD between $X(t)$ and $Y(t)$, and $S_{YY}(f)$ is the PSD of $Y(t)$. If $Y(t) = X(t) + N(t)$, where $N(t)$ is additive noise that is uncorrelated with $X(t)$, then the filter transfer function reduces to

$$H(f) = \frac{S_{XX}(f)}{S_{XX}(f) + S_{NN}(f)} \quad (8.9.10)$$

and the minimum MSE (MMSE) is given by

$$\text{MMSE} = \int_{-\infty}^{\infty} \frac{S_{XX}(f)S_{NN}(f)}{S_{XX}(f) + S_{NN}(f)} df \quad (8.9.11)$$

The filter given in Equation (8.9.10) simply weights each spectral component in proportion to the relative strength of the signal and noise components at each frequency.

Equation (8.9.11) can also be written as

$$\begin{aligned} \text{MMSE} = & \int_{-\infty}^{\infty} [1 - H(f)]S_{XX}(f)[1 - H^*(f)] df \\ & + \int_{-\infty}^{\infty} H(f)S_{NN}(f)H^*(f) df \end{aligned} \quad (8.9.12)$$

where $H(f)$ is given by (8.9.10); the first term on the right-hand side represents the signal distortion, and the second term represents the power in the noise term at the filter output. The sum of the power in the signal distortion term and the noise term equals the MSE.

The filter given by (8.9.9) is called the Wiener filter, and it is noncausal. Causal versions of the filter can be derived using spectral factorization techniques,^(85,86) or by truncating and delaying the impulse response of the nonrealizable filter by an appropriate amount.

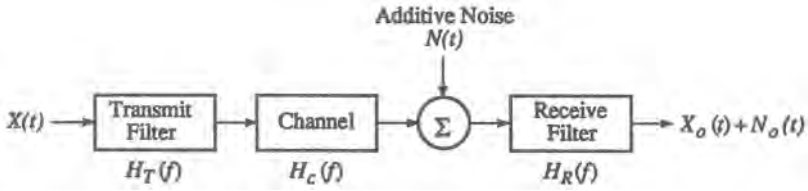


Figure 8.42. Channel model showing preemphasis (or transmit) filter and deemphasis (or receive) filter.

8.9.4. Filters for Minimizing Noise and Distortion

While the filter given in Equation (8.9.9) minimizes the MSE, it introduces a distortion in the signal component. If we can use two filters in the system, a transmit (preemphasis) filter $H_T(f)$ and a receive (deemphasis) filter $H_R(f)$ as shown in Figure 8.42 then we can choose their transfer functions such that $X_0(t) = kX(t - t_d)$ (where k is a constant and t_d is an arbitrary delay) and $E[N_0^2(t)]$ is minimized. That is, we can minimize the noise power at the output of the receive filter subject to the constraint of no signal distortion. With two filters we can meet the dual criteria of distortionless transmission and minimum noise power (or maximum signal-to-noise ratio).

The transfer function of $H_T(f)$ and $H_R(f)$ can be shown to be equal to⁽¹⁴⁾

$$|H_R(f)|^2 = \left[\frac{S_{XX}(f)}{S_{NN}(f)} \right]^{1/2} \frac{1}{|H_c(f)|} \quad (8.9.13a)$$

$$|H_T(f)|^2 = \left[\frac{S_{NN}(f)}{S_{XX}(f)} \right]^{1/2} \frac{1}{|H_c(f)|} \quad (8.9.13b)$$

with $k = 1$. The phase response of each of the filters is chosen to be linear. Note that spectral factorization is involved in obtaining $H_T(f)$ and $H_R(f)$ from $|H_T(f)|^2$ and $|H_R(f)|^2$.

The transmit filter given in (8.9.13a) amplifies or emphasizes weaker spectral components of the signal that might be masked by strong spectral components of the noise. The receive filter deemphasizes the same components inversely, so that the net result preserves the spectral properties of the input signal $X(t)$. This arrangement of preemphasis/deemphasis filters is used in FM broadcast and audio recording.

In digital transmission systems, the transmit and receive filters used to shape an input pulse waveform

$$X_T(t) = \sum_{k=-\infty}^{\infty} a_k p_T(t - kT_b)$$

into an output waveform

$$X_0(t) = \sum_{k=-\infty}^{\infty} A_k p_R(t - kT_b), \quad A_k = K_c a_k$$

while maximizing $E[A_k^2]/E[N_0^2(t)]$ have the following transfer functions⁽¹⁴⁾:

$$|H_R(f)|^2 = \frac{K|P_R(f)|}{|H_c(f)||S_{NN}(f)|^{1/2}} \quad (8.9.14a)$$

$$|H_T(f)|^2 = \frac{K_c^2 |P_R(f)| [S_{NN}(f)]^{1/2}}{K |P_T(f)|^2 |H_c(f)|} \quad (8.9.14b)$$

where K is an arbitrary positive constant. Figure 8.42 also applies topologically to this situation, but the interpretation and form of the transfer functions are evidently different for the case that minimizes the MSE.

If $p_T(t) = \delta(t)$ and $S_{NN}(f)$ is constant over the bandwidth of $P_R(f)$, then it can be shown that

$$|H_R(f)|^2 = K_1 \frac{|P_R(f)|}{|H_c(f)|} \quad (8.9.15a)$$

$$|H_T(f)|^2 = K_2 \frac{|P_R(f)|}{|H_c(f)|} \quad (8.9.15b)$$

If $H_c(f)$ is constant over the bandwidth of $P_R(f)$, then

$$|H_T(f)| = |H_R(f)| = K_3 [P_R(f)]^{1/2}$$

which is precisely the “square-root filtering” discussed in Section 8.9.2. Of course, in order to produce the desirable zero-crossing properties mentioned there, $P_R(f)$ must be a member of the raised cosine family (or satisfy the Nyquist criterion).

8.9.5. Matched Filters

Consider the problem of detecting the presence or absence of a pulse with a known shape $p(t)$ by observing $p(t) + N(t)$, when $N(t)$ is zero-mean, stationary additive noise.

Let us assume that $p(t)$ is of duration T in the interval $[0, T]$ and

$$Y(t) = \begin{cases} p(t) + N(t) \\ \text{or} \\ N(t) \end{cases}$$

We observe $Y(t)$ for T seconds in the interval $[0, T]$ and determine whether or not $p(t)$ is present by processing $Y(t)$. This is a fundamental problem in digital communication systems, radar, and sonar.

Suppose we base our decision about the presence or absence of $p(t)$ based on the output of a filter at time $t = T$, that is, we process $Y(t)$ and obtain

$$Z = \int_0^T Y(\tau) h(T - \tau) d\tau$$

and base our decision on Z .

It can be shown that the filter that minimizes the average probability of incorrect decisions has the transfer function^(13,14)

$$H(f) = K \frac{P^*(f) \exp(j2\pi fT)}{S_{NN}(f)} \quad (8.9.16)$$

where K is a real, positive constant. If $N(t)$ is white noise, then

$$H(f) = KP^*(f) \exp(j2\pi fT) \quad (8.9.17)$$

By taking the inverse transform of $H(f)$, it can be shown that the impulse response of the filter is

$$h(t) = p(T - t) \quad (8.9.18)$$

so that

$$Z = \int_0^T Y(\tau)h(T - \tau) d\tau = \int_0^T Y(\tau)p(\tau) d\tau \quad (8.9.19)$$

Equations (8.9.18) and (8.9.19) show that the impulse response of the filter is “matched” to $p(t)$ and for this reason this filter is called the matched filter. Note that when $p(t)$ is the unit rectangular pulse, $p(t) = 1$ for $0 \leq t \leq T$ and zero elsewhere, Z is just the integral of the input. If a decision has to be made in successive intervals, (8.9.19) implies that this integration starts anew at the beginning of each such interval. Thus, the value of the integral at the end of the preceding interval must be discarded (or “dumped”). A mechanism for this process is the well-known integrate-and-dump (I&D) filter.

The solution (8.9.19) has been cast in the context of detection, but it obviously applies equally to decisioning for data transmission; the pulse detection problem is completely equivalent to on-off keying modulation and a little thought shows it is also equivalent to antipodal signaling.

All digital communication systems include a lowpass filter in the receiver which is intended to perform the task of matched filtering. However, the matched filter must be matched to the received pulse, not the transmitted pulse. The former is inevitably a distorted version of the latter. Thus, even if a rectangular pulse is sent, it will be received differently, and the I&D filter is no longer the matched filter. Furthermore, as we have seen (e.g., Figures 8.5 and 8.6), a digital signal is not always representable as a sequence of pulses with a single shape. Even if that were the case, ISI and nonlinearities would combine to produce the equivalent of multiple pulse shapes at the receiver. Thus, in practice, the idea of a matched filter to a single pulse shape is not realistic in many cases. If it were possible actually to determine a single received pulse, one could still in principle determine the matched filter from (8.9.18), but such a filter may not be easy to realize. Most distorted pulses can, in fact, be matched relatively closely by a low-order classical filter like a two-pole or three-pole Butterworth or Chebyshev filter set at an appropriate bandwidth. For simulation purposes, therefore, we recommend having these filters, as well as the classical I&D, available as choices for “matched” filtering when doing system studies.

Matched filtering can in principle compensate for linear distortion in pulses when that distortion is known, but it cannot compensate for intersymbol interference. As we have seen, one could control ISI (ideally eliminate it) if it were possible to design all the filters so that the received pulse belongs to the raised cosine family. An examination of (8.9.14) shows that such filters may not be simple to synthesize especially since they depend on the “channel” transfer function $H_c(f)$. The latter can represent a complex system as well as the intervening medium; moreover, the system may be nonlinear, for which we cannot strictly speak of a transfer function. Furthermore, the system and channel may not be well defined or may be time-varying. This suggests looking for a self-adaptive solution to the problem discussed, and this is taken up in the next section.

8.9.6. Adaptive Filtering (Equalization)

As we mentioned, the transfer functions given in (8.9.14) and (8.9.15) assumed that the channel response $H_c(f)$ is known. In many communications systems the channel response is only partially known and is often time-varying (e.g., fading in microwave, mobile, and HF channels; changes in channel response in switched telephone circuits). In order to compensate for these changes, one can conceive of an equalizing filter at the receiver with transfer function $H_{eq}(f, t)$ such that

$$H_{eq}(f, t) = [H_c(f, t)]^{-1} \quad (8.9.20)$$

that is, its frequency response can be adjusted to compensate for variations in the channel response.

The most commonly used equalizer filter structure is the tapped delay line (TDL) filter shown in Figure 8.43. (It may be noted that a TDL is an FIR filter.) The input to the equalizer, $Y(t)$, is passed through a sequence of delays and the output, $Z(t)$, is formed as the weighted sum of the delayed values of the input,

$$Z(t) = \sum_{k=-M}^M C_k Y(t - kT') \quad (8.9.21)$$

where the C_k , $k = -M, -M + 1, \dots, 0, 1, \dots, M$, are $2M + 1$ “tap gains,” and T' is the delay of each stage.

By taking the Fourier transform of both sides of (8.9.21) (assuming for now that the C_k are fixed), we obtain

$$Z(f) = \sum_{k=-M}^M C_k Y(f) \exp(-j2\pi k f T')$$

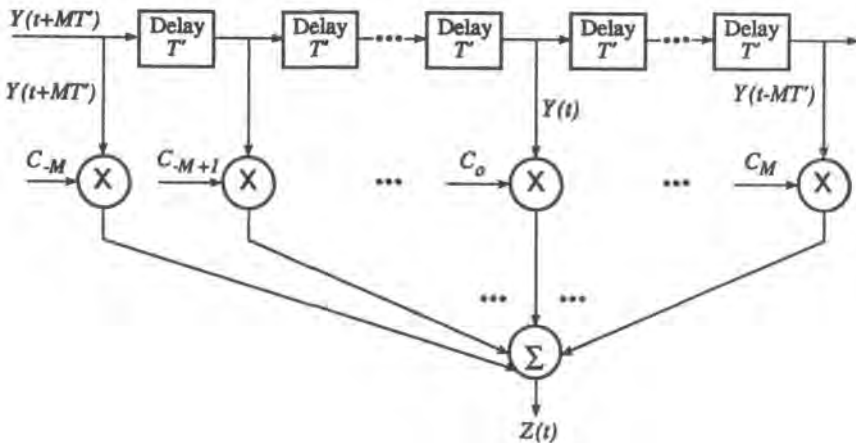


Figure 8.43. Tapped-delay-line (TDL) filter.

Rearranging the previous equation, we can express the transfer function of the filter as

$$H_{eq}(f) = \frac{Z(f)}{Y(f)} = \sum_{k=-M}^M C_k \exp(-j2\pi k f T') \quad (8.9.22)$$

The right-hand side of (8.9.22) is an expansion for $H_{eq}(f)$ as a (truncated) Fourier series in the frequency domain in the interval $[-1/2T', 1/2T']$ and it shows that the tap gains are the coefficients of this Fourier series expansion (Figure 8.44). Equation (8.9.22) also implies that the TDL structure can be used to synthesize filters, apart from considerations of time variability. In fact, it turns out that the optimum receiver for an arbitrary, but known and time-invariant, linear channel is exactly the matched filter previously discussed, followed by a (generally infinitely long) TDL filter.⁽¹⁰⁾ The delay per tap T' is a design choice. When $T' = T$, the symbol duration, the equalizer is termed a *synchronous* equalizer; when $T' = T/m$, m an integer, it is called a *fractionally spaced* equalizer: the most common case is $m = 2$. Advantages and disadvantages of synchronous and fractional equalizers are discussed in Ref. 11.

One of the best features of the TDL structure is the relative ease with which it can be modified to compensate either for time-varying channel characteristics or for lack of complete knowledge about such characteristics, even if they are time-invariant. Clearly, if the tap gains are changed, then from (8.9.22) we see that the transfer function of the TDL filter changes accordingly. The question is how to mechanize these changes in the correct way, since it may be quite difficult to measure accurately the channel characteristics. The answer is to make the equalizer *adaptive* by implementing a form of feedback algorithm that permits the equalizer to “know” that it is compensating properly. If the channel is continuously time-varying slowly, then the equalizer will continuously adapt.

One way to implement the adaptive version of the equalizer filter is to vary the tap gains so as to minimize the MSE between the actual equalizer output and the desired output (ideal output). Of course, in reality, unless we use a training sequence, the receiver does not know the ideal output, but under certain conditions one can assume that this output is almost known. For example, in the digital case the ideal output would be the error-free digital waveform input to the transmitter. If the channel does not introduce too much noise and distortion, then it is safe to say that the decisions are “almost” always correct and hence can be taken as the ideal output for the equalizer’s purposes. Other approaches to training the

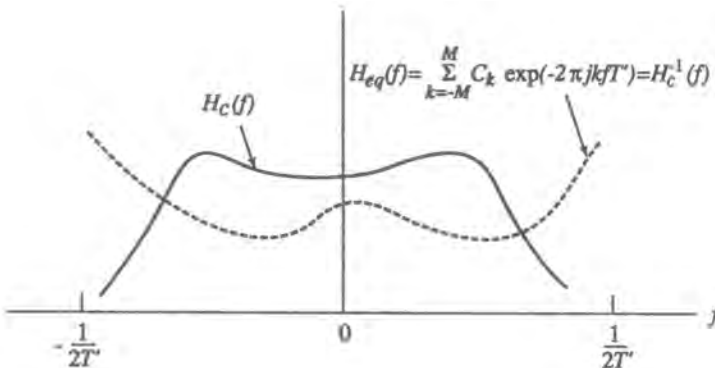


Figure 8.44. Relationship between the TDL transfer function, the tap gains, and the tap spacing.

equalizer without knowing the transmitted sequence (known as *blind* equalization) are discussed in Ref. 11.

There are different criteria that may be used to measure the degree of closeness between the desired signal and the equalized signal. Perhaps the most commonly used one (though not the only one) is the mean-square error (MSE). In the next subsection, we provide a brief exposition of the nature of adaptation algorithms designed to minimize the MSE. The section following provides a short derivation for obtaining the optimum tap gains (in the sense of minimizing the MSE) analytically when the channel is varying very slowly with respect to the information rate. Finally, we summarize the options for simulating equalizers and some of the related considerations.

8.9.6.1. Tap-Gain Adaptation for Minimizing MSE

Suppose we want to choose $\{C_k(t)\}$ such that $[X(t) - Z(t)]^2$ is minimized, where $X(t)$ is the desired output. Initially, we assume $X(t)$ and $Z(t)$ are given. {If $X(t)$ and $Z(t)$ are random processes, we will minimize $E[X(t) - Z(t)]^2$.} Commonly used adaptation algorithms are based on the gradient technique, which can be derived as follows. Starting with

$$\epsilon^2(t) = [X(t) - Z(t)]^2 = \left[X(t) - \sum_{k=-M}^M C_k Y(t - kT') \right]^2 \quad (8.9.23)$$

we can differentiate both sides to obtain

$$\begin{aligned} \frac{\partial \epsilon^2(t)}{\partial C_n} &= -2 \left[X(t) - \sum_k C_k Y(t - kT') \right] Y(t - nT') \\ &= -2\epsilon(t) Y(t - nT') \end{aligned} \quad (8.9.24)$$

The gradient algorithm for obtaining the optimum weights requires that their values be adjusted according to

$$\frac{dC_n(t)}{dt} = -\Delta' \frac{\partial \epsilon^2(t)}{\partial C_n} \quad (8.9.25)$$

where Δ' is a positive constant. Combining Equations (8.9.24) and (8.9.25) yields

$$\frac{dC_n(t)}{dt} = 2\Delta' \epsilon(t) Y(t - nT')$$

or

$$C_n(t) = C_n(t_0) + \Delta \int_{t_0}^t \epsilon(\tau) Y(\tau - nT') d\tau \quad (8.9.26)$$

where Δ is a positive constant which can be used to control the “step size” of the weight adjustment. The integration operation in (8.9.26) is a finite-time cross-correlation between the error signal and the time-shifted output.

If the error (8.9.23) is a convex function of the tap gains, the gradient algorithm guarantees convergence, namely,

$$\frac{\partial \epsilon(t)}{\partial C_n} \rightarrow 0$$

and hence

$$\frac{dC_n(t)}{dt} \rightarrow 0$$

when the optimum weight values are reached. Implementation of the adaptive tapped delay line filter is shown in Figure 8.45. It can be shown that the MSE is indeed a convex function of the tap gains (Problem P8.15). This means that the algorithm always converges eventually to within the accuracy permitted by the step size.

The convergence properties of the algorithm, such as the speed, depend on the nature of the error surface in the weight space and on the step size. To give an idea of the relationship of the step size to the dynamics of the process and the equalizer structure, one author⁽¹¹⁾ suggests that Δ be chosen as

$$\Delta = \frac{0.2}{(2M + 1)\langle Y^2(t) \rangle} \quad (8.9.27)$$

A variable step size, starting with larger values and tapering to smaller values as time progresses, can also be used to accelerate convergence and reduce granularity as the tap gains converge. For details of the gradient algorithm, the reader is referred to Refs. 87 and 88.

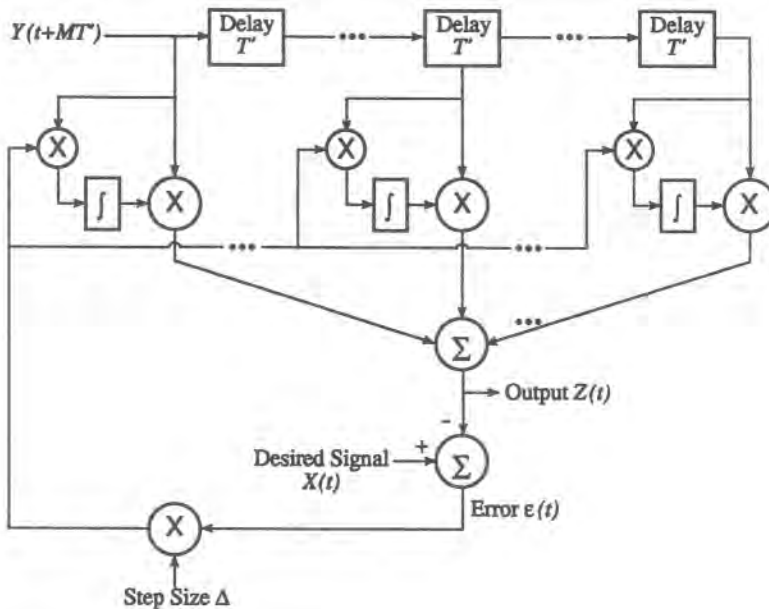


Figure 8.45. Adaptive tapped-delay-line filter.

When an adaptive equalizer is used in a QAM system, the baseband equalizer structure actually consists of four tapped delay lines.⁽⁸⁹⁾ One TDL compensates for ISI within the I channel; one compensates for ISI in the Q channel; the third and fourth TDL compensate, respectively, for crosstalk from the I into the Q channel and for crosstalk from the Q into the I channel. For simulation purposes, the equalizer can be represented as one TDL with complex coefficients operating on the complex envelope. Since the C_k will be complex-valued, (8.9.26) should be modified to

$$C_n(t) = C_n(t_0) + \Delta \int_{t_0}^t \epsilon(\tau) \tilde{Y}^*(t - nT') d\tau \quad (8.9.28)$$

where $\tilde{Y}^*$ represents the complex conjugate of the complex envelope.

In the case of random processes, as mentioned above, we want to minimize the expected value of the squared error. It follows that the equivalent of (8.9.24) in this case is

$$\frac{\partial \epsilon^2(t)}{\partial C_n} = -2E[\epsilon(t)Y(t - nT')] \quad (8.9.29)$$

It also follows that (8.9.26) becomes

$$C_n(t) = C_n(t_0) + \Delta \int_{t_0}^t E[\epsilon(\tau)Y(\tau - nT')] d\tau \quad (8.9.30)$$

But while (8.9.26) has a straightforward interpretation, (8.9.30) must be carefully interpreted. By design, the error signal is nonstationary since it is the equalizer's purpose to drive it to zero. It is also difficult to determine *a priori* the expectation inside the integral. However, this expectation can be approximated by a time average over an interval that must be long enough to approximate expectation (i.e., an average), yet short enough not to overly stretch the local stationarity assumption. This generally can be done because the control signal [the integral of $\epsilon(t)$] is relatively slowly varying. Thus, in hardware as in simulation one can approximate (8.9.30) by removing the expectation and simply integrating over an appropriate time interval.

The main point to recognize is that for the gradient algorithm to converge, an exact value of the integral in (8.9.30) is not necessary. What we need to make sure of is that the direction (sign) of the gradient at any stage is correct since Δ actually controls the amount of movement. Hence, instead of integrating for a fixed period, say, we might use a sequential algorithm, whereby C_n is incremented by a certain amount only when the value of the integral exceeds a certain threshold. In fact, there are many versions of adaptation algorithms, or at least many versions of the implementation of such algorithms.^(11,87,88) These versions typically have a particular objective such as higher speed of convergence, better performance (smaller MSE), better stability, or ease of hardware implementation. Depending upon data rate, we may use analog circuitry or digital approximations. We might implement the equalizer at baseband or at intermediate frequency (IF). If it is our objective to investigate by simulation the properties of a particular structure, algorithm, and implementation, then it is these specifics that we must mimic in software. If we are only interested in a generic equalizer, then we are free to choose the parameters in a reasonable way.

8.9.6.2. Covariance Matrix Inversion Method

In some situations, the channel or environmental conditions may vary quite slowly with respect to the information rate. In Chapter 2 we termed this condition “quasistatic.” This condition is also assumed to hold for the microwave radio channel in Case Study I of Chapter 12. In this situation an equalizer can be assumed to adapt rapidly enough that the tap-gain values reached at some point in time remain relatively fixed for a large number of symbols. We now obtain the values of these optimum tap gains as a function of the channel particulars. The ultimate calculation involves inversion of a matrix that we call the channel covariance matrix.

We consider a slightly generalized development of (8.9.23), where we assume $X(t)$, $Y(t)$, and $\{C_k\}$ can be complex. The statistical MSE criterion is now expressed as minimizing $E[\epsilon^2(t)] = E|X(t) - Z(t)|^2$. It can be shown that $\epsilon^2(t)$ is minimized when

$$E\{[X(t) - \sum_k C_k Y(t - kT')]Y^*(t - nT')\} = 0$$

for $-M \leq k, n \leq M$

(8.9.31)

We assume that the channel output $Y(t)$ can be written as

$$Y(t) = \sum_i S_i p(t - iT) + N(t)$$
(8.9.32)

where $p(t)$ is the characteristic single-pulse response of the channel, and $N(t)$ is zero-mean additive Gaussian noise with autocovariance $\rho(\tau)$ or autocorrelation function $R(\tau) = \sigma_n^2 \rho(\tau)$. We also assume that the signal sequence $\{S_i\}$ is stationary and i.i.d., so that $E|S_i|^2 = E|S|^2 = \sigma_s^2$. We assume for specificity that we deal with the zeroth symbol; that is, the desired signal $X(t_0) = S_0 s(t_0) = S_0$, i.e., $s(t)$ is normalized to unity at the sampling time t_0 . Substituting (8.9.32) into (8.9.31), with the preceding stipulations, yields

$$\sum_{k=-M}^M C_k \{\sigma_s^2 \gamma(n, k) + \sigma_n^2 \rho(n, k)\} = \sigma_s^2 p^*(t_0 - nT') \quad (8.9.33)$$

where

$$\gamma(n, k) = \sum_{j=-\infty}^{\infty} p[(t_0 + jT - kT')p^*(t_0 + jT - nT')]$$

for $-M \leq k, n \leq M$

(8.9.34a)

and

$$\sigma^2 \rho(n, k) = E[N(t_0 - kT')N^*(t_0 - nT')] \quad (8.9.34b)$$

In the special case when values of $N(t)$ spaced by T' are uncorrelated, $\rho(n, k) = \delta(n - k)$, where $\delta(m) = 1$ for $m = 0$ and zero otherwise.

These equations can be reset in matrix form:

$$\mathbf{\Gamma} \mathbf{C}_{\text{opt}} = \mathbf{r}^* \quad (8.9.35)$$

where $\mathbf{\Gamma}$ is a $(2M + 1) \times (2M + 1)$ matrix whose i, j element is $\gamma(i, j) + (\sigma_n/\sigma_s)^2 \rho(i, j)$, $\mathbf{r}$ is a $(2M + 1)$ column vector whose j th element is $p(t_0 - jT')$, and $\mathbf{C}_{\text{opt}}$ is the $(2M + 1)$ column vector of optimum tap gains, given by inverting (8.9.36),

$$\mathbf{C}_{\text{opt}} = \mathbf{\Gamma}^{-1} \mathbf{r}^* \quad (8.9.36)$$

In order to solve (8.9.36), we need to know the channel response $p(t)$, which can be obtained directly from the simulation itself. If the pulse duration can be truncated to fairly short duration with little error (which is often the case), then (8.9.36) can be computed quickly. The matrix inversion is equivalent to the training of an equalizer. The method outlined here is the basis for computing the equalizer tap gains in the “quasistatic” scenario of Case Study I in Chapter 12, where we assume a sequence of random multipath channels, each of which can be considered almost time-invariant with respect to a large number of symbols.

Since the tap gains that minimize the MSE are given by (8.9.36), the actual value of the minimum MSE (MMSE) is obtained by substituting (8.9.36) into the expression for the MSE, $E|X(t) - Z(t)|^2$. It can be shown (Problem 8.21) that the MMSE is given by

$$\text{MMSE} = \sigma_s^2 [1 - \mathbf{r}^T \mathbf{\Gamma}^{-1} \mathbf{r}^*] \quad (8.9.37)$$

where superscript T denotes transpose. We should note that the MMSE, like the tap gains themselves, are implicitly dependent upon symbol synchronization, for the matrix γ depends upon the sampling epoch t_0 . The sensitivity of the MMSE to symbol timing depends in general on the tap spacing T' .

8.9.6.3. Simulation Considerations

As with other subsystems, there can be different ways to take account of equalization in the simulation context. There are two major options, outlined in the previous two subsections, which we may label dynamic and static (or quasistatic). In the first, we simply let the equalizer “be itself.” In the second, we pre-converge the tap gains and treat the equalizer as an LTI filter. The choice of approach depends in part on the nature of the system, the objective of the simulation, and the need to effect complexity reduction in order to reduce run time. In terms of objective, we may be interested in obtaining the (equalized) BER performance of a system under a specified set of channel conditions, given an equalizer design; we may wish to investigate the convergence properties of alternative algorithms; or we may be interested in “tweaking” the parameters of a nominal design to study the performance/hardware tradeoffs.

For the sake of run-time economy, our general guideline is always to strive toward reduction of complexity (of the simulation). Hence, if our objective is to investigate properties of the equalizer per se, we need not burden the simulation with a full system model. A relatively simple system model and test signal should suffice, at least for purposes of revealing sensitivity to design choices.

Evaluation of system (BER) performance with an equalizer will generally require an *as-is* emulation of the adaptation algorithm. This poses no difficulties in principle: one merely follows the “recipe” prescribed by the algorithm. However, in the simulation environment, it also becomes possible to investigate the effect of imperfections in the implementation of the equalizer. According to the modeling paradigm outlined in the Introduction, a “realistic” model can be developed by starting with an idealized model and then hypothesizing devia-

tions from it in ways that appear to be sensible and physically realizable. Also as mentioned before, the discrete or logical elements of the algorithm can be assumed faultless, whereas the continuous or analog portions can be modeled as imperfect. While most implementations are in fact digital in nature, not all aspects need be. For example, the tap spacing T' is usually considered to be perfectly implemented. In reality, this is not possible, and in addition, the spacing from tap to tap will not be exactly the same. Thus, the tap spacing might be represented as $T' + \Delta T'$ where $\Delta T'$ could be modeled as a random quantity, consistent with the manufacturing process. For low to moderate data rates this refinement may not be necessary, but for high bit rates it may well be.

As mentioned, it is straightforward to imitate an adaptive equalizer's operation given the specifics of its design. It is proper to attempt this emulation if there is some question as to the speed of adaptation relative to the rate of change of channel conditions. There is, of course, some computational penalty in doing so, which is inherent in the approach. There is also a bit of "startup" inefficiency because the tap gains are usually initialized with unity at the center tap and zero elsewhere; hence the initial training period should not be used for performance evaluation. If, however, the fading rate is very slow compared to the convergence time and the data rate, we can use the quasistatic methodology outlined in the preceding section, where we can assume the tap gains are converged appropriately to the assumed channel conditions, and the tap-gain values are obtained through the channel covariance matrix inversion. This avoids the continual update computations that otherwise would have to be made.

8.9.7. Filters Specified by Simple Functions in the Frequency Domain

There is often the need in simulation (as in analysis) to use certain idealized forms of filtering as an approximation to reality. Such a situation may arise, for example, in the early stages of the system design when the actual transfer functions have not been finalized, or when the actual filters have not been built (hence cannot be measured). In such a case, as is discussed in Chapter 2, it is customary to describe filters by *specifications*. These specifications might take the form of an amplitude mask, or might comprise bounds on a small set of parameters such as gain slope, amplitude and phase ripple, parabolic phase, cubic phase, etc. It is convenient in such instances to synthesize a transfer function directly from these specifications. We shall refer to the individual elements of specifications as *functional filters* because they are described by analytical functions, usually of very simple form. These filters are purely mathematical constructs that are typically not realizable. Functional filters can also be used as approximations to physical effects. For example, in Chapter 9 we show that the effect of the "well-behaved" ionosphere on the phase of a signal can be approximated by a parabolic phase "filter." Another use of functional filters is as models of imperfections in otherwise specified designs.

Some of the more useful functional filters are given below. These filters can be interpreted either as lowpass (when the appropriate conditions are met) or as complex lowpass equivalents. In some instances, for example, when an amplitude mask is given as a specification, even though an analytical description can be given, it is often simpler merely to describe such filters in tabular form. This will be taken up in Section 8.9.8.

Amplitude Ripple Filter: One can model such a filter as

$$H(f) = M + R \cos(2\pi\nu_a f + \theta) \quad (8.9.38)$$

where $\mathbf{v}_a$ is the “frequency” of the amplitude ripple and θ determines the symmetry of the ripple with respect to $f = 0$. The ripple is often specified as peak-to-peak (p-p) ripple, in decibels, and is related to the parameters M and R in the following way:

$$\text{Peak-to-peak ripple (dB)} = 20 \log[(M + R)/(M - R)] \quad (8.9.39)$$

Amplitude Tilt Filter: This filter models gain slope across a passband. For generality, the model permits independent specification of the slope on either side of zero frequency, since it is sometimes observed that there is a droop on both sides of the center at the passband. Because tilt is always described in decibels, we define this filter directly in decibel units. That is, we have

$$G(f) = 20 \log H(f) = \begin{cases} m_1 f, & f \geq 0 \\ m_2 f, & f < 0 \end{cases} \quad (8.9.40)$$

Of course, the simulation must use $H(f)$ and perform the necessary antilog operation.

Parabolic Gain Filter: For the purpose of modeling gain droop, an alternative to the tilt filter above which has a gentler rolloff is a “parabolic gain” filter. Such a filter has the form

$$G(f) = 20 \log H(f) = \alpha f^2 \quad (8.9.41)$$

where α has units of dB/Hz^2 , and of course α is the droop in decibels at $f = 1$.

Phase Ripple Filter: Phase ripple is a good model for imperfections owing to reflections. A reasonable and simple model is

$$H(f) = \exp[jP \cos(2\pi \mathbf{v}_p f + \delta)] \quad (8.9.42)$$

where $\mathbf{v}_p$ is the “frequency” of the ripple, δ establishes the symmetry of the ripple, and P is the peak ripple in radians.

Parabolic Phase (Linear Time-Delay) Filter: This description is self-explanatory. The filter is given by

$$H(f) = \exp(j\theta_p \omega^2), \quad \omega = 2\pi f \quad (8.9.43)$$

where θ_p is the parabolic phase (radians) at a reference frequency of $\omega = 1 \text{ rad/s}$.

Cubic Phase (Parabolic Time-Delay) Filter: This filter is given by

$$H(f) = \exp(j\theta_c \omega^3), \quad \omega = 2\pi f \quad (8.9.44)$$

where θ_c is the cubic phase (radians) at a reference frequency of $\omega = 1 \text{ rad/s}$.

In simulation one often uses normalized frequency, typically normalized to the symbol rate in the digital case. Thus, $f = 1$ would correspond to the normalization parameter. The above filters have the same form, normalized or not, but the value of the coefficients does reflect the normalization. Functional filters are normally implemented as FIR filters. In order to avoid aliasing, it is understood that these filters will be used in conjunction with a bandlimiting filter that defines a bandwidth appropriate for the simulation sampling rate.

8.9.8. Tabular Filters for Masks and Measurements

Sometimes the amplitude or phase characteristics of a filter or system are specified by a “mask,” meaning that the amplitude or phase is bounded from above and below. These bounds form constraints on the transfer function, but do not actually specify it. In order to do a simulation run, it is necessary to have an explicit $H(f)$. One can then choose more or less arbitrarily a transfer function (or several such) within the mask. This procedure is naturally graphical and it is more sensible to describe one’s choice in tabular form than in analytic form. That is, having traced a characteristic, one then picks points on the curve and constructs a table $H(k\Delta f)$, $k = 0, \pm 1, \pm 2, \dots, \pm N$. When a transfer function is given in the frequency domain, one has the option of generating a best recursive approximation or treating it as an FIR filter, as discussed in Chapter 4. In the latter instance FFT processing is usually, though not always, done, depending upon the length of the response. Figure 8.46 shows the relationship between $H(k\Delta f)$ and other relevant parameters.

Measurements on filters are invariably performed in the frequency domain, and reported either as (seemingly) continuous curves or as tables. Hence, the preceding discussion applies to this case also. Generally, the measurements will not have been done at a frequency separation that corresponds to the resolution of the FFT algorithm, and in that case interpolation must first be done. This is frequently implemented by a cubic spline interpolation and resampling.

Tabular filters may also be preferable in some instances even if the transfer function can be analytically specified. For example, if the analytical description is complicated, it may be computationally more efficient to calculate the transfer function once (rather than repetitively) and store equally spaced values as a tabular filter. This is the case for some of the communication channels such as the optical fiber, which is represented by a transfer function that requires time-consuming numerical integration. This would generally be the approach used if

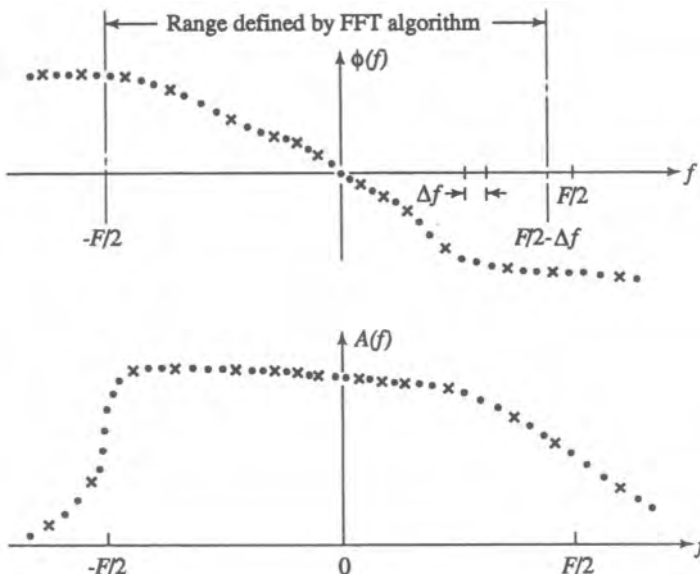


Figure 8.46. Illustration of measured amplitude and phase points (x) and interpolated points (•). $\Delta f = 1/T$, where T is the length of the input data block.

one were to implement filtering via an FFT implementation, as described in Chapter 4. It is recommended that tabular filters be implemented with scaling options that permit the user to change the variable scale or the frequency scale. This provides the flexibility to experiment with the quality of the filters.

8.10. Multiplexing/Multiple Access

The terms “multiplexing” and “multiple access” both refer essentially to the same basic notion, namely, the sharing of a communication resource among a multiplicity of users or signals. While there are shades of difference between the two, these differences are not essential for present purposes. Multiple access does imply the existence of a network control scheme. By and large, however, the network level is transparent to the link-level considerations, which are our focus here. Networking considerations form an entire field in themselves; the reader is referred to Refs. 90–92, among many others.

Multiple-access techniques themselves encompass many forms and related system considerations. It is again beyond our scope to consider these in detail (see, e.g., Refs. 17 and 93–97). Our main objective here is to outline the simulation-related issues in dealing with the various possibilities. We will thus briefly address these issues as they relate to the following types of multiple access:

- Space-division multiple access (SDMA)
- Polarization-division multiple access (PDMA)
- Frequency-division multiple access (FDMA)
- Time-division multiple access (TDMA)
- Code-division multiple access (CDMA)

The first two are not standardly thought of as multiple-access categories, but they play an important role in certain situations, especially space applications. The simulation implications are straightforward and will be considered shortly.

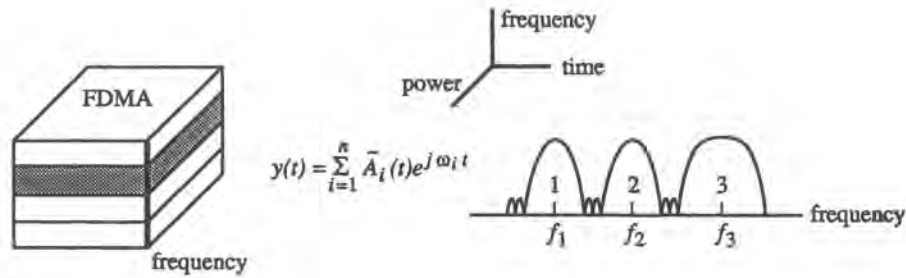
The last three on the list are the standard multiple-access approaches. A multiple-access technique has an operative domain which permits the separation of the multiple signals from one another. Thus, in FDMA the domain is frequency, in TDMA it is time, and in CDMA it is “code.” The last is not particularly suggestive. It is the structure of the code that permits this separation, and in particular the zero or low correlation among the user codes. We shall prefer, then, to attribute the signal separability to a “correlation” domain. Figure 8.47 provides an illustration of the nature of these multiple-access methods.

8.10.1. Issues in the Simulation of Multiple-Access Methods

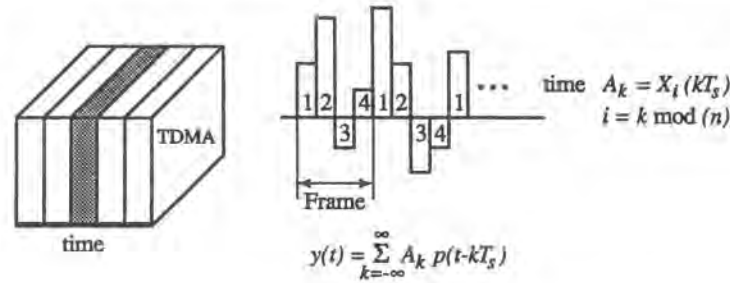
We now review the simulation implications of the different multiple-access methods.

8.10.1.1. SDMA and PDMA

We discuss these methods together because the essence of the issues is the same for both, namely accounting for the presence of other interfering signals, which are essentially cochannel. The “space” in SDMA refers to physical space. In space applications, signals from different satellites are separable because they arrive from different points in space, which



Demultiplexing: filter, frequency translation.



Demultiplexing: switch/commutator/distributor.

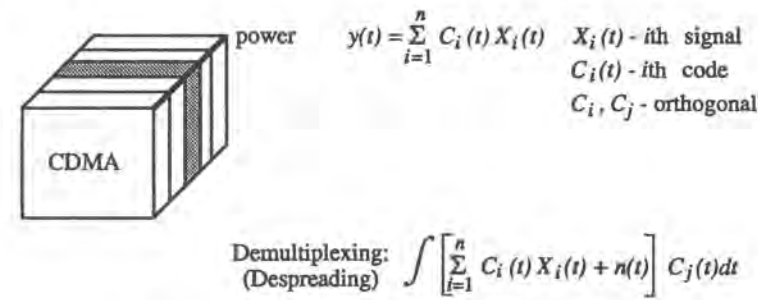


Figure 8.47. Capsule description of multiple-access schemes.

really means different angles of arrival. The antenna patterns discriminate between wanted and unwanted signals. The latter will presumably be intercepted by the lower gain antenna sidelobes, hence produce lower power levels in the receiver. Polarization division (or diversity) has just two dimensions. But the “leakage” of energy from one polarization to another will appear in the other polarization channel as interference.[†]

[†]For both SDMA and PDMA, the bulk of the interference will be cochannel for most operational scenarios. There can be adjacent-channel interference as well, but this case amounts to the FDMA situation.

To simulate a scenario using SDMA or PDMA, we simply need to generate the interfering signals at their proper power levels and frequencies and add them to the desired signal.

8.10.1.2. FDMA

As is well known, and suggested by Figure 8.47, a multiplicity of carriers share some band of frequencies, the spectrum for each carrier being nominally nonoverlapping with that of any other. Of course, in actuality there *is* some overlap, which is part of the reason why we also have to simulate some of the channels other than the particular one in which we may be interested. In a linear system, it is safe to assume that only the adjacent channels have to be accounted for. In a nonlinear channel, this is no longer the case. Intermodulation products among the various carriers can overlap onto any of them, hence in such a system it would generally be necessary to simulate the presence of all carriers. (See also the related discussion in Chapter 5.)

For illustrative purposes we shall now consider just three carriers, the spectra of which are shown in Figure 8.48. We shall suppose that the center channel is the one of interest. Figure 8.48b shows the spectrum of the complex envelope sampled at frequency F_s . It is clear that to avoid aliasing, F_s must be greater than the bandwidth of all the signals, not just the one of interest. Therefore, if we were interested in just one “typical” signal, we would nevertheless be forced to sample at a higher rate. There are actually different ways to assemble the three signals, one or the other of which may be better, depending on circumstance.

Let the i th carrier be written as

$$S_i(t) = A_i(t) \cos[2\pi f_i t + \phi_i(t) + \theta_i] \quad (8.10.1)$$

Then, the complex envelope of their sum is

$$\tilde{S}(t) = \sum_{i=0}^N A_i(t) \exp\{j[2\pi v_i t + \phi_i(t) + \theta_i]\} \quad (8.10.2)$$

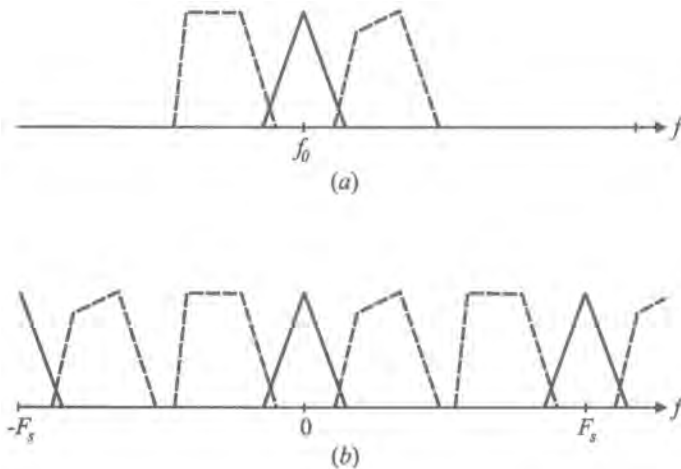


Figure 8.48. Illustration of sampling FDM signals, (a) Spectrum of three signals; (b) spectrum of sampled complex envelope.

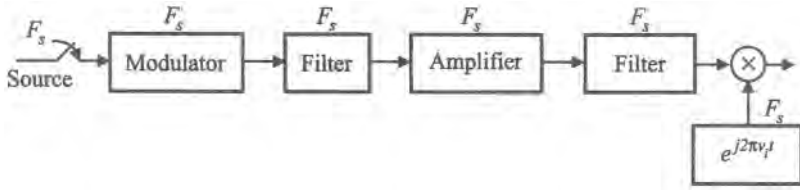


Figure 8.49. One approach to sampling and combining a set of FDM signals.

where $v_i = f_i - f_0$ and f_0 is the center of the band considered. [Equation (8.10.2) applies to any N , though we are using $N = 2$ in our example.] If we know the $S_i(t)$ in analytical form, then we simply provide to the simulation the sampled sequence $\tilde{S}(kT_s)$ for $T_s^{-1} = F_s$ equal to or greater than the combined bandwidth of the signals.

More often, we will have the sampled sequence of the individual signals, $\tilde{S}_i(kT_s)$. Noting that the sampled summands in (8.10.2) can be written as the product of two functions

$$u_i(kT_s) = \tilde{S}_i(kT_s) \exp(j2\pi v_i kT_s)$$

we can visualize a block diagram for generating $u_i(kT_s)$ as shown in Figure 8.49. For $i = 0, 1, \dots, N$, then, the output of the several multipliers would be summed. Note that the individual sources are sampled at the high rate F_s . This imposes a burden in the sense that this rate is generally much higher than needed to represent any one of the individual sources.

An alternative approach is to sample individual sources at a rate, say f_{si} , commensurate with their bandwidth, and then upsample before combining. This approach is illustrated in Figure 8.50. Of course, part of the advantage of sampling at a lower rate is offset by the need to interpolate. Whether this approach or the preceding one is preferred will be case-specific.

At the receiver, it will be generally advantageous to deal with the sampling rate f_{si} for the i th signal because downsampling requires little computational effort. To recover the desired signal from the high-sample-rate sequence, we first extract that signal with the appropriate filter (see Figure 8.51), and downsample the result to the correct sample rate.

8.10.1.3. TDMA

The situation for TDMA is illustrated in the simplified diagram of Figure 8.47. There are many possibilities and ramifications to the problem, depending upon whether the various sources are colocated or distributed, stationary or mobile, and whether the common resource also is stationary or not. A detailed look at the synchronization, timing, and buffering aspects is beyond our intended scope.

If we assume that the user symbol streams are perfectly synchronous and their arrivals at the common resource perfectly timed, then from the point of view of transmission performance through that common resource, the fact of TDMA, per se has no bearing. In other words, to evaluate the BER performance at any receiver, for example, we could treat the situation as if there were only a single user. The simulation of the individual users' transmitting chains requires no new considerations. Complexities arise only if we wish to look at a lower level of detail into the imperfections of the synchronization, and timing aspects in general. These aspects, however, are not easily accessible with a system-level simulation. The reader is referred to Ref. 17, among others, for details.

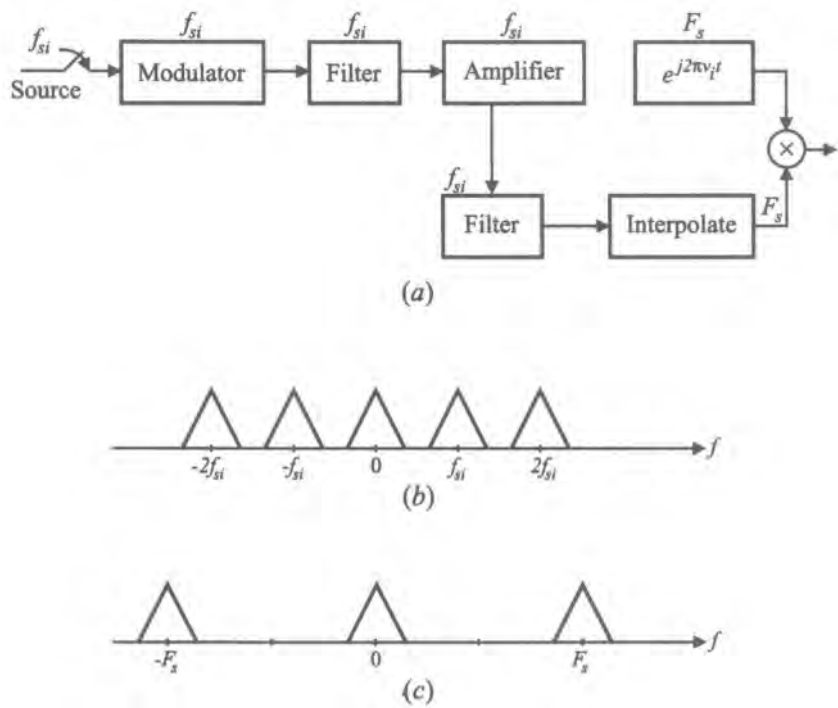


Figure 8.50. A second approach to sampling and combining a set of FDM signals.

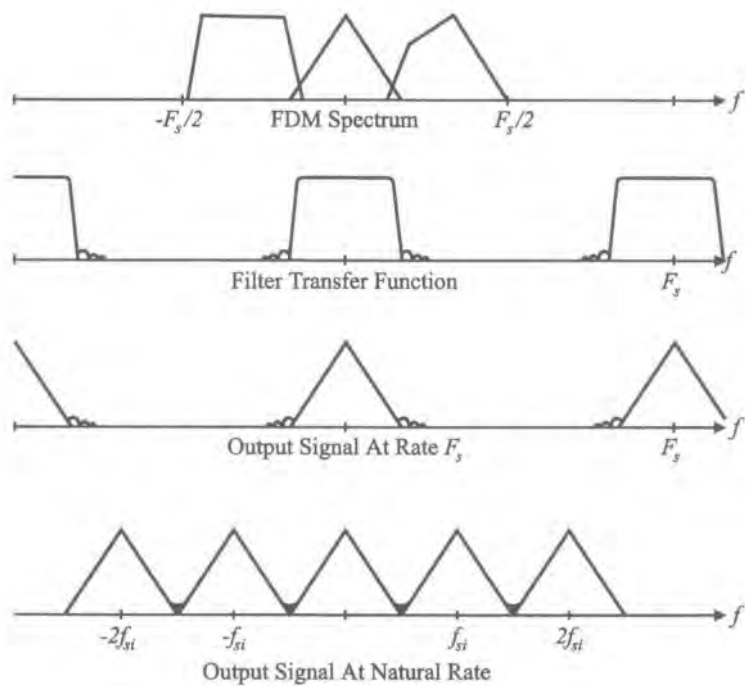


Figure 8.51. Recovering one signal from an FDM set.

8.10.1.4. CDMA (Spread Spectrum Techniques)

Here again, we have a wide-ranging discipline that is a field in itself. The reader is referred to the cited references, among many others on the subject. Our objective, stated before, is to highlight the simulation implications that might be particular to CDMA. We note that “code division” implies a method of multiple access, while the actual means of implementation are referred to as “spread spectrum” techniques. The latter term is more suggestive of the labels “frequency spread” and “frequency despread” that appear in two of the blocks of Figure 8.1. A popular technique is called *direct sequence* (DS) spread spectrum, whose implementation is illustrated in the block diagram of Figure 8.52, an abbreviated version of Figure 8.1. If the modulator output is given by

$$S_1(t) = \text{Re}\{\tilde{D}(t)e^{j\omega_1 t + \theta_1}\}$$

and the code generator output by

$$S_2(t) = \text{Re}\{\tilde{C}(t)e^{j\omega_2 t + \theta_2}\}$$

then the transmitted signal is

$$S(t) = \text{Re}\{\tilde{D}(t)\tilde{C}(t)e^{j\omega_c t + \theta}\} \quad (8.10.3a)$$

$$= \text{Re}\{\tilde{S}(t)e^{j\omega_c t + \theta}\} \quad (8.10.3b)$$

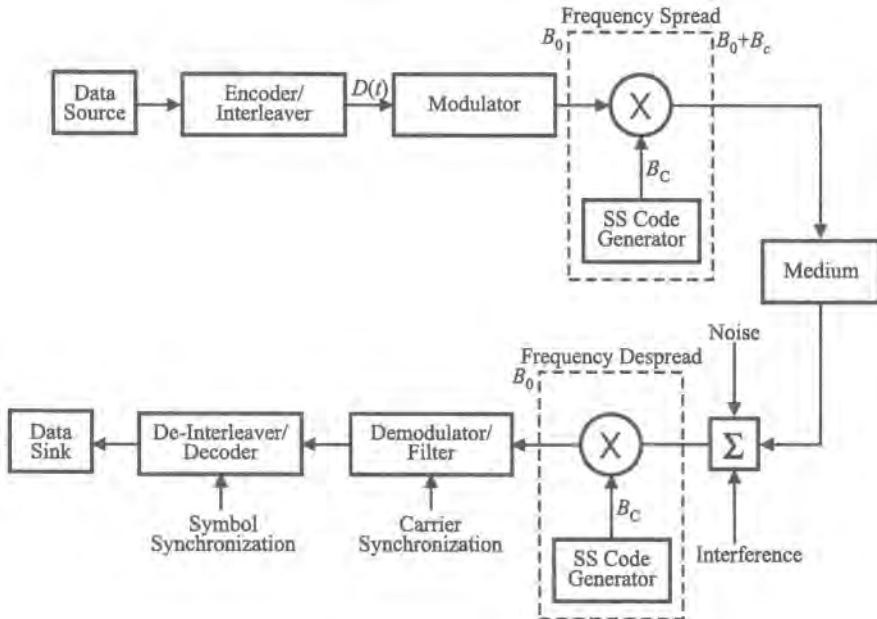


Figure 8.52. A typical direct-sequence spread-spectrum system.

where $\omega_c = \omega_1 + \omega_2$ and $\theta_1 + \theta_2 = \theta$. The multiplication of the data by the code, $\tilde{D}(t)\tilde{C}(t)$, spreads the spectrum (bandwidth) from that of the data, say B_0 , to the sum of the two $B_0 + B_c$, the latter being the bandwidth of the code, with (typically) $B_c \gg B_0$. Illustrative sketches of the data and code waveforms are shown in Figure 8.53 assuming they are real.

Suppose we have a set of users, $i = 0, 1, \dots, N$, and let $i = 0$, say, label the user of interest. The complex envelope of the received waveform at the latter's receiver is

$$\tilde{R}(t) = \tilde{S}_0(t) + \sum_{i=1}^N \alpha_i \tilde{S}_i(t) + \tilde{N}(t) \quad (8.10.4)$$

where the α_i are the relative magnitudes of the various signals. The first step in the recovery of $\tilde{D}_0(t)$ is to correlate (multiply) the received waveform with a replica of that user's "signature," the code $\tilde{C}_0(t)$. The success of the operation depends in the first place on the cross-correlation properties among the various codes,

$$C_{ij}(\tau) = \int_{t \in P} \tilde{C}_i(t) \tilde{C}_j^*(t - \tau) dt \quad (8.10.5)$$

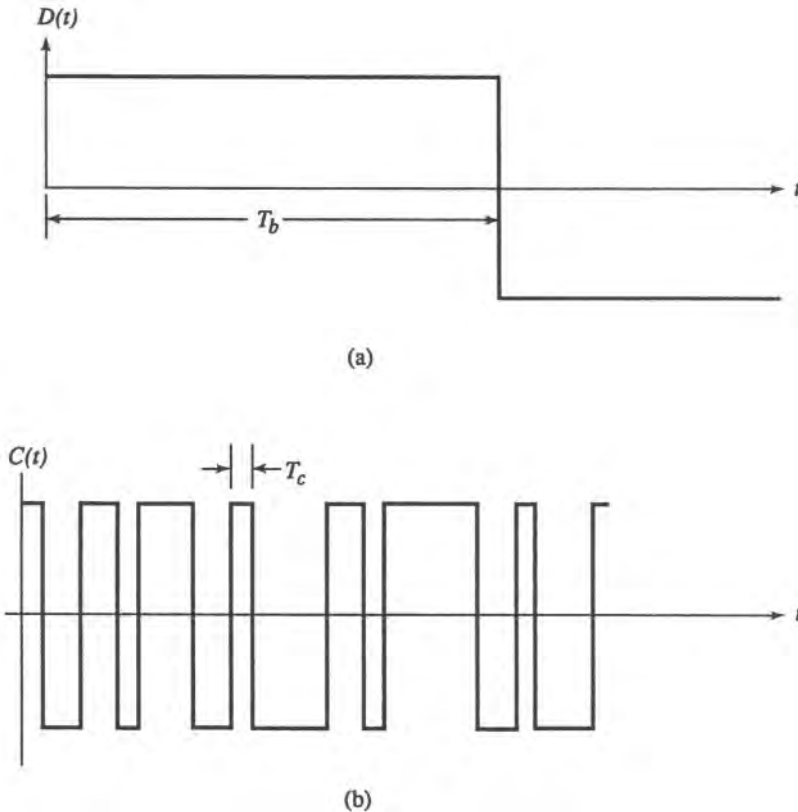


Figure 8.53. Illustration of data and code waveforms. (a) data (or channel-encoded data) stream; (b) pseudorandom code sequence.

where P is the (common) period of the codes and τ is some relative delay. Extraction of any one of the signals from the sum (10.8.4) requires that $C_{ij}(\tau)$ be very small for $i \neq j$. The operation to recover $\tilde{S}_0(t)$, say, requires multiplication at the receiver by the (synchronized) transmitter code (Figure 8.52) followed by filtering:

$$\tilde{S}'_0(t) = \left\{ \tilde{R}(t) \times \tilde{C}_0(t) \right\} * h_d(t)$$

where the prime indicates estimate and $h_d(t)$ is a data filter. Further processing follows to recover the data.

The preceding sketch indicates that the bandwidth to be processed between the transmitter and receiver code generators is larger, usually much larger, than the bandwidth of the information. We have already encountered this situation. The simulation between those points must run at the corresponding sampling rate in order to avoid aliasing. One could of course run the entire simulation at that rate. But, if B_c/B_0 were on the order of 10^3 , say, a not untypical number, this means in essence that the simulation would run 1000 times longer than it would in the absence of spreading. Because of this extremely high oversampling, multirate techniques can be used to advantage, even after accounting for the interpolation overhead. That is, the transmitter operations before the code generator and the receiver operations after the code generator can run at a sampling rate commensurate with the information bandwidth. These are the main considerations to be dealt with that specifically relate to simulation methodology. In any given CDMA system there will be, of course, some situation-specific modeling issues. Case Study IV in Chapter 12 presents a simulation study of a particular type of CDMA mobile cellular system operating in a fading environment.

8.11. Radiofrequency and Optical Sources

An ideal carrier of information at radiofrequency or at optical frequency is a pure sinusoid. Since a simulation model for a modulated carrier (the lowpass equivalent) is transparent to the actual carrier frequency, ideal (or nearly so) frequency sources need not be represented at all in the simulation. In actuality, of course, physical sources exhibit impairments or characteristic behavior that depend on the type of source.

8.11.1. Radiofrequency Sources

Practical radiofrequency sources have a spectrum of nonzero width, which can be attributed primarily to phase (rather than amplitude) modulation of the carrier by oscillator instability, frequently referred to as *phase noise*. Thus, a modulated carrier can be modeled as

$$S(t) = A(t) \cos[\omega_c t + \phi(t) + \delta(t)] \quad (8.11.1)$$

where $A(t)$ and $\phi(t)$ represent information and $\delta(t)$ is the phase noise. The lowpass equivalent is simply

$$\tilde{S}(t) = A(t) \exp[j\phi(t) + j\delta(t)] \quad (8.11.2)$$

This would be straightforward to implement if we had a workable model for $\delta(t)$. In fact, the physical process $\delta(t)$ is quite difficult to describe (see also Case Study II, Chapter 12). As an alternative, one can represent $\delta(t)$ by a process that is easy to generate, such as a Gaussian process, whose average power and spectral density are the same (to the extent possible) as that of the actual process.

The effect of phase noise is usually of most concern in coherent digital systems. There, however, it is not $\delta(t)$ itself that is of direct interest but rather the “net” phase noise that remains after demodulation, because part of $\delta(t)$ is “tracked out” by the receiver carrier tracking structure. Thus, for simulation purposes it is often more efficient to model directly this net phase noise. These considerations will be discussed again in the next section and in Chapter 12.

8.11.2. Optical Sources

An optical source is a laser, of which there are various types. We consider here only semiconductor lasers operating under certain conditions. The dynamic characteristics of semiconductor lasers have been described by numerous authors to various levels of complexity. An early modeling technique made use of measurements of the complex envelope as a function of a test modulating pattern; the characteristic (pattern-dependent) waveforms could then be recalled from lookup tables as the real modulating sequence evolved. However, by making some reasonable assumptions, we can develop a more fundamental description. In particular, if we assume single longitudinal and lateral mode of operation, and further assume that the photon density and the carrier (electron) density are uniform over the cavity length, then we can obtain a simple, yet accurate and fairly general, model that is valid for most semiconductor lasers. Under the preceding assumptions, then, the laser model is described by the following pair of rate equations⁽⁹⁸⁾:

$$\frac{dN(t)}{dt} = \frac{I(t)}{eV} - \frac{N(t)}{\tau_n} - g_0(N(t) - N_0) \frac{S(t)}{1 + \epsilon S(t)} \quad (8.11.3a)$$

$$\frac{dS(t)}{dt} = \Gamma g_0(N(t) - N_0) \frac{S(t)}{1 + \epsilon S(t)} - \frac{S(t)}{\tau_p} + \Gamma \beta \frac{N(t)}{\tau_n} \quad (8.11.3b)$$

where $N(t)$ and $S(t)$ are the electron and photon densities, respectively, g_0 is the differential gain, N_0 is the carrier density for transparency, τ_n and τ_p are the spontaneous electron lifetime and photon lifetime, respectively, Γ is the optical confinement factor, ϵ is a parameter characterizing the nonlinear gain, β is the fraction of spontaneous emission coupled into the lasing mode, $I(t)$ is the current through the active layer, e is the electron charge, and V is the volume of the active layer. We point out that in (8.11.3), $N(t)$, $S(t)$, and $I(t)$ are functions of time, while the other quantities are constants for a given laser. The current $I(t)$ carries the information to be modulated onto the source, which is usually superimposed on a constant (bias) current.

The derivation of the above rate equations is lengthy and can be found, for example, in Ref. 98. However, one can rationalize these equations very briefly by the following physical reasoning. The first equation simply explains phenomenologically all the mechanisms by which the carriers are generated and lost inside the active region. The first term on the right accounts for the rate of new carriers entering into the active region; the second term governs

the carrier loss due to spontaneous emission; and the last term governs the loss due to stimulated emission.

The derivation for the second equation starts with Maxwell's equations and then phenomenologically adds on the right side all terms responsible for photon generation, such as stimulated emission (first term), photon loss through the cavity (second term), and photon generation by spontaneous emission (third term).

The above two equations are simultaneously solved numerically for an arbitrary current waveform $I(t)$ to generate the following complex electric field at the output of the laser, i.e., the complex envelope:

$$\tilde{E}_{\text{out}}(t) = k[S(t)]^{1/2} \exp[j\phi(t)] = [P(t)]^{1/2} \exp[j\phi(t)] \quad (8.11.4)$$

Here $P(t) = k^2 S(t)$, $k^2 = V\eta h\nu/2\Gamma\tau_p$, η is the quantum efficiency of the intrinsic laser, and h is Planck's constant.

The phase term $\phi(t)$ is given by

$$\phi(t) = 2\pi \int_0^t \Delta\nu(t') dt' \quad (8.11.5)$$

where

$$\Delta\nu(t) = \frac{\alpha\Gamma g_0}{4\pi} [N(t) - \bar{N}] \quad (8.11.6)$$

α is the linewidth enhancement factor and $\bar{N}$ is the average value of the carrier density.

The instantaneous frequency $\Delta\nu(t)$ is the departure from the nominal optical frequency. In the absence of modulation, the optical frequency is, ideally, constant. In reality, there is some nonzero spectral width which can be identified as phase noise, $\delta(t)$, so that (8.9.4) can be more fully expressed as

$$\tilde{E}_{\text{out}}(t) = [P(t)]^{1/2} \exp[j\{\phi(t) + \delta(t)\}] \quad (8.11.7)$$

A symbolic block diagram of the model (without phase noise) is shown in Figure 8.54. The crux of the model is, of course, the solution of the differential equations (8.11.3). This can be accomplished using the techniques of Section 5.4. An explicit simulation diagram is shown in

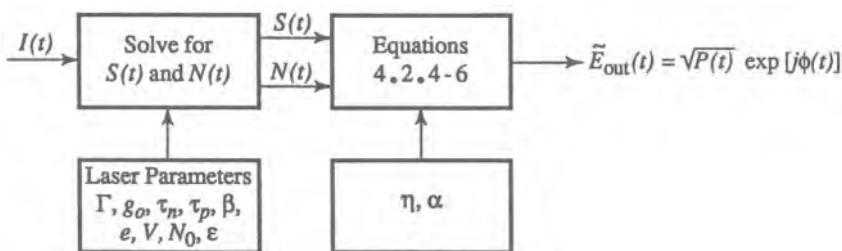


Figure 8.54. Symbolic block diagram for simulation model of laser source.

8.12. Synchronization

In communication systems that are termed “coherent,” the demodulator (receiver) needs a local carrier reference whose phase is a close approximation to the phase of the incoming carrier (ideally, it would be identical). Furthermore, if the information transmitted is digital in nature, then the receiver also needs to generate a symbol timing (clock) signal to control the sampling times at which the matched filter output is to be sampled. The process of generating carrier and timing references at the receiver is referred to generically as synchronization, or also as carrier and (symbol) timing recovery. Synchronization is one of the major functions established at the receiver and accounts for much of the demodulator complexity. Hence, this material should be considered intimately connected with that in Section 8.8.

There are several levels of synchronization in large communication systems. Besides carrier and symbol timing, which is the topic of this section, other synchronization problems include frame, word, code, and packet synchronization. A feature that distinguishes these problems from carrier and symbol synchronization is that they are usually achieved through the use of special sequences of symbols that are normally multiplexed with the information sequence.

Carrier and symbol synchronization can be accomplished with or without the use of auxiliary signals. It is generally preferred to achieve synchronization without having to multiplex special synchronization signals (such as an unmodulated carrier or an alternating bit pattern) which would use up part of the available channel capacity. Perhaps most digital systems use “self-synchronization” schemes in which the carrier and symbol timing references are derived from the received waveform, which does not contain any ancillary information that is transmitted solely to aid in synchronization. However, in some mobile applications, the simplest system solution is to transmit synchronization “pilots” in a small part of the band that can be accessed by all users. In any event, much of the following discussion applies to either case.

Synchronization techniques vary widely depending on the modulation schemes employed, the operational environment, and the degree of accuracy desired. The performance of timing and carrier recovery is typically measured in terms of biases and rms jitters. The definitions of bias and jitter in the recovered symbol timing waveform are shown in Figure 8.56. Similar measures for carrier recovery schemes can be computed based on the zero crossings of the recovered carrier.

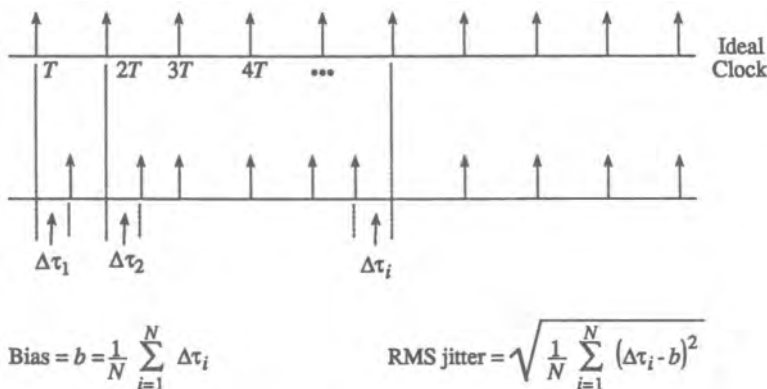


Figure 8.56. Bias and jitter in a timing waveform.

In general it is difficult to compute the amount of jitter introduced by synchronization schemes since almost all such schemes involve nonlinear processing of the signal corrupted not only by Gaussian noise, but also by oscillator noise, as well as by transmission distortion. Some well-known analytical results exist under certain assumptions that usually simplify actual situations, and these results are certainly useful as approximations. But because of the analytical difficulties that arise in the general case, one often needs to resort to simulation for a more definitive solution.

It is important, however, to distinguish between two situations. One is where the object of study is the synchronization system itself, in our parlance a subsystem. For example, we might be interested in the acquisition time of a carrier recovery circuit, which is essentially dictated by the design of that circuit and unrelated to the nature of the surrounding system. In this case, if the synchronization system is not amenable to formula-based analysis, then simulation is the proper and perhaps only tool for investigating its properties. The second situation is where the object of study is a larger communication system, two of whose *subsystems* are the carrier and timing synchronization functions. The system performance metric, such as error probability for a digital system, of course, depends in part on the characteristics of the synchronization subsystems. For computational economy, however, it is desirable to avoid explicit simulation of the synchronization subsystems while estimating error rates and other end-to-end performance measures. It turns out that one can often dispense with such explicit simulation and still obtain an adequate approximation to the performance measure. There are, in fact, several options available to the simulator with respect to the treatment of synchronization, and these will be outlined in the subsection following (and expanded upon subsequently). Regardless of the option chosen, a basic understanding of the functioning of synchronization systems is needed, and this introduction is provided here.

An appreciation of the role that carrier and timing recovery play in system performance may be had through the following simple example. Consider a binary phase-shift-keying (BPSK) system, signaling over an AWGN channel. The received signal is of the form

$$S(t) = \pm Ap(t) \cos(\omega_c t + \theta) \quad (8.12.1)$$

in the interval $0 \leq t \leq T$, where $p(t)$ is the basic rectangular pulse, $p(t) = 1$, $0 \leq t \leq T$, and zero elsewhere. To this signal Gaussian receiver noise, $n(t)$ is added[†] Establishing carrier synchronization means generating at the receiver a “local oscillator” (LO) of the form

$$C(t) = L \cos(\omega_c t + \hat{\theta}) \quad (8.12.2)$$

where $\hat{\theta}$ is an estimate of θ (ideally, $\hat{\theta} = \theta$) and L is the amplitude of the LO, which for later simplicity (and without loss of generality) we set equal to 2.

The demodulated output $d(t)$ is given by

$$d(t) = \{[S(t) + N(t)]C(t)\}_{\text{lowpass}} = \pm Ap(t) \cos \epsilon + N_d(t) \quad (8.12.3)$$

where $\epsilon = \theta - \hat{\theta}$ and $N_d(t)$ is the demodulated noise. It is obvious that a phase error reduces the signal by $\cos \epsilon$ or the power by $\cos^2 \epsilon$; in some modulation schemes, in addition to power

[†]We ignore the standard inconsistency inherent in assuming $n(t)$ is of finite power while $p(t)$ can be still rectangular, as it is not central to the following discussion.

reduction, phase error also induces crosstalk (interference). The effect of phase error is often visualized through a phasor diagram of the kind illustrated in Figure 8.57a.

In the simplified system in question, the output signal pulse is still rectangular, and the matched filter is an integrate and dump (I&D) filter (Section 8.9). The I&D needs timing information as to when to begin and end integrating. Let us assume the beginning of a pulse is at $t = 0$ and denote by $\hat{t}$ the instant when the “begin” timing pulse occurs, and let us assume the integration period is T . In the worst case, which means here an alternating “1” and “0” bit pattern, the I&D output after one integrating period is reduced by $(T - 2\hat{t})/T$. (Refer to Figure 8.57.) The power is of course reduced by $(T - 2\hat{t})^2/T^2 = (1 - 2\hat{t}/T)^2$. In the case

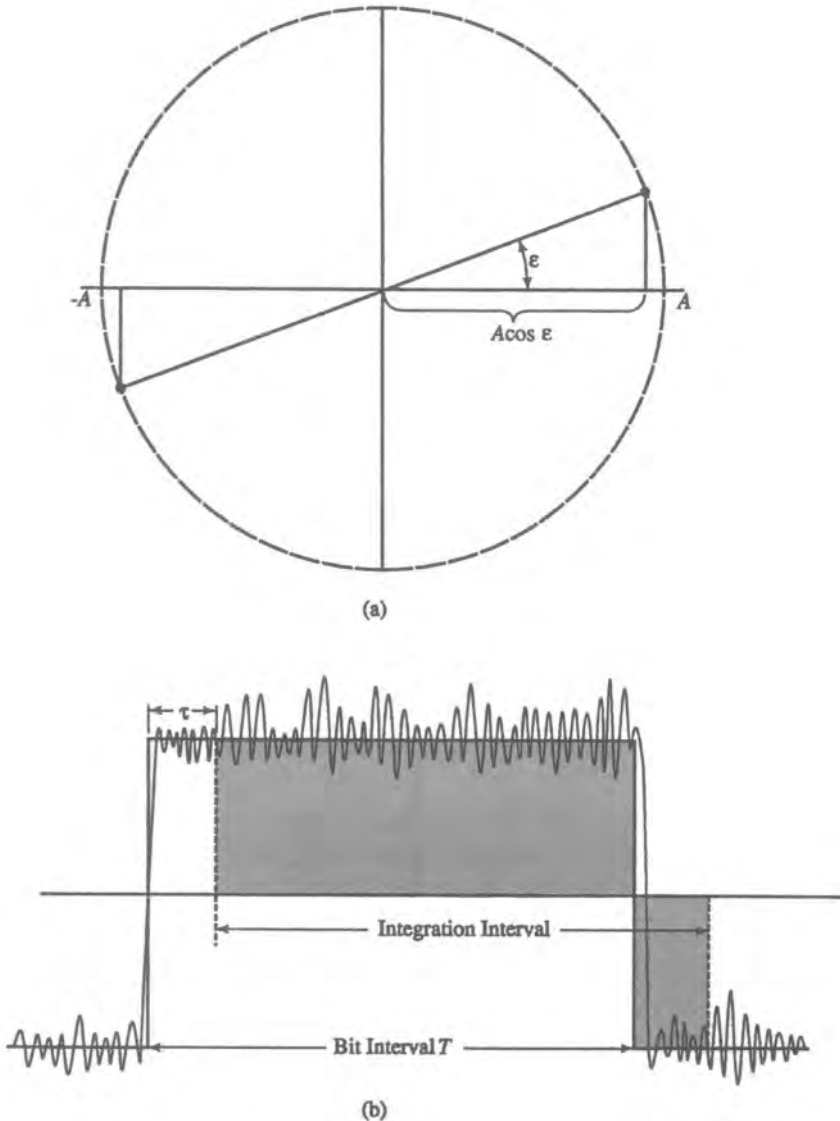


Figure 8.57. (a) Effect of phase offset; equivalent to reducing power by $\cos^2 \epsilon$. (b) Effect of timing error; equivalent to reducing power by $(1 - 2\tau/T)^2$.

considered, timing error has a more severe effect than phase error of equal relative magnitude. For example, if $\epsilon = 18^\circ$, which is 10% of the 180° separation between the two phasors, the power penalty is $\cos^2(18^\circ) \approx 0.5$ dB. On the other hand, a timing error of 10%, $\hat{\tau}/T = 0.1$, results in a loss of $(1 - 0.2)^2 \approx 2.0$ dB. For more complex modulation schemes, such as higher order QAM, the effects of phase and timing errors become increasingly severe, and phase errors become relatively more significant than indicated above. Unfortunately, it is also for these complex signal constellations that we find it most difficult to extract accurate carrier and timing references.

In general, of course, $\hat{\theta}$ and $\hat{\tau}$ are not fixed, but are really random processes whose properties will typically depend on the specific structures that implement the synchronization functions. There are in fact a great many such structures that have evolved both for carrier and timing recovery; some schemes have been developed for *joint* carrier and timing recovery, and others have made use of the detected information (decision-directed or data-aided) as well as the received signal in an attempt to derive optimal estimates. It is beyond our intended scope to summarize here the vast body of related literature (see, e.g., Refs. 17 and 100–109), to which the reader is referred. We will, however, look at a few synchronization schemes, with the primary objective of clarifying the simulation issues. From the simulation standpoint, however, the inclusion of the synchronization functions does not necessarily require an explicit replication of a physical hardware structure. In fact, as mentioned earlier, there are a number of options available to the simulator. These will be outlined next, following which a more detailed discussion of these possible approaches will be given.

8.12.1. Approaches to Including Synchronization in Simulation

We noted above that simulation of synchronization functions could be done for two reasons. One would be to study the behavior of specific implementations to see if certain performance criteria would be satisfied. In other words, the synchronization *subsystem* is the object of study. In this case it is inherent that the specific hardware or software implementation of interest must be simulated. In such subsystem studies, the performance criteria typically relate to a structure's transient behavior: such measures as acquisition time or pull-in range, as a function of the parameters of transient inputs, would be the types of items of interest. We will spend some time looking at this aspect for perhaps the most widely known acquisition circuit, the second-order phase-locked loop (PLL): this will be done in Section 8.12.8. The second reason for needing to represent synchronization is simply to account for their effect within a *system-level* simulation, where the system-level performance measure (such as BER) is the one of interest. In this case the synchronization subsystem's transient behavior is less important than its steady-state performance, which is measured in terms of the bias and jitter mentioned above. To set the stage for discussing the latter, let us return to (8.12.3) and define a decision operator Ξ such that $\Xi(d) = P_e(\hat{\theta}, \hat{\tau})$. In other words, this is short-hand notation for saying that, for given $\hat{\theta}$ and $\hat{\tau}$, the error probability associated with the signal $d(t)$ is $P_e(\hat{\theta}, \hat{\tau})$ (Figure 8.58a). While we started with the simple example in (8.12.1)–(8.12.3), we can certainly visualize the signal (8.12.1) input to a more or less arbitrary system (Figure 8.58b), the performance of which can be described parametrically also as $P_e(\hat{\theta}, \hat{\tau})$.

Thus, we can define different types of simulation approaches, depending on the way in which $(\hat{\theta}, \hat{\tau})$ are described, what is assumed to be known about them, and the particular way in which they are obtained. Some of the approaches are “structural,” in the sense that they specify operations to be performed in order to extract $(\hat{\theta}, \hat{\tau})$. These approaches are treated in this section. Other approaches are based more on a statistical description of $(\hat{\theta}, \hat{\tau})$, or at least

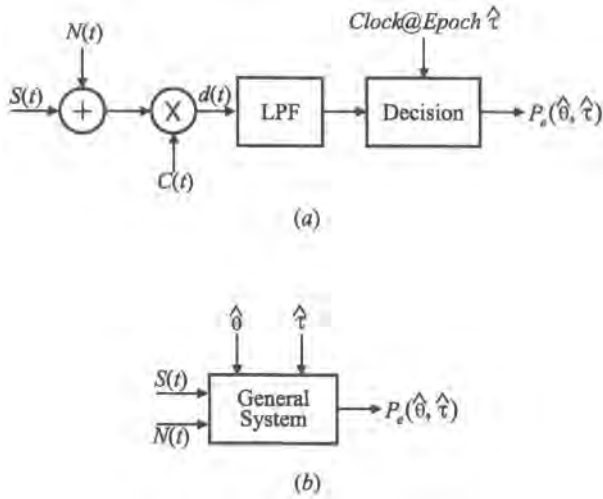


Figure 8.58. The impact of phase and timing error on BER. (a) Simple coherent detection system; (b) general system.

on an algorithm based on a statistical framework: these latter approaches are discussed in Chapter 10, which deals with estimation of parameters.

To illustrate one approach, which is suitable in the early stages of system development, when no specific designs have yet been decided, one can simply *assume* values for $(\hat{\theta}, \hat{\tau})$, or, for that matter, ranges of values, which have somehow been suitably chosen. This approach, which we refer to as *hardwired*, is developed in Section 8.3.2. An extension of this approach is to consider $(\hat{\theta}, \hat{\tau})$ to be random variables and obtain an average error probability as $P_e(\hat{\theta}, \hat{\tau})$ averaged over the distribution of $(\hat{\theta}, \hat{\tau})$. This procedure is described in Chapter 11, which treats estimation of BER per se. Relevant to the present context is the determination of $f(\hat{\theta}, \hat{\tau})$, the pdf of $(\hat{\theta}, \hat{\tau})$, but because this has a more statistical flavor, we shall discuss this topic in Chapter 10. In general, $(\hat{\theta}, \hat{\tau})$ are random processes, generally relatively slowly varying in comparison to the information rate. The time variation can be due both to the noise that is input to the system and to that which is already in the phase of the carrier. If we can hypothesize a reasonable model for the $(\hat{\theta}, \hat{\tau})$ process, one that can be fairly easily implemented by a random number generator, this RNG can in effect replace *all* of the equipment and related impairments that led to the realization of this process.[†] We call such a model an *equivalent* random process model since it represents an observable at the output of a processing chain. This approach is outlined in Section 8.3.3.

The approaches discussed thus far use abstract models for the properties of $(\hat{\theta}, \hat{\tau})$, that is, no synchronization structures are explicitly simulated. Evidently, if we did include such explicit structural models, the simulation itself would produce the evolving processes $(\hat{\theta}, \hat{\tau})$, and no particular assumptions about them would need to be made. We will illustrate this approach with some simple and well-known structures in Sections 8.12.4–8.12.7. The essential core of many structures is a phase-locked loop, which can of course also be self-standing as a synchronization circuit. In Section 8.12.8 we will look into the simulation of the PLL in some detail, both as a phase tracker and as an FM demodulator.

[†]While we speak of $(\hat{\theta}, \hat{\tau})$ for simplicity, what is of real interest is $\hat{\theta} - \theta$ and $\hat{\tau} - \tau$.

Finally, we note that there can be what we shall call “algorithmic” structures, as opposed to hardware structures, which are based on processing the input signal in a fashion that is derived from some objective statistical criterion. Typically, these algorithms attempt to find an “optimum” estimator based on a finite observation window. Because the underlying notions here tend to be more statistical in nature, we defer discussion of these *block estimators* to Chapter 10.

8.12.2. Hardwired Synchronization: Phase and Timing Bias

Let us consider a more general channel than the AWGN taken just above. Let the received waveform be expressed as

$$r(t) = \rho(t) \cos[\omega_c t + \phi(t)] \quad (8.12.4)$$

where ρ and ϕ are the envelope and phase, respectively, and these can include the effects of noise and distortion. As before, we take the local oscillator reference to be represented as

$$C(t) = 2 \cos(\omega_c t + \hat{\theta})$$

and the demodulated baseband waveform is

$$d(t) = [r(t)C(t)]_{\text{lowpass}} = \rho(t) \cos[\phi(t) - \hat{\theta}]$$

The waveform $d(t)$ is “matched” filtered and sampled at some epoch $\hat{\tau}$ to make a decision. As before, we express by $\Xi[d(t)]$ those operations that produce the error probability $P_e(\hat{\theta}, \hat{\tau})$. In the method at hand, we simply *assume* values for $(\hat{\theta}, \hat{\tau})$. It is fair to ask how one can arrive at reasonable values, and we shall discuss this presently. If we have reasonably good values to start with, we can visualize obtaining $P_e(\hat{\theta}, \hat{\tau})$ as a function of $(\hat{\theta}, \hat{\tau})$. To illustrate this point, consider $\hat{\tau}$ fixed and suppose we obtain the (conditional) BER $P_e(\hat{\theta}, \hat{\tau}|\hat{\tau})$ by varying $\hat{\theta}$. Of course, in actuality we must use a discrete set of values for $\hat{\theta}$, say NP in number. Typically, the values would be equally separated by perhaps 2° to 5° . The result might appear as shown in Figure 8.59, where it is implied that the NP values of P_e have been interpolated into a smooth curve. There is a $\hat{\theta} = \hat{\theta}_{\text{opt}}$ which minimizes the BER and we can interpret the result as the BER sensitivity to phase reference *bias*, defined as $\hat{\theta} - \hat{\theta}_{\text{opt}}$. A curve such as that in Figure 8.59 can be used to set a specification on phase bias, or static phase error, as it is also called. Such a curve is also useful in extracting the best performance that *any* carrier recovery structure is capable of.[†] If we were to simulate an actual carrier recovery structure, we could also observe how close to $\hat{\theta}_{\text{opt}}$ its average phase estimate would be. The assumption of a fixed phase reference can be thought of as achieved by coupling the transmitter oscillator (ideally pure) directly to the receiver, which is where the terminology *hardwired* comes from. In practice one attempts to “trim” delay lines or cable lengths to effect close to zero bias, but one could again in principle deliberately vary the bias.

As mentioned, the detection process takes place by sampling the matched filtered version of $d(t)$ at epoch $\hat{\tau}$. We are also interested in the sensitivity of BER to $\hat{\tau}$, and so, in analogy

[†]There is a side issue related to the duration of time (number of symbols) over which we take $\hat{\theta}_{\text{opt}}$ to have a fixed value. In other words, in a long simulation, we could divide the record into subintervals, over each of which different values of $\hat{\theta}_{\text{opt}}$ might apply.

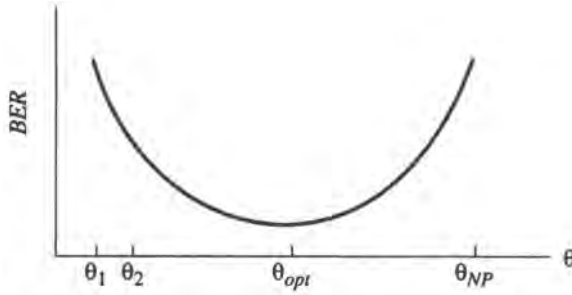


Figure 8.59. Illustration of BER as a function of phase reference for a fixed sampling instant.

with the above, we can sample the detected signal at a set of values of $\hat{\tau}$, say NS in number, to obtain the conditional BER $P_e(\hat{\theta}, \hat{\tau}|\hat{\theta})$. The set of $\hat{\tau}$ values is sometimes called a *sampling comb*; typically, the values would be equally separated by perhaps 0.01–0.05 of a symbol duration. The result might look like Figure 8.60, which again implies interpolation has been carried out. Very much the same things as were said before about carrier synchronization can be said here with respect to the BER sensitivity to timing bias, here obviously defined as $\hat{\tau} - \hat{\tau}_{opt}$. A hardwired timing reference refers to the situation when the transmitter clock is coupled directly to the receiver's detector.

Clearly, the BER is a function both of $\hat{\theta}$ and $\hat{\tau}$, hence a global view of this two-dimensional dependence is a three-dimensional surface. Such a surface provides an immediate sense of the sensitivity of error rate to departure from the jointly optimum values. Note that the $\hat{\theta}_{opt}$ obtained for different values of $\hat{\tau}$ may vary, and similarly for $\hat{\tau}_{opt}$ for different $\hat{\theta}$. An illustration of the surface $P_e(\hat{\theta}, \hat{\tau})$ is given in Figure 8.61. The utility of such a presentation is discussed in Ref. 110.

■ *Remark.* Two points are worthwhile making concerning, first, the computational cost of this approach, and second, finding good initial values for $(\hat{\theta}, \hat{\tau})$.

1. *Cost of Multiple Phase/Timing Trials.* In practice it is relatively inexpensive, in terms of overall run time, to use a fairly large number of phase and timing trials (NP and NS , respectively). This is because one does not rerun the simulation for each pair of values. The waveform at the input to the receiver is of course the same irrespective of the phase and timing particulars, and does not have to be recomputed. In many problems, the bulk of the computation takes place prior to the receiver, and therefore the additional computation incurred by multiple phase/timing trials adds a relatively small percentage to the total run time. Values of NP or NS on the order of 10 are quite practical.

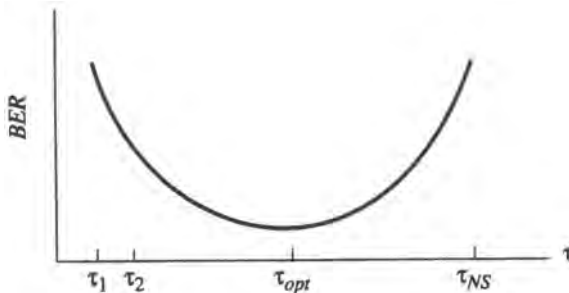


Figure 8.60. Illustration of BER as a function of sampling instant for a fixed phase reference.

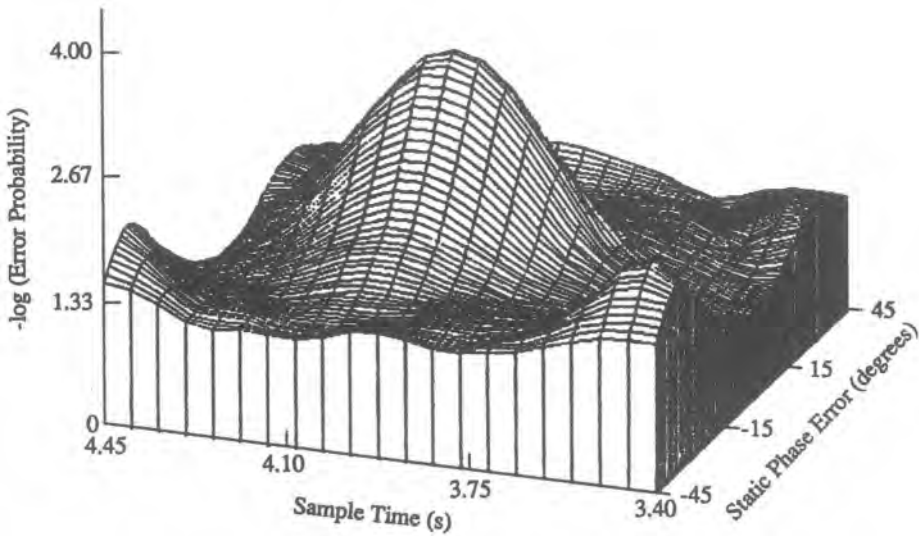


Figure 8.61. Example of BER surface as a function of phase reference and sampling instant.

2. Finding Initial Values for $(\hat{\theta}, \hat{\tau})$. As mentioned above, we have to somehow “center” the NP or NS values around appropriate starting points. There are several ways of finding “good” starting values. In Chapter 10 we shall consider a particular one. Here we point out a very simple way of doing so, based only on a knowledge of the system’s phase characteristic, say $\psi(f)$, assuming the system is linear. No computation, as such, is required. The phase $\psi(f_c)$ at the center frequency [or $\psi(0)$ if ψ is the lowpass equivalent] is roughly the phase rotation experienced by the carrier, and hence forms a good starting point for exploring $\hat{\theta}$. The negative of the slope of $\psi(f)$, which could be determined by a first-order least-squares fit or simply “eyeballed,” is clearly a good starting point for $\hat{\tau}$ since delay is defined as $-(1/2\pi)d\psi(f)/df$. ■

The preceding approach in effect treats $(\hat{\theta}, \hat{\tau})$ as if they were fixed, but unknown. In actuality, they are slowly varying processes that can usually be assumed unchanging over many symbol periods. In that case, at least over such periods, only the first-order pdf is an adequate description, and one is usually interested in P_e averaged over these distributions. Possible distributions for $(\hat{\theta}, \hat{\tau})$ are discussed in Chapter 10, and the average mentioned is further applied in Chapter 11.

8.12.3. Synchronization Using an Equivalent Random Process Model

The statistical averaging just mentioned will not reproduce any aspect of the temporal behavior of errors. For this, we must emulate $(\hat{\theta}, \hat{\tau})$ as evolving processes having, or approximating, the actual properties. Evidently, one way of doing this is by imitating the actual equipment that produces them. If we were to do this with high fidelity, it would imply modeling and simulating every frequency or clock source (which generally consists of a basic oscillator and up- or downconverters) as well as the associated tracking structures. This significantly complicates a simulation, and it is a complication that is generally avoidable if our objective is to estimate a system-level performance objective, such as BER. For, in this case, it is not the details of how $(\hat{\theta}, \hat{\tau})$ are produced that is of interest, but the net effect on the

decision variable. For this purpose it is only necessary to define an equivalent process to be injected at the receiver. This equivalent process replaces all of the sources and equipment that lead up to it. If we could precisely simulate this equivalent process, we would in fact be making no approximation insofar as the detected baseband waveform is concerned. In reality, it is too much to expect that the equivalent process' properties could be precisely known, but it may be possible to define an equivalent random process (erp) which is a good enough imitation of the real one, at least with respect to its effect on the BER. In order for this replacement to be practical, we also need to have an efficient way to pseudorandomly generate this process.[†] In order to illustrate this idea, consider now another variation of the formulation presented above. Let the received waveform be of the form

$$r(t) = S(t) + N(t)$$

where $N(t)$ is thermal (additive white Gaussian) noise, and the signal

$$s(t) = \rho(t) \cos[\omega_c t + \phi(t) + \delta(t) + \theta] \quad (8.12.5)$$

includes a wanted phase term $\phi(t)$ as well as an extraneous phase term $\delta(t)$ which represents oscillator phase noise and the effects of distortion. With standard trigonometric manipulations, we can express $r(t)$ as

$$r(t) = \rho'(t) \cos[\omega_c t + \phi(t) + \alpha(t) + \delta(t) + \theta]$$

which reflects the presence of the noise. We postulate a carrier recovery circuit which is of the "modulation wipeoff" type. That is, it attempts to "follow" the phase of the incoming carrier, exclusive of the modulation. We can thus represent the local oscillator reference as

$$C(t) = 2 \cos[\omega_c t + \hat{\delta}(t) + \hat{\theta} + \epsilon(t)]$$

where $\epsilon(t)$ is due to the presence of thermal noise, and the other phase terms are the carrier recovery structure's estimates of the corresponding quantities in (8.12.5). Multiplying $r(t)$ by $C(t)$ and taking the lowpass portion as before yields the demodulated waveform

$$d(t) = \rho(t) \cos[\phi(t) + \alpha(t) + \mu(t)] \quad (8.12.6)$$

where $\mu(t) = \delta(t) - \hat{\delta}(t) - \epsilon(t) + \theta - \hat{\theta}$. Thus, the demodulated waveform depends on the *untracked* phase "noise" $\delta(t) - \hat{\delta}(t)$ and the *tracked* thermal noise $\epsilon(t)$. (The *static phase error* $\theta - \hat{\theta}$ is simply a constant, which can be assumed or made a parameter, just as in the hardwired approach. In any case, there are no subtle modeling issues there.) Ignoring the static phase error, we find that $\mu(t)$ is the equivalent random process[‡] to which we have been alluding. Hence, the system can be simulated without phase noise and $\mu(t)$ can simply be inserted directly into the demodulated signal, as indicated in (8.12.6). In Case Study II, Chapter 12, we will show that a reasonable approximation for $\mu(t)$ is a Gaussian random process (GRP) having a certain power spectral density with a given rms value, usually

[†]This is an example of the equivalent random process methodology introduced in Section 2.3.3. The basic idea behind this methodology is to replace a set of elements and the source into the first of these elements by the process that would be observed at the output of the last of the elements, when such a replacement is feasible and sensible.

[‡]In actuality, we would synthesize two processes, one for $\epsilon(t)$ and one for $\delta(t) - \hat{\delta}(t)$, the sum of which is $\mu(t)$.

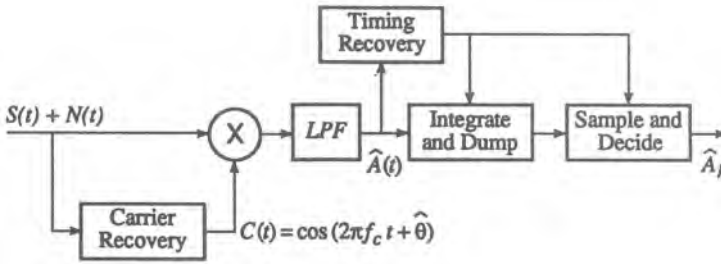


Figure 8.62. Simplified receiver block diagram showing synchronization functions.

referred to as the rms phase jitter. Methods for generating a GRP have been discussed in Chapter 7.

Generally, $\mu(t)$ is a “slow” process in comparison to the information process, hence its values separated by T_s will be correlated to some extent, and often highly so. This has two main implications. First, it implies a possible inefficiency if indeed successive values are highly correlated. This inefficiency can be mitigated to some extent by using multirate techniques (Chapter 3). The second implication that one must be mindful of is that the simulation run must be long enough with respect to the bandwidth of the process to allow a “representative” segment of the process to evolve. Otherwise, the simulation results will be biased.

The preceding discussion applies virtually unchanged to the symbol synchronization problem, except that the “jitter,” instead of being expressed in degrees, is expressed in the time domain, typically referenced to a symbol duration.[†] Hence we can simulate the effect of clock jitter and bit sync recovery by transmitting a perfectly synchronous data stream and sampling the matched filter output with a noisy clock, the noisiness of which is represented by another erp, which can also be a Gaussian random process with a specified PSD.

8.12.4. Carrier Recovery—BPSK

In this subsection and the next few, we will look at specific structures and how they can be simulated. We begin in this subsection with carrier recovery for binary PSK. An ideal BPSK signal is of the form

$$S(t) = A(t) \cos(2\pi f_c t + \theta) \quad (8.12.7)$$

where $A(t)$ is a random binary waveform,

$$A(t) = \sum_k A_k p(t - kT_b - D) \quad (8.12.8)$$

with $A_k = \pm 1$ and $p(t)$ a unit-amplitude pulse of duration T_b . As noted earlier, in order to demodulate $s(t)$ and extract A_k , the receiver (Figure 8.62) needs a local carrier $C(t)$ of the form

$$C(t) \cos(2\pi f_c t + \hat{\theta}) \quad (8.12.9)$$

and a local clock that has transitions at $kT_b + \hat{D}$, i.e., the receiver has to estimate the carrier phase θ and the clock phase D .

[†]Actually, it is also common to express clock jitter in degrees, 360° implying a symbol duration.

The spectrum of $S(t)$ has no carrier component because $A(t)$ has no dc term. To see how we can generate a carrier component, suppose we take the undistorted and noiseless modulated waveform $S(t)$ and square it to obtain

$$\begin{aligned} S^2(t) &= A^2(t) \cos^2(2\pi f_c t + \theta) = \frac{A^2(t)}{2} [1 + \cos(4\pi f_c t + 2\theta)] \\ &= \frac{A^2(t)}{2} + z_2(t) \end{aligned}$$

where we have obviously defined the bandpass term $z_2(t)$ as (the subscript 2 is a reminder of the squaring operation)

$$z_2(t) = \frac{A^2(t)}{2} \cos(4\pi f_c t + 2\theta)$$

Since $\tilde{S}(t) = A(t) \exp(j\theta)$, it is clear that the complex envelope of $z_2(t)$ is

$$\tilde{z}_2(t) = \frac{1}{2} \tilde{S}^2(t) \quad (8.12.10)$$

In the ideal case, $A(t) = \pm 1$, $A^2(t) = 1$, and hence we have

$$S^2(t) = \frac{1}{2} + \frac{1}{2} \cos(4\pi f_c t + 2\theta)$$

or $z_2(t)$ is simply the double-frequency term $\frac{1}{2} \cos(4\pi f_c t + 2\theta)$.

By passing $S^2(t)$ through a bandpass filter, we can isolate $z_2(t)$, and by processing this component through a frequency divider circuit (divide-by-two), we can extract the reference carrier $\cos(2\pi f_c t + \theta)$. This sequence of operations removes the signal (modulation) component from the modulated carrier to produce an unmodulated local carrier with the correct phase. This local carrier can be used to coherently demodulate $S(t)$ and extract $A(t)$, which can be further processed through an integrate-and-dump filter to extract A_k .

In an actual system, $A(t)$ will be a filtered version of a random binary waveform and $S(t)$ will be accompanied by noise at the receiver input. These factors lead to errors in the recovered phase. The rms value of the jitter will be a function of the noise power, fluctuation in $A(t)$ due to filtering, and the bandwidth of the bandpass filter. In the model above, then, we should replace θ by $\theta(t)$, indicating its time-varying nature.

In principle, the structure shown in Figure 8.63 would accomplish the task of generating a carrier reference if the input BPSK signal were undistorted and noiseless, the carrier frequency were precisely (or very close to) f_c , and the carrier phase θ were constant. In actuality, this scheme is normally unsatisfactory because of conflicting requirements on the bandpass filter to be narrow enough to admit only a small amount of noise, yet wide enough to accommodate instabilities in the incoming carrier frequency. The standard solution is to use a tracking filter, or phase-locked loop (PLL), as indicated in Figure 8.64. This scheme is one

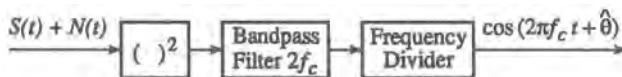


Figure 8.63. A carrier recovery scheme for BPSK (open loop).

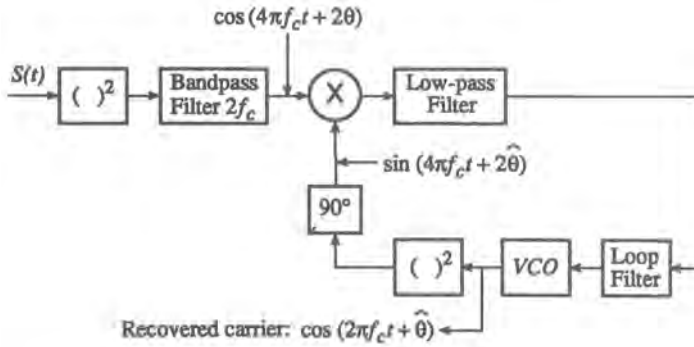


Figure 8.64. One implementation of the squaring loop for BPSK carrier recovery.

variation of a number of realizations generally referred to as a squaring loop. The loop filter following the multiplier will also eliminate the double-frequency term.

Two alternatives to the squaring loop are shown in Figure 8.65: (a) what is known as a Costas loop, and (b) a decision-directed loop, which utilizes the detected bit stream. It should be noted that the squaring loop and the Costas loop extract a carrier reference without regard to the bit timing, while the decision-directed loop cannot work without simultaneous symbol synchronization. Other variations in structure can be found in the literature.

The complex lowpass-equivalent version of the structure in Figure 8.64 is shown in Figure 8.66. Used as a subsystem in a system-level simulation, the input to the loop is the complex envelope $\tilde{S}(t)$ at the input to the receiver, as shown. As indicated in the preceding subsection, processes within synchronization circuits will generally be much slower than information processes. That is, $\hat{\theta}(t)$ will vary more slowly, and often much more slowly, than $A(t)$. This could result in computational inefficiency, since the waveform into the PLL may be greatly oversampled. The multirate techniques discussed in Chapter 3 may then be applied to advantage. However, this situation does not necessarily exist to the same degree for every synchronization structure. For example, in the Costas loop, we see that the actual signal enters the loop in both branches. The only portion of the loop where the waveform is slowly varying is after the loop filter.

Not only is the PLL a device of fundamental importance in synchronization, but, as we have seen in Section 8.8, it is also used as a demodulator for angle-modulated signals. In Section 8.12.8 we will examine the simulation of the PLL in some detail.

8.12.5. Timing Recovery—BPSK

To simplify the exposition, we assume that carrier recovery and timing recovery are independent. We also assume that modulation and demodulation are ideally implemented. Consequently, although we began with BPSK modulation, the timing recovery scheme is actually independent of the carrier modulation method. When that is the case, the transmitted baseband signal, namely

$$A(t) = \sum_k A_k p(t - kT_b - D) \quad (8.12.11)$$

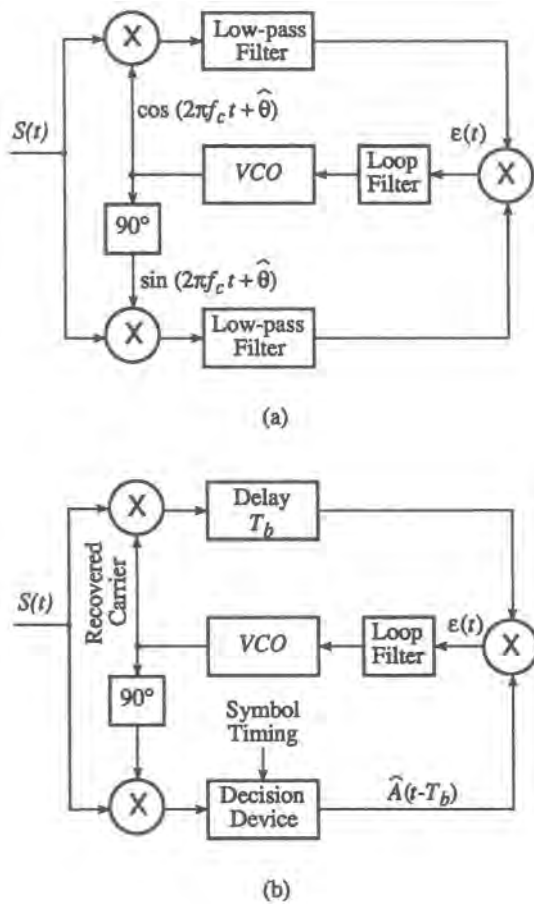


Figure 8.65. Alternative carrier recovery schemes for BPSK. (a) The Costas loop for BPSK; (b) a decision-feedback loop for BPSK.

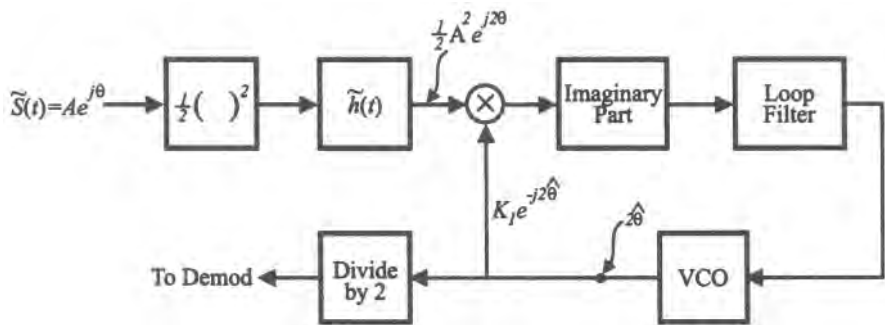


Figure 8.66. The complex lowpass equivalent of the structure in Figure 8.64.

is recovered in the baseband of the receiver exactly in the same form. In order to recover the *information*, the $\{A_k\}$, it is implied that the receiver has available a *clock* which is in synchronism with the received signal. A clock is any periodic signal $P(t)$ whose period is T_b and whose zero crossings occur at $\{kT_b - D\}$, that is,

$$P(t) = \sum_k g(t - kT_b - D) \quad (8.12.12)$$

where $g(t)$ is a pulse of duration $< T_b$. The leading edge of the clock pulse is used to control the observation of the decision device. For example, in the simple I&D filter example in the beginning of this section it was implied that a timing pulse initiates the integration period and the following timing pulse terminates that period and initiates the next. We also saw how important it is for these pulses to be close to the correct epochs (instants).

There is a useful conceptual analogy between carrier recovery and timing recovery. In Section 8.7 we called a waveform such as (8.12.11) baseband (pulse)-modulated signal. Thus, timing recovery is analogous to trying to reconstruct an “unmodulated carrier” waveform at the clock rate (instead of at f_c) and with a phase D (instead of θ). However, there are important differences. For example, if we attempted to imitate the squaring loop operation in the present context, it would not work for the basic $p(t)$ postulated because $A^2(t)$ would be unity for all t , hence would contain no zero crossings.

Thus, solutions to the timing recovery problem depend in part on the nature of $p(t)$, and conversely one might design $p(t)$ to lead to good solutions. There are in fact many possible solutions^(9,17,103,104) to the tuning recovery problem, some not necessarily connected at all with the shape of $p(t)$.

We present one possibility for the square-pulse case, known as the delay-and-multiply method. Instead of $A^2(t)$, consider the product $A(t)A(t - T_d)$, which accounts for the name of the method. T_d is a fixed delay whose value can be chosen appropriately. The basis of this technique can be understood by referring to Figure 8.67. The result of the product

$$M(t) = A(t)A(t - T_d) \quad (8.12.13a)$$

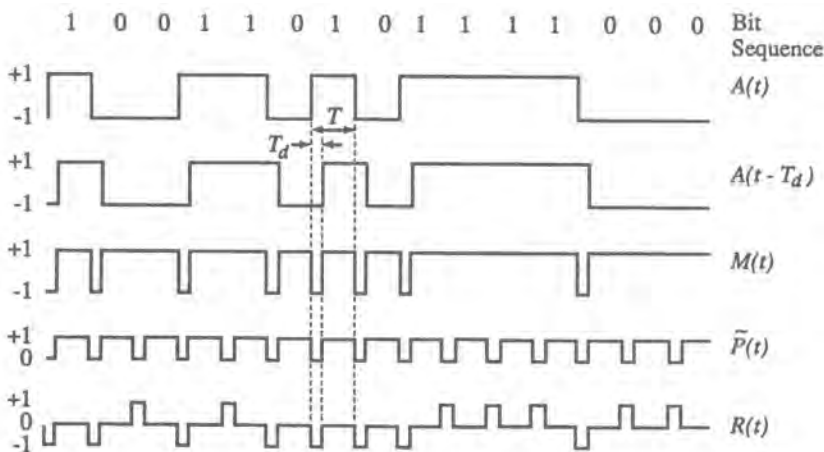


Figure 8.67. Illustration of the delay-and-multiply method for $T_d = T_b/4$ (from W. J. Gill and J. J. Spilker, Jr., An interesting decomposition property for the self products of random and pseudorandom binary sequences, ©IEEE Trans. Commun. Systems, volume CS-11, no. 2 pp. 246–247, June 1963. $A(t)A(t - T_d) = \tilde{P}(t) + R(t)$, where $\tilde{P}(t)$ is a periodic signal and $R(t)$ is random.

can be decomposed into a periodic and an aperiodic (random) component,

$$M(t) = \tilde{P}(t) + R(t) \quad (8.12.13b)$$

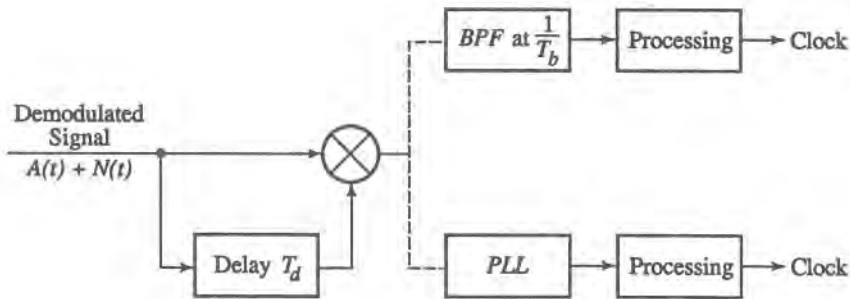
as also shown in the figure. The periodic component $\tilde{P}(t)$ has harmonics at multiples of $1/T_b$, hence the tone at $f = T_b^{-1}$ can be filtered out to provide the basis for a clock, as indicated in Figure 8.68. Note that the filtered tone $\cos(2\pi T_b^{-1}t)$ is not yet in the form (8.12.12). Some further (nonlinear) processing must be done to recover a clock. A hypothetical sequence of operations is shown in Figure 8.68b. In this case $g(t) = \delta(t)$, an impulse.

If there were only a periodic component, without noise, the open-loop implementation (upper branch of Figure 8.68a) would in principle suffice. In actuality, the random component $R(t)$ acts as a type of (pattern-dependent) noise. Hence, an important attribute of the method is the SNR at the BPF output. It can be shown⁽¹⁷⁾ that the power in the tone at T_b^{-1} is given by

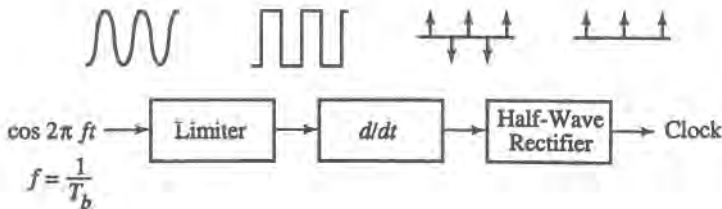
$$\left(\frac{T_d}{T_b}\right)^2 \left[\frac{\sin(\pi T_d/T_b)}{\pi T_d/T_b} \right]^2$$

whereas the power in the random component is given by

$$\left(\frac{T_d^2}{T_b}\right) \left[\frac{\sin(\pi T_d/T_b)}{\pi T_d/T_b} \right]^2 B$$



(a)



(b)

Figure 8.68. Clock recovery scheme for rectangular baseband signal. (a) Block diagram of timing recovery scheme; dashed lines indicate choice of upper or lower branch; (b) a hypothetical sequence of nonlinear processing operations.

where B is the bandwidth of the bandpass filter (or loop) around $f = T_b^{-1}$. It turns out that the best value of $T_d = T_b/2$, and it can also be seen that there is an irreducible amount of “self-noise.” There is of course thermal noise present as well. Both of these noises induce jitter in the zero crossings of the clock, which can be better reduced by tracking the tone at $f = 1/T_b$ with a phase-locked loop.

As mentioned, there are many possible schemes for timing recovery. For convenient reference we provide in Figure 8.69 the block diagram for a popular approach, called the early-late-gate bit synchronizer.⁽¹⁰³⁾ The nonlinearity can take a variety of forms, but is typically a square-law device or an absolute-value nonlinearity. This synchronizer depends for its operation on the nature of the autocorrelation function of the basic pulse $p(t)$, and it works best when $p(t)$ is rectangular.

For simulation purposes, if a synchronizer structure is given, and it is important to evaluate this specific synchronizer or to evaluate system performance including this synchronizer, then one has no choice but to try to emulate that given structure. Otherwise, a simpler (statistical) model will usually be quite adequate. As discussed in earlier sections, an operational model of the clock would consist of a bias term and a statistical description of the jitter. In order to emulate the clock process in time, one would need at least the autocorrelation of the jitter. If one assumes a Gaussian process, this information can be deduced from the bandwidth of the loop or bandpass filter. For estimating average performance, the first-order pdf of the normalized jitter ϵ suffices. (The normalized jitter is the actual jitter divided by the symbol duration.) A pdf that is often assumed for this purpose is presented in Section 10.6.

8.12.6. Carrier Recovery—QPSK

The BPSK case has established the main ideas connected with synchronization. We briefly consider the QPSK case to indicate the flavor of the extension to higher order modulation schemes. A QPSK signal can be written in the form

$$S(t) = a(t) \cos(2\pi f_c t + \theta) + b(t) \sin(2\pi f_c t) \quad (8.12.14a)$$

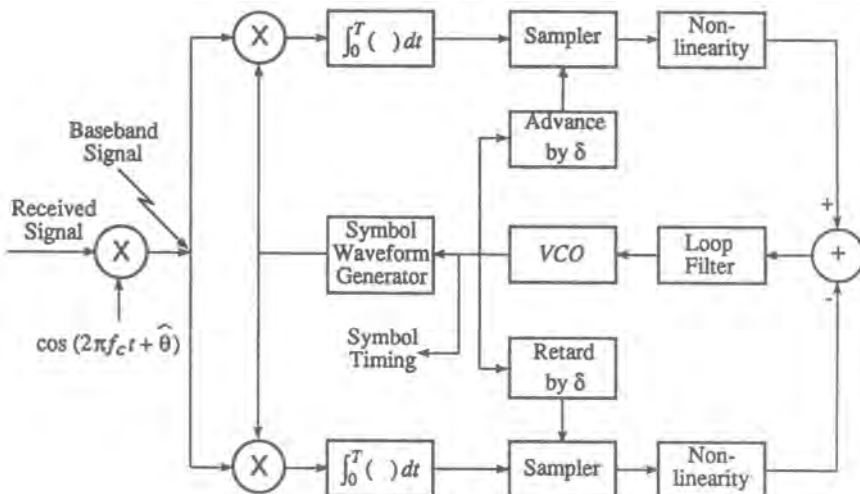


Figure 8.69. Block diagram of one implementation of early-late-gate synchronizer.

where $a(t)$ and $b(t)$ are two independent, random binary waveforms, and θ is a random, but constant phase angle. In any symbol interval, we can also express $S(t)$ in the form

$$S(t) = A \cos(2\pi f_c t + \phi_k + \theta), \quad kT_s \leq t \leq (k+1)T_s \quad (8.12.14b)$$

where A is a constant and ϕ_k takes on one of the values $0, \pi/2, \pi, 3\pi/2$. The complex envelope representation of $S(t)$ is

$$S(t) = \text{Re}\{\tilde{S}(t)e^{j2\pi f_c t}\} \quad (8.12.15)$$

where, from (8.12.14a) or (8.12.14b), the complex envelope is given in either form

$$\tilde{S}(t) = [a(t) + b(t)]e^{j\theta} \quad (8.12.16a)$$

$$= Ae^{j(\phi_k + \theta)}, \quad kT_s \leq t \leq (k+1)T_s \quad (8.12.16b)$$

One way to obtain carrier synchronization for a QPSK signal is through a generalization of the squaring-loop method, called the fourth-power, or quadrupling loop, which itself is a special case of the M th-power loop, which applies to M -ary PSK. Since the idea can just as easily be exposed for any M , we shall begin in this fashion. An M -ary PSK signal has the generic form of (8.2.14b),

$$x(t) = A \cos(2\pi f_c t + \phi_k + \theta), \quad kT_s \leq t \leq (k+1)T_s$$

where T_s is the symbol time and ϕ_k is the angle corresponding to the k th transmitted symbol. ϕ_k has one of the M values

$$\phi_k = 0, \frac{2\pi}{M}, 2\frac{2\pi}{M}, 3\frac{2\pi}{M}, \dots, (M-1)\frac{2\pi}{M}$$

Now, if we raise $x(t)$ to the power M and use (5.2.21), we have

$$x^M(t) = \frac{1}{2^M} \sum_{k=0}^M \binom{M}{k} [\tilde{x}(t)]^k [\tilde{x}^*(t)]^{M-k} e^{j2\pi f_c (2k-M)t}$$

Let us consider the sum of the $k=0$ and $k=M$ terms from the preceding equation. Calling this sum $z_m(t)$, we have

$$z_M(t) = \frac{1}{2^{M-1}} A^M \cos(2\pi M f_c t + M\phi_k + M\theta), \quad kT_s \leq t \leq (k+1)T_s \quad (8.12.17)$$

or

$$\tilde{z}_M(t) = \frac{1}{2^{M-1}} A^M e^{j(M\phi_k + M\theta)} = \frac{1}{2^{M-1}} \tilde{S}^M(t) \quad (8.12.18)$$

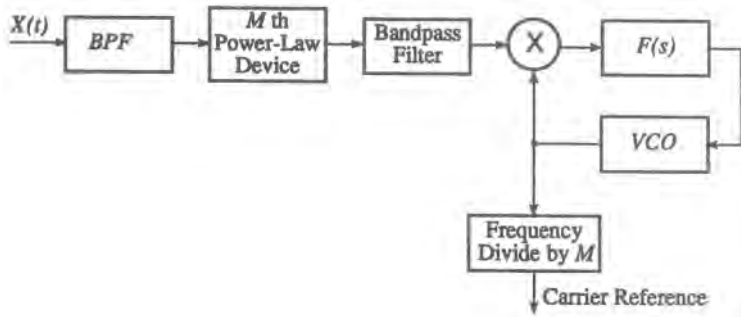


Figure 8.70. The M th power loop for recovery of M -ary PSK.

Using the fact that $M\phi_k = \text{integer multiple of } 2\pi$, we find that (8.12.17) and (8.12.18) become, respectively,

$$z_M(t) = \frac{1}{2^{M-1}} A^M \cos(2\pi M f_c t + M\theta)$$

and

$$\tilde{z}_M(t) = \frac{1}{2^{M-1}} A^M e^{jM\theta}$$

which shows that a spectral line is created at Mf_c ; a subsequent frequency divider yields a reference at f_c (see Figure 8.70). When $M = 4$, we have the specific result

$$\tilde{z}_4(t) = \frac{1}{8} \tilde{S}^4(t) = \frac{A^4}{8} e^{j4\theta} \quad (8.12.19)$$

Thus the complex envelope of $S^4(t)$ in the vicinity of $4f_c$ is proportional to $\tilde{S}^4(t)$. The remaining operations of the carrier recovery can also be expressed in terms of complex envelope representation as shown in Figure 8.71, which also indicates the correspondence between the actual and the complex lowpass operations.

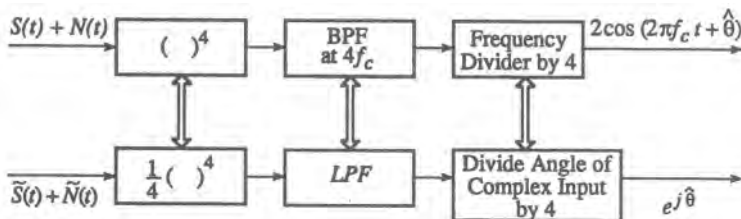


Figure 8.71. Carrier recovery for QPSK showing connection between bandpass representation and lowpass equivalent.

8.12.7. Timing Recovery—QPSK

A “delay-and-multiply”-type timing recovery scheme can be used for QPSK signals also. The product signal

$$Z(t) = S(t)S(t - T_d)$$

with $T_d \approx T_s/2$ has a strong periodic component at the clock rate. This component can be extracted using a bandpass filter or a VCO tuned to the clock rate.

The clock recovery operation can also be described in terms of the complex envelope representation as follows. Starting with (8.12.15) and (8.12.16), we can show that

$$\begin{aligned} Z(t) = & \frac{1}{2} \{A(t)A(t - T_d) + B(t)B(t - T_d)\} \cos \theta_d \\ & - \frac{1}{2} \{B(t)A(t - T_d) - A(t)B(t - T_d)\} \sin \theta_d \\ & + \text{double frequency terms at } 2f_c \end{aligned} \quad (8.12.20)$$

where $\theta_d = 2\pi f_c T_d$. Ignoring the double-frequency terms, we can express $Z(t)$ as

$$Z(t) = \text{Real part of } \{\tilde{S}(t)\tilde{S}^*(t - T_d) e^{j\theta_d}\} \quad (8.12.21)$$

The clock component in the product term can be extracted using a bandpass filter centered at the symbol rate f_s , or a VCO can be used to obtain better performance.

Bandpass and complex envelope representation of the timing recovery schemes are shown in Figure 8.72.

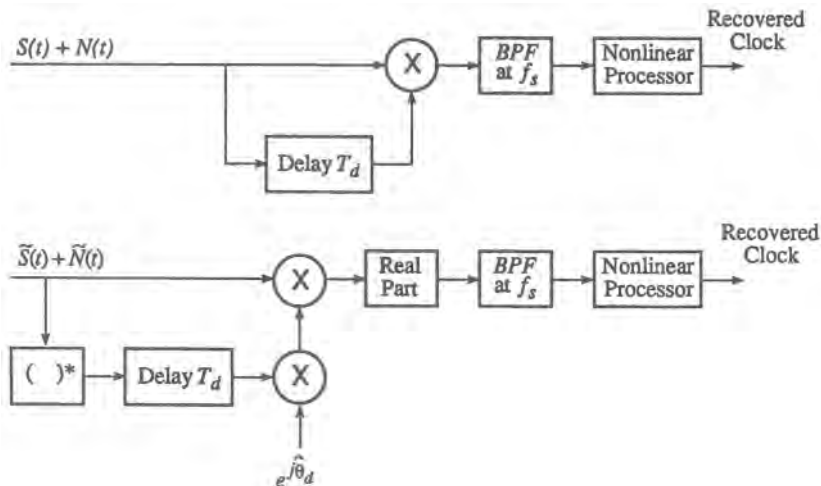


Figure 8.72. Timing recovery for QPSK showing the bandpass representation and the complex lowpass equivalent; f_s is the symbol rate.

8.12.8. Simulation of Feedback Loops: Application to the Phase-Locked Loop, Phase-Locked Demodulator, and Costas Loop

Although in principle one could use open-loop techniques, as pointed out earlier, in practice it is virtually universal to employ structures that use closed-loop (feedback) circuits such as the phase-locked loop. The simulation of feedback loops raises a number of issues that are also of interest more generally. As we have seen, a central element of many synchronization structures is the PLL. To focus the discussion, and because of its relative tractability, we shall confine ourselves almost exclusively to the classical second-order phase-locked loop, which we shall consider in its two most common applications, as a phase tracker for carrier synchronization in coherent systems, and as an FM demodulator. In the final section we will give a brief example of a structure less amenable to analysis, the Costas loop in a “modified” form.

8.12.8.1. Modeling Considerations for the PLL

The essential fact about a PLL is that it is governed by a nonlinear differential equation (NLDE). It is difficult to gain global insight into the behavior of such equations, and we are generally reduced to numerical or empirical (simulation) solutions. As we noted in Chapter 5, two quite distinct methodologies are available for simulating a device represented by an NLDE, which we referred to as *stand-alone* and *assembled* models. We pointed out also that a stand-alone model is generally superior in terms of stability and accuracy for a given T_s . However, if such a model is not available in the simulator’s library, one is forced to assemble the NLDE model from lower level models that are available. In situations where design variations in individual blocks are being investigated, it is generally more practical to use an assembled model, since otherwise we would have to develop a stand-alone model for each possible variation.

Whether or not one uses stand-alone or assembled models, aside from what may be available in the simulation library, depends also on the degree to which the requisite accuracy of modeling controls the run time of the simulation as a whole. We note that both approaches will yield indistinguishable results as T_s becomes arbitrarily small. But what is generally desired is to make T_s as *large* as possible, consistent with the desired accuracy. If it is important to optimize T_s , a stand-alone model will generally be preferable.

We note, however, that if the PLL is embedded within a larger system, the extent to which the required PLL sampling interval is likely to dominate the simulation run time depends basically on the ratio of loop bandwidth B_L to the data rate R . If $R \gg B_L$, as is often the case, then a sampling rate that is commensurate with R will usually be much larger than that needed for the loop itself, regardless of whether one uses a stand-alone or an assembled model. In fact, in that case, it may be advantageous to employ multirate techniques, that is, use a larger sampling interval internal to the PLL model.

A block diagram of the second-order loop and its phase domain version is shown in Figure 8.73. The “phase domain” is essentially the same as the lowpass equivalent, which has been shown in Figure 8.40. In either case, the amplitude has been indicated as the constant A . If the amplitude is in fact time-varying, it can easily be reflected as such in the block diagrams, or a hard-limiter could have been applied prior to the PLL. Such a limiter is easy to account for: we simply set the modulus of the complex envelope samples to a constant. If the PLL is actually part of a larger structure, like a quadrupling loop, we assume here that this operation has already been done. In order further to simplify our scope, we will assume the

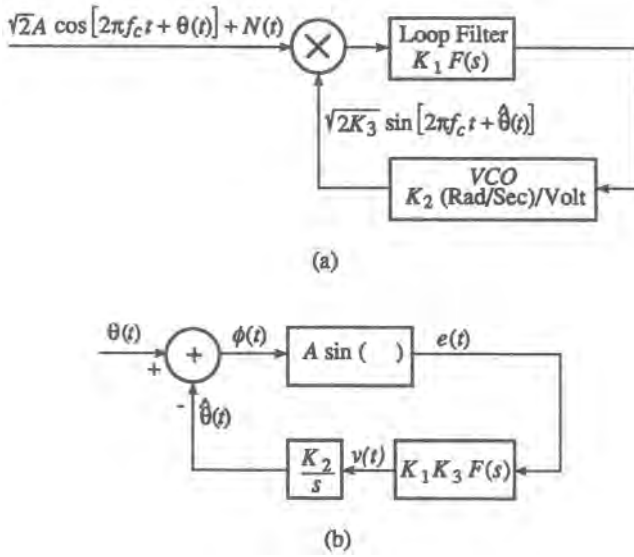


Figure 8.73. Phase-locked loop representations and related definitions. (a) Basic PLL block diagram. (b). Mathematical equivalent in “phase” domain. Loop parameter definitions for second-order loop: loop gain, $K = K_1 K_2 K_3$; natural frequency, $\omega_n = (AK/\tau_1)^{1/2}$; damping factor, $\zeta = \frac{1}{2} \omega_n \tau_1 (1 + \alpha_1/AK\tau_2)$.

local oscillator rest frequency is the same as that of the incoming carrier and we will also assume a noiseless environment, implying also that the local oscillator noise is negligible. (In Case Study II, Chapter 12, we will revisit this model with noise sources included.) Notice that the model of Figure 8.73 specifies explicitly all operations, except for the loop filter transfer function $F(s)$, which for the second-order loop we can write as

$$F(s) = \frac{s\tau_2 + 1}{s\tau_1 + \alpha_1} \quad (8.12.22)$$

where one of two cases holds for α_1 ,

$$\alpha_1 = \begin{cases} 1, & \text{passive (lag-lead) filter} \\ 0, & \text{active filter} \end{cases} \quad (8.12.23)$$

We now consider in turn the stand-alone and assembled model methodologies.

8.12.8.2. Stand-Alone PLL Model

In order to develop the stand-alone model, we follow the general procedure outlined in Section 5.4. As a first step we need to identify appropriate state variables. To that end, we recast (8.12.22) as

$$F(s) = \frac{1}{\tau_1} \frac{\tau_2 + 1/s}{1 + \alpha_1/\tau_1 s} \quad (8.12.24)$$

Substituting (8.12.28) into (8.12.29) and making use of (8.12.30), we get

$$\ddot{y}(t) = A \sin[\theta(t) + c_1 \dot{y}(t) + c_2 y(t)] + c_3 \dot{y}(t) \quad (8.12.31)$$

This is the governing second-order NLDE in terms of the y variable and the input $\theta(t)$.

For simplicity of notation we henceforth suppress the t dependence, which will be implied. We now define the state variables $\mathbf{q} = (q_1, q_2)$, as

$$q_1 = y \quad (8.12.32a)$$

$$q_2 = \dot{y} \quad (8.12.32b)$$

so that $\dot{\mathbf{q}} = (\dot{q}_1, \dot{q}_2)$ is given by

$$\dot{q}_1 = q_2 \quad (8.12.33a)$$

$$\dot{q}_2 = A \sin(\theta + c_1 q_2 + c_2 q_1) + c_3 q_2 \quad (8.12.33b)$$

The original variables, in terms of the state variables, are

$$\phi = \theta + c_1 q_2 + c_2 q_1 \quad (8.12.34)$$

$$e = A \sin(\theta + c_1 q_2 + c_2 q_1) \quad (8.12.35)$$

$$v = (AK_1 K_2 \tau_2 / \tau_1) \sin(\theta + c_1 q_2 + c_2 q_1) + (c_3 + K_1 K_3 / \tau_1) q_2 \quad (8.12.36)$$

$$\hat{\theta} = -c_1 q_2 - c_2 q_1 \quad (8.12.37)$$

An equivalent block diagram in state space form is given in Figure 8.76. Numerical solutions based on four of the methods defined in Section 5.4 are outlined below.

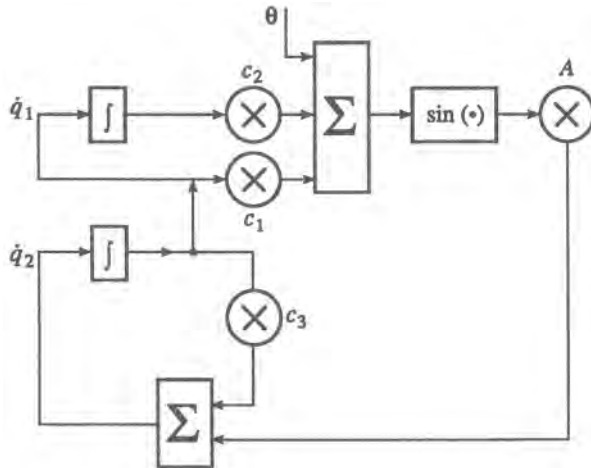


Figure 8.76. Schematic representation of PLL model in state-variable form.

Explicit Method 1: Forward Euler (FE). From Section 5.4, the FE method formulates a solution in the form

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\dot{\mathbf{q}}_n \quad (8.12.38)$$

Inserting $\dot{\mathbf{q}}_n$ from (8.12.33), we get

$$q_{1,n+1} = q_{1,n} + hq_{2,n} \quad (8.12.39a)$$

$$q_{2,n+1} = q_{2,n} + h[A \sin(\theta_n + c_1 q_{2,n} + c_2 q_{1,n}) + c_3 q_{2,n}] \quad (8.12.39b)$$

Equations (8.12.39) are straightforwardly solved iteratively.

Explicit Method 2: Adams–Bashforth (AB2). From Section 5.4, the Adams–Bashforth second-order formula leads to

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{h}{2}(3\dot{\mathbf{q}}_n - \dot{\mathbf{q}}_{n-1}) \quad (8.12.40)$$

which, upon using (8.12.33), takes the form

$$q_{1,n+1} = q_{1,n} + \frac{h}{2}(3q_{2,n} - q_{2,n-1}) \quad (8.12.41a)$$

$$q_{2,n+1} = q_{2,n} + \frac{h}{2}\{3A \sin(\theta_n + c_1 q_{2,n} + c_2 q_{1,n}) + 3c_1 q_{2,n} - A \sin(\theta_{n-1} + c_1 q_{2,n-1} + c_2 q_{1,n-1}) - c_3 q_{2,n-1}\} \quad (8.12.41b)$$

Again, the latter equations are straightforwardly solved by iterative means.

Implicit Method 1: Backward Euler (BE). The BE integration formula specifies the following relationship:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\dot{\mathbf{q}}_{n+1} \quad (8.12.42)$$

Substituting (8.12.33) yields

$$q_{1,n+1} = q_{1,n} + hq_{2,n+1} \quad (8.12.43a)$$

$$q_{2,n+1} = q_{2,n} + h[A \sin(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) + c_3 q_{2,n+1}] \quad (8.12.43b)$$

To solve these implicit equations, we use the Newton–Raphson method. As discussed in Section 5.4, we first form the functions F_1, F_2 as follows:

$$F_1(q_{1,n+1}, q_{2,n+1}) = q_{1,n+1} - q_{1,n} - hq_{2,n+1} = 0 \quad (8.12.44a)$$

$$\begin{aligned} F_2(q_{1,n+1}, q_{2,n+1}) &= q_{2,n+1} - q_{2,n} \\ &\quad - h[A \sin(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) + c_3 q_{2,n+1}] \\ &= 0 \end{aligned} \quad (8.12.44b)$$

The elements of the Jacobian are given by

$$\begin{aligned}\frac{\partial F_1}{\partial q_{1,n+1}} &= 1, & \frac{\partial F_1}{\partial q_{2,n+1}} &= -h \\ \frac{\partial F_2}{\partial q_{1,n+1}} &= -c_2 h A \cos(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) \\ \frac{\partial F_2}{\partial q_{2,n+1}} &= 1 - h[A c_1 \cos(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) + c_3]\end{aligned}$$

These elements would be used in the recursion shown in Section 5.4.

Implicit Method 2: Trapezoidal Integration (TR). From Section 5.4, this method is defined by

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{h}{2}(\dot{\mathbf{q}}_{n+1} + \dot{\mathbf{q}}_n) \quad (8.12.45)$$

Substituting (8.12.33) leads to

$$q_{1,n+1} = q_{1,n} + \frac{h}{2}(q_{2,n+1} + q_{2,n}) \quad (8.12.46a)$$

$$\begin{aligned}q_{2,n+1} &= q_{2,n} + \frac{h}{2} \{ [A \sin(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) + c_3 q_{2,n+1}] \\ &\quad + [A \sin(\theta_n + c_1 q_{2,n} + c_2 q_{1,n}) + c_3 q_{2,n}] \} \quad (8.12.46b)\end{aligned}$$

Again, toward solution by the NR technique, we form F_1, F_2 such that

$$F_1(q_{1,n+1}, q_{2,n+1}) = q_{1,n+1} - q_{1,n} - \frac{h}{2}(q_{2,n+1} + q_{2,n}) \quad (8.12.47a)$$

$$\begin{aligned}F_2(q_{1,n+1}, q_{2,n+1}) &= q_{2,n+1} - q_{2,n} \\ &\quad - \frac{h}{2} \{ [A \sin(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) + c_3 q_{2,n+1}] \\ &\quad + [A \sin(\theta_n + c_1 q_{2,n} + c_2 q_{1,n}) + c_3 q_{2,n}] \} \quad (8.12.47b)\end{aligned}$$

The Jacobian terms are therefore given by

$$\begin{aligned}\frac{\partial F_1}{\partial q_{1,n+1}} &= 1, & \frac{\partial F_1}{\partial q_{2,n+1}} &= -\frac{h}{2} \\ \frac{\partial F_2}{\partial q_{1,n+1}} &= -\frac{h}{2} A c_2 \cos(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) \\ \frac{\partial F_2}{\partial q_{2,n+1}} &= 1 - \frac{h}{2} [A c_1 \cos(\theta_{n+1} + c_1 q_{2,n+1} + c_2 q_{1,n+1}) + c_3]\end{aligned}$$

The Jacobian is inserted into the NR iteration to provide a solution.

We used TR to compute solutions for the second-order PLL for a number of examples shown in Figures 8.77-8.81; the solutions are indicated as SA (for stand-alone). The other curves arise from assembled models, discussed in the next subsection.

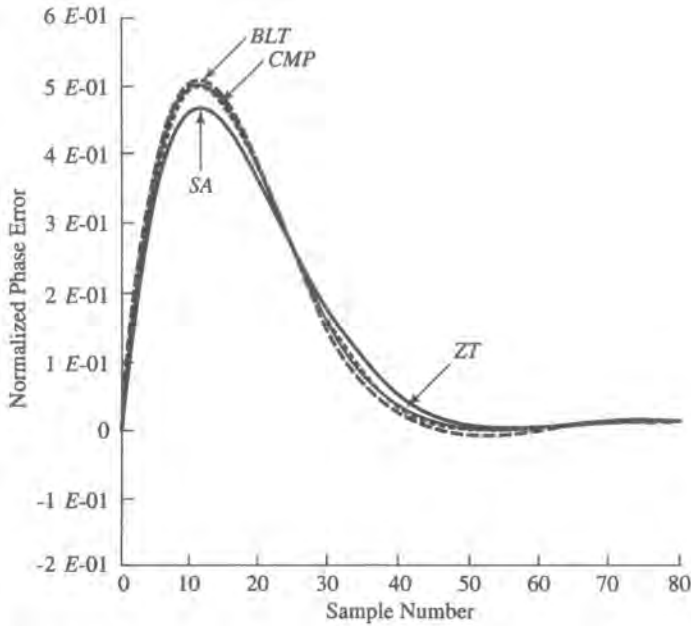


Figure 8.77. Comparison of some digital realization techniques for an assembled model; SA is the stand-alone model solution. Loop parameters: $\pi/4$ frequency step input; passive loop filter; **loop gain** = 50 ($A = 1$); $\omega_n = 1.0$; $\zeta = 0.707$; $\omega_{VCO} = 0.0$; $\hat{\theta}(0) = 0.0$; BLT, bilinear transformation on $F(s)$; ZT, z transform on $F(s)$; CMP, z transform on $(1/s)F(s)$; bilinear and z transform on $F(s)$ followed by trapezoidal integration for VCO; $T_s = 0.1$.

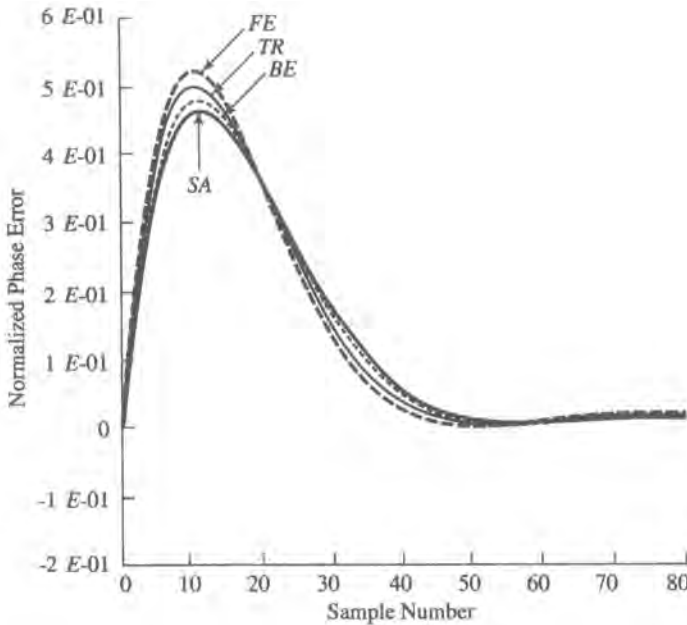


Figure 8.78. Further comparison of digital realization techniques for an assembled model. Loop parameters: $\pi/4$ frequency step input; passive loop filter (z transform); **loop gain** = 50 ($A = 1$); $\omega_n = 1.0$; $\zeta = 0.707$; $T_s = 0.1$. Integration formula for VCO: TR, trapezoidal; FE, forward Euler; BE, backward Euler.

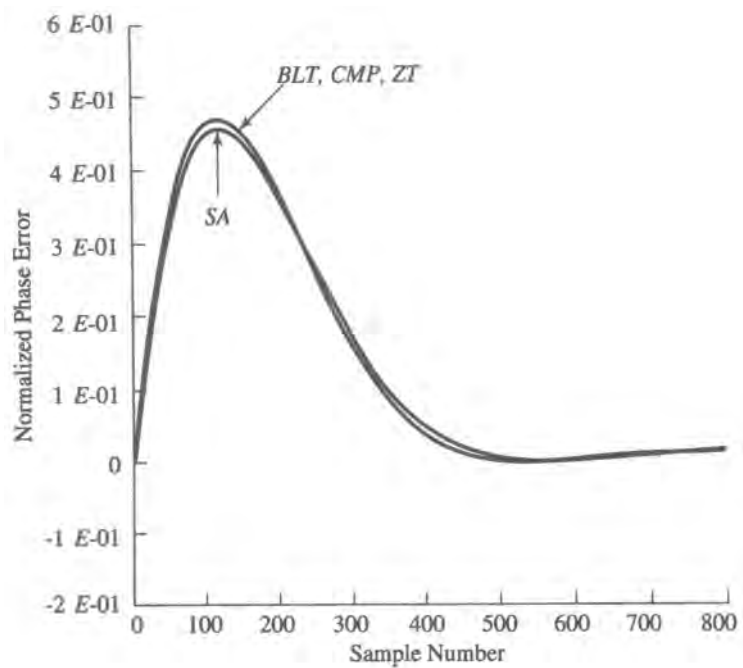


Figure 8.79. Phase error versus number of samples for $\omega_n T_s = 0.01$ (same conditions as Figure 8.77).

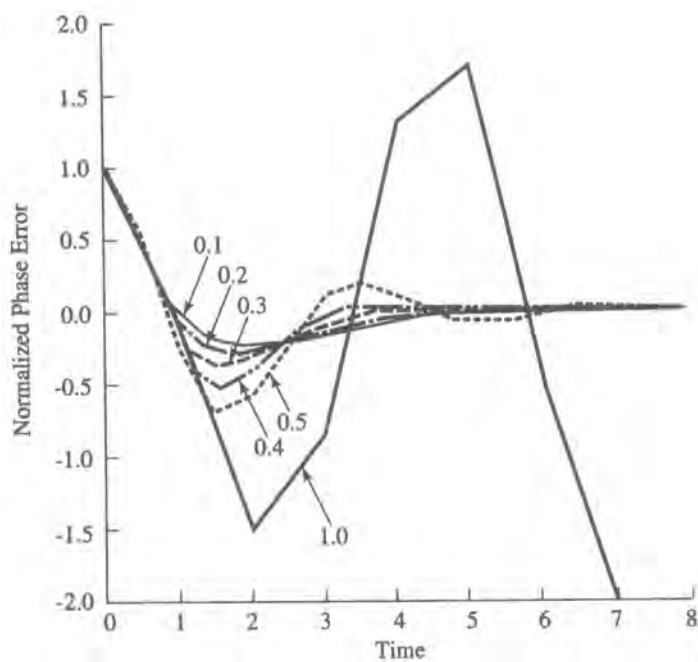


Figure 8.80. Phase error response to $\pi/4$ phase step input for different values of $\omega_n T_s$. Loop parameters: gain = 50; $\omega_n = 1.0$; $\zeta = 0.707$. Modeling choices: active loop filter; trapezoidal integration for VCO.

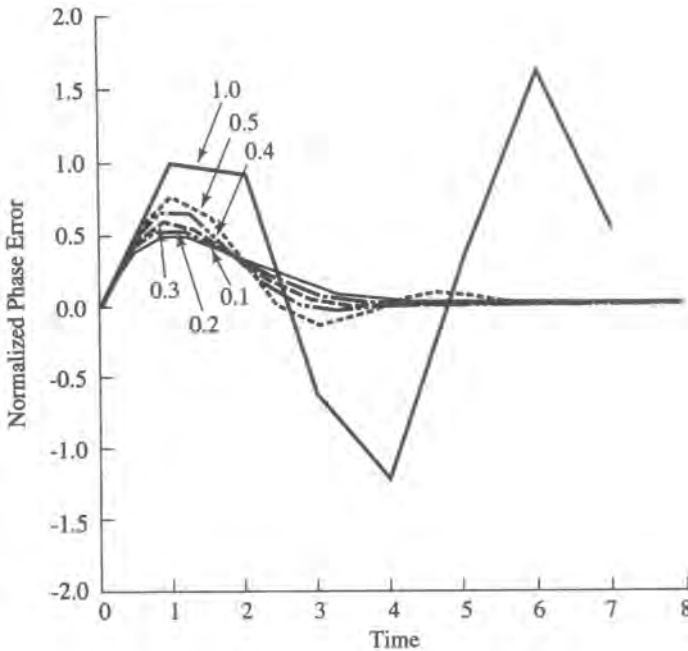


Figure 8.81. Phase error response to frequency step for different values of $\omega_n T_s$ (same conditions as Figure 8.80).

8.12.8.3. Assembled PLL Model

In the assembled model we represent the individual physical blocks as distinct entities. It is important to realize that (a) different modeling choices for each block may be available in a given library, (b) consequently, there may be a number of combinations of these choices resulting in a structure as a whole, and (c) each combination will have somewhat different properties. We remark that although each combination is effectively a digital version of an analog PLL, this is not generally equivalent to emulating a digital PLL, although there are points in common.

To illustrate the points (a)–(c) made above we list in Table 8.6 some of the possible modeling options for the loop filter, for the VCO, and for their combination.

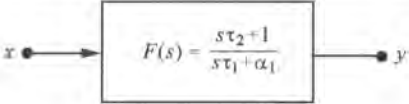
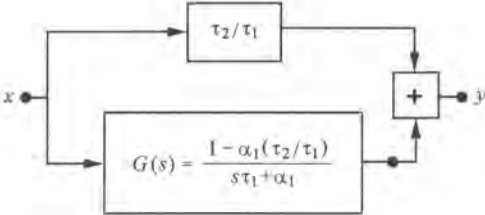
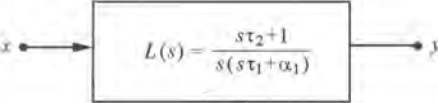
Order of Computation. One of the significant features of an assembled model is that a delay in the feedback branch is necessary, either explicitly or implicitly, depending on the specific models used for the individual blocks. We alluded to this facet briefly in the discussion of nonlinear differential equations in Section 5.4. Since delay in a feedback loop alters its dynamics and is a potential source of instability, such delay must be treated carefully. We elaborate a bit more on this point below.

The nonlinear differential equation governing PLL behavior is given, in operational form, by

$$\hat{\theta}(t) = AK \frac{F(p)}{p} \sin[\theta(t) - \hat{\theta}(t)] \quad (8.12.48)$$

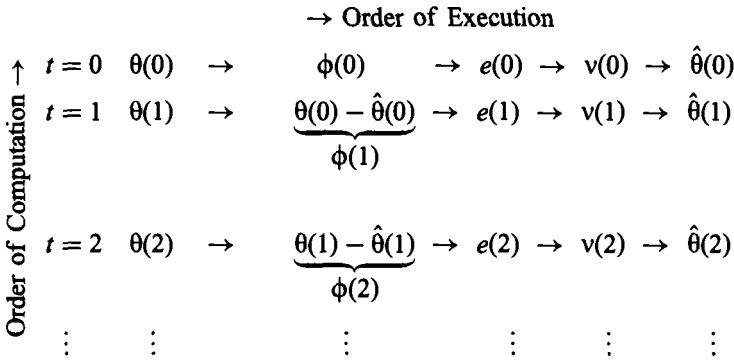
where $\theta(t)$ is the input phase and $\hat{\theta}(t)$ is its estimate returned by the loop. Equation (8.12.48) is an implicit relation: we cannot solve for $\hat{\theta}$ on the left-hand side without knowing its value in

Table 8.6 Different Modeling Options (Digital Realizations) for Phase-Locked Loop Elements

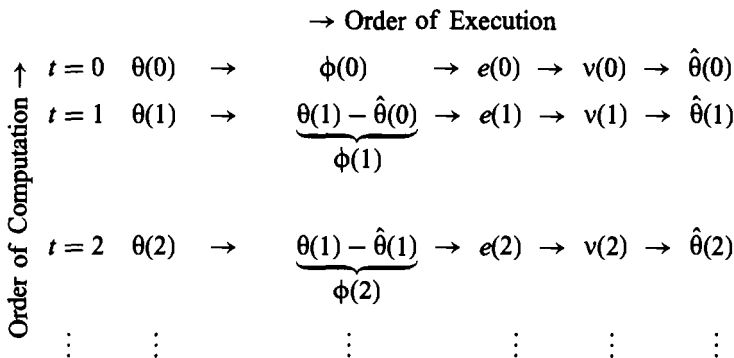
Loop filter	Structural representation	Difference equation
Passive or active loop filter; bilinear z transform		$y(n) = a_0 x(n) + a_1 x(n-1) - b_1 y(n-1)$ $a_0 = \frac{T_s + 2\tau_2}{\alpha_1 T_s + 2\tau_1}, \quad a_1 = \frac{T_s - 2\tau_2}{\alpha_1 T_s + 2\tau_1}$ $b_1 = \frac{\alpha_1 T_s - 2\tau_1}{\alpha_1 T_s + 2\tau_1}$
Passive or active loop filter; z transform		$y(n) = (\tau_2/\tau_1)x(n) + g(n)$ $g(n) = (1 - \alpha_1 \tau_2/\tau_1)(1/\tau_1)x(n) + e^{-\alpha_1 T_s/\tau_1} g(n-1)$
VCO		(a) $y(n) = T_s x(n-1) + y(n-1)$ (Forward Euler) (b) $y(n) = T_s x(n) + y(n-1)$ (Backward Euler) (c) $y(n) = \frac{T_s}{2} [x(n) + x(n-1)] + y(n-1)$ (Trapezoidal)
Composite: loop filter/VCO cascade; z transform		$y(n) = T_s (\tau_2/\tau_1) x(n) + T_s [(1 - e^{-T_s/\tau_1}) - (\tau_2/\tau_1)] x(n-1) + 1 + (e^{-T_s/\tau_1}) y(n-1) - e^{-T_s/\tau_1} y(n-2)$

the right-hand side, and vice versa, and we cannot isolate $\hat{\theta}$ to make an explicit relation because of the nonlinearity $\sin(\cdot)$. In order for a discrete-time simulation to autonomously “home in” on a solution, it is necessary to devise an explicit form of (8.12.48), and this forces a delay, as we will show.

One way to see this is to write down the actual computations as they might progress in a logical sequence that we prescribe. In the grid below, we show the various quantities indicated in Figure 8.73b for the first few stages of computation:



For simplicity we show time in unit increments, and of course each value can thus be thought of as the index in a computational sequence, for example, the index in a DO or FOR loop. We call this indexing the order of computation. At each stage of computation (a fixed index) a certain set of calculations is performed in a certain order as indicated by the arrows; the latter we refer to as the order of execution. In the grid shown it is implied that at $t = 0^-$ there is a set of given initial conditions that are mutually consistent with the governing equations; we usually assume all values are equal to zero. In the method of computation shown, there is clearly a delay of one unit in the phase error. The computational order shown assumes that each execution at time n can depend only on values of variables known at time $n - 1$. However, since a source has no inputs, we could, without violating any laws, just as well assume that the output of a source at time n is known instantaneously. (Similarly, it is sensible to assume that the contents of a delay or storage element are known at the current tick of the clock, since such an element acts as a source.) Thus, a modified computational order could appear as follows:



Here it can be seen that, effectively, the phase estimate is delayed by one unit. The delay is implicit in the sense that it is embedded in the imposed computational order.

When constructing an assembled model from off-the-shelf building blocks, another type of problem can arise. Any individual block in a library is normally designed to be “free-standing,” i.e., its input/output relationship is not context-dependent. Such blocks are exemplified by those in Table 8.6. But when we connect such blocks, system-level constraints must be satisfied, and this may not occur automatically without some sort of consistency checker. To illustrate these remarks, consider the simulation of a PLL where the loop filter difference equation is given by (Table 8.6)

$$v(n) = a_0 e(n) + a_1 e(n-1) - b_1 v(n-1) \quad (8.12.49)$$

and the VCO difference equation is given by (trapezoidal integration)

$$\hat{\theta}(n) = \frac{T_s}{2} [v(n) + v(n-1)] + \hat{\theta}(n-1) \quad (8.12.50)$$

It can be seen that, since $e(n) = \sin[\theta(n) - \hat{\theta}(n)]$, Equation (8.12.50) represents an implicit relation. This leads to a computational deadlock which would be detected by the consistency checker (ordering algorithm). This deadlock can be broken by inserting a delay into the simulation block diagram after the VCO. The resulting order of computation and execution would be precisely the same as indicated in the previous grid. But superficially the block diagram would look different because of an explicit delay block. Notice that the deadlock would not have occurred had we chosen forward Euler (explicit) integration instead of trapezoidal (implicit) integration because the former has an inherent delay.

The preceding discussion shows that some care must be exercised to set up a proper computational procedure, and that this procedure is not independent of the modeling choice, that is, the method of deriving the difference equation for a particular device. In any case, a delay of one sampling interval, whether explicit or implied, is imposed on the simulated structure. The effect of such a delay is to distort the dynamics of the APLL and, in the worst case, to result in an unstable structure. It is difficult to give a general analysis of the effect of such a delay, but in the case of small delay we can show this effect in an approximate way if we assume the loop is in lock (small phase error); we will show this analysis in Section 8.12.8.6. Generally, the effect of delay can best be seen empirically, and we will show a number of examples.

Recall that the delay is caused both by the closed-loop nature of the device and its nonlinearity. In the steady-state locked condition, it is normally assumed that the phase error is small. In this case, (8.12.48) can be approximated by

$$\hat{\theta}(t) = AK \frac{F(p)}{p} [\theta(t) - \hat{\theta}(t)] \quad (8.12.51a)$$

or

$$\hat{\theta}(t) = \frac{AKF(p)}{p + AKF(p)} \theta(t) \quad (8.12.51b)$$

Equivalently, the loop acts as a filter with closed-loop transfer function

$$H(s) = \frac{AKF(s)}{s + AKF(s)} \quad (8.12.52)$$

Basically, we now have an explicit relation between the input and output of the loop, and we can derive a corresponding difference equation with no delay. However, if we simulate a PLL structure under the assumption that (8.12.51) applies, we will replicate only the locked behavior (basically the noisiness of the phase estimate), and we cannot reproduce transient phenomena such as cycle slips. One might conceive of the possibility to sense the locked condition and switch from one difference equation (representing the transient condition) to another (representing steady state). However, software to accomplish this has to be fairly “intelligent” and may be difficult to implement.

Sampling Rate Considerations. One of the important considerations in simulation is, of course, what the sampling rate should be. Because the PLL is a nonlinear closed-loop device, special care is required in this respect. We have discussed in Chapter 5 the fact that nonlinearities increase bandwidth to some extent, and hence require correspondingly greater sampling rates. As we have just seen, the combination of the nonlinear and feedback properties of the loop induces a delay of one sampling interval in the simulation of the assembled PLL. Hence, aside from fidelity considerations, the choice of T_s affects the dynamics and stability of the loop. A good value of T_s can be tricky to find, and a good value for one set of parameters may not hold for another set. We will present shortly some simulated results to impart some appreciation of the influence of T_s . Before we present these results, however, it is instructive to show that time can be normalized with respect to the natural frequency of the loop. This indicates (as might be expected) that in an absolute sense the “bandwidth” of the loop is proportional to ω_n .

Consider the differential equation (8.12.48), which for the specific loop filter given in (8.12.22)–(8.12.23) can be expanded as

$$\ddot{\phi} + \frac{\alpha_1 + AK\tau_2 \cos \phi}{\tau_1} \dot{\phi} + \frac{AK}{\tau_1} \sin \phi = \ddot{\theta} + \frac{\alpha_1}{\tau_1} \dot{\theta} - \frac{\alpha_1}{K_2\tau_1} \omega_0 \quad (8.12.53a)$$

where ω_0 is the loop detuning (initial frequency difference between the VCO frequency and the incoming carrier frequency), and dots represent derivative with respect to time. The transformation $u = \omega_n t = (AK/\tau_1)^{1/2} t$ yields

$$\phi'' + \frac{\alpha_1 + AK\tau_2 \cos \phi}{\omega_n \tau_1} \phi' + \sin \phi = \theta'' + \frac{1}{\omega_n \tau_1} \theta' - \frac{\alpha_1}{AK_2 K} \omega_0 \quad (8.12.53b)$$

where the primes represent differentiation with respect to u , which demonstrates the scaling in question. That is, the value of $\phi(t)$ at time t is the same as that of $\theta(u)$ at normalized time $u = \omega_n t$. This allows one to normalize discrete time in increments of $\omega_n T_s$.

The effect of T_s both on stability and accuracy illustrated in the next example.

■ *Example 8.12.1.* In this example we illustrate some of the points made above concerning modeling choices and the effect of sampling interval on accuracy and stability.

We simulated a second-order PLL using both the stand-alone and assembled model methodologies. In the next three figures the label SA indicates a “stand-alone” solution, which we produced only for the trapezoidal integration case. Figures 8.77 and 8.78 illustrate the different behaviors that occur with the different modeling options shown in Table 8.6. In particular, the figures show the response of the loop (the phase error) to a frequency step input $\theta = (\pi/4)t$ versus time (sample number) for the conditions shown in the captions. The sample spacing $T_s = 0.1$ is normalized with respect to the natural frequency of the loop ω_n .

Figure 8.77 shows a comparison of results for several digital realization techniques, as

indicated in the figure. The difference equations for these various realizations are obtained from Table 8.6. Figure 8.78 shows a related set of results where the comparison is made only among different integration formulas that might be used to represent the VCO, as listed in Table 8.6.

The main point of these two figures is not to determine what is the best assembling technique. In fact, we see there is not a uniformly best one. The point, rather, is that all modeling approaches represent different kinds of approximations and that one should validate one's choices as satisfying a certain degree of fidelity. Taking the SA solution as the "correct" one, we see that the other combinations all give a very good approximation. As we might expect, these approximations improve to the point of indistinguishability as T_s becomes very small, as illustrated in Figure 8.79.

We can expect the opposite effect if T_s becomes larger. Indeed, as discussed earlier, if T_s increases sufficiently, stability becomes the issue, rather than accuracy. Figures 8.80 and 8.81 illustrate this situation for two different cases (in both figures the number of steps is equal to the abscissa divided by $\omega_n T_s$). What is clear is that a fairly small value of $\omega_n T_s$ (say, not larger than between 0.1 and 0.2) is necessary for a well-behaved response, and it is obvious that when $\omega_n T_s$ is large enough the simulated system becomes unstable. Depending upon how close we want to match the continuous response, we might have to use a considerably smaller value, say 0.01–0.05. If a PLL is explicitly embedded within a system simulation, the typical ratio of information rate to loop bandwidth is such that the sampling rate required to avoid aliasing of the signal will be a very small fraction of ω_n , perhaps on the order of 0.01–0.001, so that stability considerations need not enter. ■

The classical PLL offers the possibility of stand-alone or assembled modeling approaches. While the stand-alone approach is theoretically superior, it is also aided by the relative tractability of the corresponding NLDE. Certain tracking structures which are now common in many contemporary systems are considerably more mathematically complex, for example, structures that perform simultaneous carrier and symbol acquisition, or that use decision feedback internally. For such subsystems, simulation appears to offer at least a more manageable alternative. The "modified" Costas loop is an example of such a structure. The term refers to a modification of the original Costas loop to include decisioning within the loop, and consequently the addition of symbol synchronization as well.

■ *Example 8.12.2.* A version of the modified Costas loop is shown in Figure 8.82. The input filter F_i is a six-pole Butterworth with IF (3-dB) bandwidth of $1.3R$, where R is the symbol rate. The arm filters F_a are two-pole Butterworth filters with 3-dB frequencies equal to $0.55R$. The clock filters F_s are simulated as actual bandpass filters, even though the rest of the structure is simulated in the lowpass-equivalent domain. The center frequency of F_s is R . Some care must be taken with simulating F_s since its bandwidth is very small with respect to the data rate, hence its impulse response is very long. The delays τ shown are necessary to account for the delay in the decision loop. The VCO's transfer function $1/s$ in continuous time is approximated with trapezoidal integration, and the loop filter $F(s) = (\sigma\tau_2 + 1)/\sigma\tau_1$. The parameters are chosen so that the overall loop response is equivalent to one having $\zeta = 0.707$, and $B_L = 0.0065R$.

In the simulation results shown below, 16 samples per bit were used. Note that while the loop behaves as a very narrow bandwidth filter relative to the symbol rate, almost everywhere in the loop we have to deal with the actual information waveform. Therefore, there is essentially no opportunity to take advantage of multirate sampling.

Some illustrative simulation results are given in Figures 8.83–8.85. These results are obtained without noise, the presence of which would, of course, alter the appearance of the curves. The loop was initially detuned, as can be seen in the figures. The frequency axis is normalized with respect to R . Thus, the initial frequency offset was about $18 \times 10^{-3}R$.

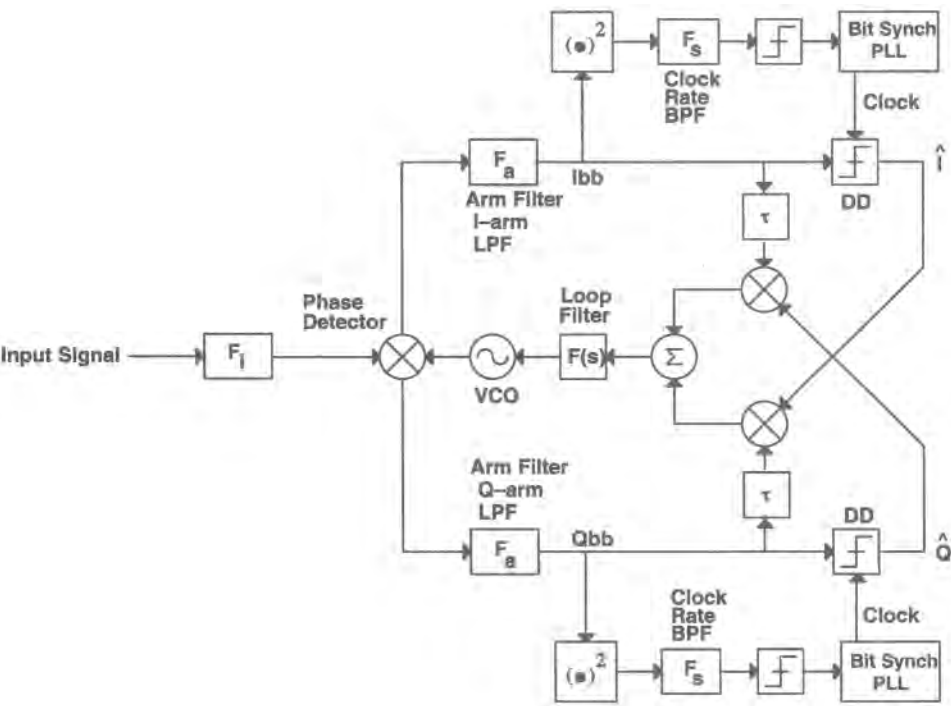


Figure 8.82. Block diagram for “modified” Costas loop.

Figures 8.83 and 8.84 show, respectively, the evolution of the phase and frequency from the initial application of the signal, which is an offset QPSK carrier. Figure 8.85 displays a “classic” phase plane plot. The beginning of the traces is at the point marked O (origin), from which the trajectory moves rightward until it hits the rightmost boundary π , then flips to $-\pi$ at the same frequency, and continues in the same manner until it achieves steady-state at the point marked D (destination). The trajectory of Figure 8.8S is similar to, but differs somewhat from, the “classic” ones for a PLL tracking a sinusoid (as in Ref. 99, for example), the main qualitative difference being the raggedness due to “data noise” induced by circulating the signal itself in the loop. ■

8.12.8.4. The Phase-Locked Loop as a Phase Tracker

One of the most important functions of the PLL is as a “phase tracker,” that is, to estimate the phase of an incoming digitally modulated carrier in order to effect (almost) coherent demodulation. Unless there is an auxiliary carrier or an unmodulated carrier component, the loop must be preceded by some nonlinearity, e.g., an m th-power device (Section 8.12.6). We can of course simulate the loop simply by assuming the form of input that we wish to study or expect to have. However, if we want to simulate the entire carrier recovery operation, the M th-power device must also be simulated. Since this device operates on the modulated carrier, as mentioned earlier, the sampling rate that is consistent with the information rate may be significantly larger than the one required for the loop.

Probably the most convenient feature of simulation with respect to (assembled) feedback loops is the ability to quickly change a design parameter and observe the corresponding effect

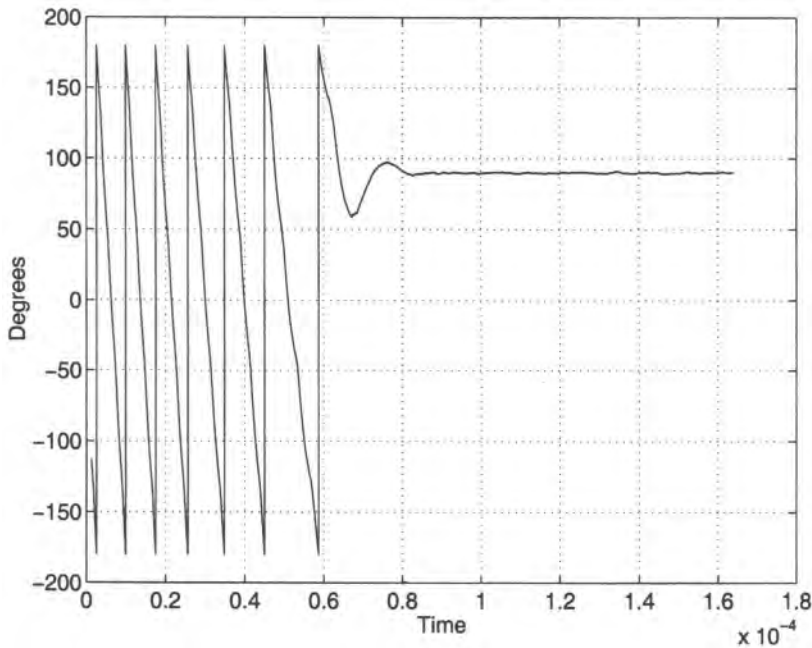


Figure 8.83. Phase estimate of modified Costas loop for Example 8.12.2.

on the transient response, in the manner illustrated in Figure 8.86. These simulations were carried out using the “phase-domain” representation in Figure 8.73b, which is essentially the same as the lowpass equivalent. One of the advantages of this representation is that we do not have to simulate explicitly a lowpass filter to reject the double-frequency term out of the phase detector. Although we have not done so here, noise can easily be introduced at the PLL input to observe its effect on the performance measures of interest.

8.12.8.5. The Phase-Locked Loop as an FM Demodulator

Another important application of the PLL is as a demodulator for frequency-modulated (FM) signals (Section 8.8). PLL demodulators are widely used in today’s communication systems because of their superior performance, ease of alignment, and ease of implementation using inexpensive integrated circuits. The significant distinction between this application and the previous one considered is that in the current case the loop is intended to follow the modulation, whereas in the phase-tracking application it is the carrier exclusive of the modulation that we wish to follow. Heuristically, we might expect the FM demodulator application to be more demanding in some sense because the modulation is a dynamic process, whereas the phase is relatively slowly varying. In fact, we can think of this case as one closely related to the input step-frequency transient illustrated in some of the earlier figures. The FM input can be thought of as a continuous series of input frequency changes with infinitesimal step size. If the frequency changes at a rate that is slower than the time required for the PLL to establish lock, then we can expect that the loop will be able to track the FM signal.

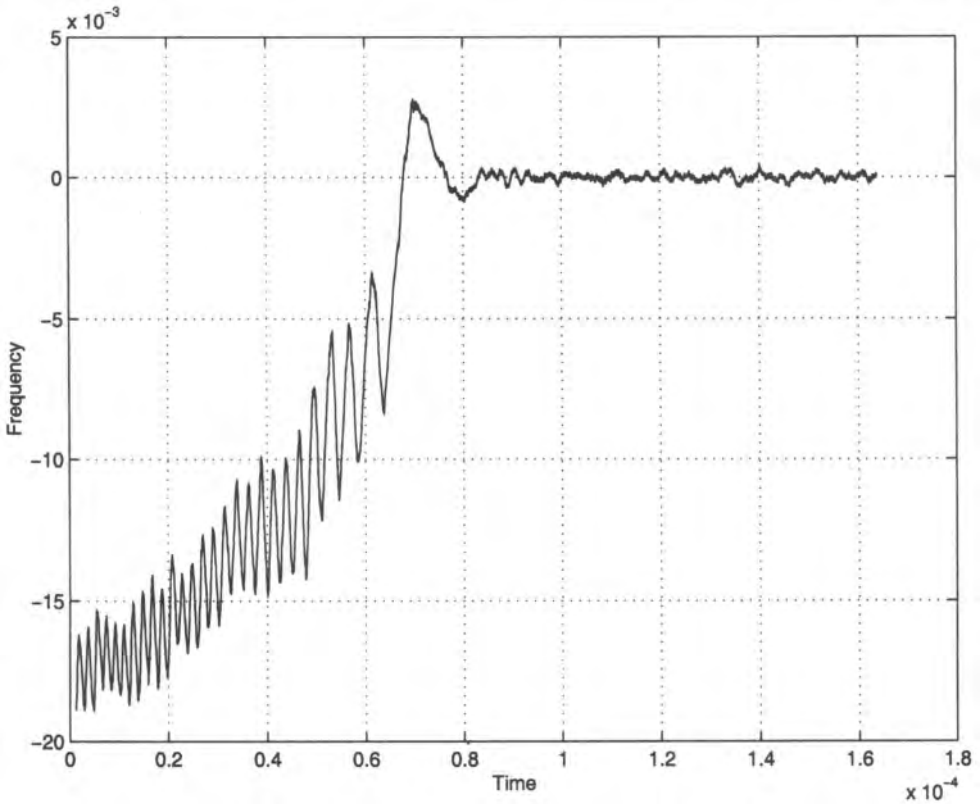


Figure 8.84. Frequency estimate of modified Costas loop for Example 8.12.2.

If we label the modulating signal $m(t)$, then the modulated carrier is

$$S(t) = A \cos \left[\omega_c t + 2\pi \Delta f \int m(\tau) d\tau \right] \quad (8.12.54)$$

Usually $|m(t)|$ is normalized to a maximum of unity, and then Δf is the peak frequency deviation. The input phase, exclusive of the carrier, is therefore

$$\theta(t) = 2\pi \Delta f \int m(\tau) d\tau \quad (8.12.55)$$

That the PLL can act as an FM demodulator can be seen heuristically from the fact that under the desired operating conditions, the phase error is small, i.e., $\hat{\theta}(t) \approx \theta(t)$. But the VCO output is given by

$$\hat{\theta}(t) = K_2 \int v(t) dt$$

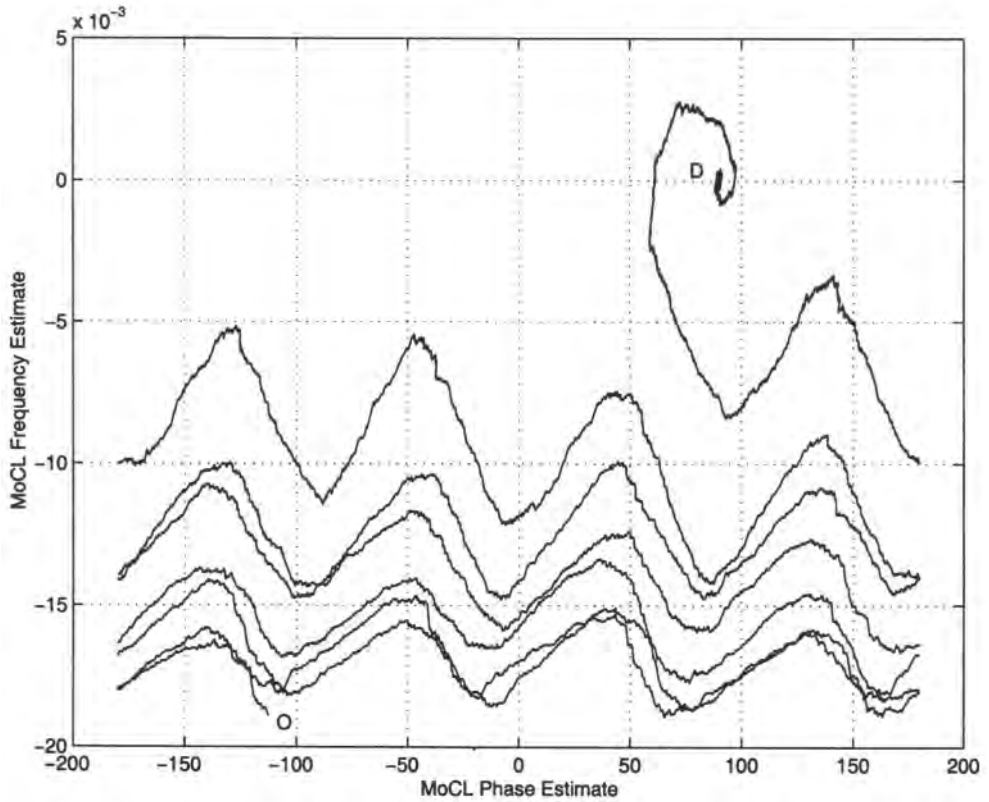


Figure 8.85. Phase plane trajectory for modified Costas loop of Example 8.12.2.

so that

$$\frac{d\hat{\theta}(t)}{dt} = K_2 v(t) \approx \frac{d\theta(t)}{dt} = 2\pi\Delta f m(t) \quad (8.12.56)$$

which means $v(t)$ is proportional to $m(t)$. Hence the output for FM demodulation, $v(t)$, is the output of the loop filter, which is the input to the VCO. Thus, to find the FM signal we observe the loop at a particular point, but the loop functions in the same way as before (for the phase-tracking application). We note that one advantage of the assembled model is that all internal points of the loop can be examined simultaneously at no additional cost. The stand-alone model for an FM detector, however, must be a separate model from the stand-alone phase-tracker model.

8.12.8.6. Effect of Delay on the Performance of the Assembled PLL Model

If we assume that the loop is in lock, with small phase error, we can show analytically the (approximate) effect of the computationally imposed delay. With small phase error we can assume linear operation, so that, from (8.12.52),

$$\frac{\hat{\theta}(s)}{\theta(s)} = \frac{AKF(s)}{s + AKF(s)} \quad (8.12.57)$$

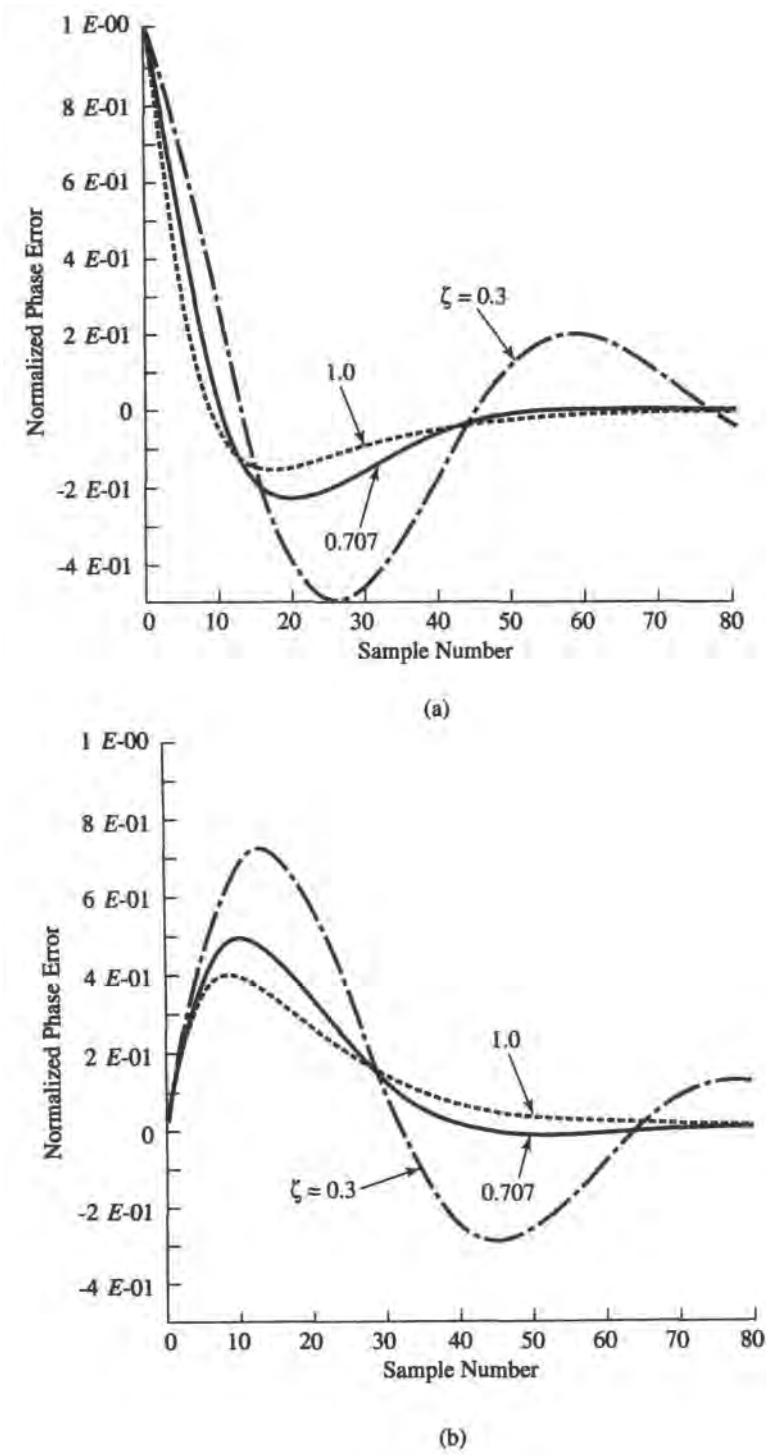
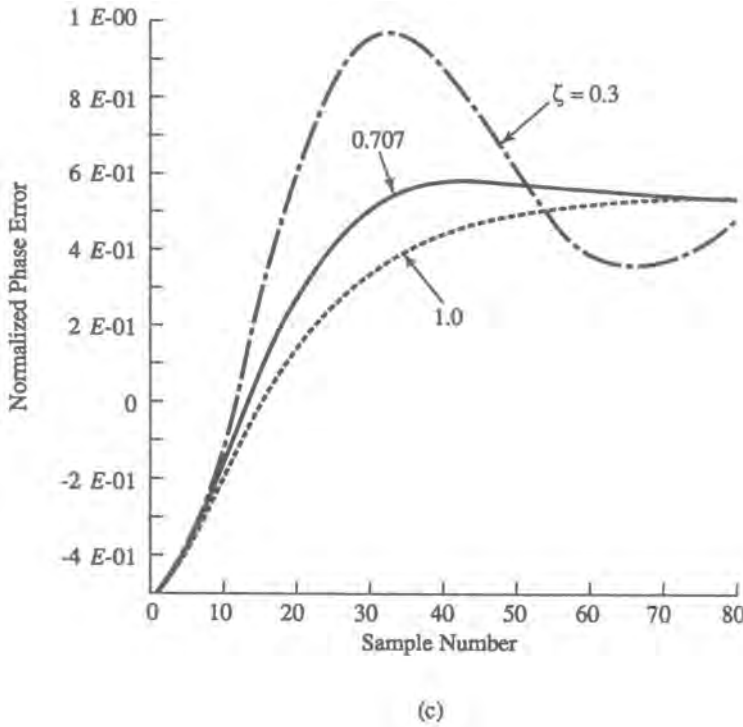


Figure 8.86. Typical PLL transient response as a function of damping factor, (a) Phase step of $\pi/4$. Loop parameters: gain = 50; $\omega_n = 1.0$. Modeling choices: active loop filter, trapezoidal integration; $\omega_n T_s = 0.1$. (b) Frequency step of $(\pi/4)t$.

Figure 8.86. (c) Frequency ramp $(\pi/16)t^2$.

If we insert a unit delay element after the VCO, its transfer function is $\exp(-sT_s)$, which for small T_s can be approximated by $1 - sT_s$. This modifies (8.12.57) to

$$\frac{\hat{\theta}(s)}{\theta(s)} = \frac{AKF(s)}{s + AKF(s)(1 - sT_s)}$$

and if we substitute the loop filter (8.12.22) for $F(s)$, we find

$$\frac{\hat{\theta}(s)}{\theta(s)} = \frac{AK(s\tau_2 + 1)/(\tau_1 - AK\tau_2 T_s)}{s^2 + s \frac{AK(\tau_2 - T_s) + \alpha_1}{\tau_1 - AK\tau_2 T_s} + \frac{AK}{\tau_1 - AK\tau_2 T_s}} \quad (8.12.58)$$

The denominator is usually written in the form

$$D(s) = s^2 + 2\zeta\omega_n s + \omega_n^2 \quad (8.12.59)$$

where (without delay) $\omega_n^2 = AK/\tau_1$ and $\zeta = \frac{1}{2}\omega_n\tau_2(1 + \alpha_1/AK\tau_2)$. We can see, therefore, that in the case of (8.12.58), the same form as (8.12.59) applies if we set

$$D(s) = s^2 + 2\hat{\zeta}\hat{\omega}_n s + \hat{\omega}_n^2 \quad (8.12.60a)$$

where

$$\hat{\omega}_n^2 = \frac{AK}{\tau_1 - AK\tau_2 T_s} = \frac{\omega_n^2}{1 - \omega_n^2 \tau_2 T_s} \quad (8.12.60b)$$

$$\hat{\xi} = \frac{1}{2} \hat{\omega}_n \tau_2 [(1 - T_s/\tau_2) + \alpha_1/AK\tau_2] \quad (8.12.60c)$$

Thus, the delay clearly affects the roots of $D(s)$, and therefore its transient dynamics and stability. Note that these effects apply whether we use the loop for phase tracking or for demodulation. An idea of the distorting effect of the delay can be seen from the following computation, assuming $\tau_1 = 0.1$, $\tau_2 = 0.04$, $\omega_n = 10$, and $AK = 10$:

T_s	$\hat{\omega}_n$	$\hat{\xi}$
0	10.0	0.7
0.001	10.02	0.696
0.005	10.1	0.682
0.01	10.2	0.663
0.02	10.426	0.625

where, of course, the values of $\hat{\omega}_n$, $\hat{\xi}$ for $T_s = 0$ are the true values. Depending on the required fidelity, it can be seen that a significantly large sampling rate may be required.

8.13. Calibration of Simulations

8.13.1. Introduction

The measurement of performance parameters, or indeed any system attribute, implicitly requires knowledge of the conditions under which the measurement was made in order to interpret it properly. Calibration is the term for the process of determining these conditions. For example, the measurement of BER is meaningful only when accompanied by the corresponding E_b/N_0 . Thus, we need to “calibrate” E_b/N_0 ; in plain words, we need to know what it is. The value (or, generally, the set of values) of E_b/N_0 is thus one of the conditions of the measurement. In this particular situation, the value of E_b/N_0 is normally preset by the simulation practitioner through some sort of procedure discussed later. In some cases, the conditions of interest may be determined as the simulation proceeds (“on the fly”) or after the simulation is terminated (postprocessing).

In the BER case, the value of E_b/N_0 is just one of the conditions that influences it. For example, the performance of the symbol synchronizer, say its rms timing jitter, is also an important factor in fixing the actual value of the BER. Here, depending on our model of the symbol synchronizer, the rms jitter may again be a quantity that is preset by the practitioner, or it may be a consequence of other conditions and it might be measured during or after the simulation.

Continuing with our example, it is self-evident that the BER depends on *every* design feature and parameter of a system. Thus, the conditions under which the BER is measured include filter bandwidths, amplifier characteristics, code rate, etc. Clearly, these kinds of design elements are givens of the problem and are not “conditions” to be determined in a calibration procedure. We may, however, look upon calibration as an implicit part of the modeling of communication systems, as it is the process that reports or sets the values of

certain parameters of models. Additionally, the calibration process may also be viewed as the modeling, proper, of the real instrumentation that may be used to measure the conditions of interest. It is clearly important to understand the relationship between the hardware implementation and the calibration procedures used in simulation when comparing simulated and measured results. The reader is also referred back to Chapter 2 for a related qualitative discussion on the differences in the type of error between simulation and measurements.

The parameters or conditions that we are typically concerned with calibrating, whether preset or measured, are signal levels, noise levels, interference levels, phase and timing references (delays and phase offsets), timing jitter and phase jitter, and biases in general. Depending on the problem and the models we use, the quantities we define may be simply fixed values, average values, rms values, or distributions. We have in fact already discussed (or will in later chapters) a number of procedures that implicitly are calibration. In previous sections of this chapter, we discussed some possible ways of establishing preset values of phase and timing references and their variability (jitters). In Chapter 10 we will present a different procedure also for obtaining either preset values of timing and phase reference or for determining such values on a sliding-window basis. Some of these simulation procedures may be similar to hardware calibration techniques. For example, the hard-wired references have their counterpart in hardware procedures. Of course, in most operational systems the timing and phase references are generated on the fly by synchronization structures and are not preset. If we model these structures directly, calibration can only be taken in the sense of monitoring important parameters as they unfold, or by postprocessing.

In Chapter 10 we will also look at the estimation of the average level of a waveform and of its average power. Our focus there is on the statistical properties, and the measurement of these quantities can also be thought of as *post hoc* calibration. Another implicit calibration procedure was discussed in Chapter 5 in conjunction with setting the operating point of an amplifier either with a test signal or via an AGC.

In the subsection following, we focus on a calibration aspect not discussed elsewhere, namely the establishment of the *ratio* of signal and noise (properly defined) in the context of digital transmission.

8.13.2. Calibration of Signal-to-Noise Ratio or E_b/N_0 for Digital Signaling

As mentioned, we develop here procedures to calibrate the signal and noise levels with the aim, in particular, of establishing some measure of their ratio. These procedures are typically used to preset this quantity prior to a simulation run, and would thus be established in separate (usually short) simulation runs, or in some cases can be precomputed without any simulation at all. The measure of interest is generically called a “signal-to-noise ratio” (SNR), and more usually in this context E_b/N_0 . This measure captures strictly the power content that may be present, and is to be distinguished from the situation where waveform fidelity is the performance criterion of interest. In this case, “signal-to-noise ratio” has a more specialized meaning, which is studied in Chapter 11.

8.13.2.1. Signal Power Level

Consider a bandpass signal

$$Z(t) = A(t) \cos(\omega_c t) + B(t) \sin(\omega_c t) \quad (8.13.1)$$

at a certain point in the system, which may be at the input to a receiver. Its average power is given by

$$P_Z = \lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} Z^2(t) dt = \lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} \frac{1}{2} [A^2(t) + B^2(t)] dt \quad (8.13.2)$$

Since we typically deal with the complex envelope, (8.13.2) is equivalent to

$$P_Z = \lim_{T_0 \rightarrow \infty} \frac{1}{2T_0} \int_{-T_0}^{T_0} |\tilde{Z}(t)|^2 dt \quad (8.13.3)$$

In the simulation version, we can only approximate P_Z both because we must deal in discrete time and the observation interval is finite,

$$P_Z \approx \frac{1}{2N} \sum_{n=1}^N |\tilde{Z}(nT_s)|^2 \quad (8.13.4)$$

where N is the number of waveform samples. If the transmitted signal carries k bits per signaling interval T , the energy per symbol E_s and the energy per bit E_b are simply given by

$$E_b = kE_s = kP_Z T \quad (8.13.5)$$

There are different philosophies in choosing the signal $Z(t)$ to be calibrated. One line of thought is to generate $Z(t)$ as close to an operational signal as possible. For example, $Z(t)$ may arrive from a previous hop in the system, where it may have been processed nonlinearly. In this case, $Z(t)$ is difficult to obtain except by direct simulation. For a statistically balanced signal, N does not have to be very large to obtain a good estimate of P_Z . In fact, if the system memory can be assumed to be finite and of moderate duration m (symbols), it is customary in this context to use a PN sequence as the transmitter. In that case, if N corresponds to a full period of the sequence, P_Z as obtained from (8.13.4) will be precisely correct, to within aliasing error. For an L -ary alphabet the PN sequence length is L^m , which may be an inconveniently large number. Another possible problem with this approach is that any measurement will necessarily reflect the particular design choices that have been made. Changes in the design will necessitate recalibration.

A simpler system model, if applicable, permits the calibration to be done analytically. Suppose that the modulating signals in (8.13.1) can be expressed as

$$A(t) = \sum A_n g(t - nT); \quad B(t) = \sum B_n g(t - nT) \quad (8.13.6)$$

where A_n, B_n are each drawn from some identical L -ary alphabet, and $g(t)$ is a basic pulse waveform. Then, it is known⁽¹¹⁾ that P_A , the average power in $A(t)$, is given by

$$P_A = \frac{E(A_n^2)}{T} \mathcal{E}_g \quad (8.13.7)$$

where

$$\mathcal{E}_g = \int_{-\infty}^{\infty} g^2(t) dt = \int_{-\infty}^{\infty} |G(f)|^2 df \quad (8.13.8)$$

is the pulse energy. An identical result holds for P_B , the average power of $B(t)$. Thus,

$$P_Z = \frac{E(A_n^2)}{T} \mathcal{E}_g \quad (8.13.9)$$

In many practical cases, (8.13.9) can be precomputed. For example, if $A_n \in \{\pm A, \pm 3A, \dots, \pm(L-1)A\}$, then $E(A_n^2) = A^2(L^2 - 1)/3$. Similarly, $\mathcal{E}_g$ can be found for many simple pulse shapes. The answer for a unit-amplitude rectangular pulse T seconds long is obvious, namely $\mathcal{E}_g = T$. As another pair of simple examples, it can be shown (Problem 8.28) that $\mathcal{E}_g = (1/R)(1 - \beta/2R)$ for the raised cosine pulse, where $R = T^{-1}$ and β is defined in (8.9.5), and $\mathcal{E}_g = 1$ for the square-root raised cosine pulse. Of course, E_b is still given by (8.13.5).

This last case is more convenient than the first one discussed, but is still dependent on design choices, namely the signaling alphabet and the pulse shape. A still more convenient choice, which is independent of implementation and has a fundamental implication, is to use an unmodulated carrier

$$Z(t) = A \cos \omega_c t$$

where

$$P_Z = \frac{1}{2} A^2 \quad (8.13.10)$$

Here, A can be set, for example, at the peak sustainable power of the output amplifier. In simulation this is determined merely by inspection of the amplifier gain (AM/AM) characteristic. The significance of (8.3.10) is that it is the maximum available power, and (assuming that number is fixed) provides an unchanging reference with respect to which BER curves can be displayed and properly compared. Better performance can be attributed to particular design choices without possible obfuscation due to changes in calibration. For a QAM signal, the power in (8.3.10) would be split between the two channels. Thus, $\frac{1}{4} A^2 T$ would represent the maximum energy per symbol per channel. For a constant-envelope signal, $\frac{1}{2} A^2 T$ would represent the energy per symbol.

We have been implicitly dealing with a test point where the signal is meaningfully viewed as a waveform, which is anywhere in the system prior to detection proper. If the signal is matched filtered, the output of interest is only the value sampled once per symbol. In this case, the average power in question should be the average of these symbol-spaced samples.

We should also note that for detection purposes we also need to calibrate the decision regions properly, which involves establishing *voltage* levels and/or angular regions corresponding to the different transmitted symbols. This calibration procedure is addressed in Problem 8.24.

8.13.2.2. Noise Power Level

We now consider setting the noise, or noise spectral power density, levels. The bandpass noise at the receiver input is given in the usual representation

$$N(t) = N_c(t) \cos \omega_c t - N_s(t) \sin \omega_c t \quad (8.13.11)$$

whose average power over any specified bandwidth B is given by

$$\begin{aligned} P_N &= \frac{1}{2} [P_{N_c} + P_{N_s}] \\ &= N_0 B \end{aligned} \quad (8.13.12)$$

where N_0 is the one-sided noise power spectral density (at RF or IF). It is convenient to assume that the bandpass noise has a finite (one-sided) bandwidth W which is much larger than the system bandwidth. $N_c(t)$ and $N_s(t)$ are equivalent lowpass processes with (two-sided) noise PSD also each equal to N_0 for $-W/2 \leq f \leq W/2$ (this is true for a particular value of the demodulator constant; see Problem 8.25).

In simulation we generate (8.3.11) in its lowpass-equivalent form, namely

$$\tilde{N}(t) = N_c(t) + jN_s(t) \quad (8.13.13)$$

Thus, in order to emulate (8.3.11), we need to generate two Gaussian lowpass sequences, each with two-sided PSD N_0 . We have seen (Chapter 7) that a random number generator (RNG) emitting samples spaced by T_s imitates “white” noise over $-f_s/2 \leq f \leq f_s/2$ with two-sided PSD v if we set the variance of the RNG to $v f_s$. Thus, for a particular number N_0 , we need only set the variances of the generators for $N_c(t)$ and $N_s(t)$, respectively, to $\sigma_c^2 = \sigma_s^2 = N_0 f_s$. The noise power at any test point is then simply $N_0 B_N$, where B_N is an appropriately defined noise bandwidth; this depends somewhat on where the test point is (see also Problem 8.26).

8.13.2.3. Calibrating Signal-to-Noise Ratio and E_b/N_0

It is now a simple matter to calibrate the ratio of signal to noise in either of the measures of interest. Suppose we are interested in setting the E_b/N_0 . This is a measure frequently calibrated at the input to the receiver, for, as alluded to before, it is then numerically independent of the design details following that point. So, let $\gamma = E_b/N_0$ be some number to which we want to set this parameter. Then,

$$N_0 = E_b/\gamma$$

and, taking E_b from (8.13.5),

$$N_0 = k P_Z T / \gamma \quad (8.13.14)$$

where k , T , and γ are all predetermined quantities and P_Z is also either precalculable or determined from the calibration run itself. We then calibrate the RNG by setting $\sigma_c^2 = \sigma_s^2 = N_0 f_s$, and the calibration of E_b/N_0 is complete.

To calibrate the signal-to-noise ratio, let $\rho =$ (signal power/noise power) at some test point of interest. The signal power we have named P_Z and again it is either precalculable or measured directly using one or another test signal. The main question here is whether to use T_s -spaced or T -spaced samples. The noise power at the same point is $N_0 B_N$, where B_N is measured between the point of insertion of the noise source and the test point (Problem 8.26). Two situations arise: in one, we may wish to fix the value of ρ at some point. In that case, we use

$$N_0 = \frac{P_Z}{\rho B_N} \quad (8.13.15)$$

to determine the necessary value of N_0 , which then allows us to set the variance of the RNG properly. In the second situation, we may set a value for $E_b/N_0 = \gamma$ and want to determine what the consequent value of ρ is at some point in the system. In this case we have

$$\rho = \frac{\gamma R}{k B_N} \quad (8.13.16)$$

where $R = T^{-1}$ is the symbol rate.

8.14. Summary

In this chapter we have discussed various aspects of the notion and process of the “modeling” of a communication system. A system can be considered to be composed of subsystems whose conventional representation is the usual block diagram. Each block may be called a subsystem, although the level of complexity in each block varies according to the application at hand. The important point is that the behavior of the system is represented by an input/output description of each subsystem. That description can be an equation, a lookup table, an algorithm, or any form appropriate to the subsystem at hand. In some cases it may be that the subsystem itself is best modeled by a further expansion in terms of its own blocks. However, for computational economy, we wish to model at the highest level of abstraction possible.

Generally, it is little more complex to use “realistic” models than idealized ones in the simulation context. This, of course, is one of the most useful features of simulation. Naturally, we first need to synthesize such a model. We have discussed one approach to obtaining such models, namely to identify the ways in which actual behavior may depart from the ideal and to capture such a departure parametrically. This methodology permits us to ascertain the performance of a system as a function of how it might be realized, specifically as a function of the values of specified parameters.

While we have given a fairly substantial sample of subsystem models, it is clear that a reasonably complete treatment would require a large volume in itself because of the large number of possible system elements and the variations in their implementations. These variations can have important modeling implications. It is important, therefore, that the simulation developer take account of a model’s application and its intended design and technology, to the extent that is known, when embarking upon the model development. Since much of simulation activity is carried out before implementation details are known, one should attempt to characterize such models in a sufficiently broad way to be generally

applicable. The methodology of simulating systems at this stage of development was also discussed in Chapter 2.

We concluded with a short discussion of an important practical aspect of the modeling of communication systems for simulation, namely the procedures that are necessary in order to properly set the ratio of signal to noise.

References

1. B. Sklar, A structured overview of digital communications—A tutorial review, Part I, *IEEE Commun. Mag.* **21**(August), 4–17 (1983); Part II, **21**(October), 6–21 (1983).
2. J. B. Anderson, T. Aulin, and C. E. Sundberg, *Digital Phase Modulation*, Plenum Press, New York (1986).
3. S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
4. R. E. Blahut, *Digital Transmission of Information*, Addison-Wesley, Reading, Massachusetts (1990).
5. L. W. Couch, II, *Digital and Analog Communication Systems*, 5th ed., Macmillan, New York (1997).
6. R. D. Gitlin, J. F. Hayes, and S. B. Weinstein, *Data Communications Principles*, Plenum Press, New York (1992).
7. S. Haykin, *Communication Systems*, 3rd Ed., Wiley, New York (1994).
8. I. Korn, *Digital Communications*, Van Nostrand Reinhold, New York (1985).
9. E. A. Lee and D. G. Messerschmitt, *Digital Communication*, 2nd ed., Kluwer Academic Press, Boston, Massachusetts (1994).
10. R. W. Lucky, J. Salz, and E. J. Weldon, Jr., *Principles of Data Communication*, McGraw-Hill, New York (1968).
11. J. G. Proakis, *Digital Communications*, 2nd ed., McGraw-Hill, New York (1988).
12. J. G. Proakis and M. Salehi, *Communication Systems Engineering*, Prentice-Hall, Englewood-Cliffs, New Jersey (1994).
13. M. Schwartz, *Information Transmission, Modulation, and Noise*, 3rd ed., McGraw-Hill, New York (1980).
14. K. S. Shanmugan, *Digital and Analog Communication Systems*, McGraw-Hill, New York (1979).
15. O. Shimbo, *Transmission Analysis in Communication Systems*, Vols. 1 and 2, Computer Science Press, Rockville, Maryland (1988).
16. B. Sklar, *Digital Communications: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey (1988).
17. J. J. Spilker, Jr., *Digital Communications by Satellite*, Prentice-Hall, Englewood Cliffs, New Jersey (1977).
18. H. Stark, F. B. Tuteur, and J. B. Anderson, *Modem Electrical Communications: Analog, Digital and Optical Systems*, 2nd ed. Prentice-Hall, Englewood-Cliffs, New Jersey (1988).
19. A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, New York (1979).
20. J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, Wiley, New York (1965).
21. R. E. Ziemer and W. H. Tranter, *Principles of Communications*, Houghton Mifflin, New York (1976).
22. J. B. Anderson and S. Mohan, *Source and Channel Coding: An Algorithmic Approach*, Kluwer Academic Press, Boston (1991).
23. R. E. Blahut, *Principles and Practice of Information Theory*, Addison-Wesley, Reading, Massachusetts (1987).
24. R. G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York (1968).
25. A. Gersho and V. Cuperman, Vector quantization: A pattern-matching technique for speech coding, *IEEE Commun. Mag.* **21** (9), 15–21 (1983).
26. R. M. Gray, Vector quantization, *IEEE ASSP Mag.* **1**(2), 4–29 (1984).
27. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Press, Boston, (1992).
28. B. Smith, Instantaneous companding of quantized signals, *Bell Syst. Tech. J.* **36**, 563–709 (1957).
29. D. F. Hoeschele, Jr., *Analog-to-Digital/Digital-to-Analog Conversion Techniques*, Wiley, New York (1968).
30. R. van de Plassche, *Integrated Analog-to-Digital and Digital-to-Analog Converters*, Kluwer Academic Publishers, Boston (1994).
31. M. T. Abuelma'tti, Effect of bit-threshold errors on the harmonic and intermodulation performance of successive approximation A/D converters, *IEEE Trans. Commun.* **41**(8), 1155–1160 (1993).
32. S. Pasupathy, Correlative coding: A bandwidth efficient signaling scheme, *IEEE Commun. Mag.* **15**, 4–11 (1977).

33. P. Kabal and S. Pasupathy, Partial response signaling, *IEEE Trans. Commun.* **COM-29**(9), 921–934 (1975).
34. C. E. Shannon and W. Weaver, *Mathematical Theory of Communications*, University of Illinois Press, Campaign-Urbana, Illinois (1963).
35. E. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York (1968).
36. R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, Reading, Massachusetts (1983).
37. G. C. Clark, Jr., and J. B. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, New York (1981).
38. S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey (1983).
39. W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes*, 2nd ed., MIT Press, Cambridge, Massachusetts (1972).
40. R. J. McEliece, *The Theory of Information and Coding*, Addison-Wesley, Reading, Massachusetts (1977).
41. W. Gautschi, *Numerical Analysis: An Introduction*, Birkhäuser, Boston (1997).
42. A. J. Viterbi, Error bounds for convolutional codes and an asymptotically optimal decoding algorithm, *IEEE Trans. Inf. Theory* **IT-13**(2), 260–269 (1967).
43. G. D. Forney, Jr., The Viterbi algorithm, *Proc. IEEE* **61**(3), 268–273 (1973).
44. H. Dawid, G. Fettweis, and H. Meyr, A CMOS IC for Gb/s Viterbi decoding: System design and VLSI implementation, *IEEE Trans. VLSI Systems* **4**(1), 17–31 (1996).
45. C.-L. Liu and K. Feher, $\pi/4$ -QPSK modems for satellite sound/data broadcast systems, *IEEE Trans. Broadcast.* **37**(1), 1–8 (1991).
46. K. Feher, Modems for emerging digital cellular-mobile radio systems, *IEEE Trans. Vehic. Technol.* **40**(2), 355–365 (1991).
47. L. E. Miller and J. S. Lee, BER expressions for differentially detected $\pi/4$ DQPSK modulation, *IEEE Trans. Commun.* **46**(1), 71–81 (1998).
48. J. A. C. Bingham, Multicarrier modulation: An idea whose time has come, *IEEE Commun. Mag.* **28**(5), 5–14 (1990).
49. I. Kalet, The multitone channel, *IEEE Trans. Commun.* **37**(2), 119–124 (1989).
50. L. J. Cimini, Jr., Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing, *IEEE Trans. Commun.* **COM-33**(7), 665–675 (1985).
51. M. Gudmundson, Spectral efficiency of a multitone frequency hopping system for personal communication systems, *IEEE Conf. Vehic. Technol.* 1650–1654 (1994).
52. J. S. Chow, J. C. Tu, and J. M. Cioffi, A discrete multitone transceiver for HDSL applications, *IEEE J. Select. Areas Commun.* **9**(6), 895–908 (1991).
53. S. B. Weinstein and P. M. Ebert, Data transmission by frequency-division multiplexing using the discrete Fourier transform, *IEEE Trans. Commun. Technol.* **COM-19**(5), 628–634 (1971).
54. C.-E. Sundberg, Continuous phase modulation, *IEEE Commun. Mag.* **24**(4), 25–38 (1986).
55. W. P. Osborne and M. B. Luntz, Coherent and noncoherent detection of CPFSK, *IEEE Trans. Commun.* **COM-22**, 1023–1036 (1974).
56. T. A. Schonhoff, Bandwidth vs. performance considerations for CPFSK, in *Proc. National Telecommunications Conference (NTC)*, New Orleans, Louisiana (December 1975), pp. 38:1–38:5.
57. T. A. Schonhoff, Symbol error probabilities for M -ary CPFSK: Coherent and noncoherent detection, *IEEE Trans. Commun.* **COM-24**, 644–652 (1976).
58. K. Murota and K. Hirade, GMSK, modulation for digital mobile radio telephony, *IEEE Trans. Commun.* **COM-29**(7), 1044–1050 (1981).
59. R. Steele (ed.), *Mobile Radio Communications*, Pentech Press, London (1992).
60. S. L. Sayegh, A class of optimum block codes in signal space, *IEEE Trans. Commun.* **COM-34**(10), 1043–1045 (1986).
61. M. Vanderaar, J. Budinger, and P. Wagner, Least reliable bits coding (LRBC) for high data rate satellite communications, in *AIAA 14th International Communications Satellite Systems Conference*, Washington, D.C. (March 22–24, 1992), pp. 507–514.
62. H. L. Berger and T. J. Kolze, Bit error rate performance of coded 8PSK, in *AIAA 16th International Communications Satellite Systems Conference*, Washington, D.C. (February 25–29, 1996).
63. M.-C. Lin, J.-Y. Wang, and S.-C. Ma, On block-coded modulation with interblock memory, *IEEE Trans. Commun.* **45**(11), 1401–1411 (1997).
64. R. H. Morelos-Zaragoza, T. Kasami, S. Lin, and H. Imai, On block-coded modulation using unequal error protection codes over Rayleigh-fading channels, *IEEE Trans. Commun.* **46**(1), 1–4 (1998).
65. G. Ungerboeck, Channel coding with multilevel/phase signals, *IEEE Trans. Inf. Theory* **IT-28**(1), 55–67 (1982).

66. G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, Efficient modulation for band-limited channels, *IEEE J. Select. Areas Commun.* **SAC-2**, 632–647 (1984).
67. E. Biglieri, High-level modulation and coding for nonlinear satellite channels, *IEEE Trans. Commun.* **COM-32**(5), 616–626 (1984).
68. G. Ungerboeck, Trellis-coded modulation with redundant signal sets, Part I, Introduction, *IEEE Commun. Mag.* **25**(2), 5–11 (February 1987), Part II, State of the art, **25**(2), 12–21 (February 1987).
69. E. Biglieri, D. Divsalar, P. J. McLane, and M. K. Simon, *Introduction to Trellis-Coded Modulation with Applications*, Macmillan, New York (1991).
70. S. Benedetto, M. Mondin, and G. Montorsi, Performance evaluation of trellis-coded modulation schemes, *Proc. IEEE* **82**(3), 833–855 (1994).
71. C. Schlegel, *Trellis Coding*, IEEE Press, New York (1997).
72. H. R. Mathwich, J. F. Balcewicz, and M. Hecht, The effect of tandem band and amplitude limiting on the E_b/N_0 performance of minimum (frequency) shift keying (MSK), *IEEE Trans. Commun.* **COM-22**, 1525–1540 (1974).
73. S. A. Gronemeyer and A. L. McBride, Theory and comparison of MSK and offset QPSK modulation techniques through a satellite channel, in *Proc. National Telecommunications Conference (NTC)*, New Orleans, Louisiana (December 1975), pp. 30:28–30:32.
74. F. Amoroso, Pulse and spectrum manipulation in the minimum (frequency) shift keying (MSK) format, *IEEE Trans. Commun.* **COM-24**, 381–384 (1976).
75. C. R. Ryan, A. R. Hambley, and D. E. Vogt, 760 Mbit/s serial MSK microwave modem, *IEEE Trans. Commun.* **COM-28**(5), 771–777 (1980).
76. M. J. Mohan, D. B. Vandervoet, and R. M. Fielding, Analysis and simulation of an MSK modulator in a dynamic temperature environment, in *Proc. International Conference on Communications (ICC '81)*, Vol. 1, Denver, Colorado (June 14–18, 1981).
77. R. DeBuda, Coherent demodulation of frequency-shift keying with low deviation ratio, *IEEE Trans. Commun.* **COM-20**(3), 429–435 (1972).
78. R. W. D. Booth, Carrier phase and bit sync regeneration for the coherent demodulation of MSK, in *Conference Record, National Telecommunications Conference* Vol. 1; Birmingham, Alabama (December 3–6, 1978).
79. S. O. Rice, Noise in FM receivers, in *Time Series Analysis*, M. Rosenblatt (ed.), Wiley, New York (1963).
80. G. Lindgren, Shape and duration of clicks in modulated FM transmission, *IEEE Trans. Inf. Theory* **IT-30**(5), 728–735 (1984).
81. A. V. Oppenheim and R. W. Schaffer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1975).
82. J. K. Holmes, *Coherent Spread Spectrum Systems*, Wiley, New York (1982).
83. C. K. Santhanam and J. Koerner, Transfer function synthesis as a ratio of two complex polynomials, *IEEE Trans. Aut. Control* **AC-8**, 56–58 (1963).
84. M. T. Jong and K. S. Shanmugan, Determination of transfer functions from amplitude response data, *Int. J. Control* **25**, 941–948 (1977).
85. E. Wong and B. Hajek, *Stochastic Processes in Engineering Systems*, Springer-Verlag, New York (1985).
86. W. A. Gardner, *Introduction to Random Processes*, Macmillan, New York (1986).
87. M. L. Honig and D. G. Messerschmitt, *Adaptive Filters*, Kluwer Academic Press, Boston (1984).
88. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1985).
89. J. K. Chamberlain, F. M. Clayton, H. Sari, and P. Vandamme, Receiver techniques for microwave digital radio, *IEEE Commun. Mag.* **24**(11), 43–54 (1986).
90. D. Bertsekas and R. G. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
91. M. Schwartz, *Telecommunication Networks*, Addison-Wesley, Reading, Massachusetts (1987).
92. C. H. Sauer and E. A. MacNair, *Simulation of Computer Communication Systems*, Prentice-Hall, Englewood Cliffs, New Jersey (1983).
93. R. A. Scholtz, The spread spectrum concept, *IEEE Trans. Commun.* **COM-25**(8), 748–755 (1977).
94. R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, Theory of spread-spectrum communications—A tutorial, *IEEE Trans. Commun.* **COM-30**(8), 855–884 (1982).
95. R. E. Ziemer and R. L. Peterson, *Digital Communications and Spread Spectrum Systems*, MacMillan, New York (1985).
96. M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications*, Vols. I, II, III, Computer Science Press, Rockville, Maryland (1985, 1986).
97. A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*, Addison-Wesley, Reading, Massachusetts (1995).
98. G. Agrawal and N. Dutta, *Long Wavelength Semiconductor Lasers*, Van Nostrand, New York (1986).

99. R. Vodhamel, A. Elrefaie, R. Wagner, M. Igbal, J. Gimlett, and S. Tsuji, 10 to 20 Gb/s modulation performance of **1.55 μ m** DFB lasers for FSK. systems, *IEEE J. Lightwave Technol.* **7**(10), 1454–1460 (1989).
100. A. J. Viterbi, *Principles of Coherent Communication*, McGraw-Hill, New York (1966).
101. *IEEE Journal of Selected Areas in Communication*, Special Issue on Progress in Military Communications-1, **SAC-3**(5) (September 1985).
101. R. C. Tausworthe, Theory and Practical Design of Phase-Locked Receivers, Vol. 1, Jet Propulsion Lab., Tech. Rep. No. 32–819, February 15, 1966.
102. F. M. Gardner, *Phaselock Techniques*, Wiley, New York (1966).
103. W. C. Lindsey and M. K. Simon, *Telecommunication Systems Engineering*, Prentice-Hall, Englewood Cliffs, New Jersey (1973).
104. L. E. Franks, Synchronization subsystems: Analysis and design, in *Digital Communications: Satellite/Earth Station Engineering*, Prentice-Hall, Englewood Cliffs, New Jersey (1981).
105. W. C. Lindsey and C. M. Chie, A survey of digital phase-locked loops, *Proc. IEEE* **69**(4), 410–431 (1981).
106. H. Meyr and G. Ascheid, *Synchronization in Digital Communications*, Vol. 1, Wiley, New York (1990).
107. R. Best, *Phase-Locked Loops: Theory, Design, and Applications*, 2nd ed., Mc-Graw-Hill, New York (1993).
108. J. L. Stensby, *Phase-Locked Loops: Theory and Applications*, CRC Press, Boca Raton, Florida (1997).
109. B. T. Kopp and W. P. Osborne, Phase jitter in MPSK tracking loops: Analytical, simulation, and laboratory results, *IEEE Trans. Commun.* **45**(11), 1385–1388 (1997).
110. R. A. Harris and P. Kristiansen, The generation and use of bit error rate surface plots, *IEEE J. Select. Areas Commun.* **SAC-2**(1), 185–190 (1984).
111. J. D. Gibson, *Principles of Digital and Analog Communications*, 2nd ed., Macmillan, New York (1992).

This page intentionally left blank

Communication Channels and Models

In its most general sense the word “channel” can be used to mean everything between the source and the sink of a signal. Referring back to Figure 8.1, this general definition would include all of the equipment shown there as well as the physical medium between the transmitter and the receiver through which the signal is radiated or conducted. In order to simulate such a channel when it is explicitly represented by a sequence of blocks, we need to have a model for each block (as well as for the medium), and in fact we do discuss models for all of these blocks in various parts of this book. In this chapter, however, we discuss channel *models*, which needs to be distinguished from the general definition of a channel given above. Basically, a channel model may be thought of as a *representation*, in mathematical or algorithmic form, for the transfer characteristics of any contiguous subset of the general block diagram. This representation is generally not based on the underlying physical phenomena, but rather on fitting external (empirical) observations. In practice, there are actually two quite distinct entities that are referred to as channel models. One of these is the physical medium only, which may be free space (as an idealization), the atmosphere, wires, a waveguide, or optical fibers. Often, one implicitly includes in this definition of channel certain other physical conditions or geometrical constraints that have a strong bearing on the effective transfer function that the medium creates between transmitter and receiver. Some of these conditions or constraints include the carrier frequency, the bandwidth, and the physical environment. Depending upon the particular combination of controlling factors, the “atmospheric” or “wireless” channel may be described in significantly different ways. Under some conditions, communication to or from satellites may be described as passing through an “almost free-space” channel. On the other hand, radio-relay or mobile systems are known to operate in an environment that produces multiple paths, hence is appropriately described as a “multipath” channel. In this section we shall outline some useful channel models which describe the medium through which the signal may pass. Most of our discussion will be on the multipath channel, which characterizes the medium for most existing and emerging wireless applications.

We shall also describe another type of channel model, appropriate to digital transmission. This type of model, called a finite-state channel model, can be thought of as a short-hand description of the net effects of the entire channel (including the medium) that exists between the digital source or encoder output and the digital sink (or decoder) input. Since the digital alphabets are discrete, the effect of the intervening channel can often be economically described by a set of probabilities that relate the channel output sequence to the input

sequence. This type of channel model will be described in Section 9.4 Case Study IV in Chapter 12 describes a specific scenario in which we derive the discrete model from the waveform simulation.

9.1. Fading and Multipath Channels

9.1.1. Introduction

In this section we will first describe *fading* and *multipath*, the performance-limiting phenomena that occur in wireless atmospheric radio channels, and then turn our attention to modeling and simulation of these channels⁽¹⁻⁷⁾.

Fading and Multipath occur in many radio communication systems. These effects were first observed and analyzed in troposcatter systems in the 1950s and early 1960s. In any wireless communication system there could be more than one path over which the signal can travel between the transmitter and receiver antennas. The presence of multiple paths is due to atmospheric scattering and refraction, or reflections from buildings and other objects. In a multipath situation, the signals arriving along different paths will have different attenuations and delays and they might add at the receiving antenna either constructively or destructively. If the path lengths and/or the geometry change due to changes in the transmission medium or due to relative motion of the antennas, as in the mobile case, the signal level might be subjected to wild fluctuations. Multipath fading affects the signal in two ways: dispersion (time spreading or frequency selectivity) and time-variant behavior.

Much of the current interest is in modeling and simulation of fading in mobile and indoor wireless communications. We will therefore limit our discussions to mobile wireless channels. Although the fading mechanisms may be different in different environments, the general concepts of modeling and simulation remain the same.

Mobile communication is affected, in addition to multipath (or *small-scale* fading), to another type of fading which is referred to as *shadow* or *large-scale* fading. Shadow fading reveals itself as an attenuation of the average signal power. Shadow fading is induced by prominent terrain contours (hills, buildings, etc.) between transmitter and receiver. The receiver is said to be *shadowed* by these objects. Shadow fading is described in terms of pathloss and statistical variations about the mean.

A mobile wireless moving over a large area encounters both types of fading: multipath fading superimposed on the much slower fading. Thus, the mobile channel impulse response in its complex-lowpass equivalent form is composed of two components,

$$\tilde{h}(\tau, t) = s(t) \times \tilde{c}(\tau, t) \quad (9.1.1)$$

where $s(t)$ is the shadow fading component and $\tilde{c}(\tau, t)$ is the multipath component. The meaning of this notation was discussed in Section 4.2. Figure 9.1 shows the received signal of a mobile receiver and depicts multipath fading superimposed on shadow fading.

In some mobile systems shadow fading is partially compensated by power control. Since shadow fading is slow, it can be regarded as *quasistatic*, and is usually simulated using the “snapshot” technique (see Case study I in Chapter 12). However, a problem arises during fast transients, e.g., when a mobile moves from deep shadow into a line-of-sight situation. Such transients can cause excessive interference in, e.g., spread-spectrum systems.

In the following sections we describe in some detail both shadow and multipath fading.

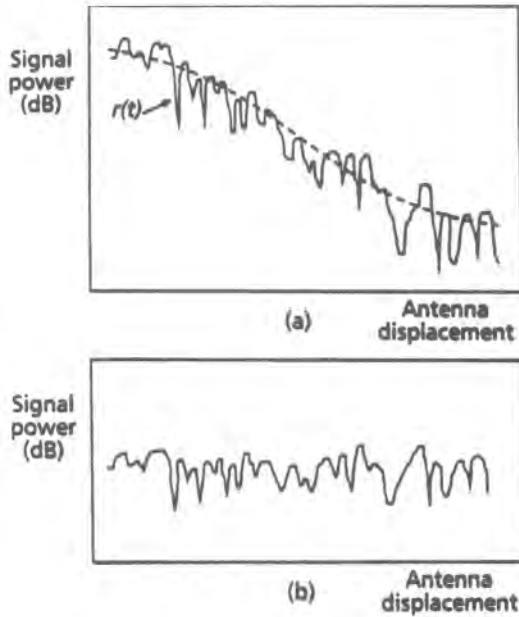


Figure 9.1. Illustration of large-scale (shadow) fading and small-scale (multipath) fading (from Ref. 5, © IEEE, 1997). (a) Combination of shadow fading and multipath components, (b) the multipath component.

9.1.2. Shadow Fading

The signal received by a mobile radio from a transmitter can be represented as

$$S_r = S_t + G_t + G_r - L_p \quad (9.1.2)$$

where S_r is the received signal in dBm, S_t is the transmitted signal in dBm, G_t is the transmit antenna gain toward the receiver in dB, G_r is the receive antenna gain toward the transmitter in dB, and L_p is the propagation loss in dB.

The gain G_t of the transmit antenna in the direction of the mobile receiver can be determined from the gain pattern, position, and orientation of the transmit antenna, along with the position of the mobile receiver. Since the mobile receiver moves relative to the transmitter, and its antenna orientation is not typically controllable, the mobile antenna is usually omnidirectional. For omnidirectional mobile antennas G_r is just the gain of that antenna, nominally constant in the horizontal plane.

Generally, the most difficult of these parameters to predict is the propagation loss L_p . Many models have been developed to try and predict this value. Some models are site-specific and require a great deal of information about the environment near and between the transmitter and receiver. Others are more statistically based and try to predict the nature of the propagation loss from some small number of parameters of the radio system and the type of environment in which it is operating. Currently, the most popular site-specific methods include ray-tracing techniques.^(8,9) These techniques were originally developed for indoor environments, (see also Section 9.1.3.6), and have been extended to dense urban outdoor areas (microcells). For the latter application, ray-tracing techniques require data about the

terrain, the buildings and their material composition, any vegetation cover, and the positions of the transmitter and receiver. In addition to the large amount of data required by these techniques, a great deal of computational effort is generally needed to determine the propagation loss between the transmitter and receiver.

The most popular of the statistical propagation models are the class of slope-intercept models. These models treat the propagation loss as being comprised of a distance-related deterministic component as well as statistical components both for the shadow and multipath fading. The deterministic component is taken to be of the form

$$L_p = \alpha + \beta \log_{10}(R) \quad [\text{dB}] \quad (9.1.3)$$

where R is the distance between the transmitter and receiver in km and α and β are parameters determined by the model. Notice that since this relation describes the propagation loss in decibels, it also implies that the received signal power in milliwatts is proportional to $R^{-0.1\beta}$. In fact, this relation can describe free-space propagation loss if $\alpha = -20 \log_{10}(\pi\lambda/4)$ and $\beta = 20$.

Values for the parameters α and β can be determined from experimental measurements of specific radio systems. Measurements are taken of received signal power averaged over several wavelengths, so that the multipath effects are averaged out at many different locations around the transmitter. These power measurements (expressed in decibel units) can be plotted against the log of the distance of the measurement sites to the transmitter. Then a linear least squares fit can be made to the data. The slope of the least squares fit determines β and the intercept of the least squares fit along with the antenna gains and transmit power determine α . In addition, if the least squares fit is subtracted from the measured data, the resulting residuals describe the statistical component of the shadow fading model.

One of the first popular slope-intercept models was the Hata⁽¹⁰⁾ model. This model was published in 1980 by Hata based on a large set of measurement data taken by Okumura *et al.*⁽¹¹⁾ in rural, suburban, and urban Japan in the 1960s. Okumura *et al.* used transmitter heights that would be comparable to today's PCS or cellular macrocell sites. Another popular slope-intercept model, developed after Hata, for transmitters at lower heights that may be used in microcells is the COST-231 model.⁽¹²⁾

In the Hata model the slope parameter β depends on the height of the transmitter above the average ground elevation around it. The intercept parameter α depends on this height, the operating frequency, and the type of environment (rural, suburban, or urban) in which the mobile is operating. These parameters are given by

$$\beta = 44.9 + 6.55 \log_{10}(h) \quad (9.1.4)$$

and the intercept

$$\alpha = \alpha_0 = 69.55 + 26.16 \log_{10}(f) - 13.82 \log_{10}(h) \quad \text{urban environment}$$

$$\alpha = \alpha_0 - [2 \log_{10}^2(f/28) + 5.4] \quad \text{suburban environment}$$

$$\alpha = \alpha_0 - [4.78 \log_{10}^2(f) - 18.33 \log_{10}(f) + 40.94] \quad \text{rural environment}$$

where h is the transmitter height above average terrain elevation in meters, and f is the frequency in MHz.

In addition, the Hata model provides for a second-order adjustment of the α parameter based on the height of the mobile receiver antenna.

The shadow fading statistical component of the slope-intercept models can be determined by examining the residuals of measurement data once the linear least squares fit to the log of the distance is removed. These residuals, in decibels, typically follow approximately a Gaussian distribution with a zero mean and a standard deviation of about 8 dB. Hence, the shadow fading component follows a log-normal distribution. The average power is predicted by the shadow fading component of the propagation model. For a given shadow fading component the multipath fading component follows a Rayleigh or Ricean distribution. In other words, the signal envelope is conditionally Rayleigh or Ricean. If there is no single dominant signal contribution to the receiver, the signal will follow a Rayleigh distribution. If there is a dominant signal contribution to the receiver, such as a line-of-sight signal, the signal will follow a Ricean signal strength distribution. In addition to the average power, the ratio of the power in the dominant component to the total power is needed to describe this distribution.

Figure 9.2 shows a scatter plot of pathloss versus distance in several German cities.⁽¹³⁾ It should be noted that the values below the slope of $n = 2$ are due to the tunneling effect present in urban streets.

9.1.3. Multipath Fading

The transmitted signal follows many different paths before arriving at the receiving antenna, and it is the aggregate of these paths that constitutes the multipath radio propagation channel (Figure 9.3). The resulting signal strength will undergo large fluctuations, which,

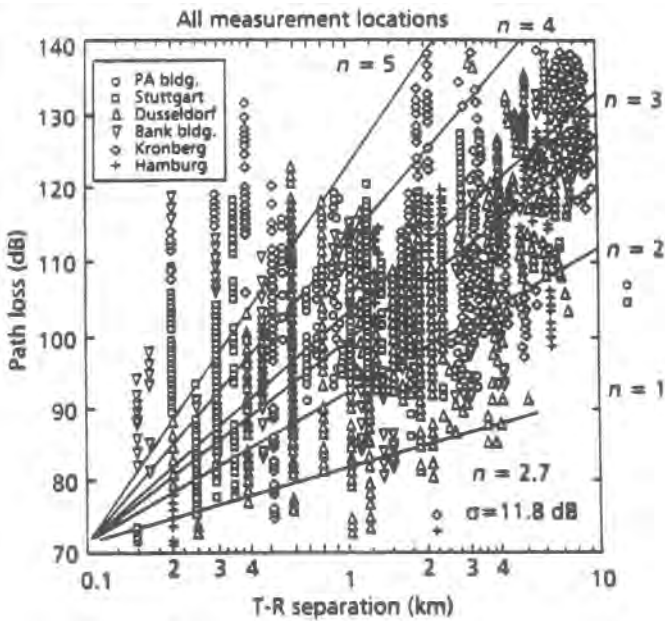


Figure 9.2. Path loss versus distance measured in several German cities (from Ref. 5, © IEEE, 1997).

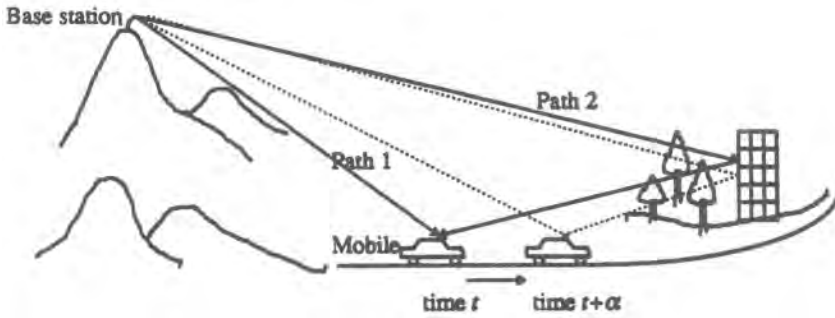


Figure 9.3. Illustration of the multipath physical environment

when the signal is small, results in a “fade.” For brevity, we will refer to this situation as *multipath fading*. The multipath fading can be assigned into two categories.

1. The multipath signal paths are made up of a relatively small and identifiable number of components reflected by small hills, houses, and other structures encountered in open areas and rural environments. This results in a channel model with a finite number of multipath components. Such a channel is referred to as a *discrete multipath channel*.
2. The multipath signal paths are generated by a large number of unresolvable reflections as might occur in a mountainous area or in a dense urban environment. This signal is composed of a continuum of unresolvable multipath components. This channel model is referred to as a *diffuse multipath channel*.

It should be noted that real measured channels may contain both discrete and diffuse components. Such channels are separated into their respective discrete and diffuse components for modeling purposes. In all the above cases the channel is modeled as a linear time-varying system with a complex lowpass-equivalent response $\tilde{\mathbf{c}}(\tau, t)$.

9.1.3.1. Lowpass-Equivalent Characterization of Multipath Channels

A simple model for discrete multipath channels has the form

$$\mathbf{y}(t) = \sum_n \mathbf{a}_n(t) \mathbf{s}(t - \tau_n(t)) \quad (9.1.5)$$

where $s(t)$ is the bandpass input signal, $\mathbf{a}_n(t)$ is the attenuation factor for the signal received on the n th path, and $\tau_n(t)$ is the corresponding propagation delay. If we express $s(t)$ as

$$\mathbf{s}(t) = \text{Re}\{\tilde{\mathbf{s}}(t)e^{j2\pi f_c t}\}$$

then we can express the channel output as

$$\mathbf{y}(t) = \text{Re}\left\{\left[\sum_n \mathbf{a}_n(t)e^{-j2\pi f_c \tau_n(t)}\tilde{\mathbf{s}}(t - \tau_n(t))\right]e^{j2\pi f_c t}\right\}$$

and it is clear that the complex envelope of the output is

$$\begin{aligned}\tilde{y}(t) &= \sum_n a_n(t) e^{-j2\pi f_c \tau_n(t)} \tilde{s}(t - \tau_n(t)) \\ &= \sum_n \tilde{a}_n(\tau_n, t) \tilde{s}(t - \tau_n(t))\end{aligned}\quad (9.1.6)$$

Equation 9.1.6 shows that we can describe the multipath channel by a time-varying, complex, lowpass-equivalent impulse response $\tilde{c}(\tau_n(t), t)$,

$$\tilde{c}(\tau_n(t), t) = \sum_n \tilde{a}_n(\tau_n(t), t) \delta(\tau - \tau_n(t)) \quad (9.1.7)$$

For the diffuse multipath channel the response is expressed in integral form as

$$\tilde{y}(t) = \int_{-\infty}^{\infty} \tilde{a}(\tau, t) \tilde{s}(t - \tau) d\tau \quad (9.1.8)$$

where $\tilde{a}(\tau, t)$ is the complex attenuation of the signal component at delay τ and time instant t . The lowpass-equivalent time-variant impulse response becomes

$$\tilde{c}(\tau, t) = \tilde{a}(\tau, t) e^{-j2\pi f_c \tau} \quad (9.1.9)$$

9.1.3.2. Statistical Characterization of Multipath Channels

The multipath fading for both the *diffuse* and *discrete* channels manifests itself in two effects:

1. Time spreading (in τ) of the symbol duration within the signal, which is equivalent to filtering and bandlimiting.
2. A time-variant behavior (in t) of the channel due to motion of the receiver or changing environment such as movement of foliage or movement of reflectors and scatterers.

Figure 9.4 shows the consequences of both effects by showing the impulse response of the *diffuse channel* versus delay τ at three instances of time t .

9.1.3.3. Statistical Characterization of the Time-Variant Behavior

The random fluctuations in the received signal due to fading can be modeled by treating $\tilde{c}(\tau, t)$ as a random process in t . Since the components of the multipath signal arise from a large number of reflections and scattering from rough or granular surfaces, then by virtue of the central limit theorem, $\tilde{c}(\tau, t)$ can be modeled as a complex Gaussian process. At any time t , the probability density functions of the real and imaginary parts are Gaussian. This model implies that for each τ or τ_k the ray is composed of a large number of unresolvable components. Hence both the *diffuse* and the *discrete* channels are complex Gaussian

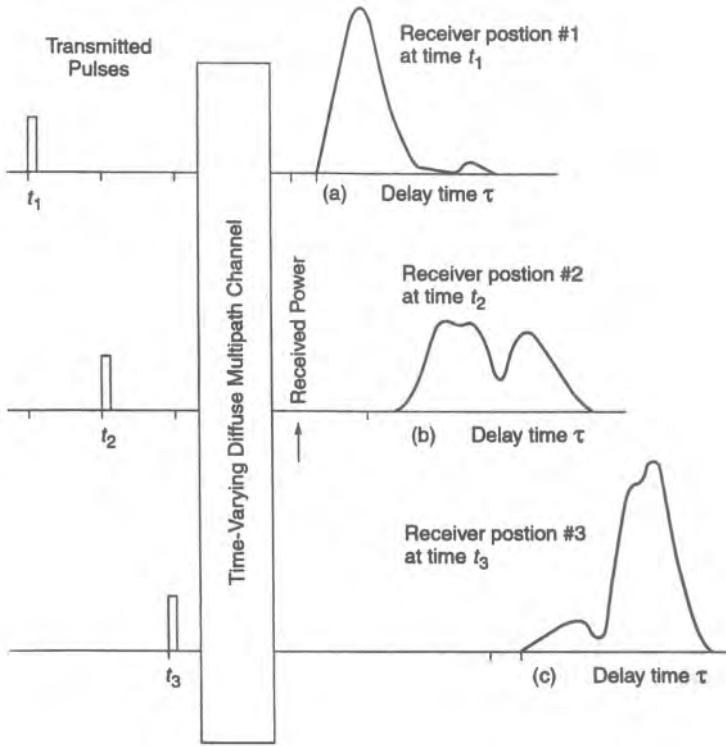


Figure 9.4. Illustration of the nature of a time-varying channel: the response to a narrow pulse versus delay at different receiver positions.

processes in t . If $\tilde{c}(\tau, t)$ has a zero mean, then the envelope $R(\tau, t) = |\tilde{c}(\tau, t)|$ has a Rayleigh probability density function

$$f_R(r) = \frac{r}{\sigma^2} e^{-r^2/(2\sigma^2)} \quad (9.1.10)$$

If $\tilde{c}(\tau, t)$ has a nonzero mean, which implies the presence of a significant specular (nonfaded) line-of-sight component, then $R(\tau, t) = |\tilde{c}(\tau, t)|$ has a Ricean probability density function

$$f_R(r) = \frac{r}{\sigma^2} I_0 \left[\frac{Ar}{\sigma^2} \right] e^{-(r^2 + A^2)/(2\sigma^2)} \quad (9.1.11)$$

where A is the nonzero mean of $\tilde{c}(\tau, t)$ and $I_0(\cdot)$ is the zeroth-order modified Bessel function of the first kind.

The ratio $K = A^2/\sigma^2$ is an indicator of the relative power in the faded and unfaded components. For values of $K \gg 1$ the channel tends to be more specular, whereas for $K \ll 1$ the channel tends to be Rayleigh. In the simulation model the specular and the Rayleigh components are separated as in Figure 9.5.

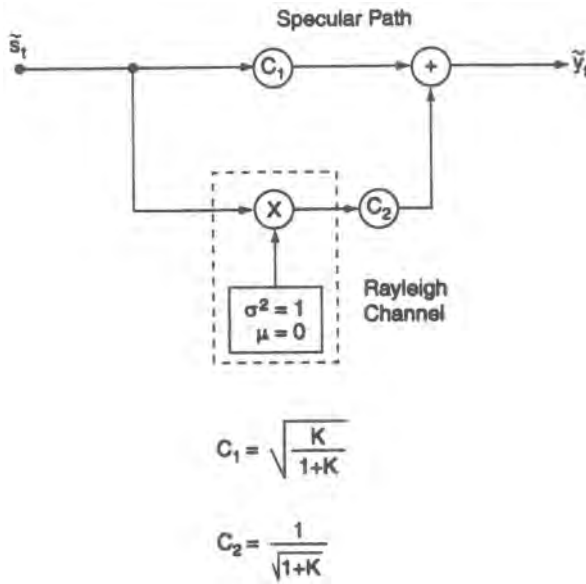


Figure 9.5. Simulation model of a Ricean channel.

While the pdf of $|\tilde{c}(\tau, t)|$ describes the distribution of the instantaneous values of the impulse response, the temporal variations are modeled by an appropriate autocorrelation function or, equivalently, by the power spectral density of the random process in the t variable. We describe these models below.

9.1.3.4. Statistical Characterization: The WSSUS Model

A model for the multipath channel that includes both the variations in t and τ was introduced by Bello.^(2-5, 14) The time-varying nature of the channel $\tilde{c}(\tau, t)$ is mathematically modeled as a *wide-sense stationary* (WSS) random process in t with an autocorrelation function

$$R_{\tilde{c}}(\tau_1, \tau_2, \Delta t) = E[\tilde{c}^*(\tau_1, t)\tilde{c}(\tau_2, t + \Delta t)] \quad (9.1.12)$$

In most multipath channels, the attenuation and phase shift associated with different delays can be assumed uncorrelated; this is the *uncorrelated scattering* (US) assumption, which leads to

$$R_{\tilde{c}}(\tau_1, \tau_2, \Delta t) = R_{\tilde{c}}(\tau_1, \Delta t)\delta(\tau_1 - \tau_2) \quad (9.1.13)$$

The equation above embodies both the WSS and US assumptions, and is referred to as the WSSUS model for fading. This autocorrelation function is denoted by $R_{\tilde{c}}(\tau, \Delta t)$,

$$R_{\tilde{c}}(\tau, \Delta t) \equiv E[\tilde{c}^*(\tau, t)\tilde{c}(\tau, t + \Delta t)] \quad (9.1.14)$$

It is apparent from (9.1.14) that the WSSUS model can be represented in either the time domain or the frequency domain by performing a Fourier transform on one or both of the

variables τ and t . From the engineer's point of view, it would be useful to have a model that simultaneously provides a description of the channel properties with respect to the delay variable τ and a frequency-domain variable (Doppler frequency) ν , whose significance will be discussed below. We obtain this model by Fourier transforming the autocorrelation function in the Δt variable:

$$S(\tau, \nu) = F_{\Delta t}[R_{\tilde{c}}(\tau, \Delta t)] = \int_{-\infty}^{\infty} R_{\tilde{c}}(\tau, \Delta t) e^{-j2\pi\nu\Delta t} d\Delta t \quad (9.1.15)$$

$S(\tau, \nu)$ is called the *scattering function* and is perhaps the most important statistical measure of the random multipath channel. It is a function of two variables, τ (delay) and a frequency-domain variable ν called the Doppler frequency. From (9.1.15) it can be seen that ν is the dual variable of Δt , hence it captures the rapidity with which the channel itself changes. Figure 9.6 shows examples of measured scattering functions. The scattering function provides a single measure of the average power output of the channel as a function of the delay τ and the Doppler frequency ν .

From the scattering function we can obtain some of the most important relationships of a channel which impact the performance of a communication system operating over that channel. The *delay-power profile*, sometimes also referred to as the *multipath intensity profile*, $p(\tau)$ is defined as

$$p(\tau) = R_{\tilde{c}}(\tau, 0) = E|\tilde{c}(\tau, t)|^2 \quad (9.1.16)$$

and represents the average received power as a function of delay τ . It can be shown that $p(\tau)$ is related to the scattering function via

$$p(\tau) = \int_{-\infty}^{\infty} S(\tau, \nu) d\nu \quad (9.1.17)$$

Another function that is useful in characterizing fading is the *Doppler power spectrum*, which is derived from the scattering function through

$$S(\nu) = \int_{-\infty}^{\infty} S(\tau, \nu) d\tau \quad (9.1.18)$$

The four functions $p(\tau)$, $S(\nu)$, and their Fourier transforms are used to characterize the behavior of the multipath channel. These four relationships, shown in Figure 9.7, are described below.

9.1.3.4.1. The Delay Power Profile An example of the delay power profile is shown in Figure 9.7a. Here, the term delay refers to *excess delay*. It represents the delay measured from the first perceptible signal that arrives at the receiver. The *maximum excess delay*, also termed *maximum delay spread*, T_m is the delay between the first and the last component of the signal during which the received power falls below some threshold level, e.g., 20 dB below the strongest component. The relationship between T_m and the symbol time T_{sym} can be viewed in terms of two different degradation criteria:

1. A channel is said to exhibit frequency-selective fading if $T_m > T_{\text{sym}}$. In this condition the multipath components extend beyond the symbol duration, which causes inter-

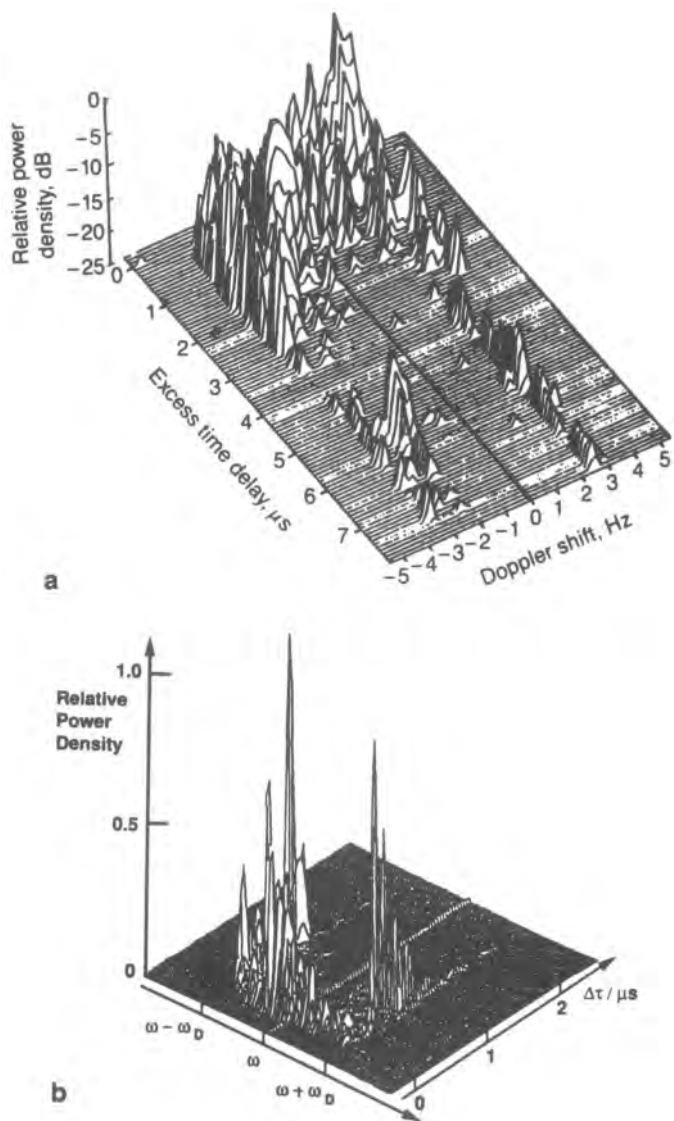


Figure 9.6. Examples of measured scattering functions. (a) Scattering function in a severe multipath area (from Ref. 2, © Halsted Press, 1992, reprinted with permission); (b) scattering function of a broadband measurement in an urban environment (from W. R. Braun and U. Dersch, A physical mobile radio channel model, © IEEE Tran. on VT, Vol. 40, No. 2, pp. 472–482, May, 1991).

symbol interference (ISI) distortion in the signal. Since the multipaths are resolvable in this case, the ISI distortion can be mitigated by rake reception (see Chapter 12) or equalization.

2. A channel is said to exhibit frequency-nonselective or flat fading if $T_m \ll T_{\text{sym}}$. In this case there is very little ISI, but we still have distortion in the system because the multipath signal can add destructively, reducing the SNR considerably. The counter-measure in such a case is power control or diversity.

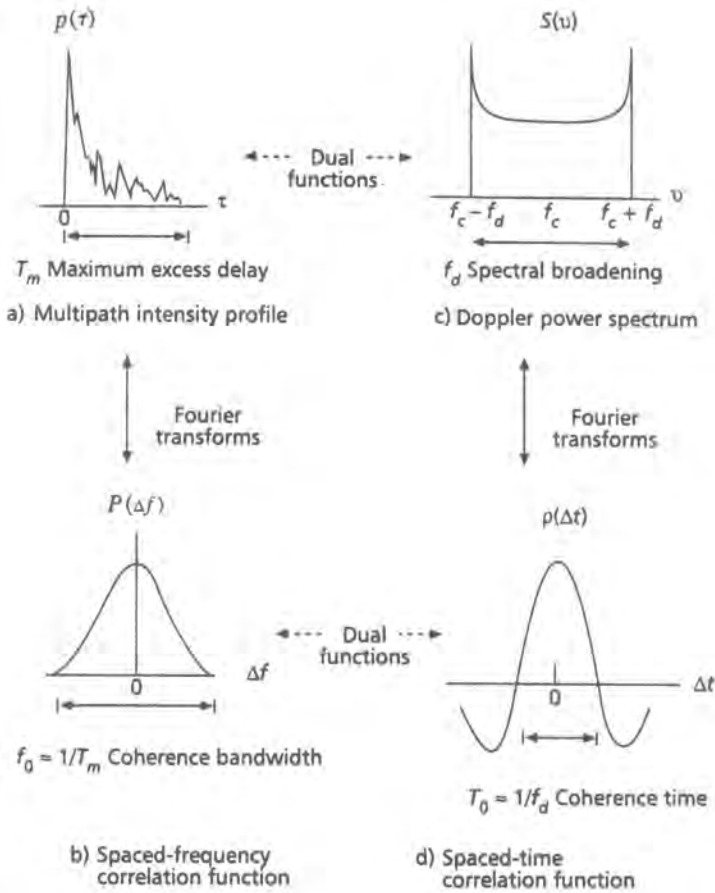


Figure 9.7. Relationships among the channel correlation functions and power density functions (from Ref. 5, © IEEE, 1997).

The dividing line between frequency-selective and flat fading is not perfectly sharp. In channels that we call flat, frequency selectivity still happens, but with smaller probability. In many applications the requirement for a channel to be frequency-nonselective is $\sigma_\tau < 0.1T_{\text{sym}}$, where σ_τ is the *rms delay spread* defined in (9.1.19) below.

In the frequency-nonselective case, the multipath components arrive within a short fraction of a symbol time, so that the channel can be well modeled by a single ray and the input/output relationship can be expressed as a multiplication. For a frequency-selective channel, the input/output relationship is a convolution:

$$\tilde{y}(t) = \tilde{c}(t)\tilde{s}(t) \quad \text{flat channel}$$

$$\tilde{y}(t) = \tilde{c}(\tau, t) * \tilde{s}(t) \quad \text{frequency-selective channel}$$

where in the second of these expressions the convolution is performed in the τ domain.

The maximum excess delay T_m is not necessarily the best indicator of how any given system will perform on the channel because some channels with the same value of T_m can

have very different delay power profiles. A more useful measurement is the *rms delay spread* σ_τ defined as

$$\sigma_\tau = \sqrt{\overline{\tau^2} - \bar{\tau}^2} \quad (9.1.19a)$$

where

$$\bar{\tau^k} = \frac{\int \tau^k p(\tau) d\tau}{\int p(\tau) d\tau} \quad (9.1.19b)$$

It has been established via simulations of systems with equalized receivers that the BER performance is practically identical if the rms delay spread is the same, even if the delay-power profiles differ widely.⁽¹⁵⁾

Since the BER performance of a communication system is more sensitive to the values of the rms and to a lesser extent to the maximum delay spread, rather than to the shape of the delay power profile, we may as well use simple profiles such as uniform or exponential in simulations or exploratory studies in general, in the absence of site-specific knowledge. The delay profiles are usually normalized to have unit area (i.e., total normalized power = 1).

Typical rms delay spreads are shown in Table 9.1.

9.1.3.4.2. The Spaced-Frequency Correlation Function A completely analogous characterization of signal dispersion can be done in the frequency domain. The function $P(\Delta f)$ shown in Figure 9.7b is defined as

$$P(\Delta f) = F(p(\tau)) \quad (9.1.20)$$

and is referred to as *spaced-frequency correlation function*. $P(\Delta f)$ represents the correlation between the channel response to two narrowband signals with the frequencies f_1 and f_2 as a function of the difference $\Delta f = f_2 - f_1$. This function can be thought of as the transfer function of the channel. Therefore, time spreading can be viewed as if it were the result of a filtering process. The *coherence bandwidth* f_0 is defined as the frequency range where all frequency component amplitudes are correlated. That is, the spectral components in that range fade together. It can be shown that f_0 and T_m are reciprocally related. As a rule of thumb, it is usually assumed that

$$f_0 \approx \frac{1}{T_m} \quad (9.1.21)$$

A general relationship between the coherence bandwidth and rms delay spread σ_τ does not exist and must be derived from actual dispersion characteristics for particular channels.

Table 9.1. Typical rms Delay Spreads

Link type	Link distance	rms delay spread
Troposcatter	1000 km	milliseconds
Outdoor mobile	1 km	microseconds
Indoor cellular	10 m	nanoseconds

For the case of mobile radio, an array of radially uniformly spaced scatterers, all with equal-magnitude reflection coefficients, but independent, randomly occurring phase angles, is widely accepted. This model is referred as the *dense scatterer* channel model^(1,16) and is commonly known as the Jakes model (described in more detail later). For this channel the coherence bandwidth is defined as the bandwidth interval over which the channel's transfer function has a correlation of at least 0.5, and can be shown to be^(7,17)

$$f_0 = \frac{0.276}{\sigma_\tau} \approx \frac{1}{5\sigma_\tau} \quad (9.1.22)$$

The two degradation criteria mentioned above for frequency-selective and flat fading can also be expressed in terms of the coherence and signal bandwidths:

1. A channel is referred to as frequency-selective if $f_0 < B$, where B is the signal bandwidth. Since the channel coherence bandwidth is smaller than the signal bandwidth, the channel acts as a filter, hence frequency-selective fading occurs.
2. Frequency-nonselective or flat fading degradation occurs whenever $f_0 \gg B$. As noted before, flat fading does not introduce ISI, but performance degradation occurs due to low SNR whenever the signal is fading. It should be noted that frequency selectivity diminishes as f_0/B is increasingly larger than 1. Therefore, the tolerable frequency selectivity depends on the application. For example, the line-of-sight Rummier channel model described in Section 9.5 is considered frequency-selective although $f_0/B \approx 8$.

Flat fading is not always desirable. For example, to achieve frequency diversity for two fading signals, it is necessary that the carrier spacing f_s between the two signals is larger than the coherence bandwidth, $f_s > f_0$, so that the two signals are uncorrelated.

9.1.3.4.3. The Time-Varying Channel. In this and the next section we will describe the properties of the channel as they relate to its time-varying nature. For mobile radio applications, the channel is time-varying because the motion between the transmitter and receiver results in propagation path changes. It should be noted that since the channel characteristics are dependent on the relative positions of the transmitter and receiver, time variance is equivalent to space variance. As mentioned previously, the time variation of the channel is characterized by the *Doppler power spectrum* $S(\nu)$, shown in Figure 9.7c.

Although the phenomena we discuss apply to any time-variant model, for the sake of simplicity we will present our exposition based on the commonly used *dense scatterer* model described above. This model has a *Doppler power spectrum* (shown in Figure 9.8a)^(1,16)

$$S(\nu) = \frac{1}{\pi f_d \sqrt{1 - (\nu/f_d)^2}}, \quad |\nu| \leq f_d \quad (9.1.23)$$

The Spaced-Time Correlation Function. The *spaced-time correlation function* $\rho(\Delta t)$ is the inverse Fourier transform of $S(\nu)$ (shown in Figure 9.7d). For the dense scatterer model (see Figure 9.8b)⁽²⁾

$$\rho(\Delta t) = F^{-1}(S(\nu)) = J_0(2\pi f_d \Delta t) \quad (9.1.24)$$

where $J_0(\cdot)$ is the zeroth-order Bessel function of the first kind.

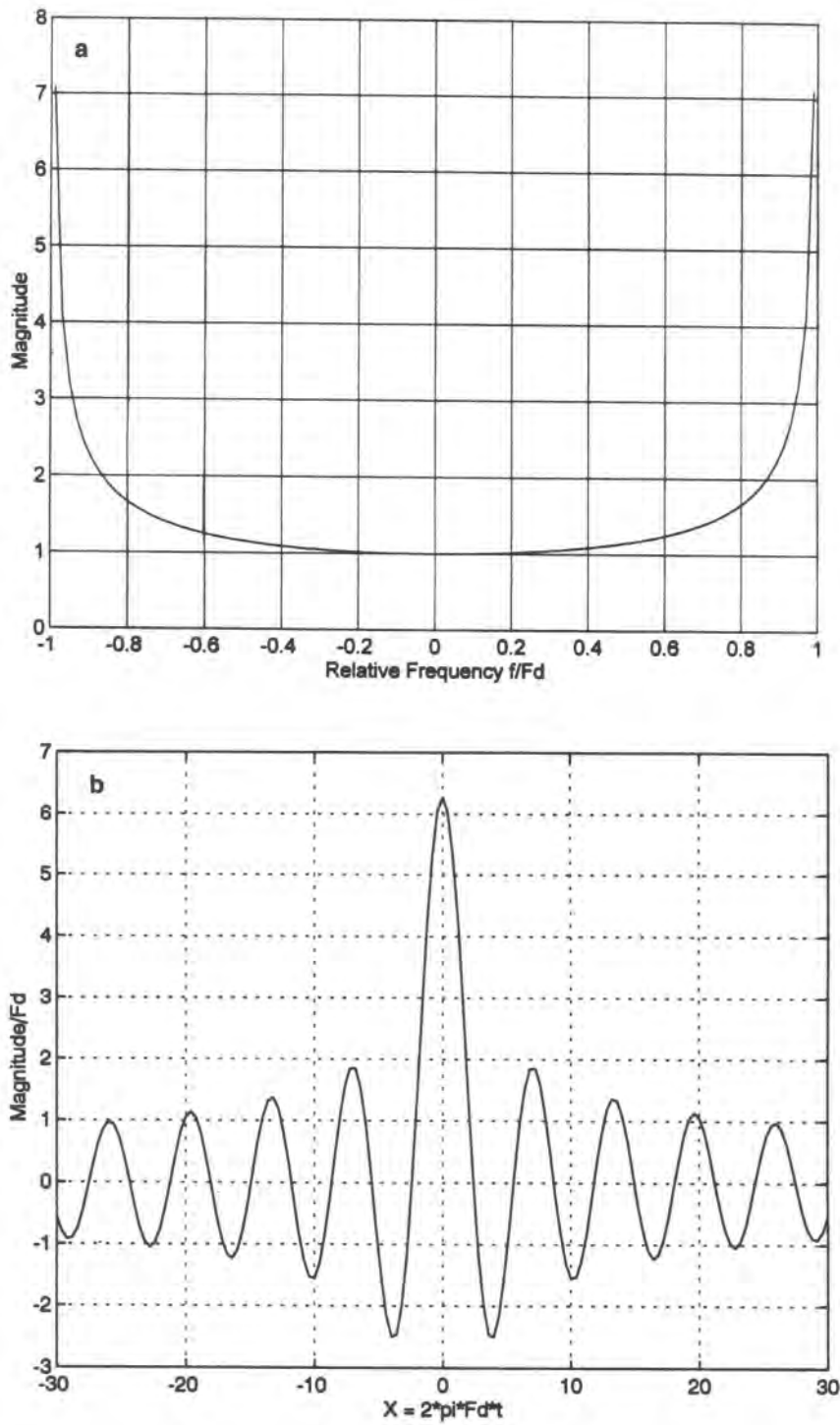


Figure 9.8. The Jakes model. (a) Power spectrum; (b) autocorrelation function.

The function $\rho(\Delta t)$ specifies the correlation between the channel's response to a narrowband (CW) signal sent at times t_1 and t_2 , where $\Delta t = t_2 - t_1$.⁽⁴⁾ The *coherence time* T_0 is the expected time duration within which the two signals remain correlated. If the channel is time-invariant, $\rho(\Delta t) = 1$. Coherence time can also be measured in distance traversed.

The time-variant behavior is categorized into *fast fading* and *slow fading*:

1. A channel is said to be fast fading if $T_0 < T_{\text{sym}}$, where T_0 is the channel coherence time and T_{sym} is the symbol time. During fast fading the baseband symbol shapes can be severely distorted, which often results in an irreducible BER and synchronization problems.
2. A channel is referred as slow fading if $T_0 > T_{\text{sym}}$. The time duration that the channel remains correlated is long compared to the transmitted symbol. The primary degradation in a slow fading channel is the loss of SNR.

If a channel is slow fading, it can be regarded as quasistatic, and the snapshot approach can be used for simulating the channel for performance estimation. See Case Study I in Chapter 12.

Doppler Power Spectrum. The Doppler power spectrum in (9.1.23) and in Figure 9.8a has been shown to match experimental data gathered for outdoor mobile radio channels.⁽¹⁾ For indoor channels a flat spectrum is used for $S(\nu)$.

The maximum Doppler frequency is

$$f_d = \frac{v}{\lambda} \quad (9.1.25)$$

where v is the velocity of the mobile and λ is the signal wavelength.

The Doppler power spectrum of the channel yields knowledge about spectral broadening of a narrowband signal (impulse in frequency) in the Doppler frequency domain. It can be regarded as a dual of $p(\tau)$, since the latter yields knowledge about the spreading of a pulse in the time domain.

$S(\nu)$ enables us to estimate the broadening imposed on the signal as a result of the channel time variations. The width of the Doppler power spectrum f_d is referred to as the *spectral broadening* or *Doppler spread*, and is also called the *fading bandwidth* of the channel. Because $S(\nu)$ and $\rho(\Delta t)$ are related through the Fourier transform, the coherence time and the Doppler spread are inversely related as

$$T_0 \propto \frac{1}{f_d} \quad (9.1.26)$$

Another popular rule of thumb is to define T_0 from the following consideration. Figure 9.9 shows the envelope of a Rayleigh-faded 900-MHz signal as a function of time. The distance between two nulls of the fading signal is half a wavelength (which is approximately the distance where the signal remains correlated). Thus the time required to traverse a distance $\lambda/2$ when traveling at velocity v is

$$T_0 \approx \frac{\lambda/2}{v} = \frac{0.5}{f_d} \quad (9.1.27)$$

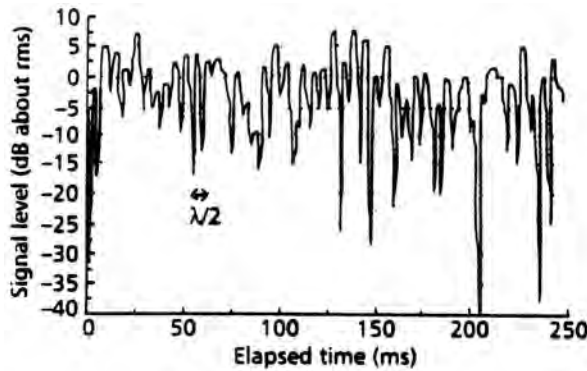


Figure 9.9. A typical Rayleigh fading envelope at 900 MHz, receiver speed, 120 km/h (from Ref. 5, © IEEE, 1997).

The degradation criteria for fast and slow fading in terms of the signal bandwidth $B \approx 1/T_{\text{sym}}$ and the fading rate $f_d \approx 1/T_0$ can be stated as follows:

1. A channel is considered fast fading if $B < f_d$. In this case, the signal can be severely distorted.
2. A channel is regarded as slow fading if $B > f_d$. No signal distortion is present, but degradation of the SNR is possible.

Wideband Channel Measurements. The multipath intensity profile is usually measured by probing the channel with a wideband RF waveform where the modulated signal is a high-rate PN sequence.⁽¹⁸⁾ By cross-correlating the receiver output against delayed versions of the PN sequence and measuring the average value of the correlator output, one can obtain the power versus delay profile. For measurements for mobile radio applications with a fixed base station and mobile user, the power delay profile is measured in short-distance increments of fractions of a wavelength. The correlation measurements made as a function of position, i.e., the spatial correlation function, can be converted into a temporal correlation function noting that $\Delta \mathbf{x} = \mathbf{v} \Delta t$, where $\Delta \mathbf{x}$ is the incremental spatial movement of the mobile and $\mathbf{v}$ is the velocity. Thus, for any vehicle speed the Doppler power spectrum can be obtained by transforming the temporal correlation function.

9.1.3.5. Structural Models for Multipath Fading Channels

In the preceding sections we looked at the statistical characterization of multipath channels. The characterization does not in itself provide a constructive way of emulating the channel. For this purpose we need to synthesize a generative model, i.e., specify explicitly the model structure and its means of implementation. In this section we shall derive such structural models.

9.1.3.5.1. Diffuse Multipath Channel Model To reiterate earlier statements, in some mobile radio channels and some other radio channels such as troposcatter channels the received signal consists of a continuum of multipath components. For mobile radio such channels can be caused by scattering and reflections from very large terrain features such as mountain ranges. We have called a channel of this type a *diffuse multipath channel*.

The complex lowpass-equivalent impulse response for the diffuse multipath channel (9.1.9) is given by

$$\tilde{c}(\tau, t) = a(\tau, t)e^{-j2\pi f_c \tau} \quad (9.1.28)$$

In analogy with (9.1.8), the received lowpass-equivalent signal is given by

$$\tilde{y}(t) = \int_{-\infty}^{\infty} \tilde{s}(t - \tau) \tilde{c}(\tau, t) d\tau \quad (9.1.29)$$

We will assume, quite reasonably, that the input to the channel is bandlimited to a bandwidth $B \propto r$ (B is the bandpass bandwidth and r is the symbol rate). We can therefore represent the lowpass input in terms of its sampled values using the minimum sampling rate of B samples per second as

$$\tilde{s}(t - \tau) = \sum_{n=-\infty}^{\infty} \tilde{s}(t - nT) \text{sinc}(B(\tau - nT)) \quad (9.1.30)$$

where $T = 1/B$ is the sampling period. The response in (9.1.29) is then

$$\begin{aligned} \tilde{y}(t) &= \int_{-\infty}^{\infty} \left[\sum_{n=-\infty}^{\infty} \tilde{s}(t - nT) \text{sinc}(B(\tau - nT)) \right] \tilde{c}(\tau, t) d\tau \\ &= \sum_{n=-\infty}^{\infty} \tilde{s}(t - nT) \int_{-\infty}^{\infty} \tilde{c}(\tau, t) \text{sinc}(B(\tau - nT)) d\tau \end{aligned} \quad (9.1.31)$$

The last expression can be stated in the form

$$\tilde{y}(t) = \sum_{n=-\infty}^{\infty} \tilde{s}(t - nT) \tilde{g}_n(t) \quad (9.1.32a)$$

where

$$\tilde{g}_n(t) = \int_{-\infty}^{\infty} \tilde{c}(\tau, t) \text{sinc}(B(\tau - nT)) d\tau \quad (9.1.32b)$$

By examining equation (9.1.32), we can make some observations:

1. The function $\tilde{y}(t)$ can be generated by passing $\tilde{s}(t)$ through a *tapped delay line* (TDL) with taps $\tilde{g}_n(t)$ spaced T seconds apart.
2. The functions $\tilde{g}_n(t)$, $-\infty < t < \infty$, are defined by (9.1.32b). This weighted integration is shown in Figure 9.10.⁽¹⁹⁾ We see that if the impulse response has length NT , then $\tilde{g}_n(t)$ in (9.1.32b) will essentially be negligibly small for $n < 0$ and $n > N$ if we assume the integral is dominated by the main lobe of the sinc function. Therefore (9.1.32a) can be approximated by

$$\tilde{y}(t) = \sum_{n=0}^N \tilde{s}(t - nT) \tilde{g}_n(t) \quad (9.1.33)$$

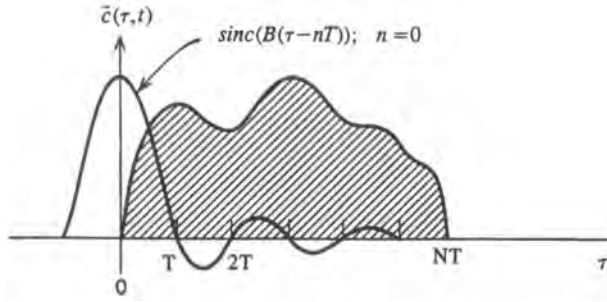


Figure 9.10. Illustration of the integration process for obtaining the tap gains of the tapped-delay-line model (from Ref. 19, © John Wiley & Sons, 1968, reprinted with permission).

3. Another interpretation of $\tilde{g}_n(t)$ is enlightening. Since $\text{sinc}(B\tau)$ is the impulse response of an ideal (brickwall) filter with bandwidth $B/2$ (where B is the RF bandwidth), $\tilde{g}_n(t)$ can be viewed as the filtered version of the channel impulse response sampled at multiples of T . Of course, this is a consequence of our original assumption on the bandlimitedness of the signal. All this follows from the exposition in Section 4.2 dealing with the TDL formulation for time-varying channels. If further we assume that $\tilde{c}(\tau, t)$ is fairly smooth over T , then (9.1.32b) can be approximated by

$$\tilde{g}_n(t) \approx T\tilde{c}(nT, t) \quad (9.1.34)$$

where $T = 1/B$.

These observations lead us to the tapped delay-line channel model in Figure 9.11.

■ *Remark.* From the point of view of model complexity, an important consideration is our selection of B , since B^{-1} dictates the tap spacing and therefore the number of taps. Of course, a physical signal cannot be strictly bandlimited, but we can define an effective bandwidth B based on some criterion, such as 99% included energy. In effect we have already made such a choice when we select the simulation interval T_s . The corresponding $B = T_s^{-1}$; recall that B here is passband bandwidth. Hence the taps in this case would be spaced T_s apart. On the other hand, the criterion for T_s , which is normally control of aliasing error, may be too stringent for purposes of the channel model. Thus, it could be legitimate to choose a larger tap spacing, but for computational ease, it should be taken as an integer multiple of T_s . ■

Statistical Tap-Gain Models. If $\tilde{c}(\tau, t)$ is assumed to be a Gaussian random process in t , as we assume here, then because (9.1.32b) is a linear operation on $\tilde{c}(\tau, t)$, the tap gains $\tilde{g}_n(t)$ are also sample functions of a complex zero-mean Gaussian process. In general, however,

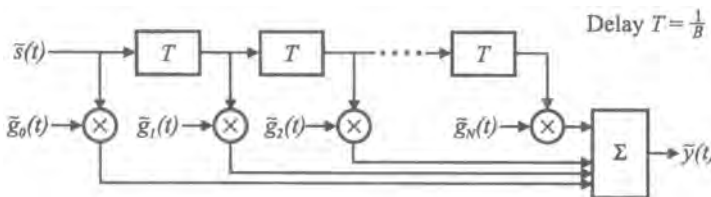


Figure 9.11. Tapped-delay-line model for diffuse multipath channels.

these tap-gain functions are correlated.^(19,20) To specify the model completely, we need their cross-covariance functions. From (9.1.32b) we define the tap-gain cross-covariances as

$$\begin{aligned} R_{kl}(\Delta t) &= E[\tilde{g}_k(\tau, t)\tilde{g}_l^*(\tau, t + \Delta t)] \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} E[\tilde{c}(\tau, t)\tilde{c}^*(\mu, t + \Delta t)] \\ &\quad \times \text{sinc}(B(\tau - kT)) \text{sinc}(B(\mu - lT)) d\tau d\mu \end{aligned} \quad (9.1.35)$$

By the uncorrelated scattering assumption [see (9.1.13) and (9.1.14)] the expectation on the integrand reduces to $R_{\tilde{c}}(\tau, \Delta t)\delta(\tau - \mu)$. Hence we obtain

$$R_{kl}(\Delta t) = \int_{-\infty}^{\infty} R_{\tilde{c}}(\tau, \Delta t) \text{sinc}(B(\tau - kT)) \text{sinc}(B(\tau - lT)) d\tau \quad (9.1.36)$$

Further discussion on particular cases of (9.1.36) and its implications in the simulation model is provided below.

a. *Uncorrelated Tap-Gain Model.* A channel cannot have an internal bandwidth constraint and be a WSSUS channel. However, the effect of a WSSUS channel on a bandlimited signal will be unchanged if the channel is preceded by an ideal bandpass filter having a bandwidth no smaller than B . This point was already implied in our discussion of $\tilde{g}_n(t)$ above.

As will become clear shortly, the simulation model is considerably simplified if we assume that the tap gains are uncorrelated, i.e., $R_{kl}(\Delta t) = 0$ for $k \neq l$. One condition for which this would be true is if $R_{\tilde{c}}(\tau, \Delta t)$ is constant for all τ since the sinc functions in (9.1.36) are orthogonal. This condition is consistent with the assumed “smoothness” of $\tilde{c}(\tau, t)$ justifying the assumption (9.1.34). The degree to which R_{kl} is near zero for $k \neq l$ will depend, of course, on the specific nature of $R_{kl}(\tau, \Delta t)$. Let us proceed for the moment on the assumption that R_{kl} is close enough to zero for $k \neq l$ that the tap gains can be considered uncorrelated. As mentioned, this is consistent with the approximation

$$\tilde{y}(t) = \sum_{n=0}^N \tilde{s}(t - nT)\tilde{g}_n(t) \approx T \sum_{n=0}^N \tilde{s}(t - nT)\tilde{c}(nT, t) \quad (9.1.37)$$

where $T = 1/B$.

Recalling (9.1.16), in the approximate case at hand we have

$$E[|\tilde{g}_n(t)|^2] = T^2 E[|\tilde{c}(nT, t)|^2] = T^2 p(nT) \quad (9.1.38a)$$

where $p(\tau)$ is the delay power profile (Figure 9.12). We can thus use this measurement to calibrate the tap gains. Since $\tilde{g}_n(t)$ is a zero-mean complex Gaussian process, we can write $\tilde{g}_n(t) = g_{n,r}(t) + jg_{n,i}(t)$, where $g_{n,r}$ and $g_{n,i}$ are both real Gaussian processes. Suppose we call the standard deviation of each σ_n ; then

$$E[|\tilde{g}_n(t)|^2] = 2\sigma_n^2 = T^2 p(nT) \quad (9.1.38b)$$

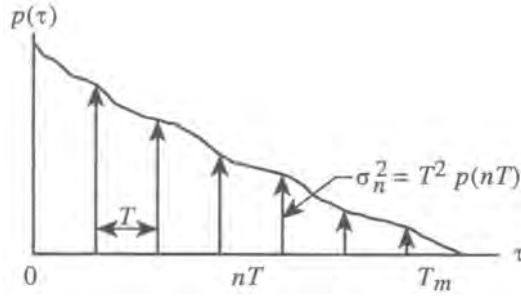


Figure 9.12. Sampled values of a delay power profile.

The utility of the uncorrelated-tap-gain assumption from the simulation standpoint is that each tap gain can be *independently* generated with a Gaussian RNG with variance given by (9.1.38b). Even when this (uncorrelated) approximation cannot be fully supported mathematically, it is convenient to use as one model for comparing different system designs. After all, there is no single “correct” model since the real channel impulse response depends on many physical variables, and the measurement of the channel response is typically an average of many measurements.

b. *Correlated Tap-Gain Models.* The approximations above led to an uncorrelated model. Without the approximation in (9.1.38a), the tap-gain functions will be correlated as in (9.1.36) and the channel is now characterized by the covariance matrix

$$\mathbf{R}(\Delta t) = \begin{bmatrix} R_{00}(\Delta t) & R_{01}(\Delta t) & \dots & R_{0N}(\Delta t) \\ R_{10}(\Delta t) & R_{11}(\Delta t) & & R_{1N}(\Delta t) \\ \vdots & & & \\ R_{N0}(\Delta t) & & & R_{NN}(\Delta t) \end{bmatrix} \quad (9.1.39)$$

where the average power of tap m is $2\sigma_m^2 = R_{mm}(0)$. We can also equate $R_{mm}(0)$ to $p(mT)$ if the delay power profile has been measured with an instrument having the same bandwidth B as has been assumed in (9.1.32b). We will discuss this point further a little below.

Generating a set of correlated random processes with an arbitrary covariance matrix as in (9.1.39) is very difficult. Here, not only do we have to generate each $\tilde{\mathbf{g}}_n(t)$ in accordance with its specified $R_{nn}(\Delta t)$, but at the same time do so to satisfy all cross-covariances $R_{kl}(\Delta t)$ for all k, l .

An approach to solving the general problem can be found in ref. 21. This method is based on fitting a vector ARMA model to tap gain processes and deriving the vector ARMA model from the given correlations and power spectral densities. The procedure for fitting the vector ARMA model is very complex and it is not clear if the extra work involved can be justified in terms of the resulting improvement in the accuracy.

An approximation, described below, that simplifies this problem makes the reasonable assumption that all tap gain functions have the same spaced-time correlation function.

Simplified Scattering Function. The Doppler PSD of each process $\tilde{\mathbf{c}}(\tau, t)$ is by definition the scattering function $\mathbf{S}(\tau, \nu)$ [see (9.1.15)] evaluated for the appropriate τ . However, because

$S(\tau, \nu)$ is often difficult to measure, a simplifying assumption is frequently made, namely, the *shape* of $S(\tau, \nu)$ is *independent* of τ ; this implies

$$S(\tau, \nu) = \psi(\tau)S(\nu) \quad (9.1.40a)$$

Then, from (9.1.17),

$$p(\tau) = \int_{-\infty}^{\infty} S(\nu) d\nu = \psi(\tau) \int_{-\infty}^{\infty} S(\nu) d\nu = a\psi(\tau)$$

An often used assumption is that both $p(\tau)$ and $S(\nu)$ are normalized to have a total power = 1; the scattering function is then given by

$$S(\tau, \nu) = p(\tau)S(\nu) \quad (9.1.40b)$$

In other words, the *delay power profile* $p(\tau)$ and the *Doppler power spectral density* $S(\nu)$ are separable.

Taking the inverse Fourier transform of (9.1.40b) with respect to ν [see (9.1.15) and (9.1.24)], we get[†]

$$R_z(\tau, \Delta t) = p(\tau)\rho(\Delta t) \quad (9.1.41)$$

where the autocorrelation has the same spaced-time correlation function for each τ , and (9.1.36) takes the form

$$R_{mn}(\Delta t) = \rho(\Delta t) \int_{-\infty}^{\infty} p(\tau) \text{sinc}(B(\tau - mT)) \text{sinc}(B(\tau - nT)) d\tau \quad (9.1.42a)$$

Assuming the tap spacing $T = B^{-1}$, we get

$$R_{mn}(\Delta t) = \rho(\Delta t) \int_{-\infty}^{\infty} p(\tau) \text{sinc}(B\tau - m) \text{sinc}(B\tau - n) d\tau \quad (9.1.42b)$$

so that the covariance matrix can now be written as

$$\mathbf{R}(\Delta t) = \mathbf{R}_0 \rho(\Delta t) \quad (9.1.43)$$

Because $p(\tau)$ is real, it is clear that the integral in (9.1.42) is real. Hence, $\mathbf{R}_0$ is real, and from the nature of the integrand, also symmetric, $R_{mn} = R_{nm}$.

The above development facilitates the construction of a practical scheme for generating the correlated tap gains $\tilde{\mathbf{g}}_n$. Since there are efficient methods for generating *independent* Gaussian processes (see Chapter 7), we are motivated to consider the generation of a set of independent processes and means of combining them in such a way that the specific covariance matrix $\mathbf{R}_{mn}(\Delta t)$ is produced. We are thus led to ask whether there exists a linear transformation $\mathbf{L}$ [an $(N+1) \times (N+1)$ matrix] such that

$$\tilde{\mathbf{g}} = \mathbf{L} \times \mathbf{Z} \quad (9.1.44)$$

[†]Loosely speaking, we can think of this simplified case as a version of a separable channel (see Section 4.2), but here in the correlation domain.

where $\tilde{\mathbf{g}} = (\tilde{g}_0(t), \dots, \tilde{g}_N(t))^T$ is the column vector of tap-gain processes and $\mathbf{Z} = (\mathbf{Z}_0(t), \dots, \mathbf{Z}_N(t))^T$ is a column vector of independent stationary complex Gaussian processes. That is, $E[\mathbf{Z}_i(t_1)\mathbf{Z}_j(t_2)] = 0$ for $i \neq j$ and any t_1, t_2 . We also stipulate that the covariance of $\mathbf{Z}_n(t)$ be of the form

$$E[\mathbf{Z}_n(t_1)\mathbf{Z}_n^*(t_2)] = \psi(\Delta t) \quad (9.1.45)$$

where $\Delta t = t_1 - t_2$ is the same for all $n = 0, 1, \dots, N$.

From (9.1.44) and (9.1.45) we have

$$E[\tilde{\mathbf{g}}(t_1)\tilde{\mathbf{g}}^\dagger(t_2)] = \mathbf{L}\psi(\Delta t)\mathbf{L}^T = \psi(\Delta t)\mathbf{L}\mathbf{L}^T \quad (9.1.46)$$

where the dagger represents complex conjugate transpose. Hence, to find the desired condition on $\mathbf{L}$, we equate (9.1.46) to (9.1.43):

$$\mathbf{R}_0\rho(\Delta t) = \psi(\Delta t)\mathbf{L}\mathbf{L}^T$$

which leads to the requirements

$$\psi(\Delta t) = \rho(\Delta t) \quad (9.1.47a)$$

$$\mathbf{R}_0 = \mathbf{L}\mathbf{L}^T \quad (9.1.47b)$$

To achieve the equality (9.1.47a), we have to shape the power spectral density of each process $\mathbf{Z}_n(t)$ to have the common Doppler spectrum $\mathcal{S}(\mathbf{v})$, which can be done by first generating a *white* process, say $\mathbf{W}_n(t)$, and passing it through a filter with transfer function $|\mathbf{H}(f)|^2 = \mathcal{S}(\mathbf{v})$ (or equivalent means) as discussed in Chapter 7 and implemented in the appendix of Case Study IV in Chapter 12 (an alternative spectral shaping technique is also considered in Case Study II).

In order to satisfy the equality (9.1.47b), $\mathbf{R}_0$ must be real symmetric, which we have shown is the case (if $\mathbf{R}_0$ were complex, the factorization would require $\mathbf{R}_0 = \mathbf{L}\mathbf{L}^\dagger$). Furthermore, if $\mathbf{R}_0$ is positive-definite, which can also be shown to be true (Problem 9.5), then the factorization (9.1.47b) can be carried out, and $\mathbf{L}$ is the upper triangular matrix

$$\mathbf{L} = \begin{bmatrix} l_{00} & l_{01} & \cdots & l_{0N} \\ 0 & l_{11} & \cdots & l_{1N} \\ \vdots & & & \\ 0 & & \cdots & l_{NN} \end{bmatrix}$$

The decomposition (9.1.47b) into upper and lower triangular matrices is called the Cholesky factorization.^(4,22,23) The resulting model is shown in Figure 9.13. We start with $N + 1$ identically and independently distributed (i.i.d.) white Gaussian complex processes. These are easy to generate. Each process is then passed through a separate filter, so that the output of each has a PSD equal to $\mathcal{S}(\cdot)$, the Doppler spectrum. The filter outputs $\mathbf{Z}_n(t)$ are transformed by $\mathbf{L}$ to produce the $\tilde{\mathbf{g}}_n(t)$.

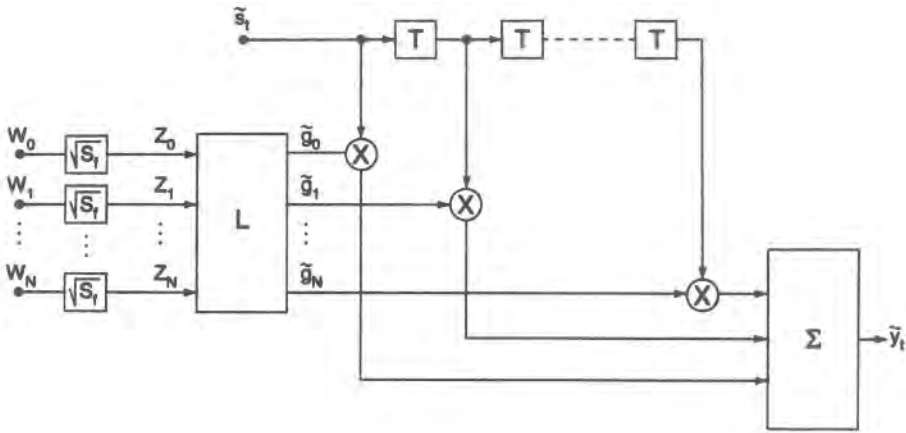


Figure 9.13. Simulation model for a particular diffuse multipath channel having correlated tap gain.

c. Filtered Delay Power Profile and Doppler Spectrum. The delay power profile and the Doppler spectrum are important properties of the channel that are central to the simulation of the model, as will be discussed in more detail below and in the next section. For example, we saw in (9.1.38) that in the situation applicable there, the strength of the tap gains is proportional to the delay power profile.

We also observed in the discussion surrounding (9.1.32) and (9.1.33) that the tap gains $\tilde{g}_n(t)$ in the tapped-delay-line model are *by construction* filtered versions of the “true” channel impulse response, the filter being an ideal brickwall filter with bandwidth equal to that of the assumed bandlimitation of the signal. The channel is therefore effectively seen through this filter. In fact, no measurement system has infinite bandwidth, so any measurements of the delay power profile or the Doppler spectrum are inherently filtered to some degree. We therefore want to investigate how the true channel properties are modified by a filtered measurement. As a related issue, suppose we want to apply a set of channel measurements obtained with a relatively wideband measuring instrument to a much narrower bandwidth system. We will also want to filter this already partially filtered channel response in order to obtain a more efficient (longer tap spacing) model. In fact, as we will see, we can *analytically* filter a measured response with any hypothetical filter whose bandwidth is less than that of the measuring instrument.

Consequently, suppose we have a measured power delay profile $p(\tau)$ [or more generally a measured channel correlation function $R_{\tilde{c}}(\tau, \Delta t)$]. Suppose we want to simulate the effect of this channel on a signal with bandwidth B . For purposes of this discussion, we initially assume the bandwidth of the measuring instrument B_M is more or less arbitrary, but of course greater than B .

Let $\tilde{h}(t)$ be the complex lowpass impulse response of such a filter. Then the complex lowpass impulse response of the filtered channel is

$$\tilde{c}_h(\tau, t) = \tilde{c}_M(\tau, t) * \tilde{h}(\tau) \quad (9.1.48)$$

where $\tilde{c}_M(\tau, t)$ is the measured channel response, and the corresponding channel correlation function is

$$R_{\tilde{c}_h}(\tau, \Delta t) = E[\tilde{c}_h(\tau, t)\tilde{c}_h^*(\tau, t + \Delta t)] \quad (9.1.49)$$

We now evaluate $R_{\tilde{c}_h}(\tau, \Delta t)$ in terms of $R_{\tilde{c}}(\tau, \Delta t)$,

$$\begin{aligned} R_{\tilde{c}_h}(\tau, \Delta t) &= E \left\{ \int_{-\infty}^{\infty} \tilde{c}_M(s, t) \tilde{h}(\tau - s) ds \int_{-\infty}^{\infty} \tilde{c}_M^*(u, t + \Delta t) \tilde{h}^*(\tau - u) du \right\} \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} E[\tilde{c}_M(s, t) \tilde{c}_M^*(u, t + \Delta t)] \tilde{h}(\tau - s) \tilde{h}^*(\tau - u) ds du \end{aligned} \quad (9.1.50)$$

To make the evaluation of (9.1.50) expeditious, let us now assume $B_M \rightarrow \infty$, i.e., the measurement is unfiltered (we will return to this point below). This means $\tilde{c}_M(\tau, t) \rightarrow \tilde{c}(\tau, t)$, and we can invoke the uncorrelated scattering assumption, so that

$$E[\tilde{c}(s, t) \tilde{c}^*(u, t + \Delta t)]$$

is nonzero only if $s = u$. Hence

$$R_{\tilde{c}_h}(\tau, \Delta t) = R_{\tilde{c}}(\tau, \Delta t) * |\tilde{h}(\tau)|^2 \quad (9.1.51)$$

In particular, if $\Delta t = 0$, we have the delay power profile

$$p_h(\tau) = p(\tau) * |\tilde{h}(\tau)|^2 \quad (9.1.52)$$

In the frequency domain this is equivalent to multiplying $P(\Delta f)$, which is the transform of $p(\tau)$, by $H(f) * H(-f)$, where $H(f)$ is the Fourier transform of $\tilde{h}(\cdot)$. Therefore, we can equate the average power of the m th tap with corresponding value of the *filtered* delay power profile, $p_h(\tau_m) = R_{mm}(0) = 2\sigma_m^2$. The number of taps can be easily determined from the plot of $p_h(\tau)$, and is usually small, typically in the range 3–10, but this will of course depend on the tap spacing.

Turning now to the question of the filtered Doppler spectrum, let us first define the filtered scattering function $S_h(\tau, \nu)$ given by

$$S_h(\tau, \nu) = F_{\Delta t}[R_{\tilde{c}_h}(\tau, \Delta t)],$$

which, by applying this definition, results in (Problem 9.6)

$$S_h(\tau, \nu) = S(\tau, \nu) * |\tilde{h}(\tau)|^2 \quad (9.1.53)$$

Hence, in general, for a given τ , the filtered Doppler spectrum $S_h(\tau, \nu)$ has a different shape than $S(\tau, \nu)$. For the simplified scattering function of the previous section. (9.1.40b), we find that (9.1.53) becomes

$$S_h(\tau, \nu) = S(\nu)p(\tau) * |\tilde{h}(\tau)|^2 = S(\nu)p_h(\tau) \quad (9.1.54)$$

In this case, the shape of the Doppler spectrum remains the same for all τ , a fact that maintains the intended simplicity of the original assumption.

We shall now return to a point hinted at above, that in practice B_M cannot be infinite. If it is quite large, perhaps we can still invoke the uncorrelated scattering assumption. Otherwise

the result (9.1.51) cannot be obtained directly. However, since the measurement $\tilde{c}_M(\tau, t)$ has been filtered by the measuring instrument's response $h_M(t)$, then we can say

$$\tilde{c}_M(\tau, t) = \tilde{c}(\tau, t) * \tilde{h}_M(\tau) \quad (9.1.55)$$

If now we want to filter this measured response, we have

$$\tilde{c}_h(\tau, t) = \tilde{c}_M(\tau, t) * \tilde{h}(\tau) = \tilde{c}(\tau, t) * \tilde{h}_M(\tau) * \tilde{h}(\tau) \quad (9.1.56)$$

When we substitute (9.1.56) into (9.1.50), we obtain

$$\begin{aligned} p_h(\tau) &= p(\tau) * |\tilde{h}_M(\tau) * \tilde{h}(\tau)|^2 \\ &= p(\tau) * |\tilde{h}_M(\tau)|^2 * |\tilde{h}(\tau)|^2 = p_M(\tau) * |\tilde{h}(\tau)|^2 \end{aligned} \quad (9.1.57)$$

where we recognize $p_M(\tau)$ as the measured power delay profile. Then, indeed, the filtered delay profile is the filtered version of *whatever* is measured.

■ **Example 9.1.1.** Diffuse Multipath Model. To illustrate the details of a diffuse multipath channel model, let us look at the exponential diffuse channel model that is often used in simulations for comparative performance evaluations of different systems:

$$p(\tau) = \frac{1}{T} e^{-0.4\tau/T}, \quad 0 \leq \tau \leq 4$$

where the tap spacing $T = 1$.

1. The magnitudes of the tap-gain processes for the uncorrelated approximation as in (9.1.49) are

$$|\tilde{g}_0| = 1.0, \quad |\tilde{g}_1| = 0.82, \quad |\tilde{g}_2| = 0.67, \quad |\tilde{g}_3| = 0.55, \quad |\tilde{g}_4| = 0.37$$

2. The correlated magnitudes of the tap gains for the above model is computed from the covariance matrix $\mathbf{R}_0$ in (9.1.42b),

$$R_{mn}(0) = \sum_{k=0}^K p(k\Delta\tau) \operatorname{sinc}\left[\frac{k\Delta\tau}{T} - m\right] \operatorname{sinc}\left[\frac{k\Delta\tau}{T} - n\right] \Delta\tau$$

where K is usually chosen so that $p(\tau)$ is sampled at 10–20 samples per symbol. The number of taps N (which is the rank of the covariance matrix) can be easily determined from the filtered delay profile $p_h(\tau)$ in (9.1.52) and is shown in Figure 9.14. Evaluating the above sum

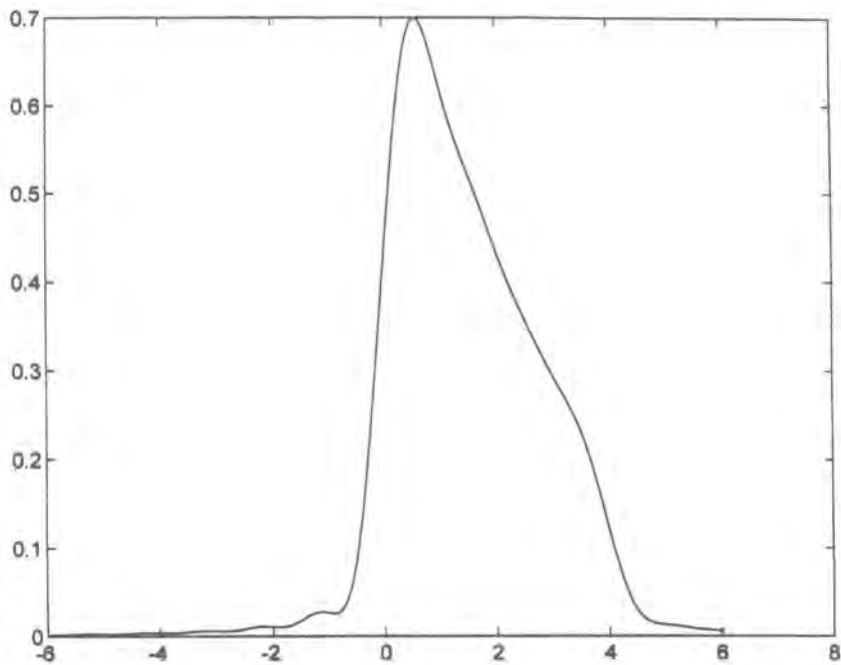


Figure 9.14. Filtered delay power profile for Example 9.1.1.

for $N = 8$, we obtain the following covariance matrix:

$$\mathbf{R}_0 = \begin{bmatrix} 0.0105 & -0.0165 & 0.0520 & 0.0136 & -0.0172 & 0.0158 & -0.0161 & 0.0092 \\ -0.0165 & 0.0264 & -0.0875 & -0.0287 & 0.0285 & -0.0239 & 0.0226 & -0.0135 \\ 0.0520 & -0.0875 & 0.4281 & 0.1448 & -0.0865 & 0.0610 & -0.0501 & 0.0337 \\ 0.0136 & -0.0287 & 0.1448 & 0.6215 & 0.0281 & -0.0190 & 0.0185 & -0.0059 \\ -0.0172 & 0.0285 & -0.0865 & 0.0281 & 0.4342 & 0.0099 & -0.0137 & -0.0015 \\ 0.0158 & -0.0239 & 0.0610 & -0.0190 & 0.0099 & 0.2938 & 0.0174 & 0.0073 \\ -0.0161 & 0.0226 & -0.0501 & 0.0185 & -0.0137 & 0.0174 & 0.1194 & -0.0320 \\ 0.0092 & -0.0135 & 0.0337 & -0.0059 & -0.0015 & 0.0073 & -0.0320 & 0.0131 \end{bmatrix}$$

The matrix $\mathbf{L}$ is evaluated using the Cholesky factorization of the matrix $\mathbf{R}_0$:

$$\mathbf{L} = \begin{bmatrix} 0.1024 & -0.1615 & 0.5077 & 0.1329 & -0.1683 & 0.1541 & -0.1567 & 0.0897 \\ 0 & 0.0186 & -0.2931 & -0.3885 & 0.0707 & 0.0550 & -0.1474 & 0.0545 \\ 0 & 0 & 0.2907 & -0.1255 & 0.0677 & -0.0038 & -0.0472 & 0.0141 \\ 0 & 0 & 0 & 0.6612 & 0.1308 & -0.0281 & -0.0361 & 0.0078 \\ 0 & 0 & 0 & 0 & 0.6158 & 0.0583 & -0.0352 & 0.0126 \\ 0 & 0 & 0 & 0 & 0 & 0.5127 & 0.0985 & -0.0195 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2422 & -0.0271 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0224 \end{bmatrix}$$

The correlated tap gains $\tilde{\mathbf{g}}_k$ are generated from independent Gaussian processes $\mathbf{Z}_k$, using the above matrix $\mathbf{L}$, by

$$\tilde{\mathbf{g}} = \mathbf{L} \times \mathbf{Z} \quad \blacksquare$$

Sampling. An aspect of the tapped delay line model that deserves additional attention is related to the sampling rate of simulations. In simulations we use sampled values of the continuous input $\tilde{s}(t)$ and output $\tilde{y}(t)$. The sampling rate used is usually 8–16 times the bandwidth, where the bandwidth includes the effect of spreading due to the time-varying nature of the system. Thus, the bandwidth at the output of the TDL is $B_y = B_s + f_D$ (see Section 4.2), where B_s is the input bandwidth and f_D is the maximum Doppler frequency. It should be pointed out, however, that in the majority of situations $B_s \gg f_D$, so that the increase in sampling rate will be minimal.

9.1.3.5.2. Discrete Multipath Channel Model. If a multipath channel is composed of a set of discrete resolvable components that originate as reflections or scattering from smaller structures, e.g., houses, small hills, etc., it is called a discrete multipath channel. The model in its most general form has, in addition to variable tap gains, variable delays and a variable number of taps. This model is applicable mostly to rapidly changing environments.

The lowpass-equivalent impulse response of a discrete multipath channel is given in (9.1.7) as

$$\tilde{c}(\tau, t) = \sum_{k=1}^{K(t)} \tilde{a}_k(\tau_k(t), t) \delta(\tau - \tau_k(t)) \quad (9.1.58a)$$

with the corresponding lowpass-equivalent output

$$\tilde{y}(t) = \sum_{k=1}^{K(t)} \tilde{a}_k(\tau_k(t), t) \tilde{s}(t - \tau_k(t)) \quad (9.1.58b)$$

For many channels it can be assumed as a reasonable approximation that the number of discrete components is constant and the delay values vary very slowly and can also be assumed constant. These assumptions are also often made for “reference” channels that are used for system studies. The model then simplifies to

$$\tilde{c}(\tau, t) = \sum_{k=1}^K \tilde{a}_k(t) \delta(\tau - \tau_k) \quad (9.1.59a)$$

with the lowpass-equivalent output

$$\tilde{y}(t) = \sum_{k=1}^K \tilde{a}_k(t) \tilde{s}(t - \tau_k) \quad (9.1.59b)$$

and is also represented as a tapped delay line, as shown in Figure 9.15.

Filtered Discrete Channel Models for Simulation. The simulation of the discrete model through (9.1.36) is quite straightforward. However, sometimes this straightforward modeling may result in inefficient simulation. The problem arises when the differential delays are small compared to the simulation sampling time T_s or are not integer multiples of T_s , or would

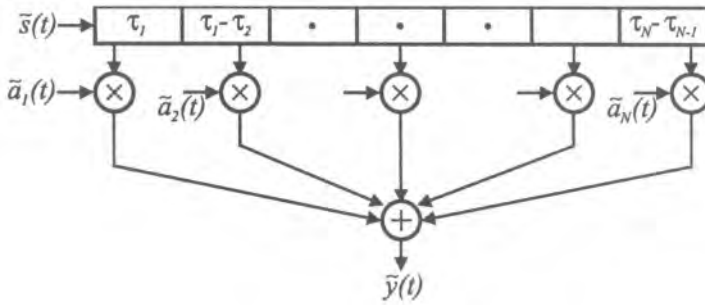


Figure 9.15. A variable-delay tapped-delay-line model for discrete multipath channels.

otherwise be numerous. In such a case it is advantageous to bandlimit the channel by design to obtain better simulation properties, namely a uniformly spaced tapped delay line. As discussed earlier, the bandlimiting filter is an ideal rectangular filter with bandwidth equal to that of the signal.

Applying the bandlimiting method used earlier for the diffuse channel, we obtain from (9.1.58b) the tap gains of a uniformly spaced TDL:

$$\tilde{g}_n(t) = \int_{-\infty}^{\infty} \tilde{c}(\tau, t) \text{sinc}(B(\tau - nT)) d\tau \quad (9.1.60)$$

Substituting the impulse response of the discrete channel from (9.1.59a) yields the tap gains

$$\tilde{g}_n(t) = \sum_{k=1}^K \tilde{a}_k(t) \text{sinc}(B(\tau_k - nT)) = \sum_{k=1}^K \tilde{a}_k(t) \alpha(k, n), \quad -N \leq n \leq N \quad (9.1.61a)$$

where, with $T = B^{-1}$

$$\alpha(k, n) = \text{sinc}\left[\frac{\tau_k}{T} - n\right] \quad (9.1.61b)$$

The $\alpha(k, n)$ decrease quite rapidly. Therefore, the number of taps needed by the bandlimited model is usually small. One way to determine the number of taps is to estimate the bandlimited delay power profile from (9.1.51) and determine the maximum delay power spread T_m at which the magnitude of the delay spread is still relevant. The model for the bandlimited discrete multipath channel is thus as shown in Figure 9.16.

The generation of the tap-gain processes for the discrete multipath channel model is straightforward. We start with a set of K independent, zero-mean, complex Gaussian white noise processes that are filtered to produce the appropriate Doppler spectrum, then scale them to produce the desired amplitude (average power) of the discrete channel components and transform them using equation (9.1.61) to produce the bandlimited tap gains $\tilde{g}_n(t)$.

■ **Example 9.1.2.** Discrete Multipath Model. To illustrate the details of the model, let us consider a two-path discrete model shown in Figure 9.17 with Rayleigh fading. This simple model is often used to evaluate the performance of communication systems in parametric form by varying the ratio of the normalized delay spread $\Delta\tau = (\tau_2 - \tau_1)/T$, where $T = 1/B$ is the symbol duration, and the ratio of relative powers in the two paths is $(\sigma_1/\sigma_2)^2$. If

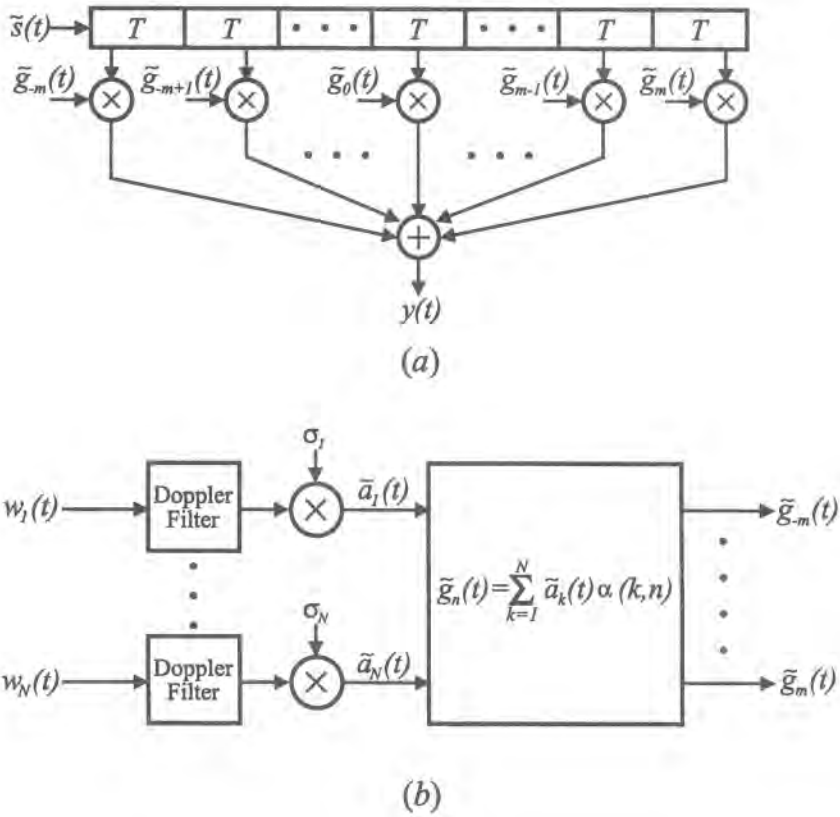


Figure 9.16. Uniformly spaced TDL model for a discrete multipath channel. (a) TDL structure; (b) generation of tap-gain processes: input processes are independent, zero-mean Gaussian processes.

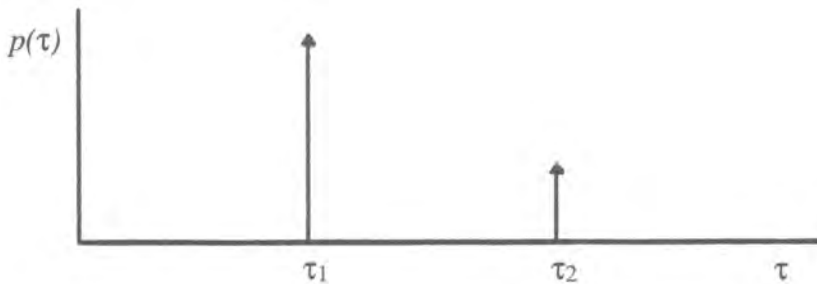


Figure 9.17. A simple two-ray model for Example 9.1.2.

$\Delta\tau \ll 1$, then the two paths can be combined and the model can be treated as frequency-nonselective; if $\Delta\tau > 0.1$ or so, there will be considerable intersymbol interference in the channel and it is treated as frequency-selective.

To illustrate the calculation of the tap-gain functions, let us assume that $\Delta\tau = 0.75$. The tap-gain functions in this case are obtained by filtering two uncorrelated white Gaussian

noise processes and then transforming them to tap-gain processes according to Equation (9.1.61) as

$$\begin{bmatrix} \tilde{g}_{-3}(t) \\ \tilde{g}_{-2}(t) \\ \tilde{g}_{-1}(t) \\ \tilde{g}_0(t) \\ \tilde{g}_1(t) \\ \tilde{g}_2(t) \\ \tilde{g}_3(t) \end{bmatrix} = \begin{bmatrix} \text{sinc}(0.0 + 3) & \text{sinc}(0.75 + 3) \\ \text{sinc}(0.0 + 2) & \text{sinc}(0.75 + 2) \\ \text{sinc}(0.0 + 1) & \text{sinc}(0.75 + 1) \\ \text{sinc}(0.0 + 0) & \text{sinc}(0.75 + 0) \\ \text{sinc}(0.0 - 1) & \text{sinc}(0.75 - 1) \\ \text{sinc}(0.0 - 2) & \text{sinc}(0.75 - 2) \\ \text{sinc}(0.0 - 3) & \text{sinc}(0.75 - 3) \end{bmatrix} \begin{bmatrix} \tilde{a}_1(t) \\ \tilde{a}_2(t) \end{bmatrix} = \begin{bmatrix} 0.0 & -0.060 \\ 0.0 & 0.082 \\ 0.0 & -0.128 \\ 1.0 & 0.300 \\ 0.0 & -0.900 \\ 0.0 & -0.180 \\ 0.0 & 0.100 \end{bmatrix} \begin{bmatrix} \tilde{a}_1(t) \\ \tilde{a}_2(t) \end{bmatrix}$$

The preceding equation shows the coefficients of the transformation for only seven taps. The reader can see that these coefficients will be negligible for higher order tap gains and hence they can be ignored and the TDL model can be truncated to seven taps. ■

9.1.3.5.3. Generation of Tap-Gain Processes The structural models considered thus far require two main steps for their implementation. First, we generate a set of white (discrete-time) Gaussian processes. The methods for doing so were discussed in detail in Chapter 7. Second, we need to shape the power spectral density of these processes to assume the shape of the Doppler spectrum at each tap location. In Chapter 7 we discussed means for generating Gaussian processes with arbitrary spectral density. Given a more or less arbitrary scattering function, we could in principle produce the shaping necessary to replicate $S(\tau, k\mathbf{B}^{-1})$, $k = 0, 1, \dots, N$. This could be a burdensome task, both in engineering manpower and computationally. For this reason, the simplified model introduced earlier is often used [as well as the fact that $S(\tau, \mathbf{v})$ may not be available]. The common Doppler spectral shape adopted in the simplified approach is also typically an even function of frequency, which means that only a real filter is necessary to do the shaping. This implies also that the real and imaginary parts of the complex white processes are independent of one another.

Thus, in the simplified model, the shaping filter (impulse response) is real and has a pure amplitude transfer function $H(f) = \sqrt{S(f)}$, where $S(f)$ is the Doppler (power) spectrum. For the reference channel models discussed latter, three spectra are specified, depending on the physical environment: a flat spectrum, a Gaussian spectrum, and Jakes spectrum. We now show how to obtain the corresponding shaping filters. The first two are rather evident, but the third is only a bit more complicated. The generic form of the model is shown in Figure 9.18.

For a flat spectrum, clearly nothing needs to be done, except for proper scaling and bandlimiting to the desired process bandwidth. If the flat spectrum is, say, $S_f(f) = A$, $|f| \leq B$, the shaping filter is simply $H_f(f) = \sqrt{A}$, $|f| \leq B$, an ideal lowpass filter.

For the Gaussian spectrum $S_G(f)$, let us say $S_G(f) = A^{-k f^2}$, with k chosen to set the desired bandwidth. The shaping filter is then $H_G(f) = \sqrt{S_G(f)} = \sqrt{A} \exp(-\frac{1}{2} k f^2)$, which is

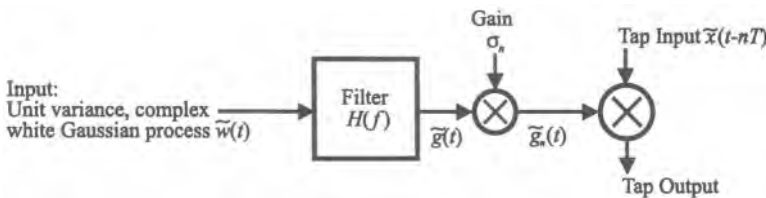


Figure 9.18. Generic block diagram for shaping the power spectrum of the nth tap-gain process.

still Gaussian, and the impulse response of which, $h_G(t)$, [inverse Fourier transform of $H_G(f)$] is also Gaussian, i.e., $h_G(t) = \sqrt{2\pi A/k} \exp[-(2\pi^2/k)t^2]$.

Finally, let us consider the Jakes spectrum [previously given in (9.1.23) and displayed in Figure 9.8a]:

$$S_J(f) = \frac{A}{[1 - (f/f_d)^2]^{1/2}} \quad (9.1.62)$$

where f_d is the largest Doppler frequency.

To obtain the shaping filter response, we have

$$S_J(f) = S_f(f)|H_J(f)|^2 \quad (9.1.63)$$

where $S_J(f)$ is the Jakes spectrum, $S_f(f)$ is the flat spectrum, and $H_J(f)$ is the Jakes frequency response.

The frequency response is then, assuming $S_f(f) = 1$,

$$H_J(f) = [S_J(f)]^{1/2} = \frac{A^{1/2}}{[1 - (f/f_d)^2]^{1/4}} \quad (9.1.64)$$

Since $H_J(f)$ is a real and symmetric function, the impulse response of the filter $h_J(t)$ can be derived from the *cosine transform*, which can be found, e.g., in Ref. 24.

The derived impulse response is then

$$\begin{aligned} h_J(t) &= F^{-1}(H_J(f)) = A^{1/2} 2^{1/4} \pi^{1/2} \Gamma(3/4) f_d x^{-1/4} J_{1/4}(x) \\ &= A^{1/2} \cdot 2.583 f_d x^{-1/4} J_{1/4}(x) \end{aligned} \quad (9.1.65)$$

where $J_{1/4}(\cdot)$ is the fractional Bessel function and $x = 2\pi f_d |t|$. Equation (9.1.65) can be well approximated by an FIR filter (see Case Study IV in Chapter 12).

The filter gain $A^{1/2}$ is chosen such that $h_J(t)$ has the normalized power of 1. The individual tap gains $\tilde{g}_n$ then have to be properly scaled to account for the different powers of the taps (see, e.g., equ. 9.1.38b). It should be noted that the bandwidth of the tap-gain processes of slowly time-varying channels will be considerably smaller than the bandwidth of the signals that flow through them. Simulation of the filter at the higher rate would likely lead to computational inefficiencies as well as stability problems. These potential difficulties can be avoided by applying multirate techniques. For example, the tap-gain filter could be modeled at a lower rate, compatible with the filter bandwidth, and interpolation would then be used at the output of the filter to produce denser samples at the rate of the signal coming into the tap. An example application is developed in Case Study IV in Chapter 12.

9.1.3.6. Indoor Wireless Channels

Fading properties of indoor wireless channels are quite different from those of mobile (vehicular) channels due to differences in the physical environment and the consequent differences in propagation mechanisms. In mobile wireless channels, 5–10 (and perhaps fewer) most significant paths are usually responsible for more than 80% of power delivered

between the transmitter and receiver antennas. Rays along those paths undergo diffraction around vertical or horizontal edges of buildings, reflection off building surfaces, ground reflection, vegetation scattering, and so forth. If the dominant path to the receiver is shadowed by an obstacle, the total power may attenuate significantly, a phenomenon that earlier we called shadow fading.

In the case of indoor wireless channels, on the other hand, the number of rays contributing to the total receiver power is much larger than in mobile wireless channels due to the fact that rays in the horizontal plane usually undergo multiple reflections and transmissions at interior or exterior walls of buildings as well as reflection off furniture. Rays in the vertical plane may also experience multiple reflections at ceilings and floors and may also bounce off furniture. Lateral diffraction is involved in propagation when the transmitter and receiver antennas are located in building hallways, but it is rare that diffracted paths overwhelm reflected or transmitted paths.

There are a number of statistical models for the indoor wireless channel, based on extensive measurements in buildings^(25–31) and also coming into use is the deterministic channel modeling technique, which combines ray tracing with a building-specific database.⁽³²⁾ By and large, proposed indoor models fall under the “discrete” multipath category, as defined earlier, and can be considered special cases of (9.1.5). However, the statistical characterization and parameter values are dependent on the specific physical environment, as might be expected.

The distinction between shadow fading and multipath (or “fast”) fading is not as appropriate in the indoor channel due to the confined environment and the much lower speed of the mobile. The latter consideration, in fact, leads to a Doppler spectrum which is modeled as flat (and generally so specified in the reference channel models for PCS; see the Appendix). In certain environments, such as an office building, propagation will almost always be “shadowed,” while in buildings with more open plans there may be either shadowed or line-of-sight (LOS) paths. However, because of the relatively low speed of the mobiles (which may also be stationary for relatively long periods of time), it is appropriate to develop models that are location-specific, depending only on whether a location is LOS to the transmitter or shadowed. In the following, a brief description of some models will be introduced as examples of modeling constructs.

9.1.3.6.1. Factory and Open-Plan-Building Model Extensive wideband measurement campaigns for indoor multipath channel modeling around 1 GHz have been made in various types of factory buildings by different workers (see, e.g., Refs. 25–28). Factories may be considered particular instances of environments that can be called “open plan,” others being warehouses, stores, supermarkets, etc. In Refs. 25 and 26, for example, the complex lowpass-equivalent channel impulse response is represented by the

$$\tilde{c}(\tau, t) = \sum_{k=1}^L a_k e^{-j\theta_k} \delta(t - \tau_k) \quad (9.1.66)$$

In (9.1.66), the path phases θ_k are *a priori* assumed to be identically and independently distributed (i.i.d.) uniformly on $[0, 2\pi]$. The other model parameters, L (the effective number of paths), the $\{a_k\}$ (the path amplitudes), and the τ_k (the path delays), are to be empirically characterized. It turns out to be convenient to develop empirical distributions in two receiver location categories: LOS and OBS (obstructed), the latter synonymous with “shadowed.” The number of multipath components L was found to have approximately a normal distribution.

The measurements also supported an empirical model for the pdf of component arrival times τ_k , which turns out to be a piecewise linear function of excess delay, assuming the multipath components in a given profile arrive independently, but with different probabilities⁽²⁶⁾ (Excess delay is simply the delay measured with respect to the first-arriving multipath component.) Amplitude fading statistics of multipath components were also obtained both in large and small scale. In a large-scale model, the cumulative distribution of the local averaged powers of individual multipath components turns out to be log-normally distributed about the mean value of local averaged powers with a standard deviation of 5.4 dB. The statistics of small-scale fading which occurs with small changes in receiver location is described by a log-normal distribution about the local mean power of individual multipath components where the standard deviation is a random variable having a distribution function⁽²⁶⁾ of exponential form.

The local averaged impulse response was used to compute path loss and multipath parameters such as mean excess delay and rms delay spread with an assumption of wide-sense stationary properties of the channel over space in the range of about 10 wavelengths. With the uniform i.i.d. assumption on the phases θ_k , the total received power will be simply the superposition of averaged power delivered along each path. This hypothesis was shown to account for the good agreement between wideband and CW path loss measurements. The total received power having a path distance of d varies randomly depending on the propagation environment between the transmitter and receiver antennas. One can express a general path loss model⁽²⁹⁾ as

$$PL(d) = PL(d_0) + 10n \log(d/d_0) + X_\sigma \quad (9.1.67)$$

where d is the distance, PL the path loss in decibels, and n the path loss index; X_σ in decibels is a random variable following approximately a Gaussian distribution with standard deviations of 5–9 dB.⁽²⁹⁾ depending on the building environment. Measured data showed that the path loss index is 2.2 on the average. Based on the measured multipath spreads ranging from 40 to 800 ns, rms delay spread was found to have values from 30 to 300 ns.⁽²⁵⁾ Measured results showed that a uniform distribution having a mean of 100 ns is a reasonable approximation.

Some sample measurements of a delay power profile are given in Figure 9.19.⁽³¹⁾ Figures 9.19a and 9.19b show single delay power profiles in a retail store. Such measurements are typically taken multiple times in the vicinity of a point, to understand the small-scale variations. Figures 9.19c and 9.19d show two such types of measurements taken in a retail store at 16 different points separated by one-quarter wavelength along a straight line in localized spots characterized as LOS and “cluttered” reception, respectively. Figure 9.20 shows some additional measurements in the same environment. A model such as (9.1.66) with appropriate statistical descriptions of the parameters will imitate measured data to the extent that the model is accurate.

9.1.3.6.2. Office Building Model In order to characterize the multipath propagation properties, wideband measurements using 10-ns, 1.5-GHz, radarlike pulses were made within a medium-size two-story office building.⁽³⁰⁾ The authors proposed a statistical model of the indoor radio channel which appears to fit the measurements very well. In this model the received signal rays arrive in clusters, where the cluster arrival times are modeled as a Poisson arrival process with some fixed rate λ . Within each cluster, subsequent rays also arrive according to a Poisson process with another fixed rate Γ with the constraint that $\lambda \gg \Gamma$.

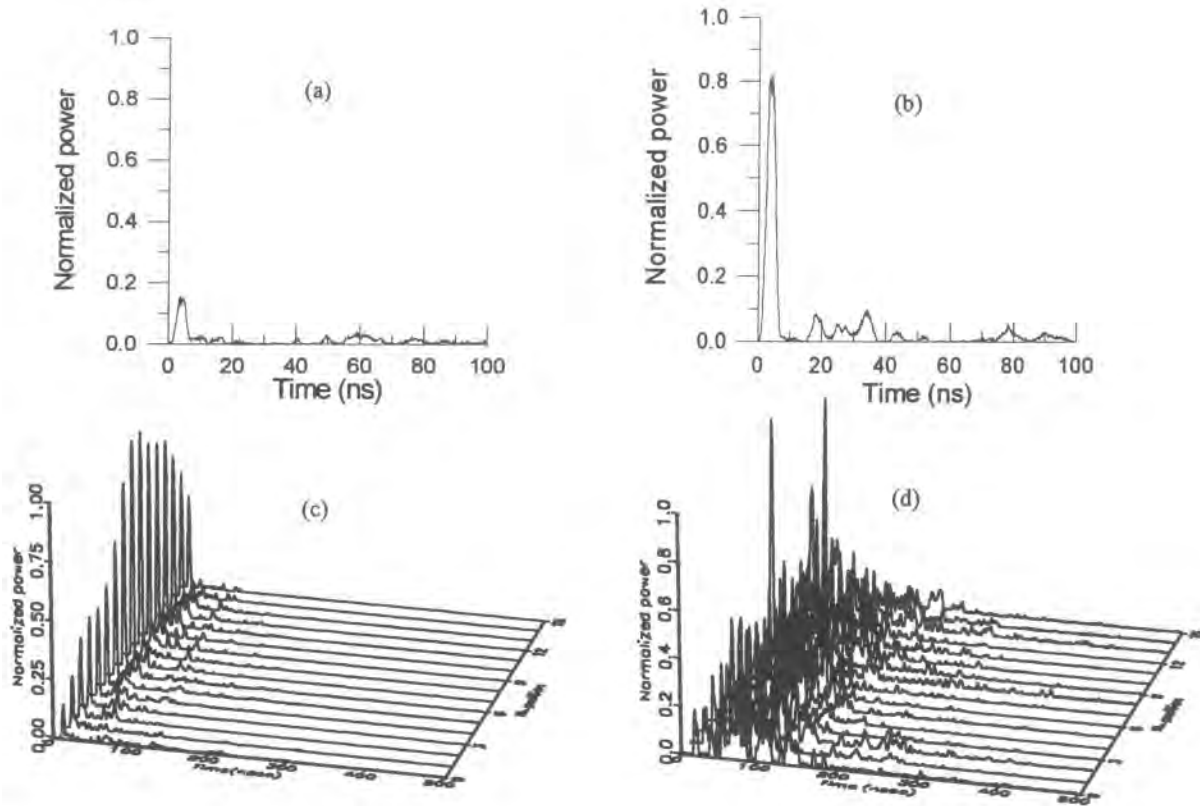


Figure 9.19. Illustration of the measurement of single delay power profiles and scattering functions in a retail store.

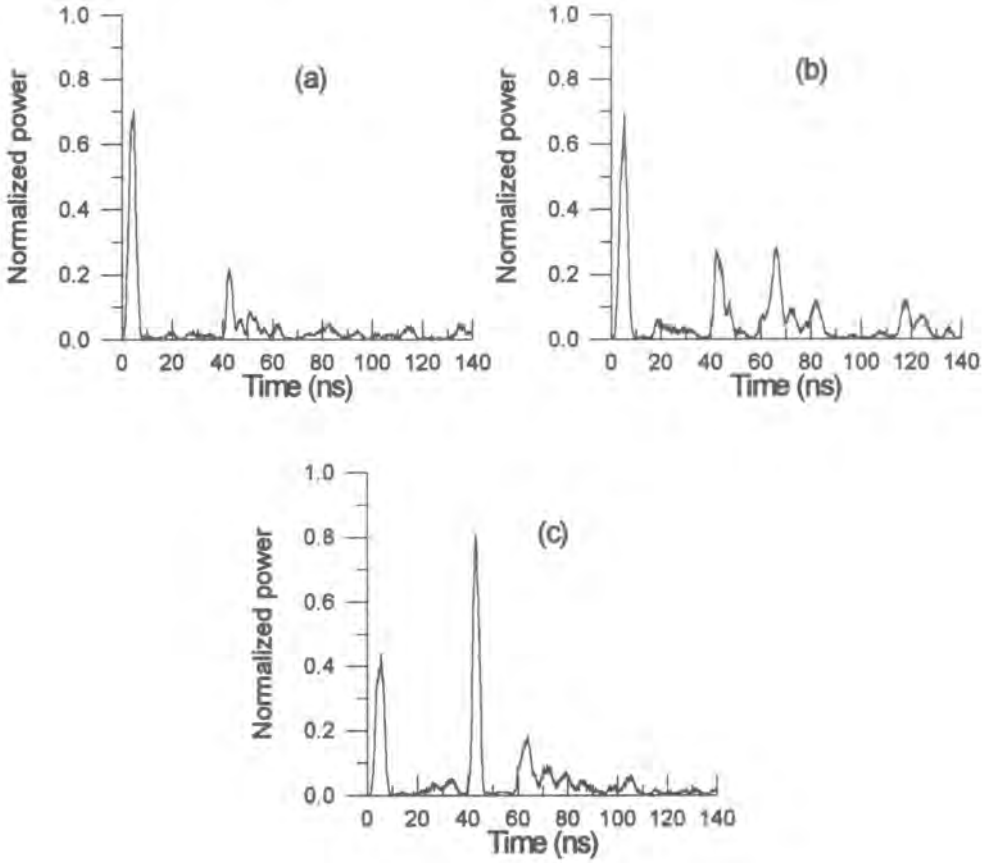


Figure 9.20. Additional measurements of delay power profiles with lightly obstructed paths in a retail store.

Hence the complex lowpass channel impulse response is given as

$$\tilde{c}(\tau, t) = \sum_{l=1}^{\infty} \sum_{k=1}^{\infty} a_{kl} e^{-j\theta_{kl}} \delta(t - T_l - \tau_{kl}) \quad (9.1.68)$$

In addition to an i.i.d. assumption on the phases θ_{kl} , uniform over $(0, 2\pi)$, the mean square values are monotonically decreasing functions of T_l and τ_{kl} as expressed in the following equation:

$$\overline{a_{kl}^2} \equiv \overline{a^2(T_l, \tau_{kl})} = \overline{a^2(0, 0)} \exp(-T_l/\Gamma) \exp(-T\tau_{kl}/\gamma) \quad (9.1.69)$$

where $\overline{a^2(0, 0)}$ is the average power of the first ray of the first cluster, and Γ and γ are power-delay time constants for the clusters and the rays, respectively. The probability distribution of the normalized amplitudes of rays is independent of the associated delays and is stated to follow the Rayleigh probability density function.⁽³⁰⁾ However, Kim *et al.*⁽³¹⁾ reported that the distribution of amplitudes is close to a Ricean distribution with K factor of 5 for LOS paths between receivers and transmitters as shown in Figure 9.21. On the other hand, the distri-

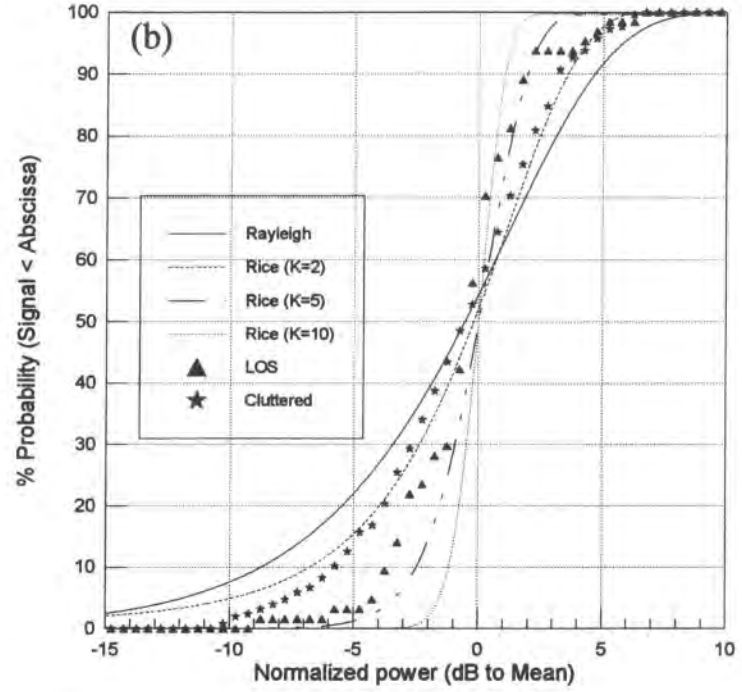
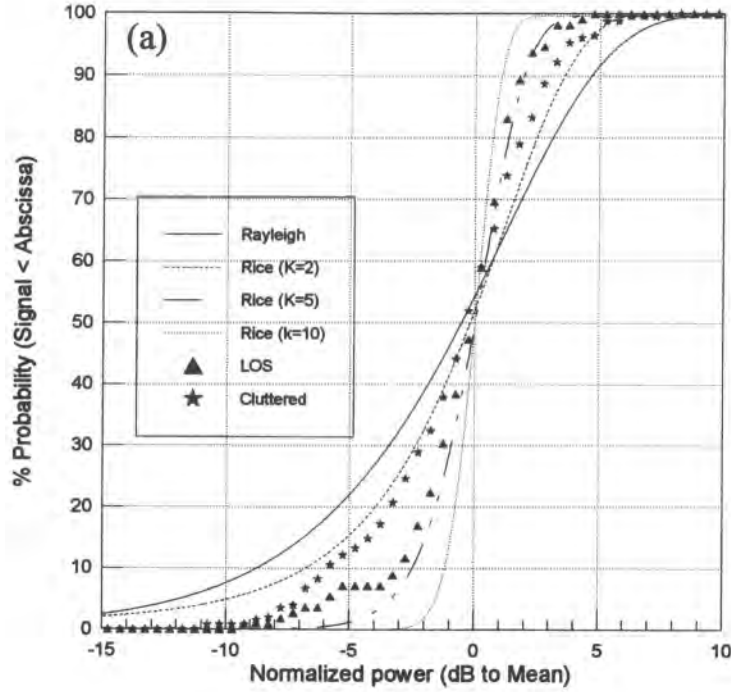


Figure 9.21. Comparison of cumulative distributions of measured amplitudes with analytical models. (a) Engineering building; (b) retail store.

bution for non-LOS paths and heavily cluttered environments around receivers and transmitters does seem to follow the Ricean distribution with a K factor of 2, which is very close to the Rayleigh distribution as suggested by Saleh and Valenzuela.⁽³⁰⁾ In principle, rays and clusters extend over an infinite time as shown in equation (9.1.67). However, the sum over l stops at L when $\exp(-T_l/\Gamma) \ll 1$, and that over k stops at K when $\exp(-\tau_k/\gamma) \ll 1$, so that in practice, one uses a truncated version:

$$\tilde{c}(\tau, t) = \sum_{l=1}^L \sum_{k=1}^K a_{kl} e^{-j\theta_{kl}} \delta(t - T_l - \tau_k) \quad (9.1.70)$$

The numbers L and K were not given a statistical characterization, although they do vary depending on the environment. Rather, the authors⁽³⁰⁾ propose to let these numbers “float”, that is, to be set by an optimization procedure that produces the closest agreement between the measurements and the model.

The measured rms delay spread within rooms of the building had a median value of 25 ns and a maximum value of 50 ns, which implies that rms delay spread is dependent on the building size as well as the extent of obstruction of line-of-sight paths. The path loss index seems to be between 3 and 4, which is higher than that of the factory model. This may be due to the existence of interior walls in the office building, which is less common in factory buildings.

Some typical examples of measured delay power profiles in an office building are shown in Figure 9.22.

9.1.3.6.3. Ray-Tracing Prediction Model An understanding of the characteristics of indoor wireless communication channels is essential for the design and deployment of wireless communication systems. Measurements of CW and pulsed signals are the most common measurement methods to model empirically the radio channel, as described in the previous sections. However, statistical analysis based on measurements may not provide for optimum deployment of the communication systems in a specific building. In order to avoid expensive measurements in individual buildings prior to installation, or adjustments afterward, theoretical prediction models have been developed to predict the path loss and delay spread from the building floor plan.

A computer-based tool called Wireless System Engineering (WiSE)⁽³²⁾ has been developed for designing an indoor and campus-sized wireless system. It uses algorithms from computational geometry and optimization to decide where to place base-station transceivers. The propagation model of WiSE is a 3D ray-tracing process that in principle follows all paths from a base station to a specific point. A brute-force approach here requires too much computation time to be feasible for real buildings. Accordingly, a modified algorithm was developed to reduce the number of path components that have to be evaluated by focusing on useful rays only. This propagation model predicts the local mean of signal power received at any given point with an i.i.d. assumption on the phase angle of each ray component. The building database, that is, the locations and radio characteristics of all walls, floors, and ceilings, must be acquired and digitized. Given building data and system parameters (for example, antenna type, antenna location and elevation, radiation power, and so forth), WiSE determines system performance (for example, received power or delay spread) throughout the building or at specific points.

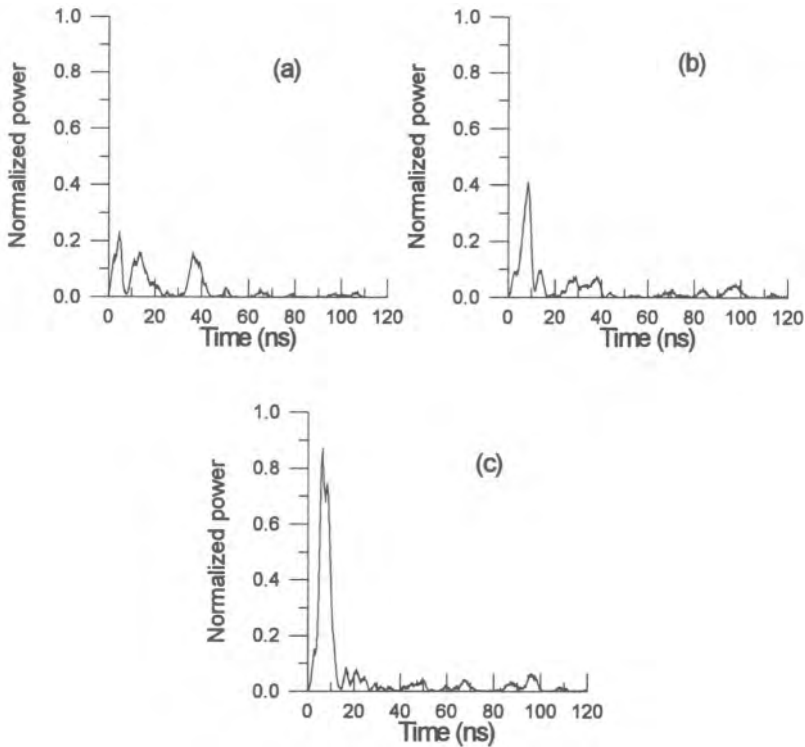


Figure 9.22. Measurements of power delay profiles for a site with lightly obstructed paths in an office building.

9.1.3.7. Radio-Relay Line-of-Sight (LOS) Discrete Multipath Fading Channel Model

There are many possible models for a multipath channel, depending on the application, the frequency band, and the physical environment. The same assumptions may not apply, or apply equally well, in all circumstances. In this section we present another type of multipath model, developed explicitly for terrestrial communication links between fixed antenna towers. There are, again, conceptually different possible models for this environment, but here we limit ourselves to the presentation of the most accepted model, the so-called Rummler model.⁽³³⁾

The channel is a line-of-sight (LOS) radio channel where the multipath fading is caused by stratification of the atmosphere under certain climatic conditions. Because of the use of larger antennas, the field of view of the antenna is limited to very small angles of arrival, and hence the smaller number of multipath components. Also, since the antennas are fixed, the only time variations in the channel characteristics are due to changes in the atmospheric conditions. These variations can be considered very slow compared to the channel bandwidths, which will be on the order of tens of MHz. Hence, the Rummler model is a multipath model with very slow fading. This model is widely used for terrestrial microwave links operating in the frequency range of 2–18 GHz between fixed towers. The model derivation is based on a set of assumptions discussed shortly. Measured data were used to obtain the values and statistical characterization of the model parameters.

Given the geometry of the link and antenna parameters, Rummler hypothesized a three-ray model of the form

$$y(t) = s(t) + \alpha s(t - \tau_1) + \beta s(t - \tau_2) \quad (9.1.71)$$

where $s(t)$ and $y(t)$ are the bandpass input and output, respectively. In terms of the complex envelopes, the model takes the form

$$\tilde{y}(t) = \tilde{s}(t) + \alpha \tilde{s}(t - \tau_1) e^{-j2\pi f_c \tau_1} + \beta \tilde{s}(t - \tau_2) e^{-j2\pi f_c \tau_2}$$

where f_c is the carrier frequency.

The lowpass-equivalent transfer function of the channel is given by

$$H(f) = 1 + \alpha e^{-j2\pi(f_c - f)\tau_1} + \beta e^{-j2\pi(f_c - f)\tau_2} \quad (9.1.72)$$

Since we are dealing with lowpass-equivalent channels, it is convenient to substitute $f \rightarrow (f - f_c)$, where f is now the lowpass frequency. Then $|f| \leq B/2$, where B is the bandwidth of the channel.

To achieve a unique fit of the model parameters to the empirical data, Rummler introduced a few modifications. The first modification of the model is based on the assumption that over the bandwidth of interest one has $B\tau_1 \ll 1$, which causes frequency nonselective (flat) fading. Hence $\exp(-j2\pi f \tau_1) \approx 1$, and (9.1.72) can be written as

$$H(f) = 1 + \alpha + \beta e^{-j2\pi f \tau_2} \quad (9.1.73)$$

The above equation has three random parameters, α , β , and τ_2 . Notice that, because of the assumptions made, the three-path model is effectively a two-path model.

Considerations related to fitting the model parameters to experimental data lead us to identify in the model a “notch” frequency parameter f_0 for which $|H(f)|$ is minimum. Thus, substituting $\tau_2 = [(f - f_0)/f]\tau$, where τ is a constant delay, we get the final form of the model

$$H(f) = a[1 - b e^{-j2\pi(f - f_0)\tau}] \quad (9.1.74)$$

where we can regard $a = 1 + \alpha$ as the overall attenuation parameter, τ the relative path delay of the second ray, and $b = -\beta/(1 + \alpha)$ and $2\pi f_0 \tau$ as the relative amplitude and phase of that ray, respectively. Since (9.1.74) is a model, not the physical reality, the choice of the parameter τ is somewhat arbitrary, so long as the ultimate model is good. It turns out that a fixed value of τ on the order of $(6B)^{-1}$ allows empirical fitting of the remaining free parameters in a way that properly accounts for the observed channel behavior. The value chosen in this model is $\tau = 6.3 \text{ ns}$.

The squared amplitude response of this channel is

$$|H(f)|^2 = a^2 \{1 + b^2 - 2b \cos[2\pi(f - f_0)\tau]\} \quad (9.1.75a)$$

and the group delay is given by the derivative of the phase characteristic,

$$D(f) = \frac{-1}{2\pi} \frac{d\phi(f)}{df} = \frac{\tau}{2\pi} \frac{b\{b - \cos[2\pi(f - f_0)\tau]\}}{1 + b^2 - 2b \cos[2\pi(f - f_0)\tau]} \quad (9.1.75b)$$

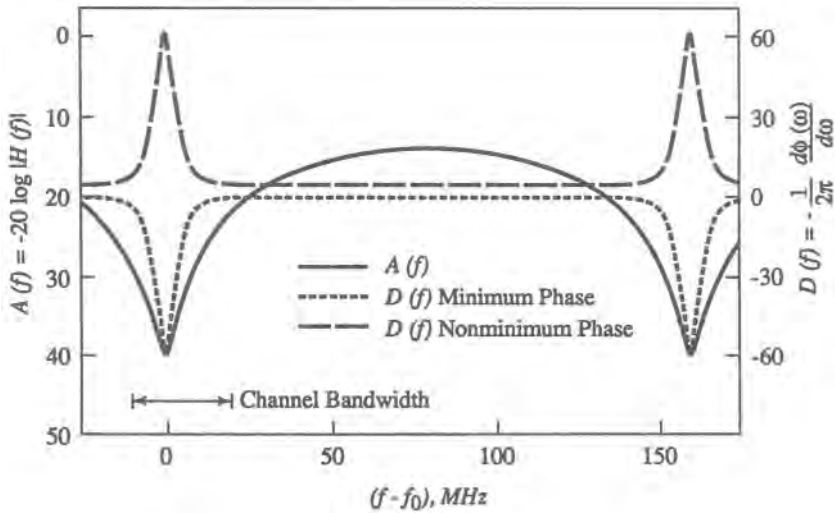


Figure 9.23. Typical amplitude and delay responses for Rummeler's LOS fading channel model.

If $b < 1$, the transfer function is *minimum-phase* since it has zeros only in the left half of the s plane. For $b \geq 1$, the transfer function is *nonminimum-phase* because its zeros are in the right half of the s plane. Typical amplitude and delay responses based on (9.1.75) are shown in Figure 9.23.

The lowpass-equivalent channel impulse response, the Fourier transform of (9.1.74), is

$$\tilde{h}(t) = a[\delta(t) - be^{-j2\pi f_0 \tau} \delta(t - \tau)] \quad (9.1.76)$$

Equation (9.1.76) corresponds to a two-tap TDL model shown in Figure 12.10 in Case Study I of Chapter 12.

The parameters a and b are random variables whose distributions have been empirically (approximately) determined. It is more convenient to describe the distributions of logarithmic forms. Thus, we define, for minimum-phase fades,

$$A_1 = -20 \log a \quad [\text{dB}] \quad (9.1.77a)$$

$$B_1 = -20 \log(1 - b) \quad [\text{dB}] \quad (9.1.77b)$$

For nonminimum-phase fades, the range of b is unbounded and it is convenient to rephrase (9.1.74) as⁽³⁴⁾

$$H'(f) = ab[e^{-j2\pi(f-f_0)\tau} - 1/b] \quad (9.1.78)$$

and we can now define

$$A_2 = -20 \log(ab) \quad [\text{dB}] \quad (9.1.79a)$$

$$B_2 = -10 \log(1 - 1/b) \quad [\text{dB}] \quad (9.1.79b)$$

The minimum- and nonminimum-phase fades are equally likely to occur and can be assumed to have identical distributions⁽³⁴⁾ as follows:

1. Parameters B_1 and B_2 are both exponentially distributed with mean 3.8 dB.
2. Parameters A_1 and A_2 are Gaussian random variables with mean

$$\mu = 24.6 \frac{B^4 + 500}{B^4 + 800} \quad [\text{dB}] \quad (9.1.80)$$

where $B = B_1$ for A_1 and $B = B_2$ for A_2 . The standard deviation of A_1 and A_2 is 5 dB. Note that, because of (9.1.80), parameters a and b are correlated, i.e., the mean of A depends on the value of B .

The distribution of the phase $\theta = 2\pi f_0 \tau$ has a range of $[-\pi, \pi]$ and a constant density on each section $|\theta| > \pi/2$ and $|\theta| < \pi/2$ with

$$P\{|\theta| < \pi/2\} = 5P\{|\theta| > \pi/2\} \quad (9.1.81)$$

The notch frequency is given by

$$f_0 = \frac{\theta}{2\pi\tau} \quad \text{where } \tau = 6.3 \text{ ns} \quad (9.1.82)$$

The application of the Rummmler model is described in Case Study I in Chapter 12. We note that although this model falls under the general description of (9.1.1), the statistical characterization is quite different from that of the mobile channel, and these again are different from the characterization of indoor channels that we discussed in the previous section.

9.2. The Almost Free-Space Channel

Virtually any channel will be benign if the signal has sufficiently small bandwidth, and conversely almost any benign channel will exhibit nontrivial distortion when the signal has sufficiently large bandwidth. Propagation in or through the atmosphere is an extremely complex phenomenon,^(35–37) which can take on a wide range of behavior depending on circumstances. At one end of this range, the atmosphere has frequently been regarded as well approximated by free space, i.e., an ideal channel. This approximation is fairly good for satellite systems using large ground antennas operating in the 4- to 6-GHz range and with elevation angles that are not too small. However, as we increase the carrier frequency and increase the bandwidth correspondingly, a nonnegligible filtering effect begins to manifest itself. We will refer to such a case as the almost free-space channel. In this channel we consider only this effective filtering to exist; that is, we assume there is no multipath or scintillation. Although this filtering characteristic is in reality time-varying, it is reasonable to treat it on a quasistatic basis because at the very wide bandwidths for which the filtering effect becomes significant the channel does vary very slowly with respect to the signal. The time variations can be separately modeled as phase noise whose spectrum is extremely narrow.⁽³⁸⁾ We consider briefly three contributors to this filtering effect: (a) the clear-air atmosphere, in which significant filtering can occur around specific absorption “lines”; (b) the rainy

atmosphere, in which absorption is a function of frequency; and (c) phase distortion due to the ionosphere. In the rainy atmosphere we also briefly consider the depolarizing effect of rain.

9.2.1. Clear-Air Atmospheric (Tropospheric) Channel

In a clear-air atmosphere, an electromagnetic wave interacts with the oxygen and water vapor that are present in a way that depends on the frequency of the wave. At certain frequencies there are resonances, resulting in peaks of absorption.⁽³⁹⁾ An example of atmospheric absorption curves is shown in Figure 9.24. Clearly, over a large enough bandwidth, the atmosphere acts as a filter. The possible effects of such a filter have been reported in Ref. 40. In order to simulate this effect, one needs to know the transfer function $H(f)$. If one has a set of measurements, it is straightforward to include them as a filter in a simulation, as discussed in Chapter 4. An analytical form is preferable, however, and for this purpose, one can use Liebe's model,⁽³⁹⁾

$$H(f) = H_0 \exp[j0.02096f(10^6 + N)L] \quad (9.2.1)$$

where N is the complex refractivity in parts per million (a function of f), $N = N_0 + D(f) + jN''(f)$; H_0 is a constant determined from table lookup; N_0 is the frequency-independent refractivity; $D(f)$ is the refractive absorption; $N''(f)$ is the absorption; and L is the distance in kilometers.

The above parameters are dependent on frequency and on atmospheric conditions, namely, temperature, barometric pressure, and relative humidity. Values of these parameters are tabulated in Ref. 39. Liquid water in clouds or fog will also affect the absorption. Of course, the major liquid water effect occurs in rain, which we discuss next.

9.2.2. The Rainy-Atmospheric Channel

At microwave frequencies, say above 10 GHz, rain can become a dominant effect on atmospheric propagation. Certainly, at high enough frequencies and rain rate, rain attenuation is much more significant than atmospheric absorption, except possibly at the resonance lines. The effect of rain is well known^(41,42) and is usually accounted for as attenuation which is

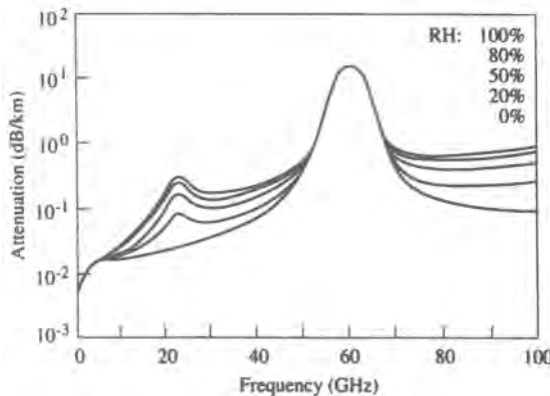


Figure 9.24. Atmospheric absorption at sea level as a function of frequency and relative humidity (from Ref. 40, © IEEE, 1984).

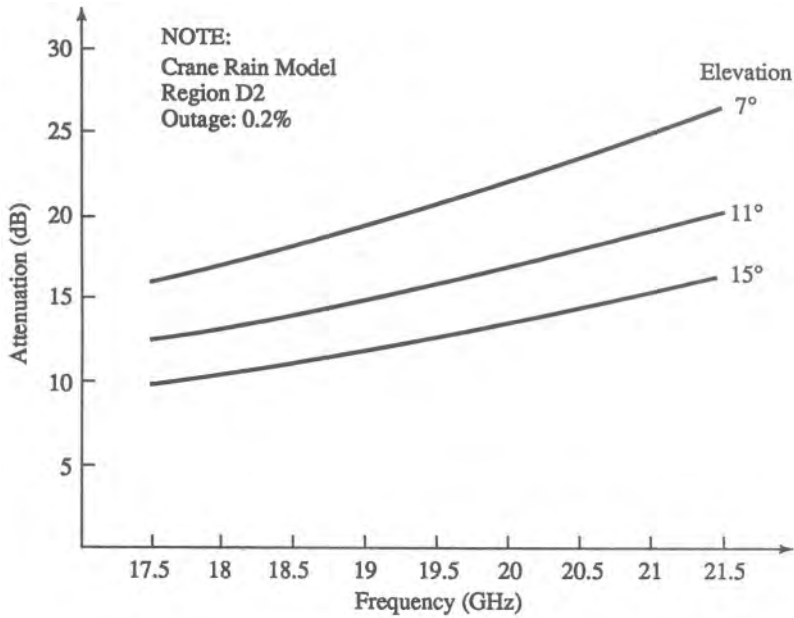


Figure 9.25. Rain attenuation as a function of frequency.

independent of frequency across the bandwidth of the signal. However, as mentioned, for sufficiently wide bandwidth this attenuation will be frequency dependent, and hence effectively act as a filter. Figure 9.25 illustrates this point. As an example, we have computed rain attenuation as a function of frequency over the band from 17.7 to 21.2 GHz allocated to the space-to-earth path for satellite communications. At these frequencies rain attenuation is not only a major attenuating factor, but, as can be seen, it exhibits a significant effective gain slope across the band. The curves in Figure 9.25 have been computed using the Crane model⁽⁴¹⁾ for three different elevation angles and rainfall corresponding to an outage probability of 0.2%. The receiving station is assumed at Washington, D.C., which has a relatively temperate climate. The effects would be more severe in rainier climates. One could interpret Figure 9.25 as a filter for use in simulation and simply input points from the curves.

Another effect of rain which can be significant in dual-polarized systems is the depolarization of radio waves.^(36,37,42) That is, some of the energy in each polarization is transformed into energy of the opposite (orthogonal) polarization. Let

$$S_1(t) = \rho_1(t) \cos[\omega_1 t + \phi_1(t)] \quad (9.2.2a)$$

$$S_2(t) = \rho_2(t) \cos[\omega_2 t + \phi_2(t)] \quad (9.2.2b)$$

represent signals of polarization 1 and 2, respectively; these polarizations could be linear, say horizontal and vertical, or left hand and right-hand circular; the carrier frequencies f_1 and f_2 may not be the same, but the spectra overlap. The simplest model for depolarization takes the form

$$R_1(t) = \alpha_{11}S_1(t) + \alpha_{21}S_2(t) \quad (9.2.3a)$$

$$R_2(t) = \alpha_{22}S_2(t) + \alpha_{12}S_1(t) \quad (9.2.3b)$$

where $R_1(t)$ and $R_2(t)$ are the signals received of polarizations 1 and 2, respectively, and the coefficients represent the relative magnitude of each term. Basically, the cross-polarized leakage introduces interference into each signal. This interference is referred to as XPI. In signal 1, for example, the XPI is $20 \log(\alpha_{11}/\alpha_{21})$. Situations of this type are easily simulated, and there is also a good deal of literature on the effect of interference on the BER of digital signals, as well as on methods to compensate for XPI (see, e.g., Ref. 43 for a brief overview of the subject). However, simulation becomes increasingly simpler than analysis as the system departs further from the ideal. Equations (9.2.3) can be generalized by replacing each of the coefficients α_{ij} by a linear frequency-dependent transfer characteristic. The result is

$$R_1(t) = h_{11}(t) * S_1(t) + h_{21}(t) * S_2(t) \quad (9.2.4a)$$

$$R_2(t) = h_{22}(t) * S_2(t) + h_{12}(t) * S_1(t) \quad (9.2.4b)$$

which, again, is straightforward to simulate if we know the impulse responses $h_{ij}(t)$.

9.2.3. The Ionospheric Phase Channel

In the lower frequency bands, say on the order of a few hundred megahertz and below, the effect of the ionosphere on propagation is extremely complex and best characterized as a time-varying multipath channel. As such, it is perhaps more appropriately modeled by the methods discussed in Section 9.1. At frequencies above a few hundred megahertz, for example, in the bands allocated to satellite communications, and assuming the absence of anomalous condition (solar flares, nuclear events, extremely low elevation angles), the ionosphere can be approximately modeled by an all-pass filter with a nonideal phase characteristic.

It can be shown⁽⁴⁴⁾ that the phase shift experienced by a wave of frequency f due to free electrons in the ionosphere, over and above the free-space propagation lag, is given by

$$\phi(f) = \frac{2\pi 40 \times 10^6}{cf} \int_{s_1}^{s_2} N_e(s) ds \quad (\text{rad}) \quad (9.2.5)$$

where c is the speed of light (cm/s), N_e is the areal electron concentration (electrons/cm²) at any point along the path s , and the integral represents the integrated ("columnar") electron density along the signal path. The differential phase shift between any two frequencies f_0 and $f_0 + \Delta f$ is therefore given by

$$\psi(\Delta f) = \phi(f_0 + \Delta f) - \phi(f_0) \quad (9.2.6a)$$

$$= \frac{2\pi 40 \times 10^6 \Delta f}{cf_0(f_0 + \Delta f)} \int_{s_1}^{s_2} N_e(s) ds \quad (9.2.6b)$$

$$= K(f_0, N_e, s) \frac{\Delta f}{f_0 + \Delta f} \quad (9.2.6c)$$

in which we have lumped into the constant K all the factors not depending on Δf . In fact, since we can consider Δf to be the departure from any given center frequency f_0 , it can be seen that (with $\Delta f \rightarrow f$)

$$\psi(f) = K \frac{f}{f_0 + f} \quad (9.2.7)$$

is the complex lowpass-equivalent filtering characteristic. We can simplify (9.2.7) still further by defining the normalized frequency $v = f/f_0$, so that

$$\psi(v) = K \frac{v}{1 + v} \quad (9.2.8)$$

Obviously, the actual phase characteristic depends in scale on f_0^{-1} and on the integrated electron density, which can vary by two or three orders of magnitude depending upon path length (elevation angle), time of day, or solar activity. The frequency-dependent characteristic $v/(1 + v)$ is very simple and can be entered directly as-is (in sampled form) for use in simulation; this function is plotted in Figure 9.26. However, it is even simpler to implement if we realize, as the figure suggests, that this function is dominated by parabolic phase. That is, if we expand $\psi(v)$ in a power series, say, or a least-squares polynomial decomposition, then the v^2 term is the major one (other than v , which does not lead to distortion) for typical

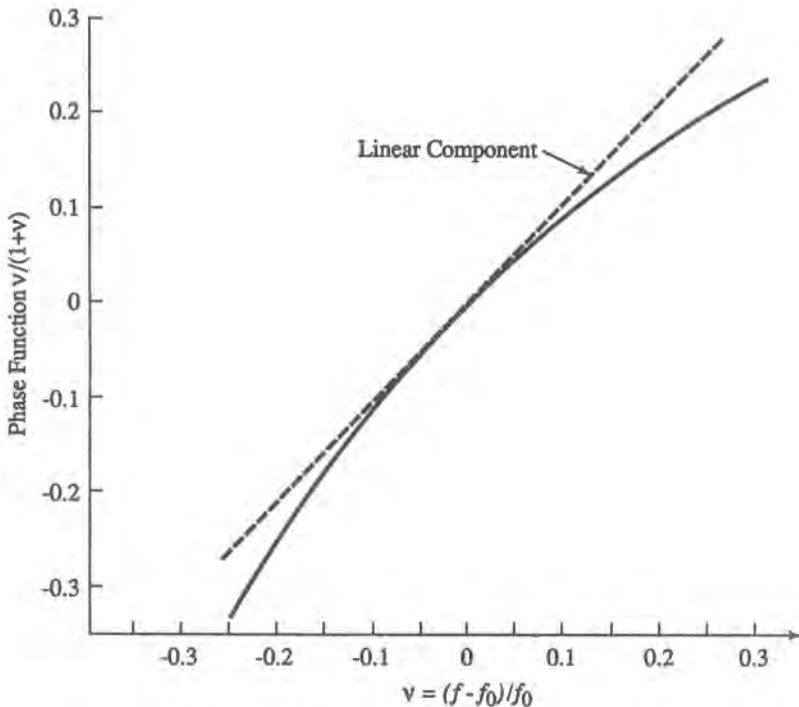


Figure 9.26. Ionospheric lowpass-equivalent phase characteristic.

fractional bandwidth values, say $|v| \leq B/f_0 < 0.1$. Because of the simple form of $\psi(v)$ it is easy to demonstrate the point using a Taylor series about $v = 0$. Thus, one obtains

$$\psi(v)/K = v/(1+v) = v - v^2 + R_3 \quad (9.2.9)$$

where the remainder is

$$R_3 \leq Mv^3/3!$$

and $M \geq |\psi'''(u)/K|$, where $|u| \leq B$. Since $\psi'''(u)/K = 6(1+u)^4$, its maximum value is reached when $u = -B$, which we can take as -0.1 . Thus, $R_3 = v^3/(0.9)^4 \approx 1.5v^3$, which means that at $v = -0.1$, the magnitude of the remainder relative to the squared term is $1.5v^3/v^2 = 1.5v = 0.15$. This means, basically, that the parabolic phase term v^2 dominates over any reasonable fractional bandwidth. Hence, a simple expedient to model the effect of the ionosphere is to replace it by a parabolic phase functional filter, as discussed in Section 8.9. Notice in this particular case that the parabola is concave down.

9.3. Conducting and Guided Wave Media

In many situations of interest the medium (channel) between transmitter and receiver physically confines the signal within or through solid boundaries. Such media include wires, cables, waveguides, or optical fibers, whose transfer functions are generally frequency dependent. Of course, these elements are also used to interconnect devices within a transmit or receiving facility. Often the effect of these media is ignored, and indeed, depending upon the bandwidth of the signal, the frequency dependence may or may not be significant. When it is significant, that frequency dependence should be taken into account when simulating a system. These media possess a wide range of physical characteristics, depending upon material, shape, and generally the specific design. In this section, for illustrative purposes, we set down the transfer function for two important cases, rectangular waveguides and optical fibers.

9.3.1. Rectangular Waveguide Medium

At microwave frequencies, waveguides often serve as the connections between various elements of a system, e.g., between a receiving antenna and a low-noise amplifier. Waveguide runs can sometimes have appreciable length, and since the transfer function is directly proportional to length, the distorting effect may not be negligible.

For rectangular waveguides, it can be shown⁽⁴⁵⁾ that the transfer function is as follows. The amplitude characteristic $A(f) = 20 \log |H(f)|$ is given by

$$A(f) = \frac{\alpha[(A/2B)(f/f_c)^{1.5} + (f/f_c)^{-0.5}]}{[(f/f_c)^2 - 1]^{1/2}} \frac{\text{dB}}{\text{m}} \quad (9.3.1)$$

where α is a constant depending upon the material and the large dimension; A is the waveguide's large dimension; B is the waveguide's small dimension; and f_c is the cutoff frequency, $f_c = c/2A$, where c is the speed of light.

The phase characteristic is given by

$$\beta(f) = (360/c)f[1 - (f_c/f)^2]^{1/2} \quad \text{deg/m} \quad (9.3.2)$$

The functions $A(f)$ and $\beta(f)$, multiplied by the waveguide length, can be used directly in simulation, for example, by sampling them in frequency and storing them as tabular filters. However, as with the ionospheric phase filter, it is more illuminating to extract the complex lowpass-equivalent filter. We do this now for the phase characteristic (9.3.2).

Assume that the bandpass signal is centered on a carrier frequency f_0 so that we can write $f = f_0 + \Delta f$. Equation (9.3.2) then transforms to

$$\psi(\Delta f) = \left(\frac{360}{c}\right)f_0\left(1 + \frac{\Delta f}{f_0}\right)\left[1 - \frac{(f_c/f_0)^2}{(1 + \Delta f/f_0)^2}\right]^{1/2} \quad (9.3.3)$$

which is the complex lowpass equivalent, and is plotted in Figure 9.27 as a function of the normalized lowpass frequency $v = \Delta f/f_0$. For plotting purposes we have assumed $f_0/f_c = 1.6$. Again, the figure suggests that this function is essentially parabolic, except for the linear phase term.

This can be demonstrated readily through the following steps, using $v = \Delta f/f_0$. First we use the fact that

$$(1 + v)^{-1} = 1 - v + v^2 - v^3 \dots \approx 1 - v \quad (v \leq 0.1)$$

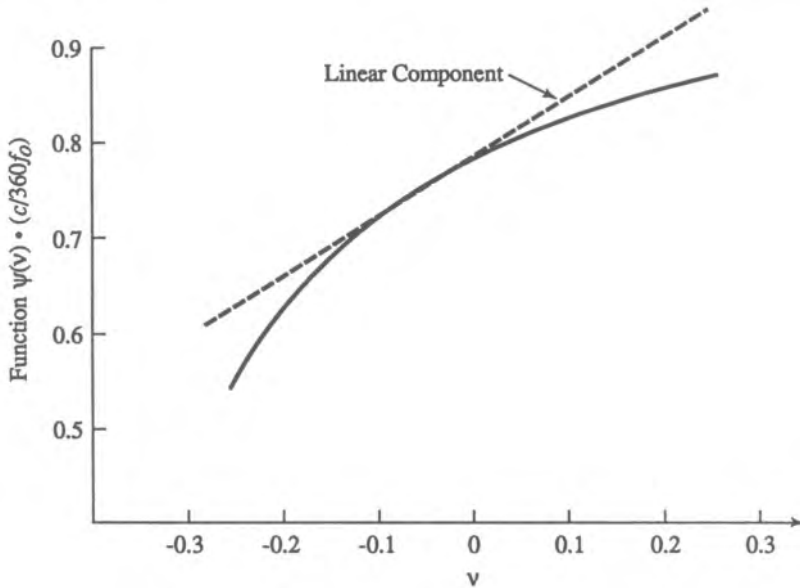


Figure 9.27. Rectangular waveguide lowpass-equivalent phase characteristic.

so that $(1 + v)^{-2} \approx (1 - v)^2$. We also use the approximation $(1 - x)^{1/2} \approx 1 - (x/2)$ for x small. Actually, the approximation is fairly good even for x not so small. Using the two approximations indicated yields

$$\Psi(v) \approx \frac{360}{c} f_0 \left\{ 1 - \frac{1}{2} v_c^2 + v + \frac{1}{2} v v_c^2 + \frac{1}{2} v^2 v_c^2 - \frac{1}{2} v^3 v_c^2 + \dots \right\}$$

where $v_c = f_c/f_0$. The constant and linear terms produce no distortion, and it can be seen that for $v \leq 0.1$ the parabolic term dominates.

9.3.2. The Fiber Optic Channel

Optical pulses propagating along an optical fiber are attenuated and distorted to an extent that depends on the chemical composition of the materials used to construct the fiber, as well as on the structure of the fiber.⁽⁴⁶⁻⁴⁸⁾ Both manifestations are important because the nature of the detected optical signal depends generally both on the detector input signal level as well as on the transfer function of the channel. The main distortion is dispersion, of which there are two types, chromatic dispersion and intermodal dispersion. The former describes the effect due to the index of refraction in the fiber, which is a nonlinear function of wavelength.⁽⁴⁷⁾ In more familiar terms, the fiber has a nonlinear phase characteristic. Intermodal dispersion is an effect that is seen in multimode fibers and results from the fact that a large number of ray path lengths is supported by the fiber, generally with different delays.⁽⁴⁷⁾ Thus, intermodal dispersion can be seen as a form of distortion due to internal multipath. Splices between fibers introduce additional mode mixing and modeling difficulties.

A mathematical model for the baseband transfer function of an optical fiber can be approximated by (see, e.g., Ref. 49)

$$H(f) = \int_{-\infty}^{\infty} S(\lambda) L(\lambda) H_{\text{im}}(f) H_c(\lambda, f) d\lambda \quad (9.3.4)$$

where $S(\lambda)$ is the source spectrum as a function of wavelength; $L(\lambda)$ is the reciprocal of loss as a function of wavelength; $H_{\text{im}}(f) = \exp(-\sigma_{\text{im}}^2 \omega^2/2 - j\omega t_d)$ (intermodal dispersion transfer function, normalized to unity gain at the center frequency); σ_{im} is the rms pulse spreading; $\omega = 2\pi f$; t_d is the fiber time delay; and $H_c(\lambda, f) = e^{-j\omega T(\lambda)/l}$ is the fiber length and $T(\lambda)$ is the group delay as a function of wavelength λ . Figure 9.28 shows for reference a typical loss curve $L^{-1}(\lambda)$ for a fiber.

The model (9.3.4) is valid when the source spectral width is large in comparison to the data rate. The model is linear in optical power (hence the name “power domain model” is sometimes used). For single-mode fibers there are no intermodal dispersion effects, so the transfer function for single-mode fibers can be written as

$$H_{\text{sm}}(f) = \int_{-\infty}^{\infty} S(\lambda) L(\lambda) \exp[-j\omega T(\lambda)] d\lambda \quad (9.3.5)$$

The simulation model of the single-mode fiber would typically be constructed from the source spectrum, the loss, and the dispersion curves in tabular form. The source spectrum function

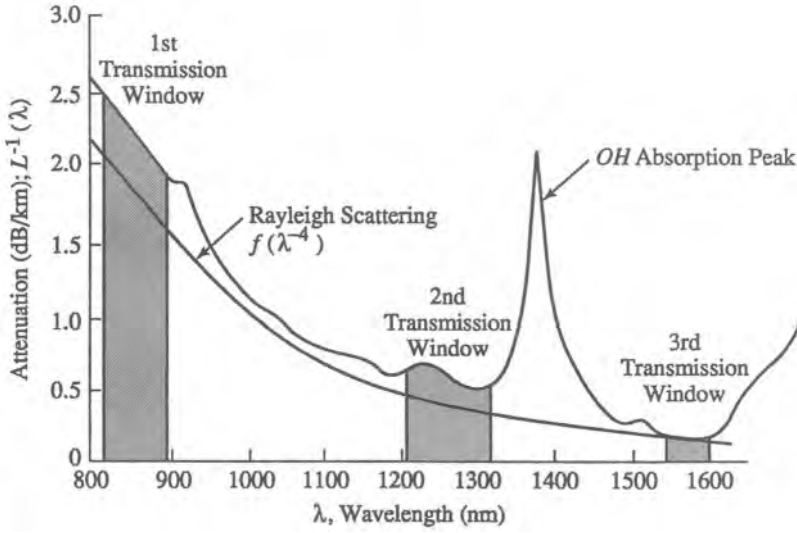


Figure 9.28. Typical loss curve as a function of wavelength for an optical fiber (from J. C. Daly, ed., *Fiber Optics*, © CRC Press, Boca Raton, Florida, 1984, reprinted with permission).

must be normalized such that $\int S(\lambda) d\lambda = 1$. For light emitting diodes LEDs, it is often appropriate to use a Gaussian shape to approximate the source spectrum,^(46,47,49,50) namely,

$$S(\lambda) = \frac{1}{\sigma_s(2\pi)^{1/2}} \exp\left(-\frac{(\lambda - \lambda_s)^2}{2\sigma_s^2}\right) \quad (9.3.6)$$

where σ_s is the rms spectral width and λ_s is the source center wavelength.

Fiber loss as a function of wavelength would typically be input from a tabular data file created by the user. The function $L(\lambda)$ in (9.3.9) can be expressed as

$$L(\lambda) = 10^{-a(\lambda)l/10}$$

where $a(\lambda)$ is the fiber loss in dB/km and l is the fiber length in km.

Chromatic dispersion $d(\lambda)$ is defined as the derivative of the group delay function $T(\lambda)$,

$$T(\lambda) = \int_{-\infty}^{\infty} d(u) du = D(\lambda) + c_0$$

where $d(\lambda)$ is the dispersion as a function of wavelength, $D'(\lambda) = d(\lambda)$, and c_0 is the constant of integration, $c_0 = T(\lambda_u) - D(\lambda_u)$. Here $T(\lambda_u)$ is a specified value of group delay at a specified wavelength λ_u . The chromatic dispersion as a function of wavelength can be modeled either as tabular data or a closed-form approximation.

Chromatic dispersion for silica fibers is sometimes approximated by (see e.g., Ref. 49)

$$\frac{dT(\lambda)}{d\lambda} = \frac{\delta}{c} \left[\frac{\lambda - \lambda_0}{\lambda^2} \right] \quad (9.3.7)$$

where δ is a dimensionless constant (typically around 0.047), c is the speed of light, and λ_0 is the zero-dispersion wavelength. The group delay for this approximation is

$$T(\lambda) = \frac{\delta}{c} \left[\ln \left(\frac{\lambda}{\lambda_0} \right) + \frac{\lambda'_0 - \lambda}{\lambda} \right] \quad (9.3.8)$$

Two closed-form expressions for dispersion that are also used as fiber models (comité consultatif international de téléphoné et telegraph CCITT standards) are as follows. For a standard (unshifted) fiber the dispersion is given by

$$d(\lambda) = \frac{1}{4} \delta_0 (\lambda - \lambda_0^4 / \lambda^3) \quad (9.3.9)$$

which results in the following group delay expression:

$$T(\lambda) = \frac{1}{8} \delta_0 (\lambda - \lambda_0^2 / \lambda)^2 \quad (9.3.10)$$

where the group delay evaluated at λ_0 is zero.

For a dispersion-shifted fiber the dispersion approximation is

$$d(\lambda) = \delta_0 (\lambda - \lambda_0) \quad (9.3.11)$$

which corresponds to the following group delay function:

$$T(\lambda) = \frac{1}{2} \delta_0 (\lambda - \lambda_0)^2 \quad (9.3.12)$$

In these CCITT standard expressions, the zero-dispersion slope is given by δ_0 . A typical value for δ_0 is 0.087 ps/nm²/km.

The transfer function model (9.3.4) or (9.3.5) can be implemented using the dispersion approximations (9.3.7), (9.3.9), or (9.3.11), or any other appropriate expression, or using tabular input. Some details on the computation of the integral can be found in Ref. 51. Of course, once the integral has been evaluated for a particular fiber, $H(f)$ can be stored as a tabular filter for simulation purposes.

When the width of the source spectrum becomes comparable to the modulation bandwidth, the power domain model becomes inadequate,⁽⁵²⁾ and a model for the single-mode fiber that assumes linearity in the electric field must be used. Coherent lightwave systems, which are under development, utilize lasers with very narrow bandwidths, making this model useful for analysis of the effects of chromatic dispersion in these systems.⁽⁵³⁾ The complex lowpass-equivalent transfer function of the fiber is then given by

$$H(f) = \exp \left(-j\pi d(\lambda_s) \frac{\lambda_s^2}{c} f^2 \right) = \exp(-j\alpha f^2) \quad (9.3.13)$$

where c is the speed of light, λ_s is the source operating wavelength, and α is a constant which lumps all of the factors in the exponent. Thus, we recognize that a model of this medium is again a parabolic phase filter.

9.4. Finite-State Channel Models

The terminology “finite-state channel” is used to denote all the elements of a communication system that lie between any two points a and b in the system where the input entering at point a is a symbol sequence $\{X_k\}$ and the output of the system at b is another symbol sequence related to the input sequence $\{Y_k\}$ (see Figure 9.29). Usually, a will be the output of the channel encoder at the transmitter and b will be the input to the decoder in the receiver. The relationship between the input and output sequences will be affected by the distortion and noise introduced by filters and other elements that are present between a and b , and the physical channel. In a binary communication system with hard decision decoding, both $\{X_k\}$ and $\{Y_k\}$ will be binary sequences and transmission errors caused by all the elements in between a and b including the physical channel will cause $\{Y_k\}$ to be different than $\{X_k\}$ at least occasionally.

The nomenclature “finite state” implies a visualization of the channel (defined between some two points a and b) as being in one of several (but typically few) identifiable conditions, or “states”. The transition from one state to another is governed by some probabilistic rule, which is part of the model. The model description is completed by specifying, for each state, an error generation mechanism. (Sometimes the terminology “discrete channel model” is also used in this context, the word “discrete” referring to the input and output alphabets at a and b . We avoid this usage here to prevent possible confusion with the use of “discrete” in the description “discrete *multipath* channels,” where, of course, it implies a countable number of reflections.) Finite state channel models fall into two categories. The first category, called *memoryless* models, can be considered a degenerate case of the general class because such models have only a single state. Memoryless models are used to model the transmission errors or the transitions from input to the output under the assumption that there is no temporal correlation in the transition mechanism. That is, the probability of transition (or error) for the n th input symbol is not affected by what happened to any other input symbol. Such models are applicable to channels in which there is no ISI or fading and the noise is AWGN. In a hard decision binary system, this assumption implies that the bit errors are uncorrelated.

The second class of finite-state models are applicable to situations where the transitions from the input symbols to the output symbols are temporally correlated, i.e., the probability of transition for the n th symbol is correlated with the transitions of the preceding and or the following symbols. This will be the case in systems that have fading and impulse noise. Temporal correlation of signal amplitude variations due to fading will cause the transmission errors to be correlated. Errors will tend to occur in bursts and these channels are referred to as *burst error channels* or *channels with memory*.

Finite-state channel models are probabilistic models which are computationally more efficient than waveform-level models. The increased efficiency comes from two factors. The finite-state models are simulated at the symbol rate, whereas a waveform-level model will be

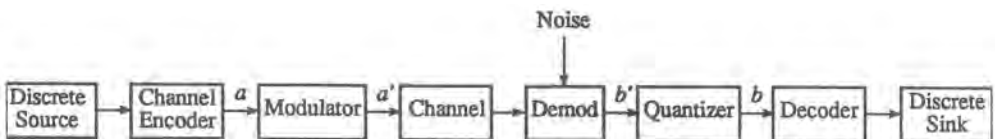


Figure 9.29. Definition of a finite-state or discrete channel: a - b ; the waveform channel lies between a' - b' .

simulated at 8–16 times the symbol rate. While each individual block is simulated in detail in a waveform-level model, there is a high level of abstraction in the finite-state model. These two factors might contribute to several orders of magnitude savings in computational burden.

In simple cases, finite-state channel models can be derived analytically from the models of the underlying components between the input a and the output b . In most cases, however, finite-state channel models are derived from simulated or measured error patterns between the input a and the output b . Finite-state channel models are used to design and analyze error control coders, interleavers, etc., and also for network simulation.

Modeling of *memoryless* channels is rather straightforward. For example, in the case of a memoryless channel with binary input and binary output, all we need to know to characterize the channel model is the bit error probability. Simulation of this channel involves drawing a single random number to decide if a given bit will suffer a transmission error or not. Thus, if we have to simulate the transmission of 1 million bits through this channel, all we have to do is generate 1 million uniform random numbers. The entire system represented by this channel is simulated efficiently. (Compare this with a waveform-level simulation, which will involve generating the sampled values of a 1 million-bit-long waveform and processing it through all the functional blocks in the channel. Of course, we might have to do this a few times to derive the BER to be used in the discrete channel model, but this is usually done at a higher BER than is typical in a coded system.)

Finite-state channels with *memory* are harder to model. The temporally correlated error generation mechanism is usually modeled by a discrete-time Markov sequence^(54–56) in which a state model is used to characterize the various states of the channel and a set of transition probabilities is used to capture the progression of the channel between various states. Each state will also have associated with it a set of input-to-output symbol transition probabilities. Thus the model now is more complex, with more parameters. The model structure and parameter values are estimated from either simulated or measured error patterns. Simulation of the finite-state model consists in generating a random number prior to the transmission of each symbol to determine the channel state and then drawing another random number to determine the input-to-output transition. While the model and parameter estimation procedures are complex, simulation of the Markov model is straightforward and very efficient.

We now look at the details of the finite-state channel models, beginning with the memoryless models. In this chapter, we only look at the construction of the model itself. The utilization of the model to derive various statistical measures of performance related to error occurrences will be taken up in Chapter 11.

9.4.1. Finite-State Memoryless Models

In a finite-state memoryless channel (recalling that it has only one state) the mapping of input to output is instantaneous and is described by a set of transition probabilities. A standard example of a simple binary memoryless channel model called the binary symmetric channel (BSC), is shown in Figure 9.30.

The input to the channel is an independent sequence of binary digits $\{X_k\}$ and the output is a binary sequence $\{Y_k\}$. A binary input of ‘0’ is received correctly as ‘0’ with a probability $1 - P_e$ and incorrectly as ‘1’ with an error probability of P_e . The channel is symmetric in that zeros and ones are affected the same way. For the BSC, we can also express the input/output relationship as

$$Y_k = X_k \oplus \varepsilon_k$$

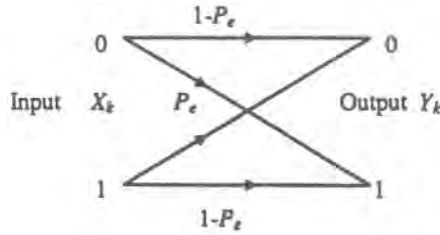


Figure 9.30. A simple discrete channel model: the binary symmetric channel (BSC).

where $\mathbf{e}_k$ is an *independent* sequence of errors, $\mathbf{e}_k = 1$ indicating a transmission error. The only parameter of this model is the probability of error P_e , which can be easily estimated via simulations or from measurements.

A slightly more complicated model for the binary channel is the nonsymmetric model (Figure 9.31a) for systems in which the probability of error may be different for zeros and ones, as in some optical communication systems. This model is characterized by a set of input-to-output transition probabilities α_{ij} , $i, j = 0, 1$, and the channel is still memoryless. These transition probabilities are estimated using a detailed waveform-level simulation using a normalized count of the crossovers. When the finite-state channel model is simulated, a random number is drawn for each bit passing through the channel to determine whether the bit will suffer a transmission error or not.

Extension of this model to the case where the input and output belong to an M -ary alphabet is shown in Figure 9.31b. Another example, shown in Figure 9.31c, is used for memoryless channels in which the output of the channel is quantized to a different number of levels than the number of input levels as in soft decision decoding. In these models the transition probabilities are estimated from simulated or measured data as

$$\hat{\alpha}_{ij} = \frac{n_{ij}}{N_i}$$

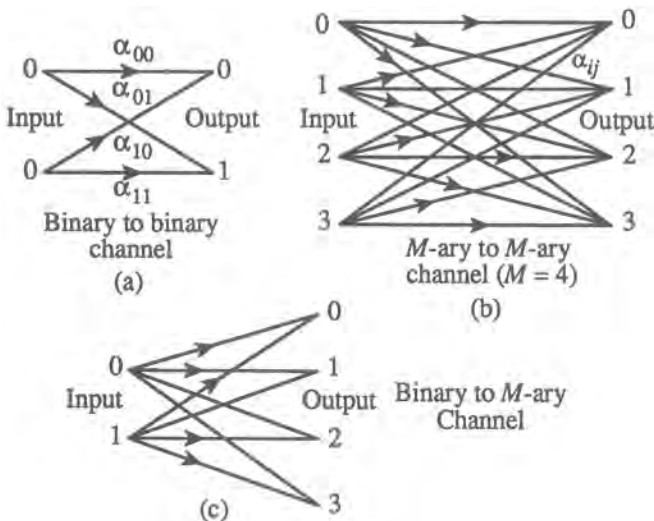


Figure 9.31. Other examples of discrete memoryless channel models.

where N_i is the number of occurrences of the i th input symbol and n_{ij} is the number of times the i th input symbol appears as the j th output.

Simulation of the channels shown in Figure 9.31 will involve once again the generation of a single random number for each input symbol to determine the input-to-output transition.

9.4.2. Finite-State Models with Memory: Hidden Markov Models (HMM)

For channels with memory the most commonly used model is the discrete-time, finite-state Markov model. There are several reasons for the popularity of Markov models. These models are analytically tractable, their theory is well established in the statistical literature, and they have been applied to communication problems, for example, to model the output of discrete information sources. Also, most importantly, computationally efficient techniques are available for estimating the parameters of Markov sequences from simulated or measured error patterns.

To set the stage for Markov models, consider a fading channel in which the received signal strength is above an acceptable threshold part of the time and below the threshold during a deep fade. Ignoring the in-between stages, we can assume the channel to be in either one of the following two “states”: (1) a good state in which the received signal level is strong enough that the probability of transmission error for a binary communication system operating over the channel is almost zero, and (2) a bad state in which the received signal level is so low that the probability of error approaches 0.5. As time progresses, the channel goes from good state to bad state and vice versa. The rate of transition and the length of stay in each of the two states will depend on the temporal correlation of the fading process. If time is measured in increments of a symbol (bit) time, then we can construct a discrete channel model as follows. This model is referred to as the Gilbert model, discussed again later.

At the beginning of each symbol (bit) interval, the channel is one of the two states. If the channel is in a good state, then the transmitted bit suffers no transmission error. On the other hand, if the channel is in a bad state, the probability of transmission error is 0.5. Prior to the transmission of each new bit, the channel may change state or remain in the current state. This transition between states takes place with a set of transition probabilities a_{ij} . A graphical representation of the state transition diagram of the model is shown in Figure 9.32

Note that when the channel is in good state, there are no transmission errors, and when the channel is in bad state, it will produce a high concentration of errors. Also, the error sequence will be statistically dependent since the state of the channel during the $(n + 1)$ th bit interval (and hence the probability of occurrence of an error) will depend on the state the channel was in during the n th bit interval. Also note that each particular state corresponds to a

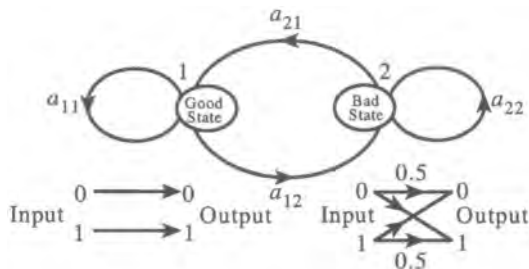


Figure 9.32. Example of the state transition diagram for a two-state Markov model (the Gilbert model).

different BSC model. By comparison, a memoryless channel has only one state and one BSC model.

A more general version of the above model can be constructed with a larger number of states, with many good and bad states with varying degrees of “goodness” or “badness” corresponding to increasingly more severity of fading and/or noise. Such an approach leads to the more general Markov model.

9.4.2.1. *N-State Markov Model*

An N -state Markov (process) model for a discrete communication channel is defined by the following parameters:

- Set of states: $\{1, 2, 3, \dots, N\}$.
- State at time $t = S_t$.
- Set of state probabilities:

$$\begin{aligned}\pi_i(t) &= \text{Probability of being in state } i \text{ at time } t \\ &= P[S_t = i], \quad i = 1, 2, \dots, N\end{aligned}$$

- Set of *state transition* probabilities:

$$\begin{aligned}\text{Probability of going from state } i \text{ at time } t \text{ to state } j \text{ at time } t + 1 \\ = a_{ij}(t) = P[S_{t+1} = j | S_t = i], \quad i, j = 1, 2, \dots, N\end{aligned}$$

(time increment is usually equal to the symbol or bit duration).

- Set of *input-to-output transition* (symbol error) probabilities for each state:

$$\begin{aligned}\text{Error symbols: } E = \{e_1, e_2, \dots, e_M\}; \quad E = \{1, 0\} \text{ in the binary case} \\ b_i(e_k) = P[\text{error symbol } e_k \text{ occurring} | S_t = i]\end{aligned}$$

These parameters define a discrete-time Markov process operating at a transition rate

of two sequences: the sequence of states $\{S_t\}$ and a sequence of error symbols $\{E_t\}$, where t is the time index, which can be indexed over the integer set $\{0, 1, 2, \dots\}$. Normally, only the input and the output of the channel and hence the error sequence can be observed, and the state sequence itself cannot be easily observed in a physical channel. Hence, the state sequence is “hidden” or not visible from external observations, and such a Markov model is called a *hidden Markov model* or HMM.

The theory of HMMs is well developed^(57,58) and HMM models have been used in a number of applications including the modeling of symbol sequences emitted by discrete information sources and speech coding. For example, the occurrence of a sequence of letters of the English alphabet in a typical passage can be modeled as a hidden Markov sequence. Similarly, the sampled values of an audio or video waveform can also be modeled by an HMM. Such models have been developed and used extensively to design source encoders in communication systems. Procedures have also been developed for estimating the parameters of such models from data. These techniques are directly applicable to modeling, analysis, and simulation of discrete (finite-state) communication channels. Also, the HMM models can be

used to evaluate the capacity of a discrete channel and for the design of optimal error control coding techniques.

9.4.2.2. First-Order Markov Process

The Markov property is defined as

$$P[S_{t+1}|S_t, S_{t-1}, \dots] = P[S_{t+1}|S_t, S_{t-1}, \dots, S_{t-m-1}] \quad (9.4.1)$$

for a Markov process with memory m . For a first-order Markov process

$$P[S_{t+1}|S_t, S_{t-1}, \dots] = P[S_{t+1}|S_t]$$

The Markov property implies that the future behavior of the system depends on the current state and the m previous states, but not on the past states beyond $t - m - 1$ or how the system got there. For channel models we will use a first-order Markov model. (For applications such as modeling the symbol sequences in an English language text, a second- or third-order model may be adequate since the influence of the preceding symbols on the probability of occurrence of a later symbol does not span more than two or three symbols.)

9.4.2.3. Stationarity

Since Stationarity is usually assumed in modeling the analog portion of the communication channel, it is common to assume Stationarity of the Markov model for discrete channels also. Stationarity implies that the parameters of the model, i.e., the probabilities $\pi_i(t)$, $a_{ij}(t)$, and $b_i(e_k)$, do not depend on t . In this case, we have

$$\begin{aligned} P[S_{t+1} = i] &= \pi_i = \sum_{k=1}^N P[S_{t+1} = i|S_t = k]P[S_t = k] \\ &= \sum_{k=1}^N a_{ki}\pi_k \end{aligned} \quad (9.4.2)$$

In terms of vector notation,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ a_{N1} & \cdot & \cdots & a_{NN} \end{bmatrix}, \quad \Pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_N \end{bmatrix}$$

we can write Equation (9.4.2) in the alternate form

$$\Pi = A^T \Pi \quad (9.4.3)$$

Using Equation (9.4.3), we can show that for a first-order Markov process, the n -stage transition matrix is given by A^n and also that $A = A^n$.

Equation (9.4.3) along with the constraint that

$$\sum_{i=1}^N \pi_i = 1$$

implies that the π_i are uniquely determined from the a_{ij} , and hence the Markov model is completely defined by the matrix A of the state transition probabilities and B , where B is the matrix of input-to-output symbol transition (i.e., error) probabilities,

$$B = \begin{bmatrix} b_{11} & \cdots & b_{1M} \\ \vdots & \vdots & \vdots \\ b_{N1} & \cdots & b_{NM} \end{bmatrix}, \quad b_{ik} = b_i(e_k)$$

that is, $b_i(e_k)$ is the probability of observing the error symbol e_k from state i .

■ *Example 9.4.1.* A Markov Channel Model. Lets assume a quaternary ($M = 4$) communication system with a channel represented by a two state ($N = 2$) Markov model as in Figure 9.32 with a *state transition matrix*

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 0.9 & 0.1 \\ 0.8 & 0.2 \end{bmatrix};$$

The *state probabilities* Π are calculated from

$$\Pi = \mathbf{A}^T \Pi$$

or

$$\begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 0.9 & 0.8 \\ 0.1 & 0.2 \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix};$$

and

$$\pi_1 + \pi_2 = 1$$

The two equations above result in $N = 2$ linearly independent equations, from which the state probabilities are evaluated:

$$\pi_1 = 8/9 \quad \text{and} \quad \pi_2 = 1/9$$

The Input-to-Output Transition Probabilities. For a quaternary system the error symbols are defined as:

$$\mathbf{E} = \{e_1, e_2, e_3, e_4\}$$

where $e_k = S_r - S_t$, and S_r = received symbol, S_t = transmitted symbol.

The error symbols are then

$$e_1 = 0 \text{ (00)} \quad (\text{no error})$$

$$e_2 = 1 \text{ (01)}$$

$$e_3 = 2 \text{ (10)}$$

$$e_4 = 3 \text{ (11)}$$

or

$$\mathbf{E} = \{0, 1, 2, 3\}$$

Let's assume that for state $\mathbf{S}_1$ (good state) the probabilities $b_{ik} = b_i(e_k)$ are

$$b_{11} = 0.8$$

$$b_{12} = 0.1$$

$$b_{13} = 0.1$$

$$b_{14} = 0.0$$

and for $\mathbf{S}_2$ (bad state) the *input-to-output transition probabilities* are:

$$b_{21} = 0.0$$

$$b_{22} = 0.4$$

$$b_{23} = 0.4$$

$$b_{24} = 0.2$$

The *input-to-output transition probabilities* $N \times M$ matrix is then:

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{bmatrix} = \begin{bmatrix} 0.8 & 0.1 & 0.1 & 0.0 \\ 0.0 & 0.4 & 0.4 & 0.2 \end{bmatrix}$$

The Average Probability of Error. The average probability of error e_k is

$$P_{\text{ave},k} = \Pi^T \mathbf{b}_k$$

where $\mathbf{b}_k$ is the k th column of the matrix $\mathbf{B}$. Thus the probabilities of the channel producing the specific error e_k are:

$$P_{\text{ave},1} = 8/9 \times 0.8 + 1/9 \times 0.0 = 0.711,$$

$$P_{\text{ave},2} = 8/9 \times 0.1 + 1/9 \times 0.4 = 0.133,$$

$$P_{\text{ave},3} = 8/9 \times 0.1 + 1/9 \times 0.4 = 0.133,$$

$$P_{\text{ave},4} = 8/9 \times 0.0 + 1/9 \times 0.2 = 0.022.$$

9.4.3. Types of Hidden Markov Models: Gilbert and Fritchman Model

The characterization of channels using discrete-time, finite-state Markov sequences has been proposed by Gilbert,⁽⁵⁹⁾ Fritchman,⁽⁶⁰⁾ and others. The Gilbert model is a two-state model with a good error-free state and a bad state with an error probability of p . This model was used by Gilbert to calculate the capacity of a channel with burst errors. Parameters of the model can be estimated from data using the procedures we will present later in this section.

The Fritchman model, first proposed in 1967, is now receiving a considerable amount of attention since it seems to be very suitable for modeling burst errors in mobile radio channels and also because it is relatively easy to estimate the parameters of the Fritchman model from burst error distributions. For binary channels, Fritchman's framework divides the state space into k good states and $N - k$ bad states. The good states represent error-free transmissions and the bad states always produce a transmission error. Hence the entries in the B matrix are zeros and ones and they need not be estimated.

The state transition matrix A in this model can be partitioned as

$$A = \begin{bmatrix} A_{GG} & A_{GB} \\ A_{BG} & A_{BB} \end{bmatrix}$$

where the submatrices represent the transition probabilities between various good and bad states. For this model it is possible to derive analytically the expressions for burst error distributions in terms of the model parameters and use these expressions to estimate the parameters of the model from empirical (measured or simulated) burst error distributions.

Let $\mathbf{O} = \{\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T\}$ be an error sequence, $\mathbf{O}_k = 1$ indicating that the k th transmitted bit suffered a transmission error and $\mathbf{O}_k = 0$ indicating error-free transmission. Also, let the notation $(\mathbf{0}^m | 1)$ denote the event of observing m or more consecutive (i.e., a burst of) error-free transmissions following an error, and $(1^m | 0)$ represent the event of observing m or more consecutive errors following a good period. Fritchman showed that the probabilities of occurrence of these two events are given by the sum of weighted exponentials

$$P(\mathbf{0}^m | 1) = \sum_{i=1}^k f_i \lambda_i^{m-1} \quad (9.4.4a)$$

and

$$P(1^m | 0) = \sum_{i=k+1}^N f_i \lambda_i^{m-1} \quad (9.4.4b)$$

where $(\lambda_1, \lambda_2, \dots, \lambda_k)$ and $(\lambda_{k+1}, \lambda_{k+2}, \dots, \lambda_N)$ are the eigenvalues of A_{GG} and A_{BB} , respectively, and the f_i are functions of the a_{ij} . From Equation (9.4.4) we can obtain the probability of obtaining exactly m zeros (or ones) as

$$P(\mathbf{0}^{m-1} | 1) - P(\mathbf{0}^m | 1) = \sum_{i=1}^k f_i \lambda_i^{m-1} (1 - \lambda_i) \quad (9.4.5a)$$

and

$$P(1^{m-1}|0) - P(1^m|0) = \sum_{i=k+1}^N f_i \lambda_i^{m-1} (1 - \lambda_i) \quad (9.4.5b)$$

The Fritchman model can be interpreted as equivalent to a Markov process with a state transition probability matrix

$$\tilde{A} = \begin{bmatrix} \Lambda_{GG} & A_{GB} \\ A_{BG} & \Lambda_{BB} \end{bmatrix}, \quad \Lambda_{GG} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} \quad \text{and} \quad \Lambda_{BB} = \begin{bmatrix} \lambda_{k+1} & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

Note that in this equivalent model there are no transitions within the set of $N - k$ good states and no transitions within the set of bad states. Such transitions are indistinguishable from the observed error sequence since a transition from one good state to another good state still produces no errors and the transition is not observable from the output.

The Fritchman model is not unique except when there is only one bad state. This is so because, in the general case, the error-free run distribution $P(0^m|1)$ and the error burst distribution $P(1^m|0)$ do not specify the statistical dependence of the error-free runs and the error bursts. In the case of a single-error-state model (Figure 9.33), Fritchman showed that

$$P(0^m|1) = \sum_{k=1}^{N-1} \frac{a_{Nk}(a_{kk})^m}{a_{kk}}, \quad m \geq 1 \quad (9.4.6)$$

From Equation (9.46) it can be seen that in the case of a single-error-state model, the error-free run uniquely specifies the $2(N - 1)$ model parameters. An empirical procedure is used to fit an $N - 1$ mixture of exponentials as in Equation (9.4.6) to the (simulated or measured) error-free run distribution. Figure 9.34 shows an example from Fritchman's paper.

While Fritchman's model is applicable to discrete channels with simple burst error distributions, it may not be adequate to characterize very complex burst error patterns, which will require more than one error state in the model. In such cases it will be very difficult to estimate the model parameters from the burst error distributions alone. Fortunately, it is possible to find a maximum likelihood estimate of the parameters using iterative techniques.

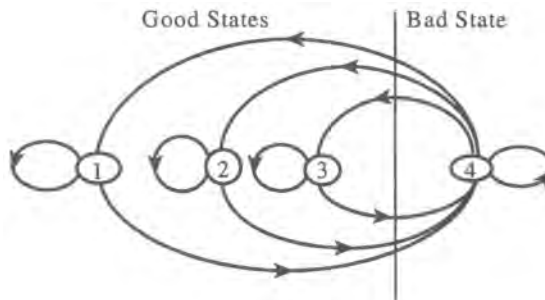


Figure 9.33. A simplified Fritchman model with a single bad state.

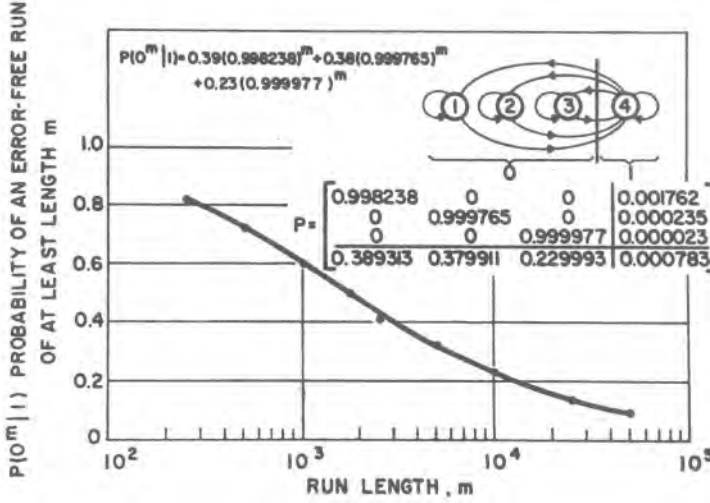


Figure 9.34. Example of model fitting to error-free run distribution (from Ref. 60, © IEEE, 1967).

9.4.4. Estimation of the Parameters of a Markov Model

The Markov model for a discrete channel is described by the $N \times N$ state transition matrix A and the $N \times M$ input-to-output transition (error probability) matrix B . An iterative procedure for estimating these parameters $\Gamma = \{A, B\}$ from a given set of a simulated or measured error sequence $\bar{O} = \{O_1, \dots, O_t, \dots, O_T\}$ was developed by Baum and Welch.⁽⁶¹⁾ This iterative algorithm, described e.g., in Ref. 62, is designed to converge to the maximum likelihood estimator of $\Gamma = \{A, B\}$, i.e., the Γ that maximizes $P(\bar{O}|\Gamma)$. The algorithm consists of the following steps.

Step 0. Start with an initial model $\Gamma = \{A, B\}$.

Step 1. With $\Gamma = \{A, B\}$ as the model, compute the “forward variables” $\alpha_t(i) = P(O_1, O_2, \dots, O_t, s_t = i | \Gamma)$ and the “backward variables” $\beta_t = P(O_{t+1}, O_{t+2}, \dots, O_T | s_t = i, \Gamma)$ for $t = 1, 2, \dots, T$ and $i = 1, 2, \dots, N$ as follows:

Forward variables (see Figure 9.35a for details)

Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad i = 1, 2, \dots, N$$

Induction:

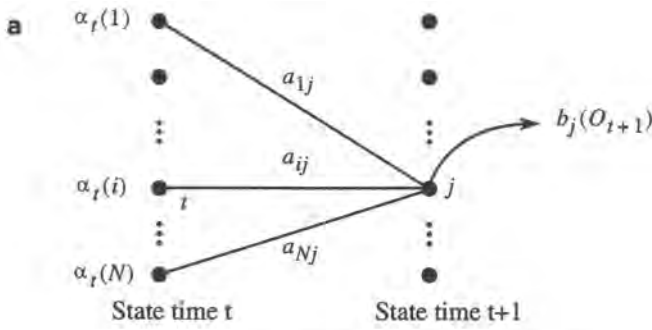
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1; \quad 1 \leq j \leq N$$

Termination:

$$P(\bar{O}|\Gamma) = \sum_{i=1}^N \alpha_T(i)$$

Note:

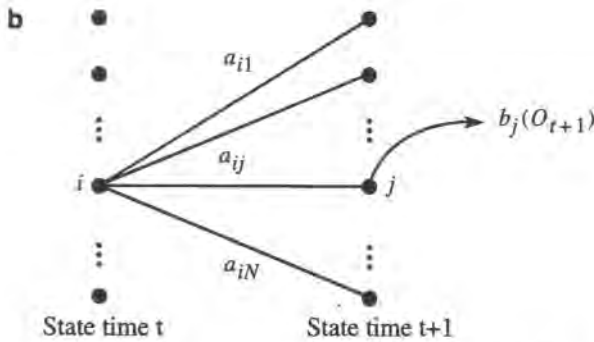
$$\sum_{i=1}^N \alpha_T(i) = \sum_{i=1}^N P(O_1, \dots, O_T, s_T = i | \Gamma) = P(\bar{O}|\Gamma)$$



$$\alpha_t(i) = P[O_1, O_2, \dots, O_t, s_t = i | \Gamma]$$

$$\text{Induction: } \alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1});$$

$$1 \leq t \leq T-1; 1 \leq j \leq N$$



$$\beta_{t+1}(j) = P[O_{t+1}, \dots, O_T | s_{t+1} = j, \Gamma]$$

Figure 9.35. The process of estimating the parameters of a hidden Markov model, (a) Computation of the forward variables; (b) computation of the backward variables.

Backward variables (see Figure 9.35b for details)

Initialization:

$$\beta_T(i) = 1, \quad i = 1, 2, \dots, N$$

Induction:

$$\beta_t(i) = \sum_{j=1}^N \beta_{t+1}(j) b_j(O_{t+1}) a_{ij}, \quad 1 \leq t \leq T-1; 1 \leq j \leq N$$

Step 2. Compute the state transition probabilities given $\bar{O}$ and Γ and update the parameters A, B according to

$$\gamma_t(i) = P[s_t = i | \bar{O}, \Gamma] = \frac{\alpha_t(i)\beta_t(i)}{P[\bar{O}|\Gamma]}, \quad i = 1, 2, \dots, N$$

$$\xi_t(i, j) = P[s_t = i, s_{t+1} = j | \bar{O}, \Gamma]$$

$$= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P[\bar{O}, \Gamma]}$$

$$\hat{a}_{ij} = \frac{\text{expected \# of transitions from } i \text{ to } j}{\text{expected \# of transitions from } i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\hat{b}_j(e_k) = \frac{\text{expected \# of times } e_k \text{ is emitted from state } j}{\text{expected number of transitions from state } j}$$

$$= \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

Step 3. Go back to step 1 with the new values of $\hat{\Gamma} = \{\hat{A}, \hat{B}\}$ (i.e., $\hat{\Gamma} = \Gamma$ obtained in step 2, and repeat until convergence, i.e., until successive values of $P[\bar{O}|\hat{\Gamma}]$ differ very little. (The Baum–Welch algorithm is guaranteed to converge to the maximum likelihood solution; proof is also given in Ref. 62.)

The Baum–Welch algorithm is one of many reestimation algorithms available for iterative computation of the likelihood ratio estimate of the model parameters. While the algorithm is computationally efficient and converges rapidly, the overall computational requirements are still high since the forward and backward variables are computed for each error symbol in the given error sequence, which could be several thousand symbols long, and potentially in the millions. Several ways of improving the computational efficiency have been proposed recently. One of the methods, proposed by Turin and Sondhi,⁽⁶³⁾ involves the computation of the forward and backward variables using the following matrix version (for the binary channel with good and bad states):

$$A = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}$$

$$\bar{O}: \begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \downarrow & \searrow & & \swarrow & \downarrow & & \swarrow \swarrow & & \downarrow & \end{array}$$

$$\text{Computation: } \pi_0 \quad [A_{00}]^4 \quad A_{01} \quad [A_{11}]^2 \quad A_{10}$$

in which the powers of submatrices involved in the computations can be precomputed and reused. Another modification⁽⁶⁴⁾ is based on the fact that for a general Markov model there is an equivalent Fritchman-like model with good states and bad states and an A matrix that has the form

$$A = \begin{bmatrix} \Lambda_{00} & A_{01} \\ A_{10} & \Lambda_{11} \end{bmatrix}$$

where Λ_{00} and Λ_{11} are diagonal matrices. With this model, the channel remains in the same state during a burst and changes state only at the end of a burst. Hence, all the variables are computed only at time steps involving the change of error symbol, i.e., at the beginning of each burst rather than once every symbol. The computations now take the form

$$\begin{array}{cccccccccc} \overline{0}: & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ & \downarrow & \searrow & & \swarrow & & & \swarrow \swarrow & & \\ \text{Computation: } \pi_0 & & [A_{00}]^4 & & A_{01} & & [A_{11}]^2 & & A_{10} & \end{array}$$

Note that the computations during long bursts of 1s and 0s now involve raising the power of a diagonal matrix rather than raising the power of arbitrary matrices, and matrix multiplications occur only at the transition from one burst to another. Thus the computations are very efficient and long error bursts can be processed without excessive storage and computational requirements.

Irrespective of the algorithms used, the Markov model for a discrete channel has to be computed for different values of the parameters of the underlying physical channel. If the discrete channel model is for a Rummmler channel, then the Markov model has to be developed from measurements or waveform-level simulations for different values of the parameters of the Rummmler channel and SNR. Thus a parametrized set of Markov models for an underlying Rummmler channel can be developed and used for the design and analysis of error control coders, interleavers, etc.

The development and use of Markov models for burst error channels (channels with memory) is currently an active area of research. The interested reader may find recent articles on this topic in the *IEEE Transactions on Communications*, the *IEEE Transactions on Vehicular technology*, and other journals and conference proceedings. Markov models are also being developed for burst errors at the higher layers of communication networks, for example, packet errors at the transport layer in a network.

Simulation of the Markov Model. Once the Markov model is derived, simulation of the model is relatively easy. The model is simulated either at the symbol rate in the system or slower. If the model is simulated at the symbol rate, then a uniform random number in the interval $[0, 1]$ is drawn at the beginning of each symbol interval to determine the state transition according to the entries in A , and then a second number is drawn to determine the input-to-output symbol transition according to the entries in the B matrix corresponding to the new state.

9.5. Methodology for Simulating Communication Systems Operating over Fading Channels

In Chapter 2, we discussed the methodology of simulation from a broader perspective. Some of the ideas discussed there will become clearer when placed in a specific context. We can illustrate various aspects of the methodology, and in particular the notion of partitioning a problem into more manageable subproblems, by considering the simulation of a communication system operating over a fading channel. Consider a typical cellular radio system for voice communication. The performance of the cellular radio interface (“air interface”) is usually evaluated at three different levels.

At the physical level the performance of the *uncoded* system is evaluated using a detailed waveform-level simulation. The main objective of this level of simulation is the comparative evaluation of various modulation, equalization, and filtering schemes and optimization of system parameters. The output of this level of simulation is the uncoded bit error rate performance and a description of the burst error characteristics of the radio channel including all the effects of waveform-level processing blocks such as modulators, filters, and equalizers. The burst error characteristic of the channel is captured in a Markov model which is parameterized in terms of fading parameters and SNR.

At this point, it is useful to elaborate a bit on our definition of “level”. Because we are dealing with a sampled analog waveform, the waveform level corresponds to a continuous-amplitude, discrete-time model. Of course, the amplitude is not truly continuous, but can be so considered for present purposes. The next level of simulation will be of the *coded* system using the Markov model. Here, the signal can be more precisely described as a discrete-amplitude, discrete-time sequence. Because the signal amplitudes belong to a discrete alphabet, we sometimes refer to this level as “discrete”. However, for the reason mentioned earlier, we avoid this usage here. Instead we will refer to this level of simulation as the *symbol level*, which in the most general case will imply not only legitimate symbol values, but quantized amplitudes as well (e.g., soft decisions). At the symbol level, comparative performance of coding, interleaving, and scrambling schemes are evaluated and the parameters of this portion of the system are optimized. The output of this level of simulation is again the BER characteristics of the coded system as a function of fading parameters and SNR.

The third level of performance evaluation deals with the speech coder and the voice quality. This will involve simulating the voice coder with a finite-state channel model for the coded system, and evaluating the voice quality subjectively. This level can also be described as symbol-level simulation. The distinction between this and the previous level will be in the nature of the finite-state model. For example, a memoryless model can often be used here under the assumption that the interleaver in conjunction with the error control coder will randomize burst errors. Otherwise, we will still need a Markov model at this level. Of course, for signals other than speech, the third level will be tailored to the signal in question, if indeed there is a third level at all.

The ultimate measure of performance for voice communication in a cellular radio system is the voice quality as measured by subjective listening tests, which produce a mean opinion survey (MOS). The MOS ranges from 1 to 5, with 5 being an excellent rating. Typically, the quality of service of a cellular radio system will be specified in terms of the requirement that 99% of the time the MOS is better than 2.0 (i.e., the probability of system outage or unacceptable quality of service below an MOS of 2 is 1%). The overall methodology used to carry

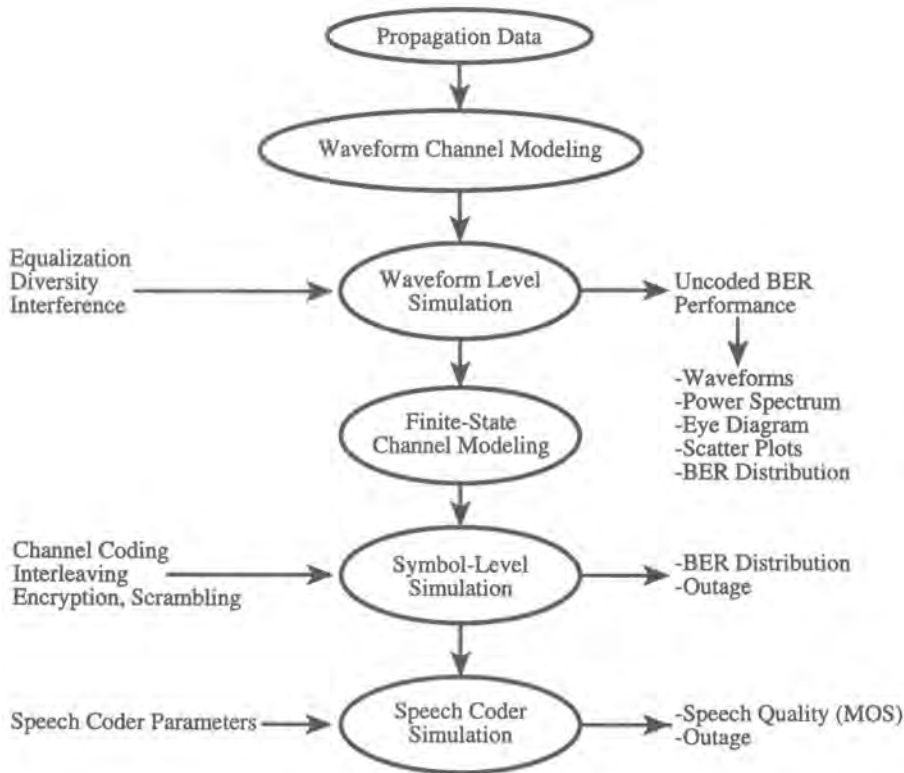


Figure 9.36. Organization of the methodology for simulating the performance of a voice communication system operating over a fading channel.

out the simulations leading up to the calculation of outage probability performance is shown in Figure 9.36; the details are as follows.

9.5.1. Waveform-Level Simulation

Waveform-level simulation of the system provides a detailed simulation of all the signal processing functions in the transmitter and receiver and it includes blocks such as modulator, demodulators, channel, and equalizer. Interference and noise are included in the simulations either through Monte Carlo or semianalytic techniques as described elsewhere in the book. The waveform-level simulation is used to compare different methods of modulation, equalization, etc., and it is also used to evaluate subsystem performance such as equalizer convergence rate. Also, optimization of parameters such as number of taps in the equalizer, tap spacing, and filter bandwidths are done using waveform-level simulations.

The output of waveform-level simulations includes spectral plots, eye diagrams, and BER curves. The performance analysis includes the following conditions: selected snapshots of the channel and optimized system components such as equalizers and filters; dynamic fading channels and optimized system components; cochannel interference, adjacent channel interference, and noise.

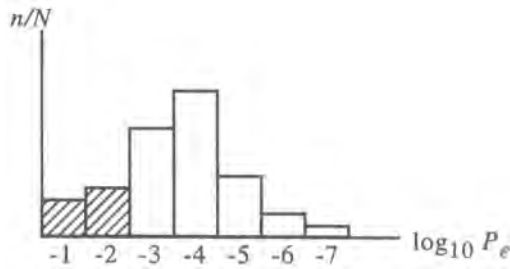


Figure 9.37. Distribution of BER in an uncoded system: the shaded area represents the outage probability at a BER threshold of 10^{-3} (i.e., n/N , where n is the number of channels out of N simulated channels that produced a $\text{BER} > 10^{-3}$).

In order to compute the overall quality of service in the system as measured by MOS and outage probability at a particular MOS level, the simulations at this level have to be carried out over thousands of static channel conditions and a distribution of the uncoded link BER has to be obtained. This distribution, usually shown as a histogram of error probabilities (see Figure 9.37 for an example), is used later in symbol-level simulations to deduce the MOS and outage probability for the entire system.

Unfortunately, this poses a significant challenge since the number of channels simulated may be large. For, if the fading statistics of the channel are described by a higher order multidimensional distribution (distributions for the number of multipath components, delay power profiles, urban versus suburban conditions, and a range of SNR values), it is easy to see that a large number of samples have to be drawn from this distribution to produce a statistically significant sample set of channel conditions. Since we have to estimate the BER for each channel condition via simulations, considerable simplification has to be made in the simulation model. Quasianalytical and other fast methods of BER estimation have to be used in order to reduce the computational burden associated with computing the BER distribution to a reasonable level.

Waveform-level simulation is also used to extract a hidden Markov type finite-state channel model for use at the next level. These finite-state channel models have to be parametrized by channel rms delay spread, mobility (Doppler bandwidth), signal power, noise, and interference as well as by what other system parameters are included in the tradeoff space. Assuming that each of the m parameters is quantized to n levels, a library of m^n finite-state channel models has to be created from the waveform-level simulations. Fortunately, the BER for the uncoded portion of the system may be as high as 10^{-2} (compared to a typical BER of 10^{-6} for the coded system) and hence each waveform-level simulation will consist of only a few thousand symbols.

9.5.2. Symbol-Level Simulation

The finite-state channel models developed through waveform-level simulations emphasize the bursty nature of errors in the physical channel due to fading, noise, and interference. These models are used to analyze the comparative performance of different types of error control coding, interleaving, scrambling, and encryption schemes. Note that all of these operations are algorithmic operations on symbol sequences and hence the finite-state channel model is most appropriate for evaluating and optimizing these operations and for

evaluating the performance of the coded portion of the communication system. While a representative set of finite-state Markov models will be useful for producing the dynamic evolution of error bursts and their impact on the dynamic performance of the coded system, burst error distributions rather than the actual error sequence may be adequate to obtain the BER distribution of the coded system. There are many analytical bounds available for computing the coded system BER as a function of the burst error distribution in the uncoded channel (see also Chapter 11).

The output of this level of simulation is the distribution of average BER in each simulated coded link, i.e., $P[P_e > P_0]$, for various levels of error probability thresholds, where P_e is the (average) decoded error probability. With an interleaver, it can usually be assumed that errors produced in the coded link are independent and hence we need only the average BER for each coded channel that is simulated. Note, however, that the notion of a BER distribution is meaningful or applicable only in a quasistatic environment, i.e., one where the channel state is relatively unchanging for a large number of symbols.

9.5.3. Speech Coder Simulation

Finally, the voice quality performance of the end-to-end link is evaluated by relating the average BER in the coded link to the MOS. This is done by sending digitized voice over the voice encoder and decoder and inserting errors in the coded voice bit stream at various BERs. The output of the decoder is recorded and played back to a set of listeners and their subjective assessment of voice quality is noted as a function of injected BER. A typical example is shown in Table 9.2. Using the MOS measure shown in this table, we can map the error probability distribution in the coded link to the distribution of MOS over the entire link. From the distribution of MOS we can calculate the outage probability at various MOS thresholds. If the underlying channel models have been parametrized in terms of rms delay spread, SNR, etc., then we can relate overall outage probability of the link to these parameters.

9.6. Summary

In this chapter we developed models, simulation techniques, and simulation methodologies for communication channels. Simulation models for communication channels take one of three forms: (1) transfer functions (2) TDL with tap-gain functions which are stationary random processes, or (3) discrete-time Markov sequences. The first two are waveform-level

Table 9.2. Mean Opinion Survey (MOS) Values as a Function of BER

Average BER	MOS
10^{-5}	5.0
10^{-4}	4.5
5×10^{-3}	3.5
10^{-3}	3.0
5×10^{-2}	2.0
10^{-2}	1.0

simulation models and the third one is used for simulating the discrete portion of the communication channel.

The most difficult part of modeling and simulation of communication systems is the modeling part. Several examples were presented with an emphasis on deriving the TDL and the Markov models. Once the models and their parameters are specified, simulation is straightforward.

In the case of fading channels, the simulation methodology used will depend on the end objective. For detailed design of subsystems and for design optimization, waveform-level simulation with dynamic channel models might be appropriate. For performance estimation (BER and outage probability), static channel models and hierarchical techniques are most appropriate, where applicable.

9.7. Appendix Reference Models for Mobile Channels

Discrete channel models are widely used for indoor and outdoor wireless system simulations. Given the large number of both mathematical and empirical models that have been proposed recently, the designer of communication systems is faced with the difficult problem of choosing a set of channel models that will represent the channels over which the communication system will operate satisfactorily. Fortunately, some guidance on the choice of which models to use has been provided by international standards bodies, which specify a set of *representative* channels for analyzing and simulating the performance of different communication systems. Such *standard* or *reference* models are also very important for comparison of performance of the different systems provided by industry. Below we present examples of reference channel models for three different applications.

9.A.1. Reference Channel Models for GSM Applications

Groupe Special Mobile, GSM, is one of the worldwide standardized systems for mobile communications, originally developed for the 1- to 2-GHz frequency range⁽⁶⁵⁾. The multiple access technique is time division multiplexing (TDMA), and the individual channels are 200 kHz wide.

Reference channel models have been developed for GSM applications, as shown in Tables 9.A.1–9.A.6. The recommended GSM models are discrete multipath models, specified for three different environments: rural, hilly, and urban. For the first, a 6-ray (path) model is

Table 9.A.1. Typical Profile for Rural Areas

Tap number	Relative time (μs)		Average relative power (dB)		Doppler spectrum
	(1)	(2)	(1)	(2)	
1	0.0	0.0	0.0	0.0	Ricean
2	0.1	0.2	−4.0	−2.0	Jakes
3	0.2	0.4	−8.0	−10.0	Jakes
4	0.3	0.6	−12.0	−20.0	Jakes
5	0.4	—	−16.0	—	Jakes
6	0.5	—	−20.0	—	Jakes

Table 9.A.2. Typical Profile for Hilly Terrain

Tap number	Relative time (μ s)		Average relative power (dB)		Doppler spectrum
	(1)	(2)	(1)	(2)	
1	0.0	0.0	-10.0	-10.0	Jakes
2	0.1	0.2	-8.0	-8.0	Jakes
3	0.3	0.4	-6.0	-6.0	Jakes
4	0.5	0.6	-4.0	-4.0	Jakes
5	0.7	0.8	0.0	0.0	Jakes
6	1.0	2.0	0.0	0.0	Jakes
7	1.3	2.4	-4.0	-4.0	Jakes
8	15.0	15.0	-8.0	-8.0	Jakes
9	15.2	15.2	-9.0	-9.0	Jakes
10	15.7	15.8	-10.0	-10.0	Jakes
11	17.2	17.2	-12.0	-12.0	Jakes
12	20.0	20.0	-14.0	-14.0	Jakes

Table 9.A.3. The Reduced Profile for Hilly Terrain (Six Taps)

Tap number	Relative time (μ s)		Average relative power (dB)		Doppler spectrum
	(1)	(2)	(1)	(2)	
1	0.0	0.0	0.0	0.0	Jakes
2	0.1	0.2	-1.5	-2.0	Jakes
3	0.3	0.4	-4.5	-4.0	Jakes
4	0.5	0.6	-7.5	-7.0	Jakes
5	15.0	15.0	-8.0	-6.0	Jakes
6	17.2	17.2	-17.7	-12.0	Jakes

Table 9.A.4. Typical Profile for Urban Area

Tap number	Relative time (μ s)		Average relative power (dB)		Doppler spectrum
	(1)	(2)	(1)	(2)	
1	0.0	0.0	-4.0	-4.0	Jakes
2	0.1	0.2	-3.0	-3.0	Jakes
3	0.3	0.4	0.0	0.0	Jakes
4	0.5	0.6	-2.6	-2.0	Jakes
5	0.8	0.8	-3.0	-3.0	Jakes
6	1.1	1.2	-5.0	-5.0	Jakes
7	1.3	1.4	-7.0	-7.0	Jakes
8	1.7	1.8	-5.0	-5.0	Jakes
9	2.3	2.4	-6.5	-6.0	Jakes
10	3.1	3.0	-8.6	-9.0	Jakes
11	3.2	3.2	-11.0	-11.0	Jakes
12	5.0	5.0	-10.0	-10.0	Jakes

Table 9.A.5. The Reduced Profile for Urban Area (Six Taps)

Tap number	Relative time (μs)		Average relative power (dB)		Doppler spectrum
	(1)	(2)	(1)	(2)	
1	0.0	0.0	-3.0	-3.0	Jakes
2	0.2	0.2	0.5	0.0	Jakes
3	0.5	0.6	-2.0	-2.0	Jakes
4	1.6	1.6	-6.0	-6.0	Jakes
5	2.3	2.4	-8.0	-8.0	Jakes
6	5.0	5.0	-10.0	-10.0	Jakes

Table 9.A.6. Profile for Equalization Test (Six Tap Setting)

Tap number	Relative time (μs)	Average relative power (dB)	Doppler spectrum
1	0.0	0.0	Jakes
2	3.2	0.0	Jakes
3	6.4	0.0	Jakes
4	9.6	0.0	Jakes
5	12.8	0.0	Jakes
6	16.0	0.0	Jakes

given, while for the latter two environments, two sets of models are offered, a 12-ray model and a 6-ray model. These correspond to a tapped-delay-line structural model with either 12 or 6 taps. The 6-tap model is offered as a computationally less demanding alternative, which may be used in particular in simulations that include interfering signals. Whenever possible, however, the full configuration (12 taps) should be used. For each model in the tables, two equivalent alternative tap settings (weights) and relative delays are indicated by (1) and (2) atop the appropriate columns.

It should be noted that for the standard signaling rate of 271 kb/s, the symbol duration is about $T_{\text{sym}} = 3.7 \mu\text{s}$, but some of the differential delays in the models are as small as $0.1 \mu\text{s}$, i.e., 74 times smaller than the symbol duration. To simulate properly such a delay would imply using a simulation sampling rate of 74 samples/symbol, which is obviously burdensome, or using an alternative approach such as outlined in the preceding section, where the model is “filtered” to produce a constant-spacing TDL.

Doppler Spectrum Types. As can be seen from Tables 9.A.1–9.A.6, the GSM channel models specify two Doppler spectra:

1. The classical (Jakes) spectrum

$$S(f) = \frac{A}{[1 - (f/f_d)^2]^{-1/2}}, \quad f \in [-f_d, f_d]$$

where $f_d = v/\lambda$, v/λ represents the vehicle speed (in m/s), and λ (in m) is the wavelength.

2. The Ricean spectrum, which is the sum of a classical Doppler spectrum and one direct line-of-sight path

$$S(f) = \frac{0.41}{2\pi f_d [1 - (f/f_d)^2]^{-1/2}} + 0.91\delta(f - 0.7f_d), \quad f \in [-f_d, f_d]$$

9.A.2. Reference Models for PCS Applications

For PCS communication systems operating in the 2-GHz band the standards bodies⁽⁶⁶⁾ have agreed on a set of discrete models for both indoor and outdoor environments. These models are summarized in Tables 9.A.7–9.A.9. In Table 9.A.7 the ray strength for each tap is defined as

$$\rho_k = E[|a_k|^2] \tag{9A.2.1}$$

and is given in Table 9.A.8. The proposed indoor models are given in Table 9.A.9. The weights of the individual rays are not specified.

It should be noted again that the differential delay of 50 ns in the indoor environment is very small compared to the symbol times proposed for PCS systems. Hence, either the fading should be treated as frequency-nonselective, or the bandlimited TDL model with symbol time spacing should be used for simulations.

Table 9.A.7. Parameters of Outdoor Models

Environment	$\tau_1 (\mu s)$	$\tau_2 (\mu s)$	$\tau_3 (\mu s)$	Spectrum	$f_d (\text{Hz})$
Pedestrian	0	1.5	14.5	Flat	12
Wireless loop	0	1.5	14.5	Gaussian	12
Vehicular	0	1.5	14.5	Jakes	180

Table 9.A.8. Relative Tap Strengths of Outdoor Models

Tap number k	Tap strength (dB) $10 \log \rho_k$
1	0
2	−3
3	−6

Table 9.A.9. Parameters of Indoor Models

Environment	Tap spacing (ns)	Number of taps	Tap spectrum Tap spectrum	Doppler $f_d (\text{Hz})$
Residential	50	2	Gaussian	3
Office	50	4	Gaussian	3
Commercial	50	12	Flat	30

9.A.3. Reference Channel Models for UMTS-IMT-2000 Applications

The next generation of wireless technology, designed to support broadband application, is in the planning stages. The International Telecommunication Union (ITU) is promoting a standard called IMT-2000, designed to support “universal mobile communications systems” (UMTS). The standard involves every aspect of system design and test and is voluminous⁽⁶⁷⁾ Here we provide an extract only of the propagation models for the terrestrial component of a system. These models include both path loss models (along the lines given in Section 9.1.2) and channel impulse response models (along the lines discussed in Section 9.1.3).

For terrestrial environments, the propagation effects are divided into three types of model: mean path loss, slow variation about the mean due to shadowing and scattering, and the rapid variation of the signal due to multipath effects. The channel impulse response is modeled using a tapped delay line as discussed in Section 9.1.3.

9.A.3.1. Path Loss Models

Equations are given for *mean* path loss as a function of distance for different terrestrial environments. The slow variation is taken to be log-normally distributed with a particular standard deviation and a specified form for the decorrelation length for the vehicular test environment.

9.A.3.1.1. Path Loss Model for Indoor Office Test Environment The indoor path loss model is given by

$$L = 37 + 30 \log_{10} R + 18.3n^{(n+2)/(n+1)-0.46} \quad [\text{dB}]$$

where R is the transmitter-receiver separation in meters and n is the number of floors in the path.

Note: L shall in no circumstances be less than the free-space loss. A log-normal shadow fading standard deviation of $\sigma = 12$ dB can be expected.

9.A.3.1.2. Path Loss Model for Outdoor-to-Indoor and Pedestrian Test Environments The following should be used for outdoor-to-indoor and pedestrian test environments:

$$L = 40 \log_{10} R + 30 \log_{10} f + 49 \quad [\text{dB}]$$

where R is the base station–mobile station separation in kilometers and f is the carrier frequency of 2000 MHz for IMT-2000/FPLMTS application.

Note: L shall in no circumstances be less than the free-space loss. This model is valid for non-line-of-sight case only and describes worst-case propagation. Log-normal shadow fading with a standard deviation of 10 dB for outdoor users and 12 dB for indoor users is assumed. The average building penetration loss is 12 dB with a standard deviation of 8 dB.

9.A.3.1.3. Path Loss Model for Vehicular Test Environments This model is applicable for the test scenarios in urban and suburban areas outside the high-rise core where structures are of nearly uniform height:

$$L = 40(1 - 4 \times 10^{-3} \Delta h_b) \log_{10} R - 18 \log_{10} \Delta h_b + 21 \log_{10} f + 80 \quad [\text{dB}]$$

where R is the base station–mobile station separation in kilometers, f is the carrier frequency of 2000 MHz for UMTS, and Δh_b is the base station antenna height, in meters, measured from the average rooftop level.

Note 1: The path loss model is valid for a range of Δh_b from 0 to 50 m.

Note 2: L shall in no circumstances be less than the free-space loss. This model is valid for the non-line-of-sight case only and describes worst-case propagation. Log-normal shadow fading with a standard deviation of 10 dB is assumed in both urban and suburban areas.

The base station antenna height is fixed at $\Delta h_b = 15$ m above the average rooftop, but each system may specify an alternate base station antenna height to optimize coverage and spectrum efficiency in their proposal. Considering a carrier of $f = 2000$ MHz and a fixed base station antenna height of $\Delta h_b = 15$ m, the above path loss formula becomes

$$L = 128.1 + 37.6 \log_{10} R \quad [\text{dB}]$$

9.A.3.1.4. Decorrelation Length of the Long-Term Fading The long-term (log-normal) fading in the logarithmic scale around the mean path loss L dB is characterized by a Gaussian distribution with zero mean and some standard deviation. Due to the slow fading process versus distance Δx , adjacent fading values are correlated. The normalized autocorrelation function $R(\Delta x)$ can be described with sufficient accuracy by an exponential function

$$R(\Delta x) = \exp \left[-\frac{|\Delta x|}{d_{\text{cor}}} \ln 2 \right]$$

The decorrelation length d_{cor} depends on the environment. This concept can be applied in the vehicular test environment with $d_{\text{cor}} = 20$ m. Although the evaluation of decorrelation length may not be fully valid in the outdoor-to-indoor and pedestrian environments, this concept is still to be applied with a $d_{\text{cor}} = 5$ m.

9.A.3.2. Channel Impulse Response Model

For each terrestrial test environment, a channel impulse response model based on a tapped delay line is given. The model is characterized by the number of taps, the time delay relative to the first tap, the average power relative to the strongest tap, and the Doppler spectrum of each tap. A majority of the time, rms delay spreads are relatively small, but occasionally, there are worst-case multipath characteristics that lead to much larger delay spreads. Measurements in outdoor environments show that rms delay spread can vary by over an order of magnitude within the same environment. Although large spreads occur relatively infrequently, they can have a major impact on system performance. To evaluate accurately the relative performance of candidate systems, it is desirable to model the variability of delay spread as well as the locations where delay spread is relatively large.

As this delay spread variability cannot be captured using a single tapped-delay-line model, up to two multipath channels are defined for each test environment. Within one test environment, channel A is the low-delay-spread case that occurs frequently, and channel B is a median-delay-spread case that also occurs frequently. Each of these two channels is expected to be encountered for some percentage of the time in a given test environment. Table 9.A.10 gives the percentage of time that the particular channel may be encountered with the associated rms average delay spread for channel A and channel B for each terrestrial test

Table 9.A.10. Parameters for Channel Impulse Response Model

Test environment	rms A (ns)	$p(A)$ (%)	rms B (ns)	$p(B)$ (%)
Indoor office	35	50	100	45
Outdoor-to-indoor and pedestrian	45	40	750	55
Vehicular, high antenna	370	40	4000	55

Table 9.A.11. Indoor Office Test Environment Tapped-Delay-Line Parameters

Tap number	Channel A		Channel B		Doppler spectrum
	Rel. delay (ns)	Avg. power (dB)	Rel. delay (ns)	Avg. power (dB)	
1	0	0	0	0	Flat
2	50	-3.0	100	-3.6	Flat
3	110	-10.0	200	-7.2	Flat
4	170	-18.0	300	-10.8	Flat
5	290	-26.0	500	-18.0	Flat
6	310	-32.0	700	-25.2	Flat

Table 9.A.12. Outdoor-to-Indoor Test Environment Tapped-Delay-Line Parameters

Tap number	Channel A		Channel B		Doppler spectrum
	Rel. delay (ns)	Avg. power (dB)	Rel. delay (ns)	Avg. power (dB)	
1	0	0	0	0	Jakes
2	110	-9.7	200	-0.9	Jakes
3	190	-19.2	800	-4.9	Jakes
4	410	-22.8	1200	-8.0	Jakes
5	—	—	2300	-7.8	Jakes
6	—	—	3700	-23.9	Jakes

Table 9.A.13. Vehicular Test Environment Tapped-Delay-Line Parameters

Tap number	Channel A		Channel B		Doppler spectrum
	Rel. delay (ns)	Avg. power (dB)	Rel. delay (ns)	Avg. power (dB)	
1	0	0.0	0	-2.5	Jakes
2	310	-1.0	300	0.0	Jakes
3	710	-9.0	8,900	-12.8	Jakes
4	1090	-10.0	12,900	-10.0	Jakes
5	1730	-15.0	17,100	-25.2	Jakes
6	2510	-20.0	20,000	-16.0	Jakes

environment. The 5% of the time not accounted for in Table corresponds to occurrences of large delay spread.

Tables 9.A.11–9.A.13 describe the tapped-delay-line parameters for each of the terrestrial test environments. For each tap of the channels three parameters are given: the time delay relative to the first tap, the average power relative to the strongest tap, and the Doppler spectrum of each tap. A small variation $\pm 3\%$ in the relative time delay is allowed so that the channel sampling rate can be made to match some multiple of the link simulation rate.

References

1. W. C. Jakes (ed.), *Microwave Mobile Communications*, Wiley, New York (1974).
2. J. D. Parsons, *The Mobile Radio Propagation Channel*, Halsted Press, New York, (1992).
3. R. Steele (ed.), *Mobile Radio Communications*, Pentech Press, London (1994).
4. J. G. Proakis, *Digital Communications*, McGraw-Hill, New York (1993).
5. B. Sklar, Rayleigh fading channels in mobile radio communications, Parts I and II, *IEEE Commun. Mag.* **35**(9), 136–155 (1997).
6. W. Y. C. Lee, *Mobile Cellular Communications*, McGraw-Hill, New York (1989).
7. T. S. Rappaport, *Wireless Communications*, Prentice-Hall, Upper Saddle River, New Jersey (1996).
8. V. Erceg *et al.*, Comparisons of a computer-based propagation prediction tool with experimental data collected in urban microcellular environments, *IEEE J. Select. Areas Commun.* **15**(4), 677–684 (1997).
9. S. C. Kim *et al.*, Radio propagation measurement and prediction using three-dimensional ray tracing in urban environments at 908 MHz and 1.9 GHz, *IEEE Trans. Vehic. Technol.* **VT-43**(3), 931–946 (1999).
10. M. Hata Empirical formulae for propagation loss in land mobile radio services, *IEEE Trans. Vehic. Technol.* **VT-29**(3), 317–325 (1980).
11. Y. Okumura, E. Ohmori, and K. Fukuda, Field strength and its variability in VHF and UHF land mobile radio service, *Rev. Elec. Commun. Lab.* **16**(9, 10), 825–873 (1968).
12. COST-231, Urban Transmission Loss Models for Mobile Radio in the 900 MHz and 1800 MHz Bands (rev. 2), COST 231 TD(90), 119 Rev. 2, Den Haag (1991).
13. S. Y. Seidel *et al.*, Path loss, scattering and multipath delay statistics in four European cities for digital cellular and microcellular radiotelephone, *IEEE Trans. Vehic. Technol.* **VT-40**(4), 721–730 (1991).
14. P. A. Bello, Characterization of randomly time-variant linear channels, *IEEE Trans. Commun. Syst.* **CS-11**(4), 360–393 1963.
15. B. Glance and L. J. Greenstein, Frequency-selective fading effects in digital mobile radio with diversity combining, *IEEE Trans. Commun.* **COM-31**(9), 625–635 (1983).
16. R. H. Clarice, A statistical theory of mobile radio reception, *Bell Syst. Tech. J.* **47**(6), 957–1000 (1968).
17. F. Amoroso, Use of DS/SS signaling to mitigate Rayleigh fading in a dense scatter environment, *IEEE Pers. Commun.* **3**(2), 52–61 (1996).
18. D. C. Cox, Delay-Doppler characteristics of multipath propagation at 910 MHz in a suburban mobile radio environment, *IEEE Trans. Ant. Prop.* **AP-20**(9), 625–635 (1972).
19. H. L. Van Trees, *Detection, Estimation and Modulation Theory*, Part 3, Wiley, New York (1968).
20. S. A. Fechtel, A novel approach to modeling and efficient simulation of frequency-selective fading radio channels, *IEEE J. Select. Areas Commun.* **11**(3), 422–431 (1993).
21. M. B. Priestly, *Spectral Analysis and Time Series*, Vol. 2, Academic Press, London (1981).
22. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed., McGraw-Hill, New York (1991).
23. B. N. Datta, *Numerical Linear Algebra and Applications*, Brooks Cole, Pacific Grove, California (1995).
24. H. Bateman, *Table of Integral Transforms*, Vol. I, McGraw-Hill, New York (1954).
25. T. S. Rappaport, Characterization of UHF multipath radio channels in factory buildings, *IEEE Trans. Ant. Prop.* **37**(8), 1058–1069 (1989).
26. T. S. Rappaport, S. Y. Seidel, and K. Takamizawa, Statistical channel impulse response models for factory and open plan building radio communication system design, *IEEE Trans. Commun.* **39**(5), 794–807 (1991).
27. R. Ganesh and K. Pahlavan, On the modeling of fading multipath indoor radio channels, in *Proceedings of Global Telecommunications Conference* pp. 1346–1350 (1989).
28. R. Ganesh and K. Pahlavan, Statistical modelling and computer simulation of Indoor radio channel, *Proc. IEEE* **138**(3), 153–161 (1991).

29. J. B. Anderson, T. S. Rappaport, and S. Yoshida, Propagation measurements and models for wireless communications channels, *IEEE Commun. Mag.* **42**–49 (1995).
30. A. A. M. Saleh and R. A. Valenzuela, A statistical model for indoor multipath propagation, *IEEE J. Select. Areas Commun.* **SAC-54**(2), 128–137 (1987).
31. S. C. Kim, H. L. Bertoni, and M. Stern, Pulse propagation characteristics at 2.4 GHz inside buildings, *IEEE Trans. Vehic. Technol.* **45**(3), 579–592 (1996).
32. S. F. Fortune *et al.*, Wise design of indoor wireless systems: Practical computation and optimization, *IEEE Comput. Sci. Eng.* **1995**(Spring), 58–68.
33. W. D. Rummler, A new selective fading model: Application to propagation data, *Bell Syst. Tech. J.* **58**(5), 1037–1071 (1979).
34. P. Balaban, Statistical models for amplitude and delay of selective fading, *ATTech. J.* **64**(10), 2525–2250 (1985).
35. G. H. Millman, Atmospheric and Extraterrestrial Effects on Radio Wave Propagation, General Electric Co., Technical Information Series, No. R61EMH29 (June 1961).
36. W. L. Flock, Propagation Effects on Satellite Systems at Frequencies below 10 GHz, NASA Reference Publication 1108 (December 1983).
37. L. J. Ippolito, R. D. Kaul, and R. G. Wallace, Propagation Effects Handbook for Satellite Systems Design, NASA Reference Publication 1082(03), (June, 1983).
38. P. Lo, J. Haddon, H. O'Neill, and E. Vilar, Computation of rain induced scintillations on satellite down-links at microwave frequencies, *IEE Conference on Communications*, Publication 219.
39. H. J. Liebe, Modeling attenuation and phase of radio waves in air at frequencies below 1000 GHz, *Radio Sci.* **16**(6), 1183–1199 (1981).
40. K. S. Shanmugan, M. S. McKinley, V. S. Frost, E. M. Friedman, and J. C. Holtzman, Wideband digital transmission through the atmosphere at EHF frequencies: Effects of refractive dispersion, *Proc. Globecom Conference*, Atlanta, Georgia (1984).
41. R. K. Crane, Prediction of attenuation by rain, *IEEE Trans. Commun.* **COM-28**, 1717–1733 (1980).
42. L. J. Ippolito, Jr., *Radiowave Propagation in Satellite Communications*, Van Nostrand Reinhold, New York (1986).
43. D. M. Jansky and M. C. Jeruchim, *Communication Satellites in the Geostationary Orbit*, 2nd ed., Artech House, Norwood, Massachusetts (1987).
44. G. H. Millman and M. C. Arabadji, Tropospheric and Ionospheric Phase Perturbations and Doppler Frequency Shift Effects, General Electric Co., Technical Information Series No. R84EMH003 (August 1984).
45. S. Ramo, J. R. Whinnery, and T. Van Duzer, *Fields and Waves in Communication Electronics*, Wiley, New York (1965).
46. J. C. Daly, (ed.), *Fiber Optics*, CRC Press, Boca Raton, Florida (1984).
47. G. Keiser, *Optical Fiber Communications*, McGraw-Hill, New York (1983).
48. S. Karp, R. M. Gagliardi, S. E. Moran, and L. B. Stotts, *Optical Channels: Fibers, Clouds, Water, and the Atmosphere*, Plenum Press, New York (1988).
49. D. G. Duff, Computer-aided design of digital lightwave systems, *IEEE J. Select. Areas Commun.* **SAC-2**(1), 171–185 (1984).
50. P. K. Cheo, *Fiber Optics Devices and Systems*, Prentice-Hall, Englewood Cliffs, New Jersey (1985).
51. J. K. Townsend, Computer simulation of digital lightwave communication links, Ph. D. thesis, University of Kansas, Lawrence, Kansas (June 1988).
52. J. Gimlett and N. Cheung, Dispersion penalty analysis for LED/single-mode fiber transmission systems, *IEEE J. Lightwave Technol.* **LT-4**, 1381–1392 (1986).
53. A. Elrefaie, R. E. Wagner, D. A. Atlas, and D. G. Daut, Chromatic dispersion limitations in coherent optical fiber transmission systems, *IEE Electron. Lett.* **23**(14), 756–758 (1987).
54. L. N. Kanal and A. R. K. Sastry, Markov models for channels with memory and their application to error control, *Proc. IEEE*, **66**, 724–744, (1978).
55. F. Swarts and H. C. Ferreira, Markov Characterization of digital fading mobile channels, *IEEE Trans. on Vehicular Tech.*, **43**, 977–985, (1994).
56. C. Chao and Y. Yao, HMM models for the burst error characteristics of Viterbi decoding, *Proceedings of ICC'93*, 751–756, (1993).
57. L. R. Rabiner and B. H. Huang, An introduction to hidden Markov models, *IEEE ASSP Mag.* **1986** (January), 4–16.
58. S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, An introduction to the theory and application of Markov processes to automatic speech recognition, *Bell Syst. Tech. J.* **62**, 1035–1074 (1983).
59. E. N. Gilbert, Capacity of a burst noise channel, *Bell Syst. Tech. J.* **39**, 1253–1265 (1960).

60. B. D. Fritchman, A binary channel characterization using partitioned Markov chains, *IEEE Trans. Inform. Theory* **IT-13**, 221–227 (1967).
61. L. E. Baum. *et. al.*, A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *Annals of Math. Statistics*, **11**, 164–171, (1970).
62. W. Turin, *Digital Transmission Systems: Performance Analysis and Modeling*, McGraw-Hill, New York (1998).
63. W. Turin and M. M. Sondhi, Modeling of error sources in digital transmission systems, *IEEE J. Sect. Areas Commun.* **11**, 340–347 (1993).
64. S. Sivaprakasam and K. Sam Shanmugan, A Markov model for burst error channels, *IEEE Trans. Commun.* **43**, 1347–1355 (1995).
65. ETSI, *GSM Recommendation 05.05, Radio Transmission and Reception, Annex 3*, 13–16 (November 1988).
66. ANSI J-STD-008, *Personal Station–Base Station Compatibility Requirements for 1.8 to 2.0 GHz Code Division Multiple Access (CDMA) Personal Communications Systems* (March 1995).
67. Recommendation ITU-R M.1225, *Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000, Annex 2* (1997).

This page intentionally left blank

Estimation of Parameters in Simulation

As mentioned in Chapter 2, a Monte Carlo simulation run can be viewed as a statistical experiment which is the software counterpart of an experiment on an actual system. The objective of the experiment is to allow us to make inferences about one or more of the properties of the signal as it passes through the system. For purposes of this book, we shall place such properties into one of two categories that we shall refer to as *parameters* and *performance measures*, respectively. By the latter, we mean measures related in some way to the fidelity of transmission. Here, too, we can subdivide this topic into two broad subtopics, one of which deals with the signal-to-noise ratio and the other with error performance in digital systems. This subject will be treated in the next chapter. In the present chapter we will deal with extracting various signal parameters that may be of interest. We are defining here “parameters” to be properties of the signal not directly phrased in term of performance measures. Such parameters may be of interest for a number of reasons. Specifically, the parameters in which we shall be interested are the average level, average power, probability distribution and density, power spectral density, delay, and phase shift.

The observations (measurements) consist of discretely spaced values of a finite-duration segment of a waveform at some point in the system. In a Monte Carlo simulation, the waveform is a sample path of a random process. Hence the measurements are inherently random. This means the inferences made can only be statistical. In Chapter 2 we alluded to alternative simulation methodologies that may not be interpretable as random experiments. Such methodologies will be particularly in evidence in the next chapter. Of course, in such cases, the standard language of statistical estimation may not apply strictly, but we may still use such terms as “estimation” in a connotative sense. More will be said about this in the next chapter.

We have emphasized the point that the main commodity consumed in simulation is run time, and, particularly in Monte Carlo simulation, this will often be the simulator’s ultimate limitation. That is, the number of waveform samples to be observed will be restricted by run-time considerations (this, too, will be examined in the next chapter), whereas in the great majority of cases, a corresponding experiment on the real system in real time will have no such limitations. Our focus in this and the next chapter, therefore, will be to highlight these run-time constraints in estimating quantities of interest.

10.1. Preliminaries

In this section we set the stage for what is to come in this and the next chapter by first reviewing some basic ideas and definitions and establishing notation.

10.1.1. Random Process Model: Stationarity and Ergodicity

Several aspects of random process theory necessary for this book have been reviewed in Chapter 6. The notions of this theory that are of particular relevance in this chapter are those of stationarity and ergodicity. These ideas allow us to attribute more general significance to a single measurement than merely the measurement itself. The key to such an interpretation is the visualization of a random process as an ensemble of functions of time and embedding a simulation run within such a model. As we know from Chapter 6, the ergodic property basically means that we can equate a time average over a particular sample function (over $-\infty < t < \infty$) with an ensemble average. Consequently, this means we can equate a time average over a particular sample function with that over any other sample function in the ensemble. (Ergodicity can therefore be thought of as establishing a transitivity property among members of the ensemble.) This is the basis for attributing “global” significance to a measurement performed on a single sample function (i.e., a single simulation run). Ergodicity requires that the process be stationary. This means that the statistical properties of estimators are time-invariant.

There is an important class of processes in communications for which ergodicity and stationarity may not be unconditionally true. This is the class for which every sample function has associated with it an explicit or implicit (hidden) periodicity. Every modulated carrier, for example, exhibits an explicit periodicity in the carrier wave itself, although, in the simulation context, this periodicity will not manifest itself whenever we use the complex envelope representation. Samples of waveforms have implicit periodicity associated with them because of the clock inherent in the sampling process. The general term given to these processes is that they are *cyclostationary*. If, for such processes, sample path properties are equal to the corresponding ensemble properties, then the processes are said to be *cycloergodic*. As before, the main issue here relates to the generalizability of a measurement. For such processes, ergodicity may or may not hold, depending upon the model assumed for certain parameters of a process. Suppose, for example, that $X(t) = A \cos(\omega_c t + \theta)$, where θ is a random variable with uniform distribution over $(-\pi, \pi)$; then $X(t)$ is ergodic in the mean and autocorrelation function (or “weakly” ergodic). Similarly, $X(t) = A(t) \cos(\omega_c t + \theta)$ is weakly ergodic if $A(t)$ is ergodic. The key is the inclusion (or interpretation) of θ , which is often overlooked. The specific issues involved have been discussed in detail by Gardner.⁽¹⁻³⁾

It is generally not simple to ascertain ergodicity, except for Gaussian processes (see Section 6.7.3). We take it as granted that any estimate computed in a single simulation run can have a meaningful general interpretation if the necessary properties, such as stationarity or ergodicity, do apply. It is important to make clear that such ensemble properties are desirable only for the purpose of placing a simulation run in a larger context. They are not necessary for the mechanization of a simulation run, nor are they required to make a simulation run useful.

10.1.2. Basic Notation and Definitions

It is useful to review the notational conventions established in Chapter 6 (Section 6.7.1) and adapt them to the present context. As we have seen, a random process can be looked upon

as a collection of time functions, and we shall denote the process $X(t)$. Any given specific function in this ensemble is labeled $x(t)$ or $\mathbf{x}_i(t)$ if the set of functions is denumerable. When we perform a simulation run, we are dealing with one such $x(t)$.

In general, we use $X(t)$ to represent the waveform of interest (voltage or current) at some point in the system. We are also often interested in some function of $X(t)$, say $Y(t) = g[X(t)]$; foreexample, $Y(t) = X^2(t)$ or $Y(t) = |X(t)|$. Our usual objective in simulation is to estimate a property or parameter Q of $Y(t)$. This we do by a suitable operation on an observation of $Y(t)$.

We define an *observation* either as a finite-time “slice” of the process

$$\mathbf{Y}_T(t) = \begin{cases} Y(t), & 0 \leq t \leq T \\ 0, & \text{elsewhere} \end{cases} \quad (10.1.1a)$$

or as the sequence of samples

$$\mathbf{Y}_N = (Y_1, Y_2, \dots, Y_N) \quad (10.1.1b)$$

where

$$Y_i = Y((i-1)T_s)$$

T_s is the simulation sampling interval, and $(N-1)T_s \approx T$. The specific interval $[0, T]$ is so denoted for convenience, but by the stationarity assumption the interval could be located anywhere in time. Unless there is reason to distinguish between the continuous-time observation (10.1.1a) and the discrete-time observation (10.1.1b), and unless there is a need to state the length (T or N) of the interval, we shall denote an observation simply as $\mathbf{Y}$ and refer to it as a vector. Note that $\mathbf{Y}$ is a random vector. Once a sample function $y(t)$ has been chosen, the corresponding vector $\mathbf{y}$ is a nonrandom quantity and can be referred to as a measurement. Of course, in the simulation context we are always obliged to deal with discrete-time observations or measurements.

Any function of the random vector $\mathbf{Y}$, $G(\mathbf{Y})$, is a random variable and is called a *statistic*. If by the operation G we attempt to estimate the characteristic Q of the process, then we say that

$$\hat{Q} = G(\mathbf{Y}) \quad (10.1.2)$$

is an estimator of Q (or, sometimes, a point estimator) and a specific value $G(\mathbf{y})$ computed from the measurement $\mathbf{y}$ is called an estimate. The caret is standardly used to denote an estimator. In principle, there can be many estimators for the same characteristic Q . There are established procedures for determining “good” estimators, and we refer the reader to the many excellent texts in statistics (e.g., Refs. 4–10). We shall therefore take it for granted that the estimators we choose have a reasonable basis.

For almost all cases of interest, the estimator can be expressed as a weighted time average:

$$\hat{Q} = \frac{1}{N} \sum_{i=1}^N w_i Y_i \quad (10.1.3)$$

We shall also consistently denote a time average with angular brackets

$$\hat{Q} = \langle \mathbf{wY} \rangle_N \quad (10.1.4)$$

where the subscript N will be made explicit only if required for clarity. The product of vectors, as in $\mathbf{wY}$, is interpreted as the ordinary scalar, or “dot” product, namely, the summation in (10.1.3). Since $\hat{Q}$ is a random variable (because it depends on the chosen sample function) it has an associated distribution and pdf, say $f_{\hat{Q}}(q; N)$, which are properties of the ensemble. Hence, all average properties (moments) of $\hat{Q}$ are determined by $f_{\hat{Q}}$, and such averages will be indicated by the expectation operator E . Since we normally expect the estimation process to improve as N increases, an essential attribute of an estimator is that it converges to the true value as $N \rightarrow \infty$. Such an estimator is termed *consistent*. By the ergodic property, $\hat{Q}$ will tend to Q as $N \rightarrow \infty$ for almost every sample function.

Of course we are always limited to a finite, and perhaps small, number of samples, and it is under these conditions that we wish to determine the quality of an estimator, where quality means closeness to the true value in some sense. We consider this question next.

10.1.3. Quality of an Estimator: Bias, Variance, Confidence Interval, and Time-Reliability Product

The quality of an estimator, as just stated, can be judged by how close it is to the true value of the parameter. Since $\hat{Q}$ is a random variable, “closeness” must be measured in a probabilistic sense. In this book, three measures of quality of an estimator are of interest: the bias, the variance, and the confidence interval. The latter two are generally used as alternatives to describe the “spread” of an estimator. The confidence interval is more descriptive, but generally difficult to obtain, especially for small samples. One then has to be content with a cruder, but still useful measure, namely, the variance. A measure that we shall find useful, the time-reliability product, combines the variance with the run-time measure.

One general point that is worthwhile remembering is that the degree to which we are able to make progress in formulating expressions and arriving at useful results is very much dependent upon what we know or are willing to assume about the properties of the processes under examination.

10.1.3.1. Bias of an Estimator

An estimator is said to be unbiased if

$$\begin{aligned} E(\hat{Q}) &= \int_{-\infty}^{\infty} q f_{\hat{Q}}(q; N) dq \\ &= \int_{-\infty}^{\infty} G(\mathbf{y}) f_Y(\mathbf{y}) d\mathbf{y} = \int_{-\infty}^{\infty} G[\mathbf{g}(\mathbf{x})] f_X(\mathbf{x}) d\mathbf{x} \end{aligned} \quad (10.1.5)$$

is equal to Q . The equality between the integrals is a consequence of the fundamental theorem of expectation.⁽¹⁰⁾ This allows us to compute averages without explicitly knowing the distribution $f_{\hat{Q}}$, which is generally hard to find, and rely only on the underlying distribution, which can be assumed known. If $E(\hat{Q}) \neq Q$, the estimator is said to be biased. The value of an unbiased estimator is equal to the true value, “on the average,” but the value of any specific

estimate obtained from a simulation run may not be equal to the true value. Unbiasedness is an important property, particularly for a BER estimator. Heuristically, that property guarantees that estimates will generally be clustered around the true value.

10.1.3.2. Variance of an Estimator

The variance of an estimator,

$$\sigma^2(\hat{Q}) = E(\hat{Q}^2) - E^2(\hat{Q}) \quad (10.1.6a)$$

$$= \int_{-\infty}^{\infty} q^2 f_{\hat{Q}}(q; N) dq - E^2(\hat{Q}) \quad (10.1.6b)$$

is a measure of the dispersion about its expected value. The smaller the variance, the better the estimator, for a given N . In fact, if $\sigma^2(\hat{Q}) \rightarrow 0$, and $\hat{Q}$ is unbiased, then every observation is arbitrarily close to the true value of the parameter. Generally, $\sigma^2(\hat{Q}) \rightarrow 0$ only as $N \rightarrow \infty$, and we are then faced with a tradeoff between estimator variance and sample size N , the latter being directly proportional to run time in the simulation context.

10.1.3.3. Confidence Interval

The confidence interval is the most descriptive measure of the quality of an estimator because it quantifies the measure of spread with an associated probability. Let $h_1(\hat{Q}), h_2(\hat{Q})$ be two functions of the estimator such that, with high probability, the interval (h_1, h_2) brackets the true value Q . The difference $h_1 - h_2$ is the width of the confidence interval. The probability associated with the condition $h_2 \leq Q \leq h_1$ is called the confidence level and is usually denoted $1 - \alpha$. Thus, a confidence interval and confidence level are defined through

$$P[h_2(\hat{Q}) \leq Q \leq h_1(\hat{Q})] = 1 - \alpha \quad (10.1.7)$$

Typical values for α are 0.05 and 0.01, corresponding to “95% confidence level” and “99% confidence level,” respectively. The left side of (10.1.7) is often labeled a “fiduciary” probability because Q is not a random variable, but an unknown parameter. The proper interpretation of (10.1.7) is that, with probability $1 - \alpha$, the random interval defined by (h_1, h_2) will bracket the true value.

To illustrate the procedure for obtaining a confidence interval, assume that the pdf of the estimator $f_{\hat{Q}}(q)$ is known. We assume that $f_{\hat{Q}}$ is unimodal and that the mode is at or near Q , as illustrated in Figure 10.1. Let $\sigma_{\hat{Q}}$ be the SD of $\hat{Q}$ and let d_1 and d_2 be positive constants such that

$$P[\hat{Q} - Q \leq -d_1 \sigma_{\hat{Q}}] = \alpha_1 \quad (10.1.8a)$$

$$P[\hat{Q} - Q \geq d_2 \sigma_{\hat{Q}}] = \alpha_2 \quad (10.1.8b)$$

as shown in the figure. Combining (10.1.8a) and (10.1.8b), we get

$$P[\hat{Q} - d_1 \sigma_{\hat{Q}} \leq Q \leq \hat{Q} + d_2 \sigma_{\hat{Q}}] = 1 - \alpha \quad (10.1.9)$$

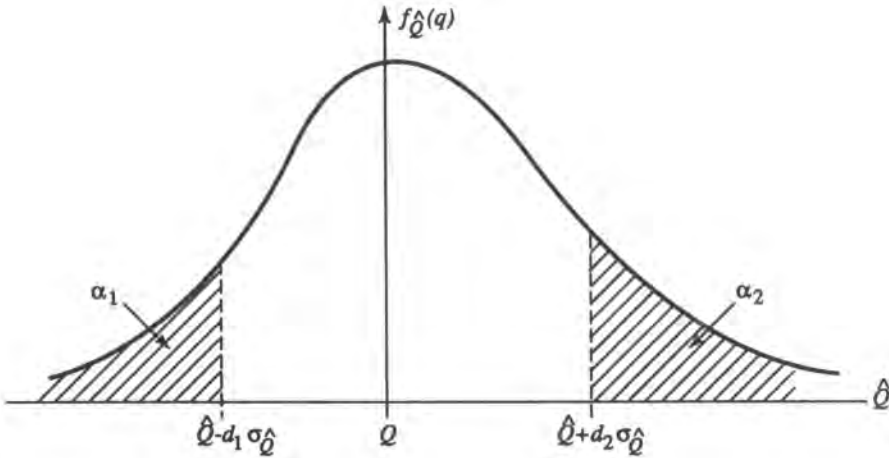


Figure 10.1. Definition of confidence interval and confidence level.

which is in the form (10.1.7), where $\alpha = \alpha_1 + \alpha_2$. The confidence interval $[\hat{q} - d_1 \sigma_{\hat{Q}}, \hat{q} + d_2 \sigma_{\hat{Q}}]$ is constructed around the observed value $\hat{q}$ and has length $(d_1 + d_2) \sigma_{\hat{Q}}$. For given $1 - \alpha$ it is desirable to find d_1, d_2 so as to minimize this length, although it is usually adequate to take $d_1 = d_2$. ($f_{\hat{Q}}$ may not be symmetric, but if it is, the interval length is minimized if $d_1 = d_2$.) If $\hat{Q}$ is the sample mean of N variates, say

$$\hat{Q} = \frac{1}{N} \sum_{k=1}^N Z_k \quad (10.1.10)$$

where the Z_k are i.i.d. with SD σ , then $\sigma_{\hat{Q}} = \sigma/\sqrt{N}$.

The construction of (10.1.9) requires knowledge of $f_{\hat{Q}}$, which is difficult to obtain for many estimates. This is why one often settles for the variance (usually much easier to find) as an estimator of spread. However, there are some important cases when the estimator distribution can be obtained exactly or approximately. One case is when the estimator is a linear function of samples of a Gaussian process because the estimator then has a normal distribution. For example, if in (10.1.10), $Z_k \in N(\mu, \sigma^2)$, then $\hat{Q} \in N(\mu, \sigma^2/N)$ and a confidence interval for the mean can be established if σ is known. In the same situation, we cannot construct a confidence interval if σ is not known. In that case the well-known alternative is to use Student's t -distribution. The variable

$$t = \frac{\hat{Q} - \mu}{S} (N - 1)^{1/2} \quad (10.1.11)$$

where $\hat{Q}$ is given by (10.1.10) and

$$S^2 = \frac{1}{N} \sum_{k=1}^N (Z_k - \hat{Q})^2 \quad (10.1.12)$$

is independent of σ , and its distribution is tabulated in many texts (see, e.g., Refs. 5, 10, and 11). We can thus find a confidence interval for μ by setting up the expression

$$P[t_1 \leq t \leq t_2] = P[\hat{Q} - t_2 S / (N - 1)^{1/2} \leq \mu \leq \hat{Q} - t_1 S / (N - 1)^{1/2}]$$

In probably most cases of interest in simulation it is not necessary to use the t -distribution because it converges rapidly to the normal $N(0, 1)$ for N greater than about 30.

In the situation when $\hat{Q}$ is the sum of a large number of variables, for example, (10.1.10) with N large, under the conditions of the central limit theorem (see Chapter 6) $\hat{Q}$ tends to the normal distribution and this distribution can be used as an approximation to obtain a confidence interval.

10.1.3.4. Time-Reliability Product

A measure of the goodness of an estimator that is also useful in the simulation context is the time-reliability product,

$$\zeta \triangleq N\sigma^2(\hat{Q})$$

which is the product of the number of observations and the variance of the estimator. This one figure embodies the tradeoff between the “reliability” of the estimator and the “time” of observation N , which is proportional to run time. It is useful for comparing one estimator against another and will be made use of in Chapter 11.

10.1.3.5. Normalized Measures

Measures of goodness such as bias and variance are most useful when referenced to the quantities being measured. For example, a bias of magnitude 10^{-6} seems very good on its face, but renders useless the estimation of a quantity (such as an error probability) suspected to be on the order of 10^{-7} . Therefore, measures of estimator goodness normalized to the true value of a parameter (or possibly its estimated value) are generally more meaningful. For example, we might define a normalized bias as $b(\hat{Q}) = (Q - \hat{Q})/Q$ or simply $\hat{Q}/Q$. Ideally, the first of these is zero, while the second is unity. Similarly, we can define a relative precision or normalized standard deviation as $\sigma(\hat{Q})/Q$. These relative measures are especially useful when the quantities to be estimated are small, as in error probability estimation, where the absolute measures may give us a false sense of comfort, as in Figure 10.1.

10.2. Estimating the Average Level of a Waveform

We may sometimes wish to estimate the average level of a waveform within or prior to a simulation run. The main utility of such a measurement will typically be for calibration purposes. For example, if two system elements are intended to be ac-coupled, the output of the coupling network should have a long-term zero average. An observation at that node during simulation will confirm this, or possibly uncover a “bug.”

10.2.1. Form of the Estimator

As before, $X(t)$ is the waveform (process) of interest. An illustrative waveform is shown in Figure 10.2, along with samples $X(kT_s) \equiv X_k$ separated by the simulation sampling interval T_s . The natural estimator of the average level is the sample mean, namely,

$$\langle \mathbf{X} \rangle = \frac{1}{N} \sum_{k=1}^N X_k \quad (10.2.1)$$

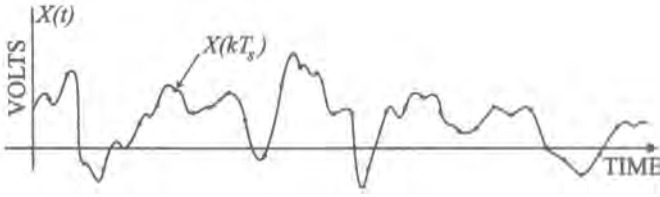


Figure 10.2. A segment of a sampled waveform $\{X(kT_s)\}$; its average is $\bar{X} = (1/N) \sum_{k=1}^N X(kT_s)$.

10.2.2. Expected (Mean) Value of the Estimator

As we discussed, an estimator is a random variable. Its expected (or mean) value is thus of interest and is given by

$$E(\langle \mathbf{X} \rangle) = E\left(\frac{1}{N} \sum_{k=1}^N X_k\right) = \frac{1}{N} \sum_{k=1}^N E(X_k) = \frac{1}{N} \sum_{k=1}^N E(X) = E(X) \quad (10.2.2)$$

where we have used successively the distributive property of expectation and stationarity. So, as is well known, the average value of the sample is the same as that of the population, irrespective of N .

Although we cannot do a continuous-time measurement in the simulation context, the objective of the discrete sum may be to estimate the average of a continuous-time process, which ideally would be estimated by

$$\langle \mathbf{X}_T \rangle = \frac{1}{T} \int_0^T X(t) dt$$

The average over all possible sample functions is simply

$$E(\langle \mathbf{X}_T \rangle) = \frac{1}{T} \int_0^T E(X(t)) dt = E(X)$$

Because of stationarity, $E(X(t))$ is a constant, so that the expectation is independent of t . Hence, the (ensemble) average of the measurement in this particular instance is the same whether a continuous-time average were possible or whether a discrete average is taken.

10.2.3. Variance of the Estimator

The variance of the estimator is the ensemble average of the square of the difference between the value of the measurement and its expected value. Symbolically we have

$$\text{Var}(\langle \mathbf{X} \rangle) = E(\langle \mathbf{X} \rangle - E(\langle \mathbf{X} \rangle))^2 \quad (10.2.3a)$$

$$= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N C_{XX}(i, j) \quad (10.2.3b)$$

where C_{XX} is the autocovariance of the process,

$$C_{XX}(i, j) = E([X_i - E(X_i)][X_j - E(X_j)]) \quad (10.2.4)$$

By virtue of stationarity, $E(X_i) = E(X_j) = E(X)$ and

$$C_{XX}(i, j) = C_{XX}(i - j) \quad (10.2.5)$$

Thus, substituting (10.2.5) into (10.2.3b) and relabeling $i - j = k$, we have

$$\text{Var}(\langle \mathbf{X} \rangle) = \frac{\sigma_X^2}{N} + \frac{2}{N^2} \sum_{k=1}^{N-1} (N - k) C_{XX}(kT_s) \quad (10.2.6)$$

where $C_{XX}(0) = \sigma_X^2$ is the variance of the process, and for clarity we have made time explicit by including the sampling interval T_s .

Not much more can be said without making specific assumptions about C_{XX} . Let us assume that $C_{XX}(\tau)$ essentially vanishes for $\tau > \tau_0$, with $\tau_0 = |k_0|T_s$. For physical processes, this will always be at least approximately true. Suppose T_s is sufficiently larger than τ_0 so that we can safely assume $C_{XX}(kT_s) = 0$ for all $k \geq 1$. Then the sum in (10.2.6) is zero, so that

$$\text{Var}(\langle \mathbf{X} \rangle) = \sigma_X^2 / N \quad (10.2.7)$$

Hence $\text{Var}(\langle \mathbf{X} \rangle) \rightarrow 0$ as $N \rightarrow \infty$, which implies, from (10.2.3a), that

$$\lim_{N \rightarrow \infty} E(\langle \mathbf{X} \rangle - E(\langle \mathbf{X} \rangle))^2 = 0$$

Heuristically, we can interpret this as saying that the average for almost every sample path tends to the ensemble average as the number of observations becomes large.

Now suppose $T_s \rightarrow 0$. Two situations present themselves here. Since $NT_s = T$, we can assume that the observation interval T remains constant, in which case $N \rightarrow \infty$, or we can assume that N remains finite, in which case $T \rightarrow 0$. Actually, neither of these cases is very meaningful in the simulation context because, in the first instance, the number of samples is infinite, and in the second, all the samples are essentially at the same instant, namely, completely redundant. So this latter case is not of much interest, but there is something to be learned from the first case ($NT_s = T$, with T fixed).

In particular, it can be shown (see Chapter 6 and Ref. 1) that, as $T_s \rightarrow 0$, Equation (10.5.3) takes the form

$$\lim_{T_s \rightarrow 0} \text{Var}(\langle \mathbf{X} \rangle) = \frac{2}{T} \int_0^T \left(1 - \frac{\tau}{T}\right) C_{XX}(\tau) d\tau \quad (10.2.8)$$

Since $C_{XX}(\tau) \leq C_{XX}(0) = \sigma_X^2$, we can bound (10.2.8) as

$$\text{Var}(\langle \mathbf{X} \rangle) \leq \frac{2\tau_0}{T} \sigma_X^2, \quad \tau_0 < T \quad (10.2.9a)$$

$$\leq 2\sigma_X^2, \quad \tau_0 \geq T \quad (10.2.9b)$$

Condition (10.2.9b) confirms the fact that the condition $\tau_0 \geq T$ is to be avoided, as might be expected. Condition (10.2.9a) may be rephrased as

$$\text{Var}(\langle \mathbf{X} \rangle) \leq \frac{\sigma_X^2}{N_e} \quad (10.2.9c)$$

where

$$N_e \triangleq T/2\tau_0$$

can be viewed as an “effective” number of independent samples, by analogy with (10.2.6). Therefore, even though there is an infinite number of samples, they do not contribute independently to the reliability of the estimate. In fact, since we are limited to a finite number of simulation observations, it seems from the foregoing that the best strategy is to space the samples far enough apart that there is little or no correlation between adjacent samples. We note that the reliability of the estimates does not explicitly depend on T_s , which would appear to give us the freedom to choose T_s large enough so that samples are independent. But, in fact, the sampling theorem (Chapter 3) constrains us from using too large a value for T_s . Thus, it is not generally possible to choose T_s so that it both satisfies the sampling theorem and produces uncorrelated samples.

However, there is one important special case where both requirements are satisfied. This case is that for which the waveform $X(t)$ has a power spectral density (PSD) given in Figure 10.3. The autocorrelation function corresponding to this PSD is given by

$$R_{XX}(\tau) = 2AB[\sin(2\pi B\tau)/2\pi B\tau] \quad (10.2.10)$$

The sampling theorem for random processes (Section 6.10) stipulates that samples separated by $1/2B$ are sufficient to represent the process. But, if $\tau = 1/2B$, $R_{XX}(1/2B) = 0$; hence the samples are uncorrelated.

What is a suitable value of $\text{Var}(\langle \mathbf{X} \rangle)$? Generally, we would like $\sigma_X/[\sqrt{N}E(\mathbf{X})]$ to be sufficiently small, which means choosing N sufficiently large. The main problem here is when $E(X) = 0$. In that case, we have to accept an absolute value of $\text{Var}(\langle \mathbf{X} \rangle)$ as being a good

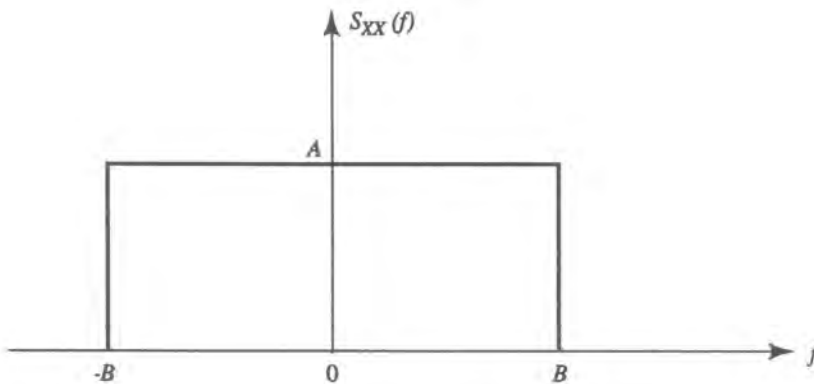


Figure 10.3. Power spectral density of a bandlimited lowpass process.

criterion, or perhaps measure $\sigma_X/\sqrt{N}$ with respect to a nonnegative parameter such as σ_X itself.

10.2.4. Mixture (Signal Plus Noise) Processes

In simulation we are almost always interested in the mixture of two or more processes, typically signal plus noise, signal plus interference, etc. Thus, if

$$X(t) = \sum_{i=1}^M V_i(t) \quad (10.2.11)$$

and if the reasonable assumption is made that the $V_i(t)$ are mutually independent, then (10.2.6) becomes

$$\text{Var}(\langle \mathbf{X} \rangle) = \sum_{i=1}^M \frac{\sigma_{V_i}^2}{N} + \frac{2}{N^2} \sum_{i=1}^M \sum_{k=1}^{N-1} (N-k) C_{VV}^{(i)}(kT_s) \quad (10.2.12)$$

where $C_{VV}^{(i)}$ is the autocovariance of $V_i(t)$. Thus, rather similar results as before can be expected, but now we have to pay attention to the properties of the component processes. Of course, we can still think in terms of a single covariance function which is the sum of the C_{VV} , and in that sense the formulation (10.2.6) is unchanged.

10.2.5. Confidence Interval Conditioned on the Signal

Let us assume that

$$\mathbf{X} = \mathbf{S} + \mathbf{N} \quad (10.2.13)$$

where $N(t)$ is a zero-mean Gaussian random process and $S(t)$ the signal process. In this case we can obtain a confidence interval for $\langle \mathbf{X} \rangle$ by conditioning upon the signal. Conditioning upon the signal means, in the conceptualization of the ensemble, that the same $s(t)$ is always generated, while the natural range of realizations of $N(t)$ is permitted.

Let $\mathbf{s} = (s_1, s_2, \dots, s_M)$ represent the vector of measured signal values. Then the conditional sample mean is

$$Z \triangleq \langle \mathbf{X} | \mathbf{s} \rangle = \frac{1}{M} \sum_{k=1}^M s_k + \frac{1}{M} \sum_{k=1}^M N_k \quad (10.2.14a)$$

$$= \langle \mathbf{s} \rangle + \frac{1}{M} \sum_{k=1}^M N_k \quad (10.2.14b)$$

Since the N_k are assumed normal, the summation in (10.2.14b) is also normal with zero mean, and variance given by

$$\sigma_Z^2 = \frac{\sigma_N^2}{M} + \frac{2}{M^2} \sum_{k=1}^{M-1} (M-k) C_{NN}(kT_s) \quad (10.2.15)$$

Thus, Z is normal with pdf

$$f_Z(z) = \frac{1}{(2\pi)^{1/2} \sigma_Z} \exp \left[-\frac{(z - \langle \mathbf{s} \rangle)^2}{2\sigma_Z^2} \right] \quad (10.2.16)$$

Hence, a confidence interval for $\langle \mathbf{s} \rangle$ can be constructed from

$$P[Z - d_1 \sigma_Z \leq \langle \mathbf{s} \rangle \leq Z + d_1 \sigma_Z] = 1 - \alpha \quad (10.2.17)$$

by using for Z the observed value z , and d_1 is given by the implicit relation

$$\frac{1}{(2\pi)^{1/2}} \int_{-d_1}^{d_1} e^{-u^2/2} du = 1 - \alpha \quad (10.2.18)$$

Note that the interval has been chosen to be symmetrical because in this case it produces the shortest interval. The *length* of the interval is $2d_1 \sigma_Z$. It is desired that this length tend to zero as N becomes large, and generally it is realistic to assume that the requisite properties hold.

■ *Remark.* In practice, one often uses *test signals* to probe a system and it is useful to think of such a signal as the one that the experiment is conditioned upon. For a digital system, the test signal is typically a PN sequence, while for an analog system it is often a sine wave, or several sine waves, or perhaps a square wave. All of these signals are, of course, periodic, so that if the observation interval is a multiple of a period and if T_s is chosen suitably, the resulting $\langle \mathbf{s} \rangle$ will be (almost) exactly the true average for that signal. Whether the test signal is representative of the ensemble, or even a member of the ensemble, is another question. Indeed, actual systems are often tested and specified in terms of one or a few test signal(s), and as far as the system engineer is concerned, the test-signal “ensemble” may be the one of real interest; it is presumed that the connection between a certain level of performance of the test signal and that of the ensemble has been established. ■

■ *Summary of the Simulation Procedure.* The procedure for obtaining the average of a waveform is quite straightforward (see Figure 10.2).

1. Collect samples $\{X(kT_s)\} = \{X_k\}$, $k = 1, 2, \dots, N$
2. Sum the samples and divide by N , as in (10.2.1). ■

10.3. Estimating the Average Power (Mean-Square Value) of a Waveform

The average power of a waveform is an important attribute, whether in analog or digital transmission. The waveform of interest will normally consist of signal and noise. From the link performance standpoint the signal-to-noise ratio (SNR) is the quantity of interest. We shall look into estimating SNR in the next section. Here, we focus on the total average power. The average power per se is generally not a performance barometer. Instead, like the average level, the average power is a parameter of interest at a more detailed level of system design. For example, it is typical to specify the input drive level to an amplifier, hence one must be able to verify that the requisite operating point is indeed achieved.

A problem that is distinct from, but closely related to, the estimation of the average power is the estimation of the energy over a specified interval T_0 . This problem is of interest

in digital communications. The average power P_{av} is the average of the finite-time average power $P_k(T_0)$, which is the energy over the k th interval divided by the duration T_0 . If the energy in the k th interval $\xi_k(T_0)$ is intended to be independent of k , as in a constant-envelope signal, there is an obvious connection between average power and energy. In actuality, the energy will fluctuate from one interval to the next because of envelope variation due to filtering and because of the presence of noise. (Of course, the energy may vary intentionally in different intervals, as in PAM or QAM constellations not on a circle.) Nevertheless, the average energy may still be of interest in order, say, to calibrate the E_b/N_0 axis of a BER curve.

In some systems the energy $\xi_k(T_0)$ is *inherently* random. For example, in a direct detection optical receiver, the output of an avalanche photodetector in a given interval is a random variable even with fixed input intensity. In such a system, while the average output may still be useful information, what is of real interest is the distribution of the output given a type of symbol. In such cases, the problem is one of estimating a distribution, not a parameter. This is addressed in Section 10.4.

10.3.1. Form of the Estimator for Average Power

As before, the waveform $X(t)$ is sampled at intervals T_s to yield $\mathbf{X}$. We choose the average power estimator to be of the following form:

$$P_N(\mathbf{X}) = \frac{1}{N} \sum_{k=1}^N X_k^2 \quad (10.3.1)$$

We can use the results of the preceding section if we define the auxiliary variable $Y(t) = X^2(t)$, in which case the problem becomes formally one of estimating the mean of $Y(t)$, namely,

$$\langle \mathbf{Y} \rangle = \frac{1}{N} \sum_{k=1}^N Y_k \quad (10.3.2)$$

Refer to Figure 10.4, which shows an illustrative segment of a waveform and its samples.

10.3.2. Expected Value of the Estimator

It is clear that

$$E\langle \mathbf{Y} \rangle = E(Y) \quad (10.3.3a)$$

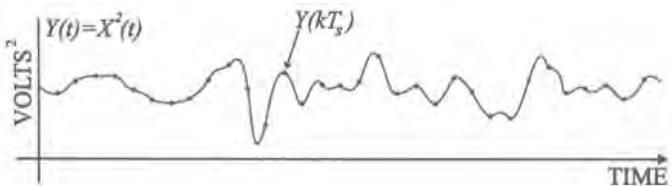


Figure 10.4. A segment of the square of a waveform $\{Y(kT_s)\}$, where $Y(t) = X^2(t)$; its average is $\bar{Y} = (1/N) \sum_{k=1}^N Y(kT_s)$.

for a stationary process. Hence

$$E(P_N(\mathbf{X})) = E(X^2) \quad (10.3.3b)$$

so that the expected value of the finite-sample estimator of average power is equal to the mean-square value of the process, independent of N .

In the continuous-time case,

$$\begin{aligned} E(\langle \mathbf{Y} \rangle) &= \frac{1}{T} \int_0^T E(Y(t)) dt \\ &= E(Y) = E(X^2) \end{aligned}$$

so that (if the underlying process is continuous) the expected value is properly given by the sampled version of the process. Hence the temporal average is an unbiased estimator of the related ensemble parameter.

10.3.3. Variance of the Estimator

We can write, directly from (10.2.6),

$$\text{Var}(\langle \mathbf{Y} \rangle) = \frac{\sigma_Y^2}{N} + \frac{2}{N^2} \sum_{k=1}^{N-1} (N-k) C_{YY}(kT_s) \quad (10.3.4)$$

so that all the properties of $\langle \mathbf{X} \rangle$ also apply to $\langle \mathbf{Y} \rangle$ if C_{YY} has the same characteristics as C_{XX} . Similarly, if we let $T_s \rightarrow 0$, $NT_s \rightarrow T$, then from (10.2.8)

$$\text{Var}(\langle \mathbf{Y} \rangle) = \frac{2}{T} \int_0^T \left(1 - \frac{\tau}{T}\right) C_{YY}(\tau) d\tau \quad (10.3.5)$$

The estimator $P_N(X)$ will converge in mean square to P_{av} for almost every sample path $x(t)$ if $\text{Var}(\langle \mathbf{Y} \rangle) \rightarrow 0$ as $N \rightarrow \infty$. This will happen in the case of (10.3.5) if⁽¹⁾

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T C_{YY}(\tau) d\tau = 0 \quad (10.3.6)$$

or, in the case of (10.3.4), if the discrete equivalent of (10.3.6) holds, namely,

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=1}^N C_{YY}(kT_s) = 0 \quad (10.3.7)$$

Of course, the constraints on C_{YY} also imply constraints on the related parameters of $X(t)$. Specifically, since $Y = X^2$, we have

$$C_{YY}(\tau) = E(X^2(t)X^2(t+\tau)) - R_{XX}^2(0) \quad (10.3.8)$$

where $R_{XX}(\tau)$ is the autocorrelation function of $X(t)$. Note that $R_{XX}(0) = E(X^2)$, the very quantity we are trying to estimate. Nevertheless, one could, in principle, evaluate C_{YY} in terms of the known properties of $X(t)$, and hence compute variance as a function of N . In practice, (10.3.8) is not simple to evaluate, and requires knowledge of the four-dimensional joint density of the process $X(t)$. If $X(t)$ is a Gaussian random process, one can show that

$$C_{YY}(\tau) = 2C_{XX}^2(\tau) + 4E^2(X)C_{XX}(\tau) \quad (10.3.9)$$

Notice that $X(t)$ and $X(t + \tau)$ become independent as $\tau \rightarrow \infty$ for most processes of interest, hence both (10.3.8) and (10.3.9) tend to zero as $\tau \rightarrow \infty$, which for most processes of interest is a necessary condition for (10.3.6) to hold. [Even for a singular process like $X(t) = a$, $-\infty < t < \infty$, for which $X(t)$ and $X(t + \tau)$ do not become independent as $\tau \rightarrow \infty$, $C_{YY}(\infty) \rightarrow 0$. In fact, for this example, $C_{YY}(\tau) = 0$.] Of course, if $C_{XX}(\tau)$ is identically zero for some $\tau > \tau_0$, then (10.3.6) is guaranteed.

■ *Example 10.3.1.* Let us suppose that NT_s and C_{YY} are such that the first term in (10.3.4) dominates. Then, the relative error is given by

$$\begin{aligned} \epsilon &= \frac{[\text{Var}(\mathbf{Y})]^{1/2}}{E(Y)} \approx \frac{\sigma_Y}{\sqrt{NE(Y)}} \\ &= \left\{ \frac{2}{N} \left[1 - \frac{E^4(X)}{R_{XX}^2(0)} \right] \right\}^{1/2} \end{aligned} \quad (10.3.10)$$

To get a numerical sense of (10.3.10), suppose $X(t) = A + N(t)$ with $N(t)$ a zero-mean process with variance σ^2 . Then, $E(X) = A$, and $E(X^2) = R_{XX}(0) = A^2 + \sigma^2$. Substituting in (10.3.10) yields

$$\epsilon = \left\{ \frac{2}{N} \left[1 - \frac{1}{1 + 2(\sigma/A)^2 + (\sigma/A)^4} \right] \right\}^{1/2}$$

which is plotted in Figure 10.5. If A is only moderately larger than the fluctuating component, e.g., $A/\sigma = 10$, it can be seen that comparatively few observations will yield a reliable estimate. In the reverse situation, e.g., $A/\sigma = 1$, a fairly large number of observations may have to be made if the fractional error is required to be small, less than 1%, say. ■

■ *Remark on Confidence Interval.* We previously mentioned the difficulty of obtaining confidence intervals. This is even more so in the present instance. Even if we condition $\mathbf{P}_N$ on the signal, we can show that the conditional variable is a sum of independent but not identically distributed noncentral gamma random variables. The labor involved with evaluating this distribution is clearly disproportionate with the (relatively low) accuracy that we generally need for an average power estimate. Hence, the variance is quite sufficient for our purposes. ■

■ *Summary of the Simulation Procedure.* The simulation procedure for estimating the average power should be quite clear from the preceding, namely:

1. Collect samples $\{X_k\}$, $k = 1, 2, \dots, N$
2. Square each sample $\{Y_k = X_k^2\}$, $k = 1, 2, \dots, N$ (see Figure 10.4).
3. Sum the samples Y_k and divide by N , as in (10.3.1). ■

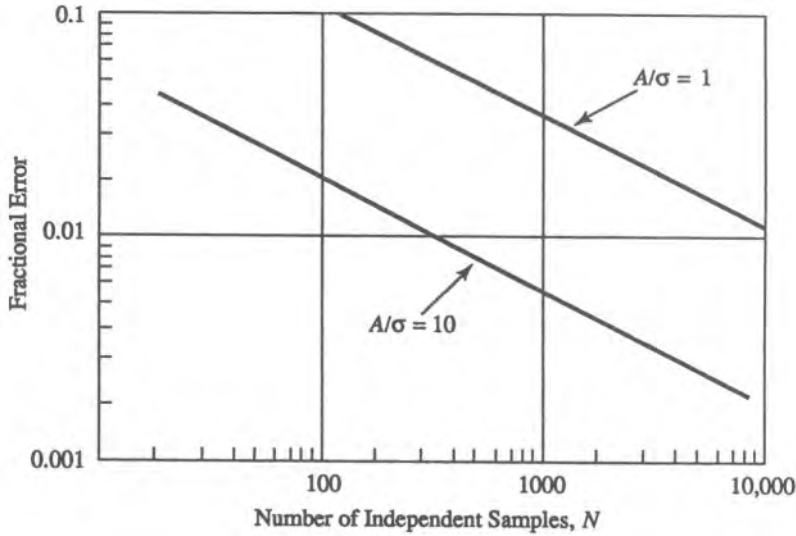


Figure 10.5. Relationship between the fractional error of the average power estimate and the number of observations for Example 10.3.1.

10.4. Estimating the Probability Density or Distribution Function of the Amplitude of a Waveform

In communications applications the probability density function (pdf) or the cumulative distribution function (CDF) of the values of a waveform is usually not, in itself, a measure of performance. Nevertheless, these functions tell us about the statistical nature of the process under observation and thus help in understanding the system. In digital communications, partial knowledge of these functions is necessary for the determination of error probability. Also of interest in characterizing some communications links is the temporal distribution of errors, for example, the probability of a given number of errors in a block of specified size. Empirically obtained distributions can impart quantitative or qualitative information. One can obtain insight by observing that a distribution “appears” to be of such and such a form. To obtain this information, only a procedure is needed. However, to make quantitative inferences we must be able to make quantitative statements about the accuracy of the estimator resulting from that procedure.

In this section we give only a short treatment of this subject, focusing on the histogram method, which is particularly simple to implement in software. More extensive treatments can be found, for example, in Refs. 5 and 11–14.

10.4.1. The Empirical Distribution

The *empirical distribution* $\hat{F}_X(\mathbf{x})$ is the classical estimator for an underlying distribution $F_X(\mathbf{x})$. (To simplify the notation in this section we shall drop the subscript X on F .) Let

$\mathbf{X} = (X_1, X_2, \dots, X_N)$ be an observation of the random vector $\mathbf{X}$ written in *increasing order* ($X_k \leq X_{k+1}$). Then, we define

$$\hat{F}(x) = \begin{cases} 0, & x < X_1 \\ k/N, & X_k \leq x < X_{k+1}, \quad k = 1, 2, \dots, N-1 \\ 1, & x \geq X_N \end{cases}$$

$\hat{F}(x)$ is a binomially distributed random variable with the probability of “success” $p = F(x)$. The distribution of $\hat{F}(x)$ is therefore

$$P[\hat{F}(x) = m/N] = \frac{N!}{m!(N-m)!} [F(x)]^m [1 - F(x)]^{N-m} \quad (10.4.1)$$

This emphasizes that $\hat{F}(x)$ is a random variable, and it is of interest to quantify how close $\hat{F}(x)$ is to $F(x)$. We do not discuss this in detail here, but simply mention the following important theorems.⁽⁵⁾

We expect that as $N \rightarrow \infty$, $\hat{F}(x)$ approaches $F(x)$ in some sense. In fact, Glivenko’s theorem⁽⁵⁾ (also known as the Glivenko–Cantelli theorem) states that

$$P\left(\lim_{N \rightarrow \infty} D_N = 0\right) = 1$$

where

$$D_N = \sup_{-\infty < x < \infty} |\hat{F}_N(x) - F(x)|$$

and $\hat{F}_N(x)$ is exactly the same as $\hat{F}(x)$ with an explicit subscript N indicating the number of samples. The theorems of Kolmogorov and Smirnov⁽⁵⁾ give explicit formulas for evaluating the limit of $\sqrt{D_N}$ as $N \rightarrow \infty$, and a very interesting formulation for the same thing has been given in terms of a “Brownian bridge”.⁽¹⁴⁾ The theorem of Renyi⁽⁵⁾ also gives explicit formulas for evaluating the more useful statistic $[\hat{F}_N(x) - F(x)]/F(x)$.

The empirical distribution when plotted has a familiar staircase-type appearance, and is easy to produce within a simulation (see Figure 10.6). Note that in general the location of the risers of the staircase will not be equally spaced. However, we can enforce equispaced transitions by grouping the data in “cells” defined accordingly. (Of course this will come about naturally if we define the cells sufficiently finely.) This grouping results in a more immediately interpretable visual display and is typically implemented as discussed below.

10.4.2. The Empirical Probability Density Function—Histogram

One is often interested in obtaining the density function directly, rather than the CDF. The “natural” empirical pdf may be thought to be the derivative of $\hat{F}(x)$. This would result in a set of delta functions located at the risers of $\hat{F}(x)$. However, an essentially equivalent but visually more pleasing alternative is the *histogram*. The histogram is basically a bar chart, as illustrated in Figure 10.7, which shows the actual or relative frequency of events observed within equispaced intervals. The histogram is an estimator of the first-order pdf. Similar ideas

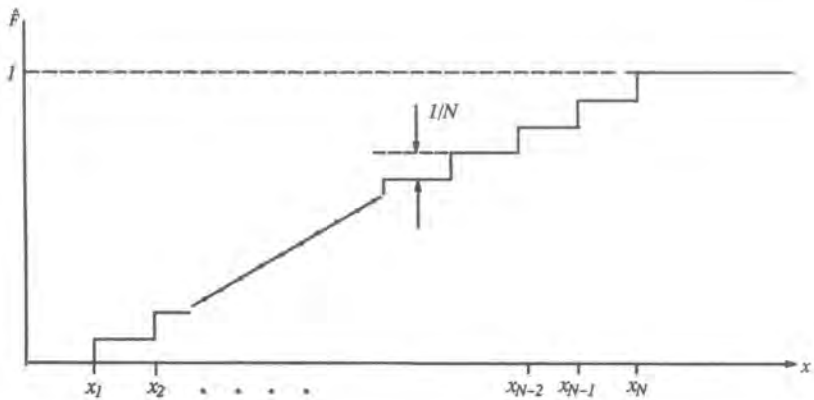


Figure 10.6. Illustration of the empirical distribution.

are used to obtain estimates of the multidimensional pdf. Sometimes we may be interested in determining if it is reasonable to suppose that the underlying population has a pdf of a given form, like a normal distribution. This is a test-of-hypothesis problem which has been briefly addressed in Chapter 6 as a “goodness-of-fit” test.

10.4.2.1. Form of the Estimator

Let the i th interval be centered on x_{c_i} , and let W be the width of the interval. Hence the i th interval will be denoted

$$\Delta_i = \left\{ x: x_{c_i} - \frac{W}{2} < x \leq x_{c_i} + \frac{W}{2} \right\}$$

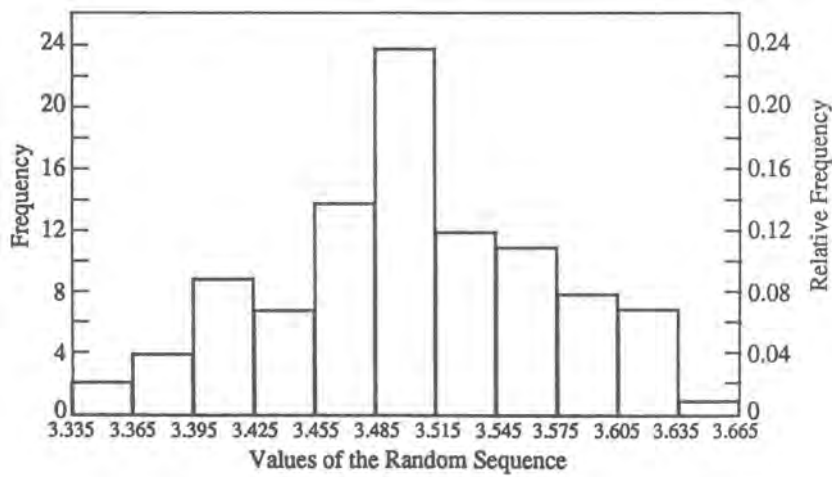


Figure 10.7. Construction of a histogram (from Ref. 11, © John Wiley & Sons, 1998, reproduced by permission).

so that the (normalized) histogram $\hat{f}(\mathbf{x})$, namely, the estimator of $f(\mathbf{x})$, is defined by

$$\hat{f}_i(\mathbf{x}) = \frac{N_i}{NW}, \quad \mathbf{x} \in \Delta_i \quad (10.4.2)$$

and N_i is the number of observations (out of N) for which $\mathbf{x} \in \Delta_i$. It is clear that $\frac{1}{W} \sum_i \hat{f}_i(\mathbf{x}) = 1$, as it should.

10.4.2.2. Expectation of the Estimator

The expected value of $\hat{f}_i(\mathbf{x})$ is given by

$$E(\hat{f}_i(\mathbf{x})) = \frac{1}{NW} E(N_i), \quad \mathbf{x} \in \Delta_i$$

From the discussion in Section 10.4.1, N_i is a binomial random variable with probability of “success” p_i given by

$$p_i = \int_{\mathbf{x} \in \Delta_i} f(\mathbf{x}) d\mathbf{x}$$

so that

$$E(N_i) = Np_i$$

whence

$$E(\hat{f}_i(\mathbf{x})) = p_i/W \quad (10.4.3)$$

In general, (10.4.3) will not equal $f(\mathbf{x})$ for all $\mathbf{x} \in \Delta_i$. This is obvious because $\hat{f}_i(\mathbf{x})$ is constant within the cell, whereas for most phenomena of interest this is not so for $f(\mathbf{x})$. Thus, $\hat{f}(\mathbf{x})$ is a biased estimator. A quantitative estimate of the bias can be obtained by approximating $f(\mathbf{x})$ by the first three terms of a Taylor series expansion about the center of the cell, namely,

$$f(\mathbf{x}) \approx f(\mathbf{x}_{c_i}) + f'(\mathbf{x}_{c_i})(\mathbf{x} - \mathbf{x}_{c_i}) + \frac{1}{2}f''(\mathbf{x}_{c_i})(\mathbf{x} - \mathbf{x}_{c_i})^2, \quad \mathbf{x} \in \Delta_i$$

Hence, we have

$$E(\hat{f}_i(\mathbf{x})) \approx \frac{1}{W} \int_{\mathbf{x} \in \Delta_i} [f(\mathbf{x}_{c_i}) + f'(\mathbf{x}_{c_i})(\mathbf{x} - \mathbf{x}_{c_i}) + \frac{1}{2}f''(\mathbf{x}_{c_i})(\mathbf{x} - \mathbf{x}_{c_i})^2] d\mathbf{x}$$

or

$$E(\hat{f}_i(\mathbf{x})) \approx f(\mathbf{x}_{c_i}) + f''(\mathbf{x}_{c_i}) \frac{W^2}{24}, \quad \mathbf{x} \in \Delta_i$$

Thus, the bias of $\hat{f}_i(x)$ as an estimator for $f(x_{c_i})$ is approximately

$$b(\hat{f}_i(x)) = E(\hat{f}_i(x) - f(x_{c_i})) \approx f''(x_{c_i}) \frac{W^2}{24} \quad (10.4.4)$$

which evidently increases with cell size width W .

■ *Example 10.4.1.* A brief example is helpful in visualizing the magnitude of the bias. Suppose $f(x)$ is Gaussian,

$$f(x) = \frac{1}{(2\pi)^{1/2}\sigma} e^{-x^2/2\sigma^2}$$

Then, we have

$$f''(x) = f(x) \left[\left(\frac{x}{\sigma^2} \right)^2 - \frac{1}{\sigma^2} \right]$$

A more meaningful indicator of bias than (10.4.4) itself is the bias relative to the true value, or the normalized bias

$$\epsilon_b = b(\hat{f}_i(x))/f(x_{c_i})$$

Hence, we have

$$\epsilon_b = \left\{ \left(\frac{x_{c_i}}{\sigma^2} \right)^2 - \frac{1}{\sigma^2} \right\} \frac{W^2}{24}$$

Let $x_{c_i} = iW$ and suppose we set $W = \alpha\sigma$. Then, we have

$$\epsilon_b = \frac{\alpha^2}{24} [(i\alpha)^2 - 1]$$

This allows one to size $W (= \alpha\sigma)$ for a desired ϵ_b and a desired location (i) in the tail of the density. ■

10.4.2.3. Variance of the Estimator

Since N_i is a binomial random variable, its variance is $Np_i(1 - p_i)$, so that the variance of $\hat{f}_i(x)$ is given by

$$\text{Var}[\hat{f}_i(x)] = \left(\frac{1}{NW} \right)^2 Np_i(1 - p_i)$$

Approximating p_i by

$$p_i = \int_{x \in \Delta_i} f(x) dx \approx Wf(x_{c_i})$$

results in

$$\text{Var}[\hat{f}_i(x)] = \frac{1}{NW} f(x_{c_i})[1 - Wf(x_{c_i})]$$

The normalized standard error ϵ_r for estimating $f(x_{c_i})$ is

$$\epsilon_r = \frac{\{\text{Var}[\hat{f}_i(x)]\}^{1/2}}{f(x_{c_i})} = \left[\frac{1 - Wf(x_{c_i})}{NWf(x_{c_i})} \right]^{1/2} \quad (10.4.5)$$

Notice that the effect of cell width is opposite to that on the bias. Here, for given x_{c_i} , as $W \rightarrow 0$ we see that $\epsilon_r \rightarrow [NWf(x_{c_i})]^{-1/2}$, hence the variance and standard error increase as W decreases. If W is fixed at a value that yields acceptable bias, then it is clear that in order to maintain ϵ_r at some given desirable value, it is necessary that N increase inversely with $f(x_{c_i})$. In other words, in order to maintain a fixed (relative) accuracy for $\hat{f}(x_{c_i})$ when $f(x_{c_i})$ is in the tails we must increase N so that $Nf(x_{c_i})$ remains constant. It may be seen that if we choose $W \propto 1/\sqrt{N}$, both the bias and standard error will tend to zero as $N \rightarrow \infty$.

■ *Remark.* In digital communications the performance criterion is error probability, which is given by the area under $f(x)$ for x in the “tail” region, say $x > x_T$, the “threshold.” This is equivalent to the value of the CDF $F(x_T)$ or $1 - F(x_T)$ depending upon the symbol detected. (This is discussed in detail in the next chapter.) It seems sometimes to be assumed that it is easier to estimate $f(x)$ and integrate the estimator $\hat{f}(x)$ than directly to estimate $F(x_T)$. This misimpression is probably due to the fact that for a relatively large N the mode of the density can be estimated quite accurately. This follows from the expression for ϵ_r in (10.4.5). This allows a good analytical fit of pdf around the mode, which perhaps leads to a false sense of security about extrapolating the form near the mode to the tails. The fact is, as we have seen, that the accuracy in the tails decreases for a given N , and is proportional to $N^{-1/2}$. This is precisely the dependence on N of the accuracy for estimating a probability directly. This is hardly surprising since a probability is the integral of a density. In fact, a probability estimated by integrating the empirical density function is nothing more than the empirical distribution $\hat{F}(x)$ evaluated at the appropriate value of x . (Actually, this last statement is approximate, but sufficiently good for moderately small values of the cell width W .) Now, the distribution of $\hat{F}(x)$ is given by (10.4), which has expected value $F(x)$ and variance $F(x)[1 - F(x)]/N$. Thus, for given x and N , we should expect the same accuracy whether estimating $\hat{F}$ directly or by integrating $\hat{f}$. ■

■ *Summary of the Simulation Procedure (Histogram).* The construction of a histogram can be summarized as follows:

1. Collect N sampled values of a waveform.
2. Choose an interval width W and a reference interval center x_{c_0} .
3. The i th interval is centered at $x_{c_0} + iW = x_{c_i}$.
4. Count the number of samples in $x_{c_i} \pm W/2$; this is N_i .
5. Calculate the estimate $\hat{f}_i = N_i/NW$, all i . ■

10.5. Estimating the Power Spectral Density (PSD) of a Process

The power spectral density (PSD) embodies the frequency-domain properties of a process. Probably the single most important frequency domain attribute is “bandwidth,”

which is derivable from the PSD. (We note that bandwidth is not uniquely defined^(15,16). Traditionally, the PSD of a particular modulation method has been obtained by analytical means, and a great deal of attention has been paid to this problem; see, e.g., Refs. 17–26. However, derivation of the PSD in an analytical or computable form can pose significant difficulties when practical equipment impairments or nonlinearities are present. Thus, it is often more straightforward to determine the PSD “experimentally,” using the simulated system. Such a measurement implies a statistical approach to determine properties of the PSD estimator.

There is a rich body of literature on spectral estimation (see, e.g., Refs. 11 and 27–31), much of which deals with “model-based” techniques that are designed to cope with “system identification” issues. One such issue is concerned with the resolution of close “spikes.” Another issue is that the signal may be observable for an interval so short (corresponding to less than a few hundred samples) that “conventional” techniques could not produce a meaningful estimate of PSD. Neither of the above issues presents a problem in simulation, where we typically have available thousands of samples. For simulation application, therefore, it is sufficient to take recourse only to the straightforwardly implemented “classical” techniques.

10.5.1. Form of the Estimator

There are two “classical” techniques of spectral estimation, based on alternative definitions of power spectral density. These two definitions are equivalent for a wide-sense stationary process.

10.5.1.1. The Correlogram or Indirect Method

The first definition of the PSD $S_{XX}(f)$ of the process $X(t)$ is as the Fourier transform of the autocorrelation function $R_{XX}(\tau)$. For a continuous function, the classical definition is^(17,32)

$$S_{XX}(f) = \int_{-\infty}^{\infty} R_{XX}(\tau) \exp(-j2\pi f\tau) d\tau, \quad -\infty < f < \infty \quad (10.5.1)$$

In simulation we must deal with samples of $X(t)$. The discrete-time version of (10.5.1) is given by⁽³⁰⁾

$$P_{XX}(f) = T_s \sum_{k=-\infty}^{\infty} R_{XX}(k) \exp(-j2\pi f k T_s), \quad |f| \leq \frac{1}{2}f_s \quad (10.5.2)$$

where $R_{XX}(k)$ is the discrete-time autocorrelation function (or sequence)

$$\begin{aligned} R_{XX}(k) &\triangleq R_{XX}(kT_s) = E[X(nT_s + kT_s)X(nT_s)] \\ &\triangleq E[X(n+k)X(n)] \end{aligned} \quad (10.5.3)$$

(In much of the literature dealing with discrete-time systems and digital signal processing, the time element is not explicit. But since we are dealing here with approximation of continuous systems, the time element is taken into account with the factor T_s , which is necessary in order

to keep the units correct.) Notice that we have changed notation from $S_{XX}(f)$ for the continuous PSD to $P_{XX}(f)$ for the discrete-time PSD. It is well known that sampling replicates the spectrum at frequency intervals to $1/T_s$ (see Chapters 3 and 6). We assume that T_s has been chosen to control the *aliasing error* to an acceptable level, and take it as our objective to determine $P_{XX}(f)$ as best we can.

In practice we have only a finite number of observations $\{X(n)\}$. A natural estimator of (10.5.2) is to replace the true autocorrelation sequence by an estimator, or *correlogram*, $\hat{R}_{XX}(k)$. Hence, the PSD estimator can be written as

$$\hat{P}_{XX}(f) = T_s \sum_{k=-L}^L \hat{R}_{XX}(k) \exp(-j2\pi f k T_s) \quad (10.5.4)$$

where the index k is referred to as the *lag*, and the maximum lag used or available is L . It should be noted that $\hat{P}_{XX}$ is periodic in f and that we are interested in the first zone, $|f| \leq 1/2T_s$. To evaluate (10.5.4), we first need the autocorrelation estimate. There are two basic approaches to this, the one most commonly used being

$$\hat{R}_{XX}(k) = \frac{1}{N} \sum_{n=0}^{N-k-1} X(n+k)X^*(n) \quad (10.5.5)$$

where $k = 0, 1, \dots, N-1$ is the number of observed samples, and X^* is the complex conjugate if X is complex. [The form (10.5.5) is customary in this context. However, if one wanted to estimate the autocorrelation function itself, (10.5.5) yields a biased estimate. To produce an unbiased estimate, replace $1/N$ in front of (10.5.5) by $1/(N-k-1)$.] Because $\hat{R}_{XX}(k) = \hat{R}_{XX}^*(-k)$, one can obtain $\hat{R}_{XX}$ for negative lags. As k increases, fewer samples are included in the summation. Hence, the estimator reliability deteriorates as k increases. For this reason, it has been suggested⁽³¹⁾ that the maximum lag L in (10.5.4) be limited to about $N/10$. The $\hat{P}_{XX}(f)$ is given for any f in the interval $|f| \leq 1/2T_s$. In practice, we calculate $\hat{P}_{XX}(f)$ only for a discrete set $f_i = i/KT_s$, $0 \leq i \leq K$, in order to take advantage of the FFT algorithm (see Section 10.5.5).

10.5.1.2. The Periodogram or Direct Method

An alternative definition for the PSD is the following⁽³²⁾:

$$S_{XX}(f) = \lim_{T \rightarrow \infty} E[\hat{S}_{XX}(f, T)] \quad (10.5.6)$$

where

$$\hat{S}_{XX}(f, T) = \frac{|X_T(f)|^2}{T} \quad (10.5.7)$$

and

$$X_T(f) = \int_0^T X(t) \exp(-j2\pi f t) dt \quad (10.5.8)$$

is the finite-time Fourier transform of a sample function of the process. When the process is wide-sense stationary, (10.5.6) is equivalent to (10.5.1). A “natural” estimate of $S_{xx}(f)$ is suggested by (10.5.6) if we simply omit the limiting and expectation operations. That is, $\hat{S}_{xx}(f, T)$ is an estimate for $S_{xx}(f)$. The $\hat{S}_{xx}(f, T)$ is referred to as the *periodogram* or the *sample spectrum*. The discrete-time version of $\hat{S}_{xx}(f, T)$ is denoted $\tilde{P}_{xx}(f)$, and is given by

$$\tilde{P}_{xx}(f) = \frac{1}{NT_s} |X_N(f)|^2 \quad (10.5.9)$$

where

$$X_N(f) = T_s \sum_{n=0}^{N-1} X(n) \exp(-j2\pi fnT_s) \quad (10.5.10)$$

is the discrete Fourier transform of the observed data sequence. (The tilde here stands for estimator, as the caret usually does. We use a different symbol here to distinguish the two estimators.) It can be seen that X_N , hence $\tilde{P}_{xx}$, is periodic in f , but we are interested only in the fundamental period. Again, to speedily compute $\tilde{P}_{xx}$ one normally relies on the FFT algorithm to determine discretely spaced values $\tilde{P}_{xx}(f_k)$, $f_k = k/KT_s$, $k = 0, 1, \dots, K-1$. If $K > N-1$, zero padding is implied.

■ *Remark on the Relationship between Direct and Indirect Methods.* If, in the indirect method the maximum lag L is set to $N-1$ the largest possible value, the estimators $\hat{P}_{xx}(f)$ and $\tilde{P}_{xx}(f)$ are identical.⁽³⁰⁾ While the same observed sequence forms the basis of both procedures, there is the possibility of further manipulation of either form of the estimator which may or may not have a ready interpretation in the other form. Although $\hat{P}_{xx}(f)$ and $\tilde{P}_{xx}(f)$ contain the essential ideas of PSD estimation, as estimators they suffer from certain drawbacks, which can be remedied to some extent by auxiliary techniques referred to as windowing and averaging (smoothing). These techniques are customarily applied in practice and they are discussed in the next section. ■

10.5.2. Modified Form of the Estimator: Windowing and Averaging

The term *window* implies a function $w(t)$ that “lets in” or colors certain portions of another function $X(t)$. Any finite observation is implicitly a windowed sample function of a process. In that case the (time) window is inescapable and induces distortion which can be mitigated to some extent by intentionally superimposing another window. There are two basic types of windows: *data windows*, denoted $d(n)$, that are used to modify directly (multiply) the observed data sequence $X(n)$; and *lag windows*, denoted $\lambda(k)$, that are used to modify directly the autocorrelation sequence $\hat{R}_{xx}(k)$. Data windows are also referred to as tapering functions. The transform of a data window is called a *frequency window* and denoted $W(f)$, and the transform of a lag window is called a *spectral window* denoted $\Omega(f)$.

If we apply a data window $d(n)$ to $X(n)$ in (10.5.10), we have

$$X_N(f; d) = T_s \sum_{n=0}^{N-1} X(n)d(n) \exp(-j2\pi fnT_s) \quad (10.5.11)$$

$$= X_N(f) \circledast W(f) \quad (10.5.12)$$

where $\circledast$ represents circular convolution, and the presence of the window is made explicit in the argument of X_N . Therefore the PSD estimator is

$$\tilde{P}_{XX}(f; d) = \frac{1}{(NT_s)\xi} |X_N(f; d)|^2 \quad (10.5.13)$$

where

$$\xi = T_s \sum_{n=0}^{N-1} d^2(n) \quad (10.5.14)$$

is the discrete-time window energy. Division by ξ maintains the expected value of the sample average power unchanged; obviously, this operation can be avoided if $d(n)$ is prenormalized so that $\xi = 1$.

The effect of the window is to convolve the discrete-time Fourier transform of the data with the frequency window. This makes it somewhat difficult to visualize the net effect on $\tilde{P}_{XX}(f)$ from sample to sample. However, a more easily interpretable result is obtainable in an average sense. One can show that

$$E[\tilde{P}_{XX}(f; d)] = \sum_{k=-\infty}^{\infty} T_s \lambda(k) R_{XX}(k) \exp(-j2\pi f k T_s) \quad (10.5.15)$$

where $R_{XX}(k)$ is the true autocorrelation sequence, and

$$\lambda(k) = \frac{1}{N} \sum_{n=-\infty}^{\infty} d(n) d(n+k) \quad (10.5.16)$$

It can be seen that $\lambda(k)$ acts as a lag window. Hence, in an average sense, the data window is equivalent to a lag window which is the data window's autocorrelation function.

Equation (10.5.15) can be expressed as

$$E[\tilde{P}_{XX}(f; d)] = P_{XX}(f) \circledast \Omega(f) \quad (10.5.17)$$

where $\Omega(f)$ is the discrete Fourier transform of $\lambda(k)$. Note that $\lambda(k)R_{XX}(k)$ is the effective autocorrelation $R_{XX}(k; d)$ of the windowed sequence, and that $R_{XX}(0, d)$ is the average power of the process. Hence, this average power will be unaffected if we set $\lambda(0) = 1$, which we can do simply by dividing (10.5.17) by

$$\lambda(0) = (1/N) \sum d^2(n) = (1/NT_s)\xi$$

It can also be shown that

$$\Omega(f) = (1/NT_s) |W(f)|^2 \quad (10.5.18)$$

Therefore, the properly normalized version of (10.5.17) is

$$E[\tilde{P}_{XX}(f; d)] = P_{XX}(f) \circledast \Omega(f) / \lambda(0) = P_{XX}(f) \circledast |W(f)|^2 / \xi \quad (10.5.19)$$

Table 10.1. Definitions of Typical N -Point Discrete-Time Windows^a

Window name	Discrete-time function $d[n]$ or $\lambda[k]$	Frequency response $W(f)$ or $\Omega(f)$
Rectangular (uniform)	1	$D_N(f)$
Triangle (Bartlett)	$1 - t[n] $	$2[(N-1)T]^{-1} D_{(N-1)/2}^2(f)$
Squared cosine (Hann)	$0.50 + 0.50 \cos(2\pi t[n])$	$0.50 D_N(f) + 0.25 [D_N(f - (NT)^{-1}) + D_N(f + (NT)^{-1})]$
Raised cosine (Hamming)	$0.54 + 0.46 \cos(2\pi t[n])$	$0.54 D_N(f) + 0.23 [D_N(f - (NT)^{-1}) + D_N(f + (NT)^{-1})]$
Weighted cosines (Nuttall, $R = 3$)	$\sum_{r=0}^R \alpha_r \cos(2\pi r t[n])$	$\sum_{r=0}^R 0.5 \alpha_r [D_N(f - r(NT)^{-1}) + D_N(f + r(NT)^{-1})]$
Truncated Gaussian ($\alpha = 2.5$)	$p[-2(\alpha t[n])^2]$	$(\sqrt{2\pi}/2\alpha) \exp[-2(\pi T/\alpha)^2] * D_N(f)$

^aFrom S. L. Marple, Jr., Frequency-Time Signal Processing, course notes. © 1991, S. L. Marple, Jr., reproduced with permission.

An alternative approach to windowing is to define a window in the lag domain explicitly, rather than in the time domain. That is, obtain the autocorrelation sequence from unwindowed data, then apply a lag window $\lambda(k)$.

Much attention has been paid in the literature to windows.^(11,27-31) For illustration we show here a few of the more common ones. Table 10.1 catalogs the formulas for some data or lag windows and their transforms, while Figure 10.8 illustrates these forms in the “sample” and frequency domains. Maple⁽³¹⁾ also provides listings and a disk for programs that compute the PSD through both approaches. In Table 10.1, T is the sampling interval. The data window $d[n]$ is defined over the index range $0 \leq n \leq N-1$ and N may be even or odd. The lag window $\lambda[k]$ is defined over the index range $-(N-1)/2 \leq k \leq (N-1)/2$ and N is always odd. The function $t[n] = 2n/(N-1) - 1$ for $d[n]$ and $t[k] = 2k/(N-1)$ for $\lambda[k]$. The function $D_N(f) = T \exp[-j2\pi f T(N-1)] \sin(\pi f T N) / \sin(\pi f T)$ for $d[n]$ and $D_N(f) = T \sin(\pi f T N) / \sin(\pi f T)$ for $\lambda[k]$.

A window distorts the true spectrum. The customary measures of that distortion are bias and variability (in the usual statistical sense) and resolution. The latter is the ability to distinguish two relatively closely spaced spectral lines one from the other. The choice of window rests on balancing bias and resolution, as improving one will generally worsen the other. In simulation we can usually obtain a long enough record to satisfy good performance on both counts. We recall that windows are also beneficial in the filtering context, as discussed in Chapter 3.

We need to introduce one more operation that is necessary for the periodogram approach, and this is the smoothing or averaging of periodograms. The operation means simply to average $\tilde{P}_{xx}(f)$ or $\tilde{P}_{xx}(f; d)$ calculated for a number of data segments. In general, the individual segments do not have to be contiguous. In fact, there appear to be advantages in using overlapped segments. The periodogram obtained by averaging unwindowed nonoverlapping segments is called the Bartlett periodogram, $\tilde{P}_B(f)$, and the periodogram obtained by averaging windowed overlapped segments is called the Welch periodogram, $\tilde{P}_w(f)$.⁽³³⁾ Let $\tilde{P}_{xx.m}(f; d)$ be the periodogram obtained from the m th data segment; then,

$$\tilde{P}_w(f) = \frac{1}{M} \sum_{m=0}^{M-1} \tilde{P}_{xx.m}(f; d) \quad (10.5.20)$$

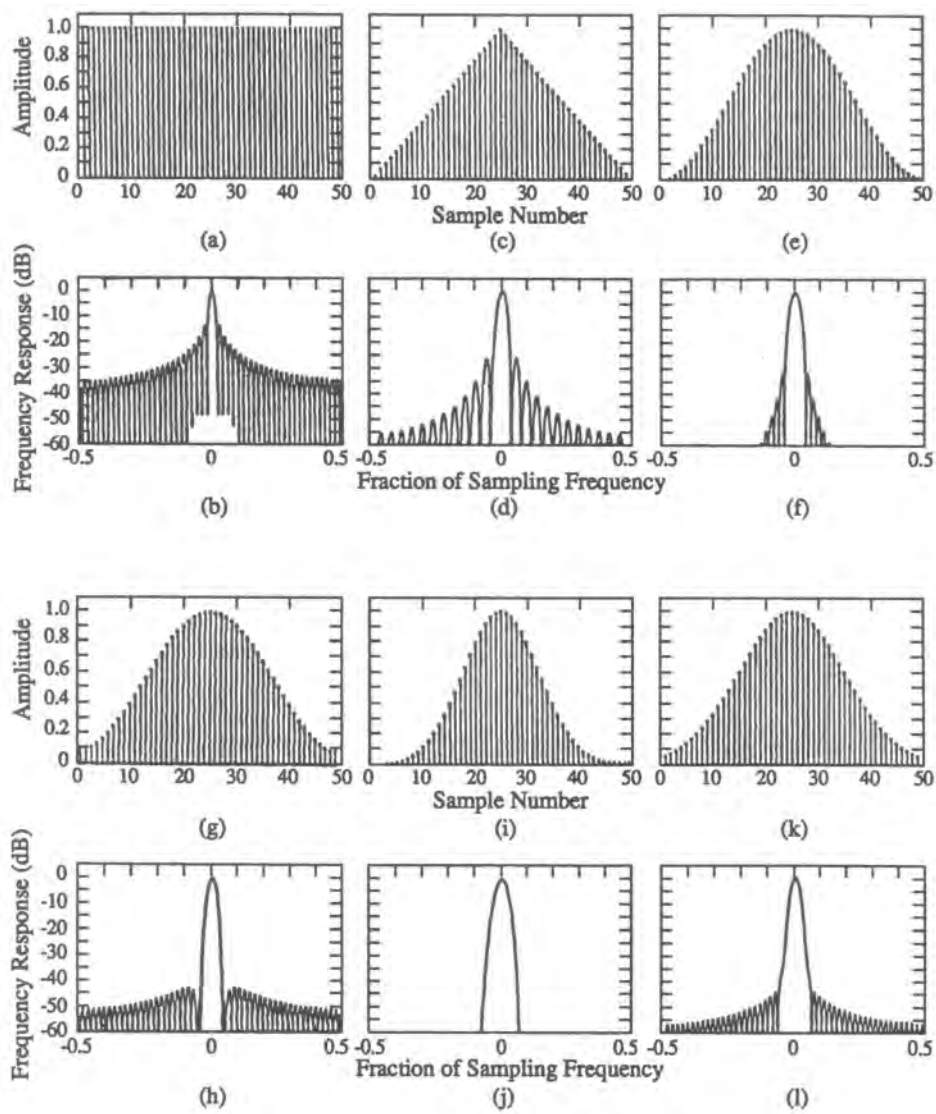


Figure 10.8. Typical discrete-time windows (upper) and log of their DTFT magnitude (lower), (a, b) Rectangular; (c, d), triangular; (e, f), Hann; (g, h), Hamming; (i, j), Nuttall; (k, l), truncated Gaussian. (From S. L. Marple, Jr., Frequency-Time Signal Processing, course notes, © 1991, S. L. Marple Jr., reproduced with permission.)

where M is the total number of segments, $M \geq N/J$, where J is the number of samples per segment. Another popular smoothing approach uses a running average in the frequency domain⁽¹¹⁾ (see also problem P10.9.)

10.5.3. Expected Value of the Estimator

According to (10.5.19), the expected value of $\tilde{P}_{xx}(f; \mathbf{d})$ is not $P_{xx}(f)$, hence the periodogram is a biased estimate. However, $\tilde{P}_{xx}(f; \mathbf{d})$ is asymptotically unbiased because for

any reasonable window, including the implicit rectangular window, $|W(f)|^2 \rightarrow \delta(f)$ as $N \rightarrow \infty$. Hence, it will generally be true that (with $\xi = 1$),

$$\lim_{N \rightarrow \infty} E[\tilde{P}_{XX}(f; d)] = P_{XX}(f) \quad (10.5.21)$$

The same statements are also true for the Bartlett or Welch periodograms, keeping in mind that the window is J samples rather than N samples long. Hence (10.5.21) is true for $\tilde{P}_B(f)$ or $\tilde{P}_w(f)$ if both $J, N \rightarrow \infty$.

It can also be shown, for the periodogram estimate with applied lag window $\lambda(k)$, that there is an effective lag window $(1 - |k|/N)\lambda(k) = \lambda_e(k)$ with transform $\Omega_e(f)$ such that

$$E[\hat{P}_{XX}(f; \lambda)] = P_{XX}(f) \otimes \Omega_e(f) \quad (10.5.22)$$

Even if no lag window is explicitly applied, there is an implied window, as before. Notice that if $L = N - 1$ and no applied window is used, then the triangular window $(1 - |k|/N)$ is precisely the convolution of the rectangular data window, as expected.

10.5.4. Variance of the Estimator

For an arbitrary process, the variance of the estimator is difficult to evaluate, but for a Gaussian process, one can show⁽³²⁾ that

$$E[\hat{S}_{XX}^2(f, T)] \geq 2\{E[S_{XX}(f, T)]\}^2 \quad (10.5.23)$$

By definition of variance we then have

$$\text{Var}[\hat{S}_{XX}(f, T)] \geq \{E[\hat{S}_{XX}(f, T)]\}^2 \quad (10.5.24)$$

which states that, *independently of* T , the standard deviation of the estimated spectrum is at least as large as the expected value of the estimated spectrum itself. Thus, the unmodified periodogram is not only a poor estimator, it cannot be improved by increasing the length of observation. Essentially, the reason is that the conventional periodogram ignores the expectation operation in (10.5.6) which makes it a consistent estimator. This is the motivation for the Bartlett and Welch periodograms, which form “pseudoensembles” over which to average. For discrete-time functions it can be shown (see, e.g., Ref. 31) that the variance of the periodogram of a (discrete) zero-mean white Gaussian process is

$$\begin{aligned} \text{Var}[\tilde{P}_{XX}(f)] &= P_{XX}^2(f) \left[1 + \left(\frac{\sin(2\pi f T_s N)}{N \sin(2\pi f T_s)} \right)^2 \right] \\ &\geq P_{XX}^2(f) \end{aligned} \quad (10.5.25)$$

which reconfirms the more general result (10.5.24).

For a Gaussian process, it is possible to find the distribution of the periodogram, from which we can calculate confidence intervals if desired. This is because, in either the continuous or discrete form, $X_T(f)$ or $X_N(f)$ is a linear combination of normal variables, hence is itself normal. The periodogram, which is proportional to $|X_T|^2$ or $|X_N|^2$ is then, by definition, chi-square distributed. (See, e.g., Refs. 11 and 30 for a derivation.)

If the Bartlett or Welch periodograms can be considered to be obtained from independent segments, then, from (10.5.20) we have

$$\text{Var}[\tilde{P}_w(f)] = \frac{1}{M} \text{Var}[\tilde{P}_{xx,m}(f; d)] \quad (10.5.26a)$$

and since, from the previous discussion, the variance of the individual periodograms $\text{Var}[\tilde{P}_{xx,m}(f; d)]$ is approximately $P_{xx}^2(f)$, we have that

$$\text{Var}[\tilde{P}_w(f)] \approx \frac{1}{M} P_{xx}^2(f) \quad (10.5.26b)$$

Thus, segmenting the data improves the variance by a factor equal to the number of segments. If there is overlap in the Welch periodogram, the segments will not be independent, but there will be more segments, hence (10.5.2b) will still be approximately correct. It should be noted that in order to drive the variance (10.5.2b) to zero as N becomes indefinitely large, M should increase also. An appropriate scheme is to choose $M \propto \sqrt{N}$.

In the correlogram approach, the above results still apply if the maximum possible lag is used, since, in that case, the estimator is identical to the periodogram. If a maximum lag $L < N - 1$ is used, the ratio N/L can be thought of as having the same effect as segmenting does in the periodogram approach, namely, reducing the variance by that number. This is the basis of the statement made earlier that L should not be larger than about $N/10$.

10.5.5. Some Considerations on Implementing PSD Estimators: Summary of the Simulation Procedure

Use of the FFT is standard because of processing speed. If we observe N samples *per segment*, or time interval NT_s , the FFT yields N frequency samples $f_i = i/NT_s$, $i = 0, 1, \dots, N - 1$. The bandwidth of a (truly bandlimited) baseband process that could be observed without aliasing is $1/2T_s$, and the resolution is $1/NT_s$. The number N in almost every FFT routine is a power of 2. If the number of observations cannot be made a power of 2, one can contrive to do so by adding zeros to the end of the observed data sequence. This is referred to as *zero-padding*. The discrete Fourier transform is unaffected at any f if we simply add zeros to $X(n)$. However, for discretely spaced f , this may not be the case and the frequency locations may not be the same, depending upon the relationship between N and N_p , the padded sequence length. If N_p/N is an integer, however, the original frequency components will appear at their original values. There will be additional components as well, which can be considered interpolated values.

For the actual simulation procedure, either the direct or indirect method can be used, although the Welch periodogram appears to be the most convenient to implement. The following summarizes the procedure to be used for either approach.

10.5.5.1. Welch Periodogram Procedure (Direct Method)

1. Collect N total data points $X(n)$, $n = 0, 1, \dots, N - 1$.
2. Select a segmentation and overlap scheme, namely, choose J samples per segment and shift S between segments so that the number of segments $M = [(N - J)/(S + 1)]$

effects the desired balance between variance and resolution. (Welch⁽³³⁾ specifically proposed 50% overlap.)

3. Select and apply a data window. (Welch⁽³³⁾ specifically proposed the Hann window.)
4. Compute M windowed periodograms and average.

10.5.5.2. Windowed Correlogram Procedure (Indirect Method)

1. Compute $\hat{R}_{XX}(k)$ directly from the definition (10.5.5) or the unbiased version $(N/N - |k|)\hat{R}_{XX}(k)$, or compute $\hat{R}_{XX}(k)$ from the following operations.
 - a. Pad $X(n)$ with zeros and create a padded sequence $X_p(n)$ with length at least $2N$, that is a power of 2. Here, this padding is necessary to avoid the circular error of the FFT discussed in Chapter 3.
 - b. Compute the discrete frequency Fourier transform with the FFT:

$$X_{2N}(i/2NT_s) = T_s \sum_{n=0}^{2N-1} X_p(n) \exp(-j2\pi in/2NT_s) \quad i = 0, 1, \dots, 2N - 1$$

- c. Obtain $\hat{R}_{XX}(k)$ from the inverse transform of the periodogram

$$\hat{R}_{XX}(k) = \frac{1}{N} \frac{1}{2N} T_s \sum_{i=0}^{2N-1} \left| X_{2N}\left(\frac{i}{2NT}\right) \right|^2 \exp\left(\frac{j2\pi ik}{2NT_s}\right)$$

This $\hat{R}_{XX}(k)$ is the biased correlogram. The unbiased version can be obtained as in step 1.

2. Apply a lag window $\lambda(k)$ and truncate to $2L + 1$ points,

$$\bar{R}_{XX}(k) = \lambda(k)\hat{R}_{XX}(k)$$

or

$$\bar{R}_{XX}(k) = (N/N - |k|)\lambda(k)\hat{R}_{XX}(k)$$

for $|k| = 0, 1, \dots, L$. To have the same averaging effect as in the periodogram approach, we should have $N/L \approx M$. Since the triangular window is implicit in $\hat{R}_{XX}(k)$, we can implicitly undo it as part of the window, or explicitly, as indicated in the last two formulas.

3. Pad $\bar{R}_{XX}(k)$ with zeros for $|k| > L$, and take the FFT,

$$\hat{P}_{XX}(f; \lambda) = T_s \sum_{k=-N'-1}^{N-1} \bar{R}_{XX}(k) \exp(-j2\pi ik/N'T_s)$$

$$f = i/N'T_s, \quad i = 0, 1, \dots, N - 1$$

Here N' is the length of the padded sequence, which may be set to N or another number if desired.

10.6. Estimating Delay and Phase

The material in this section deals with certain aspects of synchronization and could just as well have been included with the related material in Chapter 8. However, we discuss these topics here because of their statistical flavor. At least under certain simplified scenarios, the phase of a carrier and the delay of a waveform can be thought of as unknown parameters that are to be estimated. As discussed earlier, parameter estimation theory is concerned with estimating the “true” but unknown values of parameters. For almost any realistic system, there are in fact no such “true” values of delay and phase. There are “good” (and generally different) values of delay and phase which correspond to satisfying some criterion, such as minimum error probability, minimum mean-squared error, and so on. The only situation in which delay and phase are unique, and thus correspond to having “true” values, is for the very simplest system imposing only a pure time delay and phase shift upon the signal. That is, if $\tilde{s}_i(t)$ is the input complex envelope to a system, the output is assumed to be of the form

$$\tilde{s}_o(t) = g\tilde{s}_i(t - \tau) \exp(j\theta) \quad (10.6.1)$$

where we have slightly generalized the formulation by also allowing a fixed but unknown gain g , which in fact will have no effect on the result. Despite its apparent simplicity, this framework serves the useful purpose of demonstrating the fundamental role of cross-correlation in the synchronization context. That is, we will presently show that in the noiseless case a cross-correlation procedure will extract precisely the correct values τ and θ of delay and phase shift. Intuitively, we can then expect that such a procedure, or a variant of it, should work reasonably well, though not necessarily optimally, in a more realistic system.

When noise and distortion are present, the form of a good estimator becomes less obvious, and finding its distribution may not be a tractable problem. While we can follow a recipe for obtaining the estimator within a simulation, in certain performance evaluation techniques we do not attempt to generate the estimates explicitly, but instead use the distribution of the estimates as they might have been extracted by certain structures. For phase and delay estimators based on the maximum-likelihood principle, we will show the estimator structures, but the estimator distributions do not appear to be known. However, such distributions are known, or reasonable approximations thereto, for commonly used implementations (which are theoretically suboptimum) such as the phase-locked loop. We will also briefly discuss these.

10.6.1. Estimating Carrier Phase and Timing Synchronization in the Noiseless Case

We will now examine the cross-correlation procedure illustrated in Figure 10.9, assuming the distortionless system embodied by (10.6.1). The output complex envelope of the system $\tilde{s}_o(t)$ is one input to the cross-correlator and the other input is the delayed input signal $\tilde{s}_i(t - \alpha)$, where α is a delay variable. The (complex) cross-correlation is given by

$$R_{s_o s_o}(\alpha) = \langle \tilde{s}_o(t - \alpha) \tilde{s}_o^*(t) \rangle = g R_{s_i s_i}(\tau - \alpha) \exp(-j\theta)$$

where

$$R_{s_i s_i}(\tau - \alpha) = \langle \tilde{s}_i(t - \alpha) \tilde{s}_i^*(t - \tau) \rangle \quad (10.6.2)$$

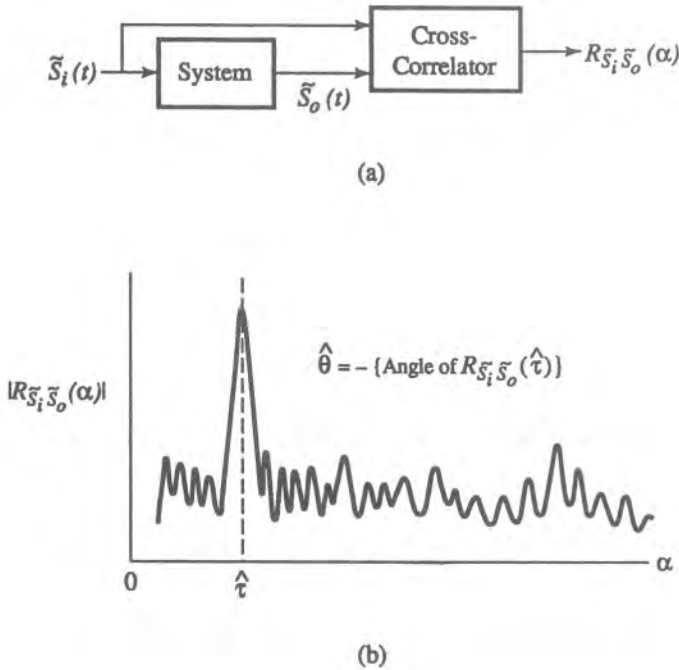


Figure 109. Estimating time delay and phase offset by the cross-correlation technique. (a) Block diagram. (b) Typical result.

Since $R_{s_i s_i}$ is an autocorrelation function, the maximum $|R_{s_i s_i}(\cdot)|$ occurs when the argument is zero, and it is known that $R_{s_i s_i}(0)$ is real-valued. Hence

$$\max_{\alpha} |R_{s_i s_o}(\alpha)| = \max |g R_{s_i s_i}(\tau - \alpha)|$$

occurs when $\alpha = \tau$, which means $\hat{\tau}$, the *estimate* of τ , is that value of α which maximizes the above expression. Furthermore, we have

$$R_{s_i s_o}(\alpha = \tau) = g R_{s_i s_i}(0) \exp(-j\theta)$$

and since $g R_{s_i s_i}$ is real, we see that the (correct) *estimate* of θ is given by

$$\hat{\theta} = -\text{angle of } R_{s_i s_o}(\alpha = \hat{\tau})$$

Thus, as stated above, the optimality of the procedure is clear for this distortionless and noiseless system, and serves at least as an intuitive justification for the cross-correlation procedure in more general contexts. In fact, it will be shown in the next chapter that the $\hat{\tau}$ that maximizes the cross-correlation between input and output is optimum in the sense of maximizing a signal-to-noise ratio measure for an *arbitrary* system. Thus, for general application we could define symbol and phase synchronization to be achieved for the $\hat{\tau}$, $\hat{\theta}$ obtained through such a procedure. As we noted above, however, in distorted systems, the notion of a unique delay or phase rotation does not apply. We are simply obtaining values of delay and phase rotation that will satisfy some performance criterion, e.g., maximizing SNR.

Of course, there is always noise in an actual system, but in the simulation context, we have the “luxury” of shutting off noise sources for the purpose of establishing some preliminary estimates of delay and phase. These estimates will generally differ from those obtained with noise present, but may be quite adequate for certain purposes. There are in fact several ways to use the cross-correlation procedure for simulation purposes. In any of the approaches to synchronization which do not use models of actual devices, it is necessary to derive in some fashion initial values of carrier phase and symbol timing. We might then search performance behavior around these initial values, as in the hard-wired approach discussed in Chapter 8. The cross-correlation procedure is one way to obtain these initial values. Some software packages offer users such a procedure as a “front-end” option prior to a simulation run to initialize delay and phase. Typically this is implemented by transmitting a relatively short PN sequence of length, say, K bits. Whenever the estimator is derived from processing a fixed block of data, as in this case, the estimator will be referred to as a *block* estimator. Of course, in the noiseless case discussed so far, the estimators are perfect, which is to say their pdf’s are delta functions.

Again, in the absence of actual or planned physical implementations, we can also employ cross-correlation as a reasonable “stand in” for real circuitry, but operating on the actual distorted and noisy waveform. The properties of these estimates would depend on the “memory” (i.e., the value of K in the above procedure) of the cross-correlation. The corresponding performance estimate would simply be accepted as the “actual” estimate. In a block-based simulation, it is natural to make the delay estimator memory K the same as the length of the block. In a stream simulation, more options are available. The delay estimator can be based on some arbitrary but appropriate memory K , which is recomputed every K symbols and applied to those symbols. Or one could use a *sliding-block* estimator, which imitates a continuous structure: here, the memory is also some number K , but the block shifts by only one symbol at a time, or possibly several symbols.

The cross-correlation is by definition a time-domain operation. It can be implemented in principle by the procedure outlined in Section 11.1 and illustrated in Figure 11.3. In many cases a “frequency-domain” or FFT-based procedure will be a more efficient way to effect the cross-correlation. This approach is also described in Section 11.1 and illustrated in Figure 11.5.[†]

10.6.2. Block Estimators

As defined above, a *block* estimator is one based on processing a segment of signal of predetermined length. In principle, the block could be a waveform (in the actual hardware) or samples of the complex envelope or of a suitably processed version of it. Of course, in simulation, “continuous” and “sampled” coalesce, although the sampling rate and numerical precision will generally differ between the discrete-time emulation of analog processing and some actual discrete-time processing. Block estimators are useful in several ways. They are natural for reproducing actual block processors, as are used in burst transmissions, e.g., in TDMA systems. They are well suited for implementing actual digital processors that perform this function. They are useful, as mentioned above, as hypothetical structures for synchronization (barring actual information), and in block-based simulation (FFT processing) they

[†] There is a small difference between the setting here and that in Chapter 11. In the latter, the waveform is assumed real, while here it is complex. The difference in the operations should be clear.

are the natural tool to use. From a theoretical standpoint, optimum synchronization structures based on the maximum *a posteriori* (MAP) principle, or the closely related maximum-likelihood principle, always begin by formulating the likelihood ratio for some fixed number of samples, i.e., a block estimator. This is true even when the intended structure is continuous, as the block estimator is often used as the basis for inferring a related continuous structure. There are many variations on the form of the estimators, depending upon a number of factors: the modulation method, independent or joint estimation of phase and clock, and what auxiliary information may be available. For example, the data could be assumed known (“data-aided,” as in a training sequence), a clock might be available when estimating phase, etc. The resulting structures are typically feedforward (or open loop) and can be used as is, but often form the basis for deriving continuous or discrete feedback (closed-loop) structures. There is a large literature on the subject, and the reader is referred to Refs. 34 and 35, among many others, as well as the related references in Chapter 8.

10.6.2.1. Block Delay Estimator

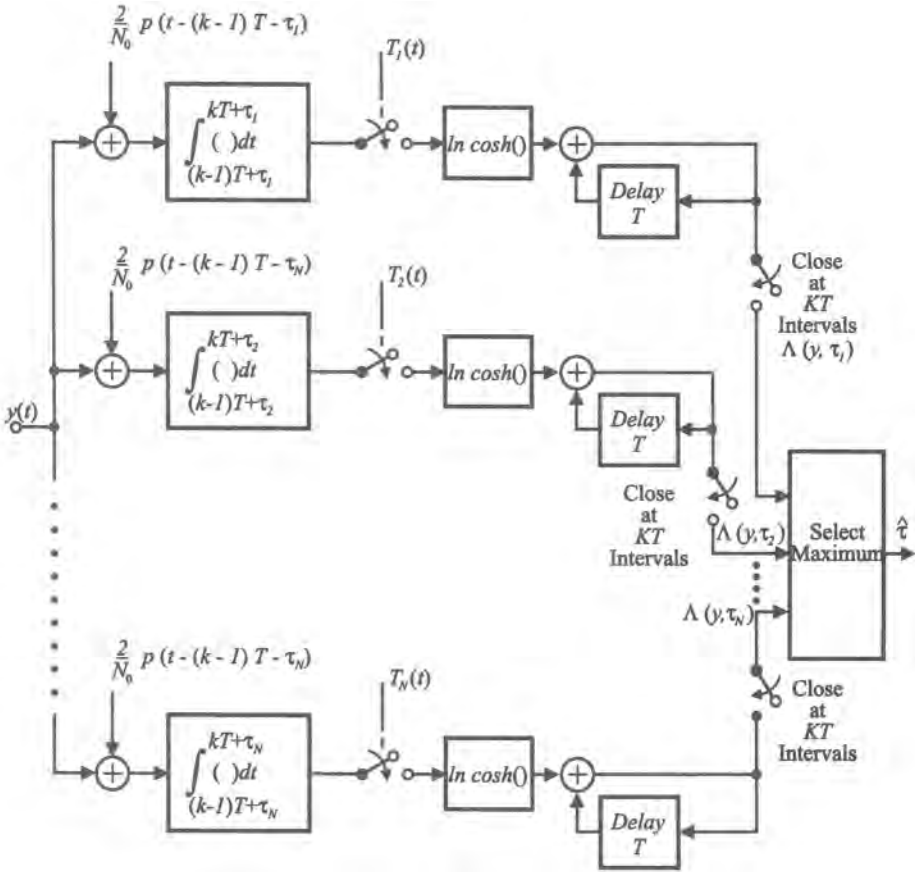
We now look at one structure of a block estimator for delay. In an actual system, synchronization estimates are invariably corrupted by noise (and distortion). Here we assume that Gaussian noise is present, but continue the distortionless assumption. We will simply state the end result of the development in Refs. 34 and 36, since the final form is mainly of interest here. For this purpose it is useful to write explicitly the received signal, namely,

$$y(t) = \sum_{k=0}^K a_k p(t - (k-1)T - \tau) + n(t)$$

where $\{a_k\}$ is the ± 1 symbol sequence, T is the signaling interval, and $p(t)$ is the basic signal pulse, assumed time-limited, i.e., $p(t) = 0$ for $t < 0$ and for $t > T$, but is otherwise arbitrary. (Note that unlike the case treated in the preceding subsection, the signal here is assumed to be real.) The delay τ is fixed but unknown *a priori*, and so can reasonably be considered a random variable uniformly distributed in the interval $(-T/2, T/2)$ modulo T . The optimum estimate (in the MAP sense) of the delay is given by that value $\hat{\tau}$ of τ which maximizes the following expression:

$$\Lambda(y, \tau) = \sum_{k=0}^K \ln \cosh \left[\frac{2}{N_0} \int_{T_k(\tau)} y(t) p(t - (k-1)T - \tau) dt \right] \quad (10.6.3)$$

where $T_k(\tau)$ is an interval defined by $(k-1)T + \tau \leq t \leq kT + \tau$. An implementation of (10.6.3) is shown in Fig. 10.10 (compare with Fig. 11.3). This is a block estimator with memory K symbols. The resulting estimate $\hat{\tau}$ is optimum in the MAP sense, given K and $p(t)$ as well as the other assumptions; it is not clear, however, what performance measure would thereby be optimized in an actual system. In any case, it is interesting to see that the receiver first performs a cross-correlation, followed by a nonlinear operation. But the cross-correlation is fundamental. This estimator is an *open-loop* solution, which does not pose any stability issues, but it is relatively computationally intensive. Various continuous structures can be derived from this basic form, as described in Ref. 34.


 Figure 10.10. Maximum *a posteriori* (MAP) symbol synchronizer for arbitrary pulse waveshape (after Ref. 34).

The estimator in (10.6.3) is not difficult to implement in simulation, though, as mentioned, we cannot assert that the corresponding system performance would be optimized. The distribution of mis estimator also appears difficult to obtain, though some measures of it have been obtained.^(34,37) For example, it can be shown⁽³⁷⁾ that for the normalized timing error ε defined as

$$\varepsilon = (\tau - \hat{\tau})/T$$

the rms value is given by

$$\sigma_{\varepsilon} = \frac{0.25}{(KE_b/N_0)^{1/2}} \quad (10.6.4)$$

Practical synchronizers will have values somewhat larger than this.

10.6.2.2. Block Phase Estimator

We now consider a block phase estimator, again for the distortionless case with additive Gaussian noise. We assume the incoming signal is a quadrature modulated carrier of the form

$$s(t) = (2\rho)^{1/2} \left\{ \left[\sum a_k f(t - kT) \right] \cos(\omega_c t + \theta_c) + \left[\sum b_k g(t - d - kT) \right] \sin(\omega_c t + \theta_c) \right\} \quad (10.6.5)$$

where $\rho = (E_s/N_0)^{1/2}$; E_s is the energy per symbol; N_0 is the single-sided noise PSD; $\{a_k\}$ and $\{b_k\}$ are the I- and Q-channel input binary sequences; $f(t)$ and $g(t)$ are the I- and Q-channel pulse shapes; T is the symbol duration; d is the channel timing offset (also called *skew*); and θ_c is the unknown phase to be estimated. Some common examples of (10.6.5) are[†]

$$\text{Standard QPSK: } f(t) = g(t) = p_{T/2}(t - T/2); \quad d = 0$$

$$\text{Offset QPSK: } f(t) = g(t) = p_{T/2}(t - T/2); \quad d = 0.5T$$

$$\text{MSK: } f(t) = g(t) = \sin(\pi t/T); \quad 0 \leq t \leq T; \quad d = 0.5T$$

Let $\{\tilde{Z}_k\}$ represent the sequence of samples taken at intervals kT at to the output of a matched (or lowpass) filter whose input is the complex envelope. For $f(t) = g(t)$, it can then be shown^(35,38) that the optimum estimate of θ_c (in the maximum likelihood [ML] sense) is given by that value of θ which is the solution to the following expression:

$$\max_{\theta} \sum_{k=1}^K \ln \left\{ \cosh[(2\rho)^{1/2} \operatorname{Re}(\tilde{Z}_k e^{-j\theta})] + \cosh[(2\rho)^{1/2} \operatorname{Im}(\tilde{Z}_k e^{-j\theta})] \right\} \quad (10.6.6)$$

where K is the number of symbols in the block and can again be thought of as the “memory” of the procedure. This specific estimator is one which is not data-aided (neither the symbol values a_k, b_k nor their estimates appear in the expression), but is “clock-aided,” i.e., the estimator implicitly depends on symbol synchronization since $\tilde{Z}_k$ is the sampled value of the matched filter output. It is interesting that, at low signal-to-noise ratio, the estimator (10.6.6) reduces to

$$\hat{\theta}_c = \frac{1}{4} \arg \left(\sum_{k=1}^K \tilde{Z}_k^4 \right) \quad (10.6.6a)$$

where one can recognize the quadrupling principle discussed in Chapter 8 for QPSK. One can again recognize the application of the cross-correlation concept in the terms $\tilde{Z}_k e^{-j\theta}$ in (10.6.6). This estimator is also an open-loop structure, which is straightforward to implement, though somewhat computationally costly. A block diagram of an implementation would have a similar structure as that of Figure 10.10, except that now the various branches would correspond to discretized values of θ . For offset formats, assuming again that $f(t) = g(t)$, the ML estimator becomes^(35,38)

$$\max_{\theta} \sum_{k=1}^K \ln \left\{ \cosh[(2\rho)^{1/2} \operatorname{Re}(\tilde{Z}_k e^{-j\theta})] + \cosh[(2\rho)^{1/2} \operatorname{Im}(\tilde{Z}_{k+1/2} e^{-j\theta})] \right\} \quad (10.6.7)$$

where the subscript $k + 1/2$ means $(kT + T/2)$. The extension to different functions for $f(t)$ and $g(t)$ is covered in Ref. 38.

[†] Recall that $p_{T/2}(t)$ is a unit-amplitude rectangular pulse for $|t| \leq T/2$. The “basic” pulse in the table is shifted by $T/2$ so that it begins at the origin. This is not fundamental, but conforms to many treatments.

We reiterate that (10.6.6) and (10.6.7) are based specifically on the assumption that the transmitted signal is corrupted only by additive Gaussian noise. Thus, in a realistic system we cannot assume that the forms provide an optimum solution, or even a good one. In fact, we can expect that in the presence of distortion we might better use an estimator that makes use of all the samples of the complex envelope (separated by T_s) rather than symbol-spaced samples. Thus, while we introduced (10.6.6) or (10.6.7) to motivate the discussion, in practice we might use a heuristic block estimator that does use **T_s -separated** samples. One advantage of such an estimator is that it is also independent of symbol synchronization. Of course, this presents a tradeoff in computation not only in simulation, but in digital hardware also.

An example of such a heuristic algorithm for QPSK based on the quadrupling principle has been suggested.⁽³⁹⁾ This algorithm returns an estimate based on the following computation:

$$\hat{\theta}_c(L) = \sum_{j=0}^L \bar{\phi}_j \quad (10.6.8)$$

where

$$\bar{\phi}_n = \frac{1}{4N} \sum_{i=1}^N \left\{ 4 \left[\phi(iT_s) - \sum_{j=0}^{n-1} \bar{\phi}_j \right] \bmod(2\pi, \pi) \right\} \quad (10.6.9)$$

$\phi(i)$ is the phase of the complex envelope, and **$\bmod(2\pi, \pi)$** means reduction modulo 2π until the absolute value of the remainder is less than π . The estimate (10.6.8) is essentially a recursively computed average of the phase over the block, exclusive of the modulation. The number of iterations L can be chosen as a fixed value or based on a differential criterion $\hat{\theta}_c(L) - \hat{\theta}_c(L-1)$. The **$\bmod(2\pi, \pi)$** operation is designed to remove the fourfold ambiguity that normally accompanies the times-four estimate. A new estimate would be provided in each contiguous block of N samples.

A block algorithm used in the simulation context might be an emulation of an actual (hardware or firmware) algorithm, or it might be a heuristic procedure, as above, intended to imitate, but not duplicate the action of a synchronization circuit. As mentioned above, a useful feature of a block algorithm is that it is naturally suited to block-based (or FFT-based) simulation architecture. The distribution of the estimators (10.6.6)–(10.6.7) is not easy to come by, nor for that matter that of the estimator (10.6.8). This is of no great concern if we apply the estimators directly, but for certain performance estimation methods, it is necessary to know the functional form of the estimator distributions. We speak to this issue briefly in the next subsection.

10.6.3. Distribution of PLL-Based Phase and Timing Estimators

In one of the performance evaluation techniques developed in the next chapter, we formulate a BER estimate as an average over the distribution $f(\hat{\theta}, \hat{\tau})$ of the phase and timing estimates $\hat{\theta}$ and $\hat{\tau}$, or equivalently the phase and timing errors. In this case, the synchronization circuitry is not explicitly simulated. Its properties are embodied in f . As we have seen, this pdf does not appear to be known for the block estimators discussed above. However, for

synchronization structures employing a phase-locked loop, good approximations for these distributions are known.

10.6.3.1. Distribution of the Phase Estimator

For the simple model where the signal input to the PLL is of the form

$$r(t) = \sqrt{2A} \cos[2\pi f_c t + \theta] + n(t) \quad (10.6.10)$$

where $n(t)$ is additive white Gaussian noise, there is no loop detuning, and the transmitter and receiver oscillators are perfectly stable, results have been derived[†] for the first-order pdf of the (modulo 2π reduced) phase error $\phi(t) = \hat{\theta}(t) - \theta$. For a first-order loop, the probability density function of $\phi(t)$ is given by⁽³⁴⁾

$$f(\phi) = \frac{\exp(\rho \cos \phi)}{2\pi I_0(\rho)}, \quad |\phi| \leq \pi \quad (10.6.11)$$

where I_0 is the zeroth-order modified Bessel function, and

$$\rho = \frac{A^2}{N_0 B_L}$$

is the “loop SNR,” which is the received signal-to-noise ratio in a bandwidth equal to B_L , the one-sided loop bandwidth (defined in Case Study II, Chapter 12), and as usual N_0 is the one-sided receiver noise PSD. For relatively large values of ρ , (10.6.11), with ϕ expressed in radians, can be approximated by⁽³⁷⁾

$$f(\phi) = \frac{\exp[-\phi^2(\rho/2)]}{(2\pi/\rho)^{1/2}} \quad (10.6.12)$$

which is to say a normal distribution with variance ρ^{-1} .

The expression for the pdf of the phase error for the second-order PLL is too complicated⁽³⁴⁾ to be useful as a model. However, it, too, can be well approximated by a normal pdf for most signal-to-noise ratios of interest. Under the conditions of the model (10.6.10) we have

$$f(\phi) = \frac{\exp(-\phi^2/2\sigma_\phi^2)}{(2\pi)^{1/2}\sigma_\phi} \quad (10.6.13)$$

where σ_ϕ is the rms phase jitter. A comparison of the normal approximation to the experimentally measured pdf is shown in Figure 10.11a for $\rho = 6.41$ dB and $r = 2$, where r is a parameter of the loop defined as $r = AK_1K_2K_3\tau_2^2/\tau_1$ and the other parameters are defined in Section 8.11.

[†] Results are also known for more complicated models, but, as expected, they are not as simple as those for the model at hand.

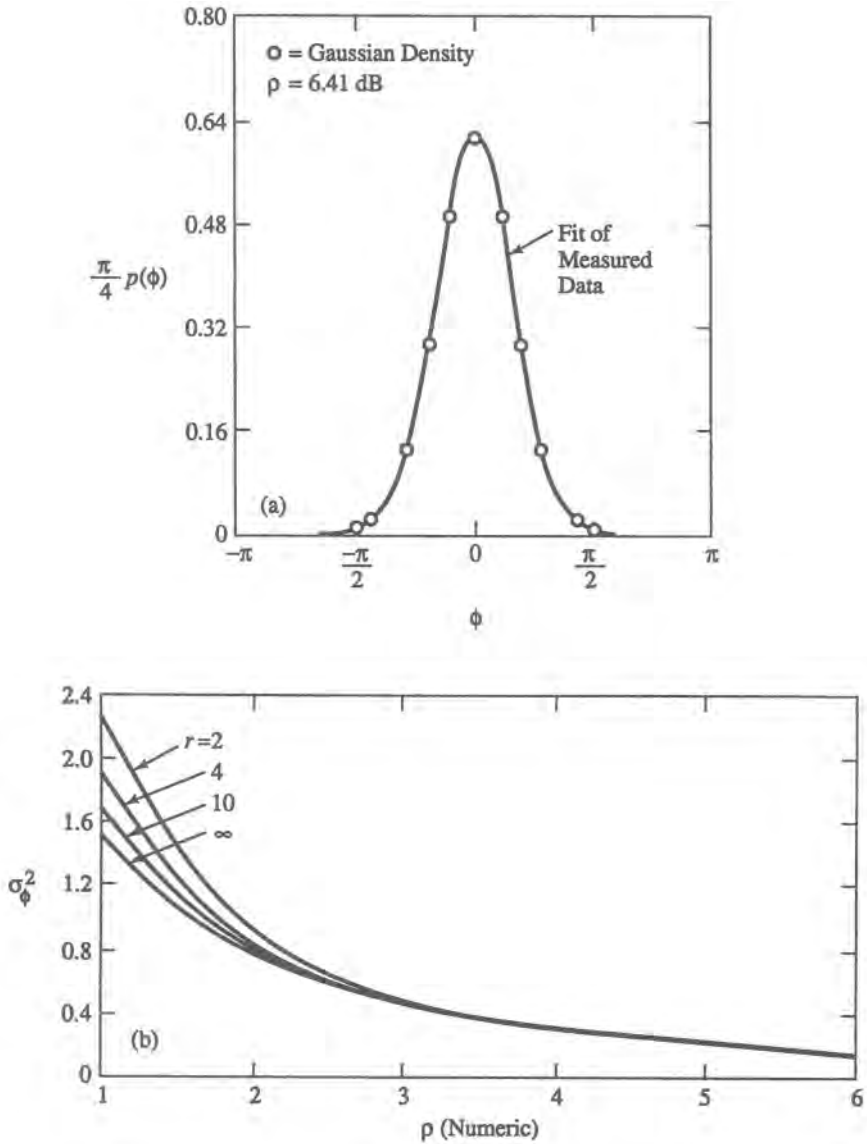


Figure 10.11. Some properties of the second-order loop, (a) Comparison of normal approximation to the measured pdf of the second-order loop (with $r = 2$); solid line is a fit of measured data (from F. J. Charles and W. C. Lindsey, Some analytical and experimental phase-locked loop results for low signal-to-noise ratios, © Proc. IEEE, Vol. 54, No. 9, pp 276–290, Sept. 1966.

As can be seen, the agreement is excellent. The exact pdf of the second-order loop⁽³⁴⁾ is also indistinguishable from the solid line in Figure 10.11 a. Therefore, one would be justified in using (10.6.13) as a model, for which we need to know only the rms phase jitter σ_ϕ . For the loop considered, Figure 10.11b shows the variance (σ_ϕ^2) as a function of ρ for the various values of the parameter r defined above.

It is important to note that the jitter for a structure that incorporates a PLL, such as the squaring loop (Section 8.12.4), is not necessarily the same as that of the loop itself because the prior nonlinear processing will generally alter the noise. For the squaring loop with an ideal rectangular input bandpass filter, it can be shown⁽³⁴⁾ that for small phase error the variance indicated above is increased by the factor $(1 + 1/\rho\gamma)$, called the squaring loss, where $\gamma = 2B_L/W$ and W is the bandwidth of the bandpass filter.

For an N th-power loop, we note that the produced carrier component will have a (mod 2π reduced) distribution of the form (10.6.13) where σ_ϕ is the rms phase jitter in the loop. When the phase estimate is divided by N to produce the local reference, say α , for demodulation, the pdf of α will have the same form as (10.6.13) with ϕ replaced by $N\alpha$, with the range of α given by $|\alpha| \leq \pi/N$.

10.6.3.2. Distribution of the Timing Estimator

As we have seen in Chapter 8, many timing recovery structures are based on the nonlinear processing of a waveform, which produces a periodic signal to be tracked by a PLL or some type of feedback circuit. As an example, consider NRZ data and as before define $\epsilon = (\tau - \hat{\tau})/T$, the normalized timing error. The corresponding pdf is given by⁽³⁴⁾

$$f(\epsilon) = \frac{\exp[(\cos 2\pi\epsilon)/(2\pi\sigma_\epsilon)^2]}{I_0[(2\pi\sigma_\epsilon)^{-2}]}, \quad |\epsilon| < \frac{1}{2} \quad (10.6.14)$$

which, under the same conditions that led from (10.6.11) to (10.6.12), can be approximated by a normal density. The resulting model is easy to apply and only requires knowledge (or an assumed value) for the rms timing jitter σ_ϵ . This model is symmetric about the “true” value of delay τ , but one can easily add to it a bias, as discussed in Section 8.11.2.

10.7. Visual Indicators of Performance

In digital communications eye diagrams and scatter diagrams are widely used as qualitative (visual) indicators of the “health” of a system.^(40,41) Although these indicators are qualitative, they have a statistical flavor in the sense that what is seen represents the system behavior at some point, over the “ensemble” of possible waveforms at that point. Eye diagrams and scatter diagrams are widely used as diagnostics and are very useful adjuncts in a simulation’s postprocessing capabilities.

10.7.1. Eye Diagrams

The quality of a communication system can be deduced qualitatively from the degree of distortion in the waveform at the demodulator output if there were no noise in the system. An eye diagram encapsulates this distortion by displaying the detected waveform in a way that is convenient for this purpose.

An eye diagram is formed in the following way. First, it is customary to use as input to the system a maximum-length (ML) pseudorandom sequence generator, say of period N symbols. The system output $P(t)$ is of course periodic with period NT , where T is a symbol duration.

Consider the set of shifted or delayed waveforms

$$\{P_k(t)\} = \{P(t - kT), k = 0, 1, \dots, N - 1\} \quad (10.7.1)$$

which consists of $P(t)$ and all delayed versions by multiples of T , up to $(N - 1)T$. We call $P_k(t)$ the K th trace of $P(t)$. Now imagine all N traces simultaneously displayed on the same set of axes. Because every $P_k(t)$ is a shifted version of $P(t)$, it, too, is periodic. A little reflection shows that the simultaneous display just mentioned is also periodic, but with period T instead of NT . That is, in every interval $mT \leq t \leq (m + 1)T$, for all integers m , the display is identical and is called the eye diagram. The preceding description corresponds to what is seen in laboratory setups, where every trace appears on the oscilloscope screen. Although eye diagrams are perfectly meaningful for random sequences, the use of a PN sequence ensures that all ISI patterns are displayed.

A computer-generated eye diagram is shown in Figure 10.12, from which the origin of the term becomes clear. Even though it is the eye diagram corresponding to the waveform at the decision device input that is of ultimate interest, the eye diagram can be obtained at various points in a system to obtain a sense of how distortion accumulates, or perhaps where a major contributor exists. Of course, in a carrier-modulated system, points prior to the demodulated output will be at IF or RF, and an imaginary demodulator must be postulated at any such point in order to obtain the eye diagram.

For the system in question, the waveform is normalized so that $+1$ or -1 represents the amplitude of the (rectangular) waveform that would be observed if there were no distortion. The decision threshold is at zero. It can be seen that some traces are farther from the threshold and some closer. The “eye opening” $E_0(\leq 1)$ is defined as the distance from the threshold to the closest trace at the (postulated) sampling instant. The eye opening can be used to derive sometimes useful performance bounds.

Figure 10.12 exhibits another phenomenon, a closing of the eye along the time axis. The width of the traces around the zero crossings ($\Delta\tau$ in the figure) is indicative of distortion and jitter. Since the eye is oblong in shape rather than rectangular, if we sample off-center, we will encounter lower signal values, and the narrower the horizontal opening, the steeper the eye will descend. Thus, the zero-crossing width in the eye pattern is an indicator of degradation

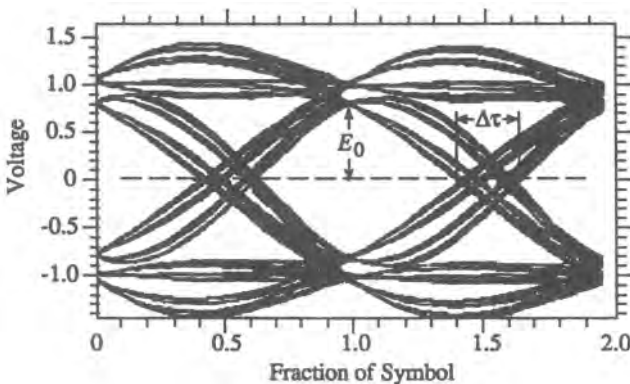


Figure 10.12. Example of an eye diagram.

that will be incurred due to sampling clock jitter. Nonsymmetries in the eye diagram indicate the presence of nonlinearities.

10.7.2. Scatter Diagrams

The term *scatter diagram* is often used to describe a plot of the values of a set of samples of a random variable against a common independent variable. The same idea applies here except that, as typically done, the variable in question is not random. This variable is the sets of values of samples of the demodulated waveform, either at the output of the actual demodulator or at the output of a hypothetical demodulator at some intermediate point. The *waveform* here is interpreted as the complex envelope, which is representable either in polar or rectangular coordinates.

As with eye diagrams, scatter diagrams are usually obtained by sending ML pseudorandom sequences in a noiseless system. Under this condition, a scatter diagram does not represent random samples. Rather, it shows the effects of imperfections in the system through a distortion of the signal constellation in signal space. This idea is illustrated in Figure 10.13, which applies to a 16-QAM rectangular constellation. The idealized constellation is shown in Figure 10.13a, in which each point (symbol) corresponds to a certain amplitude and phase of the carrier. Each of these points can also be thought of as the in-phase and quadrature voltages at the output of an ideal demodulator following an ideal modulator. As the signal progresses down the system, distortion manifests itself. But at any instant, the distortion is a function of the preceding symbols. Thus, at the sampling instant the complex envelope will take on a multiplicity of values. This is illustrated in Figure 10.13b. As can be seen, where each symbol had originally been an ideal point, it is now represented by a small area around that point, which contains all sampled values. The figure also illustrates the effect of nonlinear distortion, which induces a compression and rotation of the original constellation. The scatter diagram is especially helpful in visualizing effects such as these.

Since the scatter diagram shows all the possible values of a waveform at a particular sampling epoch, it can be seen that the scatter diagram is exactly a vertical slice of the eye diagram. Hence, as with the eye diagram, we can use the scatter diagram to bound the BER by

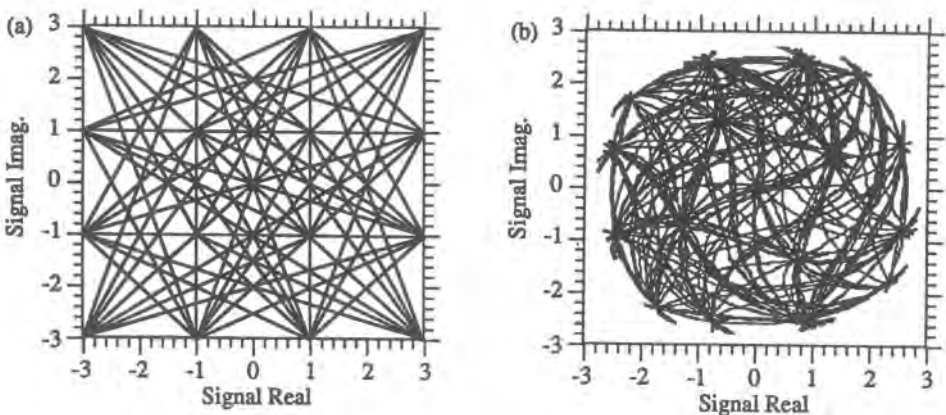


Figure 10.13. Examples of scatter diagrams for a 16-QAM constellation. (a) Ideal constellation, (b) Filtered and nonlinearly distorted constellation.

inserting into the theoretical expression for BER the closest distance between modulation states. This bound, of course, will apply only to the corresponding sampling instant.

10.8. Summary

Many properties of a signal are of interest, either as potential diagnostics or directly as the performance measure(s) of interest. The latter are discussed in the next chapter. In this chapter we looked at a number of waveform properties that may be useful to know for various reasons, such as the average level, average power, amplitude probability distribution, power spectral density, and delay or phase rotation. We described the procedures whereby the property of interest can be estimated. By and large, these procedures are straightforwardly implemented in simulation. Where applicable, we also developed the statistical properties of the estimators so as to provide the simulation user with the tradeoff between estimator reliability and computer run time.

References

1. W. A. Gardner, *Introduction to Random Processes*, Macmillan, New York (1986).
2. W. A. Gardner, Rice's representation for cyclostationary processes, *IEEE Trans. Commun.* **COM-35**(1), 74–78 (1987).
3. W. A. Gardner, Common pitfalls in the application of stationary process theory to time-sampled and modulated signals, *IEEE Trans. Commun.* **COM-35**(5), 529–534 (1987).
4. A. M. Mood, *Introduction to the Theory of Statistics*, McGraw-Hill, New York (1950).
5. M. Fisz, *Probability Theory and Mathematical Statistics*, 3rd ed., Wiley, New York (1963).
6. P. L. Meyer, *Introductory Probability and Statistical Applications*, 2nd ed., Addison-Wesley, Reading, Massachusetts (1970).
7. V. K. Rohatgi, *Statistical Inference*, Wiley, New York (1984).
8. H. Cramer, *Mathematical Methods of Statistics*, Princeton University Press, Princeton, New Jersey (1946).
9. P. Bickel and K. Doksum, *Mathematical Statistics*, Holden-Day, San Francisco (1977).
10. R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*, 4th ed., Macmillan, New York (1978).
11. K. S. Shanmugan and A. M. Breipohl, *Random Signals: Detection, Estimation and Data Analysis*, Wiley, New York, pp. 18–20 (1988).
12. E. Parzen, On estimation of a probability density and its mode, *Ann. Math. Stat.* **33**, 1065–1076 (1962).
13. R. Tapia and J. Thompson, *Nonparametric Probability Density Estimation*, Johns Hopkins University Press, Baltimore, Maryland (1978).
14. S. M. Ross, *Stochastic Processes*, Wiley, New York (1983).
15. F. Amoroso, The bandwidth of digital data signals, *IEEE Commun. Mag.* **18**(6), 13–24 (1980).
16. D. Slepian, On bandwidth, *Proc. IEEE* **64**(3), 292–300 (1976).
17. D. Middleton, *An Introduction to Statistical Communication Theory*, McGraw-Hill, New York (1960).
18. H. E. Rowe, *Signal and Noise in Communication Systems*, Van Nostrand, New York (1965).
19. N. Blachman, The power spectrum of a digital signal, *IEEE Trans. Commun.* **COM-22**(3), 349–350 (1974).
20. V. K. Prabhu and H. E. Rowe, Spectra of digital phase modulation by matrix methods, *Bell Syst. Tech. J.* **53**(5), 899–935 (1974).
21. I. Kom, *Digital Communications*, Van Nostrand Reinhold, New York (1985).
22. J. B. Andersen, T. Aulin, and C. E. Sundberg, *Digital Phase Modulation*, Plenum Press, New York (1986).
23. O. Shimbo, *Transmission Analysis in Communications*, Vol. I, Computer Science Press, Rockville, Maryland (1988).
24. G. Robinson, O. Shimbo, and R. Fang, PSK signal power spectrum spread produced by memoryless nonlinear TWTs, *COMSAT Tech. Rev.* **3**(2), 227–256 (1973).
25. C. Devieux, Analysis of PSK signal power spectrum spread with a Markov chain model, *COMSAT Tech. Rev.* **5**(2), 225–252 (1975).
26. S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).

27. F. J. Harris, On the use of windows for harmonic analysis with the discrete Fourier transform, *Proc. IEEE* **66**(1), 51-83 (1978).
28. S. M. Kay and S. L. Marple, Jr., Spectrum analysis—A modern perspective, *Proc. IEEE* **69**(11), 1380-1419 (1981).
29. Special Issue on Spectral Estimation, *Proc. IEEE* **70**(9) (1982).
30. N. Mohanty, *Random Signals, Estimation and Identification*, Van Nostrand Reinhold, New York (1986).
31. S. L. Maple, Jr., *Digital Spectral Analysis with Applications*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
32. W. B. Davenport and W. L. Root, *An Introduction of the Theory of Random Signals and Noise*, McGraw-Hill, New York (1958).
33. P. D. Welch, The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short modified periodograms. *IEEE Trans. Audio Electroacoust.* **AU-15**, 70-73 (1967).
34. W. C. Lindsey and M. K. Simon, *Telecommunication Systems Engineering*, Prentice-Hall, Englewood Cliffs, New Jersey (1973).
35. U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*, Plenum Press, New York (1997).
36. P. Wintz and E. J. Luecke, Performance of optimum and suboptimum synchronizers, *IEEE Trans. Commun. Technol.* **COM-17**(3), 380-389 (1969).
37. J. J. Spilker, Jr., *Digital Communications by Satellite*, Prentice-Hall, Englewood Cliffs, New Jersey (1977).
38. I. Richer, A block phase estimator for offset-QPSK signaling. In *Proc. National Telemetry Conference*, New Orleans, Louisiana (1975).
39. M. C. Jeruchim, Some modeling aspects in the simulation of digital links. In *Proc. International Telemetry Conference*, Los Angeles, California (1978).
40. K. Feher *et al.*, *Telecommunications Measurements, Analysis and Instrumentation*, Prentice-Hall, Englewood Cliffs, New Jersey (1987).
41. Bell Telephone Laboratories, *Transmission Systems for Communications*, 5th ed. (1982).

Estimation of Performance Measures from Simulation

As we mentioned in the previous chapter, the present chapter is concerned with extracting performance measures from simulation. Obtaining such measures is normally the primary objective of a simulation run. For analog transmission, the standard measure of performance is the signal-to-noise ratio (SNR), although that measure is also useful even for digital transmission. Estimation of the SNR is taken up in Section 11.1. For digital signaling, however, the usual performance measure is related to the errors in the received symbol stream. There are different ways to describe the error content of a digital stream, which will be discussed in Section 11.2, along with specific techniques for estimating the related measures.

It is appropriate here to elaborate a bit on a point hinted at in the previous chapter, namely the use of the terms *estimate* or *estimator*. These terms have a very specific connotation when used in a statistical sense. This is the appropriate sense for a Monte Carlo[†] simulation since, as previously pointed out, such a simulation should be viewed as a statistical experiment. In particular, an estimate is a random quantity, the output of a measurement on a system *as-is*. The uncertainty, presumably, is due only to statistical variation, the degree of which is quantified by standard statistical measures of dispersion (discussed in the previous chapter) such as the confidence interval or the variance of the estimator (assuming an unbiased estimator). There are actually relatively few instances when a simulation reproduces a system *as-is*. (In fact, it could be argued that this almost never occurs because, even when such fidelity is intended, there are inevitable modeling errors, no matter how small.) In fact, in many of the performance evaluation techniques to be described (for digital signals), departure from *as-is* representation is inherent in the technique. In such cases the use of estimate in a statistical sense does not apply strictly. Without making too fine a point of it, *performance evaluation* may be a better generic term, but when it suits us we shall also use “estimate” in a general way—the context will make clear its proper interpretation.

The notation and conventions established in Section 10.1 will be used in this chapter as well.

[†]We will henceforth interchange at will Monte Carlo and its abbreviation MC.

11.1. Estimation of Signal-to-Noise Ratio

The standard measure of performance for an analog signal is the *signal-to-noise ratio* (SNR), although SNR is also a meaningful measure of quality for any signal. For an analog signal, it is natural to take the purpose of the transmission system to reproduce the input signal as closely as possible. Therefore, it follows that one can think of the output as composed of two parts; one which “looks” like the original signal, the other part being the remainder. It is also natural to consider proper reproduction to be unaffected by amplitude scaling or delay; that is, the signal part of the output is *similar* to the input signal. Let the waveform $X(t)$ at the system output be obtained from a transformation on the input signal and noise

$$X(t) = g[S(t), N(t)] \tag{11.1.1}$$

The operation g can be quite arbitrary, and nothing that follows depends explicitly on it. According to our previous observation, we can decompose the output into a “signal” part and a “noise” part

$$\mathbf{X} = \mathbf{S}_0 + \mathbf{N}_0 \tag{11.1.2}$$

where the signal part is a scaled/delayed version of the input (refer to Figure 11.1). How to choose the scale and delay forms the essence of the problem. There is more than one reasonable way to do so, but below we shall use the common approach of defining the signal as the best mean-square fit of the output. For simplicity we shall assume that all input and output waveforms are real. The case of complex signals is a relatively straightforward extension (see Problem P11.3).

11.1.1. Derivation of the Estimator

It is convenient to proceed by assuming that we have drawn a specific realization of signal, $s(t)$, and a specific sample function of noise, $n(t)$, which together produce a specific realization of the output, $x(t)$. For these particular functions, the equivalent of (11.1.2) is

$$\mathbf{x} = \mathbf{s}_0 + \mathbf{n}_0 \tag{11.1.3}$$

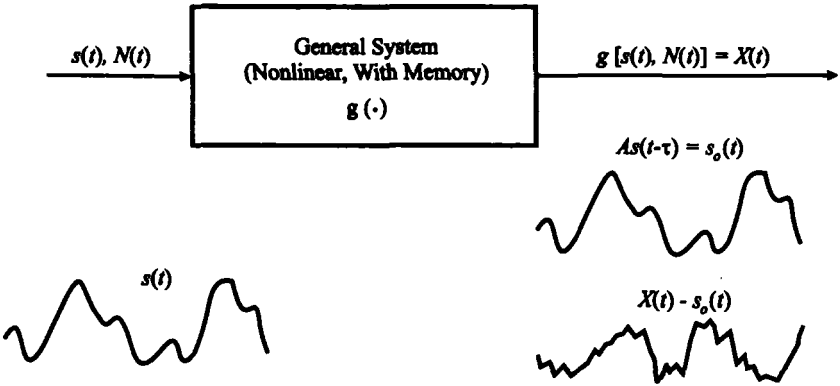


Figure 11.1. Definition of signal-to-noise ratio.

We express the signal as

$$\mathbf{s}_0 = A\mathbf{s}_\tau \quad (11.1.4)$$

where

$$\mathbf{s}_\tau = \begin{cases} s(t - \tau), & 0 \leq t \leq T \quad \text{or} \quad s(iT_s - \tau), \quad i = 1, 2, \dots, N \\ 0, & \text{elsewhere} \end{cases} \quad (11.1.5)$$

We *define* the output signal and noise to be that for which the average squared error

$$\varepsilon^2 = \langle (\mathbf{x} - A\mathbf{s}_\tau)^2 \rangle \quad (11.1.6)$$

is minimized over all A and τ . It can be shown that (11.1.6) is minimized when

$$u(\tau, A) = A^2 \langle \mathbf{s}_\tau^2 \rangle - 2AR_{\mathbf{x}\mathbf{s}}(\tau) \quad (11.1.7)$$

is also minimized, where

$$\langle \mathbf{s}_\tau^2 \rangle = \frac{1}{N} \sum_{k=1}^N s^2(kT_s - \tau) \quad (11.1.8a)$$

is the N -point average power, and

$$R_{\mathbf{x}\mathbf{s}}(\tau) = \frac{1}{N} \sum_{k=1}^N x(kT_s)s(kT_s - \tau) \quad (11.1.8b)$$

will be referred to as the *empirical cross-correlation* function. We assume there exist unique $A = A_*$ and $\tau = \tau_*$ which minimize (11.1.7). For the applicable τ_* it can be shown that

$$A_* = R_{\mathbf{x}\mathbf{s}}(\tau_*) / \langle \mathbf{s}_{\tau_*}^2 \rangle \quad (11.1.9)$$

which indicates that once τ_* is found, A_* follows automatically. For this particular measurement, therefore, a “signal” $\mathbf{s}_0$ and a “noise” ε are defined by the foregoing procedure. Hence, the SNR estimate $\hat{\rho}$ is given by

$$\hat{\rho} = \langle \mathbf{s}_0^2 \rangle / \varepsilon^2 \quad (11.1.10)$$

There is an interesting geometrical representation of the signal and noise in terms of orthogonal vectors, and a further decomposition of the “noise” into “random” noise and distortion, ⁽¹⁾ as illustrated in Figure 11.2. [In this figure, $\mathbf{V}$ is a linear subspace of the space of finite-energy signals that contains $s(t)$ and all of its translates $\mathbf{s}_\tau$. For example, if $s(t)$ is bandlimited to $|f| \leq B$, then $\mathbf{V}$ is the space of functions likewise bandlimited. The symbol $\mathcal{L}$ indicates the one-dimensional subspace spanned by $\mathbf{s}_{\tau_*}$.] In actuality, we do not know and generally cannot specify the specific noise sample function $n(t)$. But we can choose a specific $s(t)$, a test signal,

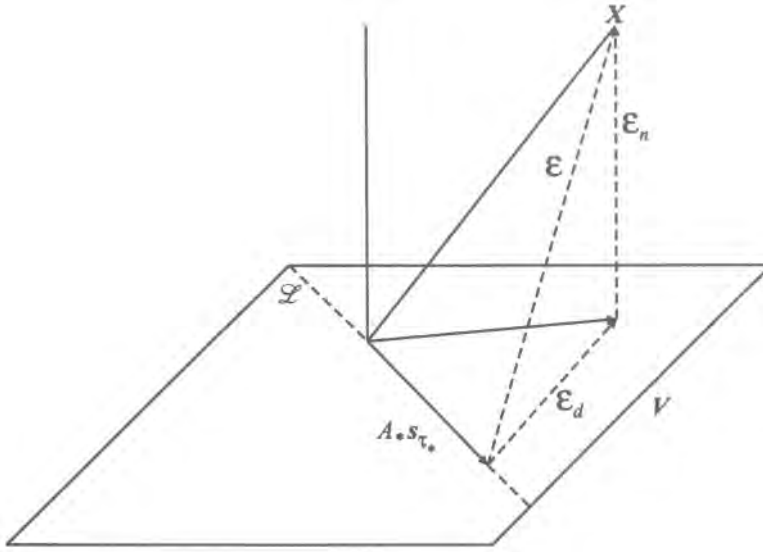


Figure 11.2. Geometric interpretation of SNR. Function-space orthogonal decompositions of system output X into signal $A \cdot s_{\tau_*}$, distortion ϵ_d , and random noise ϵ_n components: $\mathcal{L}$ is the linear span of s_{τ_*} in the space V of square-integrable functions orthogonal to ϵ_n .

and in the following we assume that $s(t)$ is given. There are good reasons for this, which will be discussed shortly. Thus, we concentrate now on

$$X(t) = G[s(t), N(t)] \quad (11.1.11)$$

If we were to repeat the previous measurement with a different $n(t)$, we would generally obtain a different τ_* and SNR. It is natural to seek a definition for the “true” SNR that is independent of the measurement. In order to do this, we also need to think of τ_* as having a true value. It is also natural to suppose that the true SNR and delay manifest themselves as $T \rightarrow \infty$. Now consider the counterpart of (11.1.7) when $X(t)$ is given by (11.1.11),

$$U(\tau, A) = A^2 \langle s_{\tau}^2 \rangle - 2AR_{Xs}(\tau) \quad (11.1.12)$$

that is, instead of the number R_{Xs} we have the random variable R_{Xs} , which we call the *semiempirical cross-correlation* function. We can expect as T becomes large that $\langle s_{\tau}^2 \rangle \rightarrow P(s)$, the true average power of $s(t)$. Hence, (11.1.12) will be minimized if R_{Xs} is maximized. We define the true value of τ_* , τ_m , to be that which maximizes R_{Xs} , the expected value of R_{Xs} as T becomes indefinitely large, and the true SNR to be that determined by R_{Xs} . Thus, we have

$$R_{Xs}(\tau_m) = \lim_{T \rightarrow \infty} E[R_{Xs}(\tau_m, T)] \quad (11.1.13)$$

where τ_m is the value that maximizes the right-hand side and E is expectation with respect to the noise ensemble.

It can now be seen why it is useful to condition the experiment on the signal, because otherwise we cannot ensure that τ_m will maximize the expected value of the semiempirical

cross-correlation for arbitrary $s(t)$. Of course, we could define the delay as that which maximizes (11.1.13) averaged further over an ensemble of S . However, the fundamental notion in SNR is waveform fidelity, and it therefore seems meaningful to condition all operations on a given input signal.

11.1.2. Form of the Estimator

The SNR estimator $\hat{\rho}$ is of the same form as (11.1.10), but with $R_{\mathbf{x}s}$ instead of $R_{\mathbf{x}}$ in (11.1.9). Making the appropriate substitutions, we obtain

$$\hat{\rho} = \frac{R_{\mathbf{x}s}^2(\tau)}{\langle \mathbf{X}^2 \rangle \langle \mathbf{s}_\tau^2 \rangle - R_{\mathbf{x}s}^2(\tau)} \quad (11.1.14)$$

where $\tau = \tau_m$. In practice, we approximate τ_m by τ_0 , which might be obtained by analysis or measurement. But the statistical properties of (11.1.14) are independent in form of the specific value of τ . With

$$R_0 = R_{\mathbf{x}s}(\tau) / (\langle \mathbf{X}^2 \rangle \langle \mathbf{s}_\tau^2 \rangle)^{1/2}$$

we can express $\hat{\rho}$ as

$$\hat{\rho} = R_0^2 / (1 - R_0^2) \quad (11.1.15)$$

Notice in this form the potential for numerical difficulty as $\hat{\rho}$ reaches high values. Then, R_0 is close to (but less than) unity, and a small error (be it observational or computational) will be magnified. For example, let $R_0 = 0.99$, which yields $\hat{\rho} = 99$. For a 1% error in R_0 , say $R_0 = 0.99/1.01$, $\hat{\rho} = 49.5$, a 50% error. While this error corresponds to 3 dB, which is not a greatly significant error in an analog SNR, the example does indicate that suitable care must be exercised to minimize errors in the measurement.

■ *Remark on the Form of $\hat{\rho}$.* The SNR estimator derived above is a least-squares estimator. Under the condition that the system is simply an unknown gain and delay, Tranter *et al.*^(2,3) showed that the maximum-likelihood (ML) estimator is precisely of the same form. This lends us comfort that the estimator is indeed a good one. For more general channels, however, there is not an ML estimator for our definition of SNR. ■

11.1.3. Statistical Properties of the Estimator

For arbitrary G it is extremely difficult to obtain any properties of $\hat{\rho}$. However, we can make some progress if we assume G is a linear operation such that the output noise samples are independent, but can otherwise be quite general. Under these conditions (and also making the nonlimiting normalization $\langle \mathbf{s}_\tau^2 \rangle = 1$), we can show that $\hat{\rho}$ is the ratio of two independent noncentral chi-square variates:

$$\hat{\rho} = \chi_1^2(\lambda_1) / \chi_{N-1}^2(\lambda_2) \quad (11.1.16)$$

The numerator is chi-square with 1 degree of freedom (d.f.) and noncentrality parameter $\lambda_1 = (N/\sigma^2)R_{\mathbf{ss}_0}^2$, where $R_{\mathbf{ss}_0}$ is the cross-correlation of the output signal and the τ -delayed

input signal. The denominator is chi-square with $N - 1$ d.f. and $\lambda_2 = (N/\sigma^2)(R_{s_0 s_0} - R_{s_0}^2)$, where $R_{s_0 s_0}$ is the output signal's autocorrelation function at zero delay. The parameter σ^2 is the variance of the noise samples. The distribution of $\hat{\rho}$ is referred to as a doubly noncentral F distribution. It is a rather complex distribution, but it turns out that we can get good approximations to the first two moments if we use the singly noncentral F approximation.⁽⁴⁾ Invoking this approximation, we arrive at

$$E(\hat{\rho}) = \frac{N^2(A - 1/N)(\gamma + 1/N)}{N^2(A - 1/N)^2 - 2N(B - 1/N)} \quad (11.1.17)$$

for the expected value of $\hat{\rho}$, and for sufficiently large N we have

$$\sigma^2(\hat{\rho}) = 2\gamma/N(\Delta + 1)^2 \quad (11.1.18)$$

for the variance, where $A = \Delta + 1$, $B = 2\Delta + 1$, $\Delta = \lambda_2/N$, and $\gamma = \lambda_1/N$. It is of interest to look at two limiting conditions, $\Delta \rightarrow 0$ and $N \rightarrow \infty$. (The conditions $\Delta = 0$ implies a distortionless system.) In the first instance we get

$$\lim_{\Delta \rightarrow 0} E(\hat{\rho}) = \frac{N}{N - 3} \left(\gamma + \frac{1}{N} \right)$$

which agrees precisely with Ref. 5 which applies under these conditions, and in that case γ is the true value of SNR. The variance expression, under the same conditions, also agrees with Ref. 5. But under these conditions, $\hat{\rho}$ is a singly noncentral F variate. For this case, confidence intervals have been computed both in Refs. 2 and 3 and in Ref. 5 to which the reader is referred, and these intervals ought to be good guides for the case in which $\Delta \neq 0$. Now, if we let N be very large, we get

$$\lim_{N \rightarrow \infty} E(\hat{\rho}) = \frac{\gamma}{1 + \Delta} \triangleq \rho_{\text{true}}$$

which we *define* to be the “true” value of SNR. This can be given an interpretation as signal power divided by noise power plus distortion power. Therefore, we can rewrite (11.1.18) as

$$\sigma^2(\hat{\rho}) = \frac{2}{N(1 + \Delta)} \rho_{\text{true}} \quad (11.1.19)$$

This means that for sufficiently large N , the $\pm K\sigma$ band about the expected value is of the form

$$\rho_{\text{true}} \left\{ 1 \pm K \left[\frac{2}{N(\Delta + 1)\rho_{\text{true}}} \right]^{1/2} \right\}$$

so that a given accuracy requirement is satisfied if $N\rho_{\text{true}}$ is a constant. It can be seen that for any value of ρ corresponding to high-fidelity systems and for large values of N , excellent estimation results. But recall from (11.1.15) that this is precisely the situation where high

numerical accuracy is required. For detection problems where the signal is “buried” in noise there may be some possibility of N being large enough to be fairly time-consuming.

11.1.4. Implementing The Estimator

There are two approaches to computing an estimate of the SNR, the time-domain and the frequency-domain approaches. We have indicated that the abstract setting is most meaningful if we specify a particular signal. Additionally, in a given simulation run, we do draw a specific realization of the noise. Thus, the experiment is best thought of as implementing the minimization of (11.1.7). The main task associated with that is the determination of the best delay τ_* . This, and the computation of the SNR estimate, can be done with the arrangement in Figure 11.3, which is the time-domain implementation. To be practical, the range of τ explored must be moderately small. In practice, it is usually straightforward to establish the range of delay to be investigated (see the related discussion in Sections 8.12 and 10.6). In Figure 11.3, we have labeled this range $\tau_{\max}$. Of course, we must also discretize τ , which means the true τ_* may not be observed. But if the increments are small enough, interpolation should yield close to the true result. Figure 11.3 implies that we need a record of $s(t)$ that is longer than that of $x(t)$ by the length of delays tried, that is, by $\tau_{\max}$, as illustrated in Figure 11.4. This is because in (11.1.8b) the delay variable τ appears in the argument of s . This is an arbitrary choice: since delay is relative, it could just as well have appeared in the argument of x . Whether we have a longer record of s or of x is ultimately immaterial, but the details of the implementation are dependent on the specific choice.

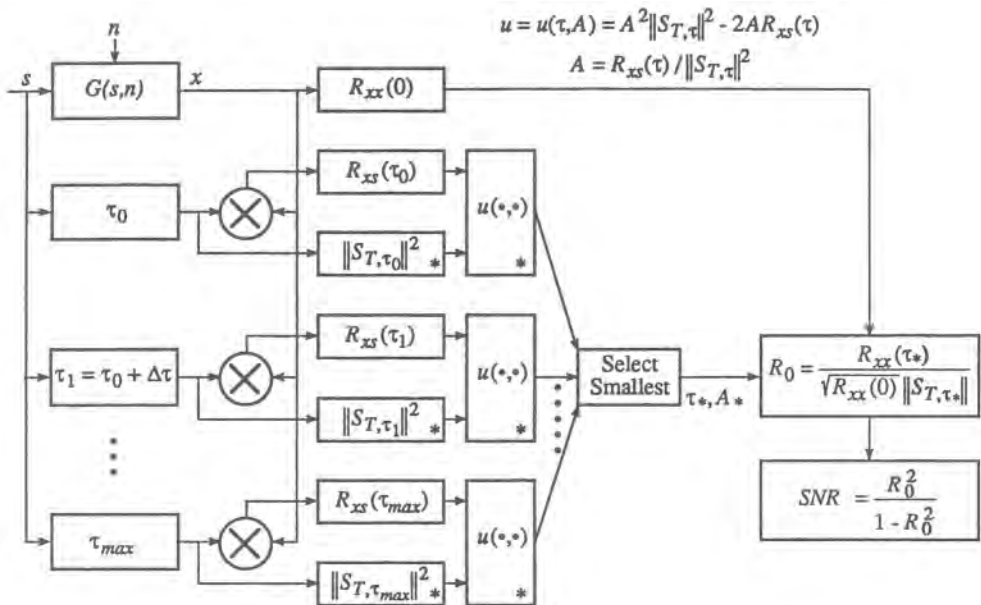


Figure 11.3. Configuration for obtaining delay parameter and SNR computation (from Ref. 1, ©IEEE, 1989). Blocks marked with asterisks are omitted in the reduced complexity algorithm, and “Select Smallest” becomes “Select Largest.” Under some circumstances it is practical and meaningful to perform the delay measurement with the noise set to zero.

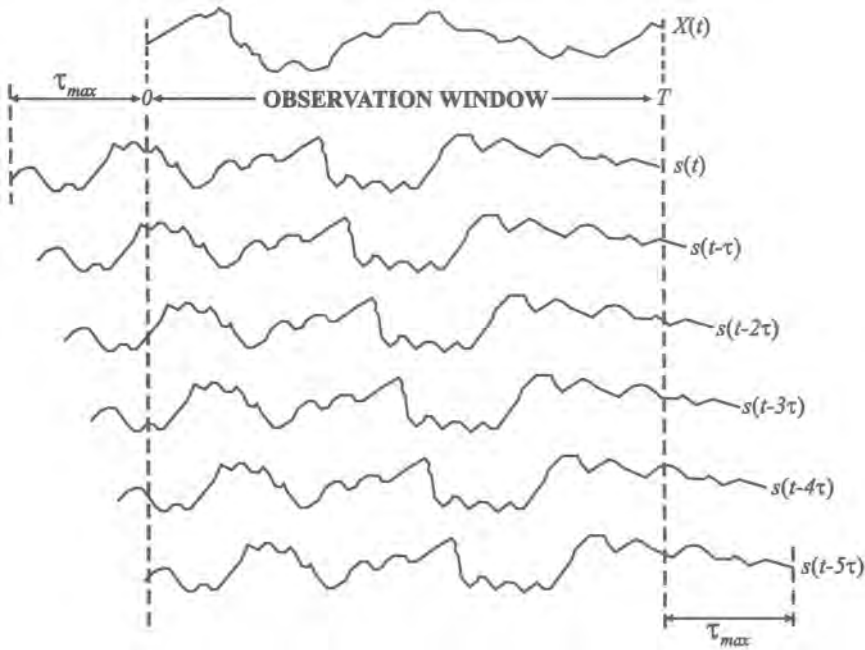


Figure 11.4. Illustration of put of the cross-correlation process.

The measurement procedure would be simplified if $\langle s^2 \rangle$ were the same for all τ . We remarked earlier that this becomes true as $T \rightarrow \infty$. But we can usually assume T large enough that $\langle s^2 \rangle$ is very nearly independent of τ . This will also be true if $s(t)$ is periodic and if T is a multiple of the period. In either case, the desired minimization occurs when the empirical cross-correlation function is maximized. Then, a simplification is possible, as indicated in the caption of Figure 11.3; that is, we can dispense with all but the cross-correlators and replace the “smallest” with a “largest” selector. Based on the latter simplification, the time-domain estimation procedure is summarized below.

■ Summary of the Time-Domain Simulation Procedure for Estimating SNR:

1. Run the simulation and obtain M samples of the output of the system $\{x(k)\}$, $k = 1, 2, \dots, M$, and $N = M + K$ samples of the input signal $\{s(k)\}$, $k = -(K - 1), -(K - 2), \dots, 0, 1, \dots, M$. The additional K samples of the input signal should be selected so as to span the expected largest delay in the system $\tau_{\max}$, and the empirical cross-correlation function always contains N points.

2. Compute the empirical cross-correlation

$$R_{\mathbf{x}\mathbf{s}}(j) = \frac{1}{N} \sum_{k=1}^N x(k)s(k-j), \quad j = -(K-1), \dots, 0$$

and find the value of j, j_* , that yields the maximum magnitude of $R_{\mathbf{x}\mathbf{s}}(j)$.

3. Compute

$$\begin{aligned} \langle \mathbf{X}^2 \rangle &= \frac{1}{N} \sum_{k=1}^N x^2(k) \\ \langle \mathbf{s}^2 \rangle &= \frac{1}{N} \sum_{k=1}^N s^2(k-j_*) \end{aligned}$$

4. Calculate $\hat{\rho}$, the estimate of SNR, from

$$\hat{\rho} = \frac{R_{\mathbf{x}\mathbf{s}}(j_*)}{\sqrt{\langle \mathbf{X}^2 \rangle \langle \mathbf{s}^2 \rangle - R_{\mathbf{x}\mathbf{x}}^2(j_*)}} \quad \blacksquare$$

From Figure 11.3, the above procedure is naturally viewed as a parallel one. But in simulation, i.e., in the computer, it will be implemented sequentially, a potentially lengthy process depending on the granularity used for dividing $\tau_{\max}$. A typically more efficient equivalent procedure can be carried out in the “frequency domain,” that is, by taking advantage of the FFT algorithm. Let $\{\mathbf{x}_n\}$, $\{\mathbf{s}_n\}$ represent sequences of sampled values in time and let $\{X_m\}$, $\{S_m\}$ be the corresponding DFTs.[†] As in Chapter 3, set $W_N = \exp(-j2\pi/N)$. Then, from Parseval’s theorem (Ref. 6 and Chapter 3), we have

$$\sum_{n=0}^{N-1} x_n^* s_n = \frac{1}{N} \sum_{m=0}^{N-1} X_m^* S_m$$

and from the shifting property $s_{n+k} \rightarrow S_m W_N^{-mk}$, we get

$$\sum_{n=0}^{N-1} x_n s_{n+k} = \frac{1}{N} \sum_{m=0}^{N-1} X_m^* S_m W_N^{-mk} = \text{IDFT}(X_m^* S_m) \quad (11.1.20)$$

so that the value of k that maximizes the cross-correlation, which corresponds to determining the optimum delay (quantized to the simulation sampling interval), is also the value that maximizes the right-hand side of (11.1.20). But the right-hand side is the inverse DFT (IDFT)[‡] of the sequence $\{X_m^* S_m\}$. Hence, we do not need to perform N convolutions, as the left-hand side might suggest, but a single FFT, and simply select that k for which the FFT coefficient is greatest in magnitude. The latter statement is also conditioned on the same assumption made earlier, namely that the average input power for a given number of samples is essentially independent of the specific samples. A summary of the corresponding simulation procedure is now given, and also illustrated in Figure 11.5

■ Summary of the Frequency-Domain Simulation Procedure for Estimating SNR:

1. As in the time-domain procedure, run the simulation and obtain M samples of the output of the system $\{x(k)\}$, $k = 1, 2, \dots, N$, and $N = M + K$ samples of the input signal $\{s(k)\}$, $k = -(K - 1), -(K - 2), \dots, 0, 1, \dots, N$. The additional K samples again span the maximum expected delay. Figure 11.6 shows the relationship of these sequences for this purpose.

2. Perform the inverse DFT (FFT) on the sequence $\{X_m^* S_m\}$, where $\{X_m^*\}$ is the conjugate of the DFT of $\{x_m\}$ and $\{S_m\}$ is the DFT of the sequence $\{s_m\}$. Note that the $\{x_m\}$ sequence is first padded with K zeros. The coefficient number k whose magnitude is greatest corresponds to the delay sought. Note that only the first $K + 1$ coefficients need be searched.

3. Evaluate Equation (11.1.14) with τ set to that found in the previous step, and, of course, $R_{\mathbf{x}\mathbf{s}}$ interpreted as $R_{\mathbf{x}\mathbf{s}}$. ■

When N is very large the FFT may not be possible, or may be undesirable because of excessive roundoff. In this case, the sequences $\{\mathbf{x}_n\}$ and $\{\mathbf{s}_n\}$ have to be segmented, as

[†]Here, X stand for the DFT of the time-domain sequence and should not be confused with its use to denote a random variable or process, as, for example, in (11.1.1).

[‡]As hinted earlier, we may take here either the DFT or the IDFT, depending on whether the longer record is of the input or the output sequence and which of these sequences is conjugated in Parseval’s theorem.

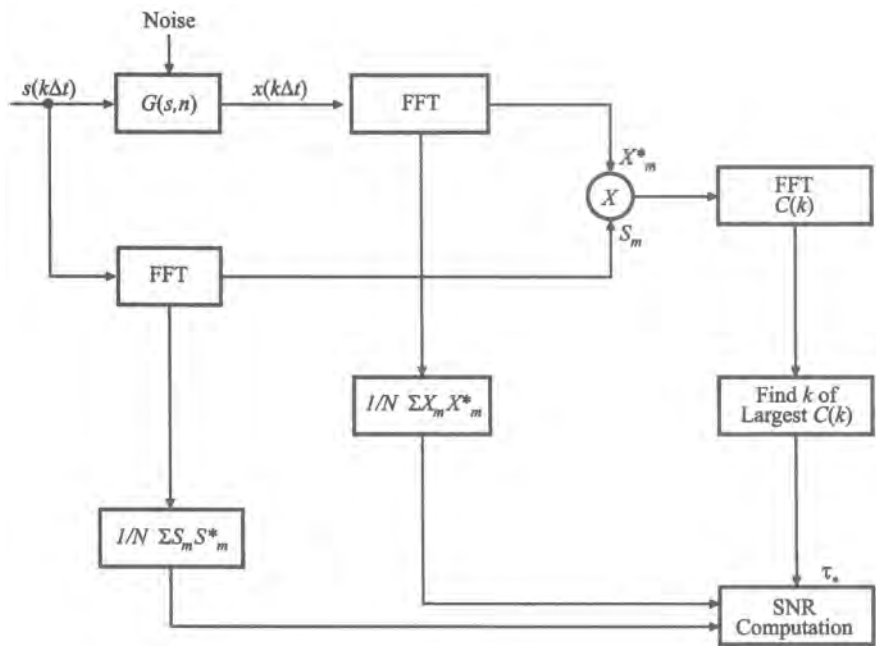


Figure 11.5. Frequency-domain implementation of SNR estimation. Note the assumption that $\sum S_m S_m^* \approx (s^2(t - \tau_s))$.

suggested by Tranter *et al.*,^(2,3) and the DFT evaluated piecewise. The reader is referred to Refs. 2 and 3 for the details of implementation. However, in today’s computing environment, where FFT sizes of the order of, say, 250,000 (but generally a power of 2) are easily computed, the segmentation just mentioned will probably not often be required.

11.2. Estimating Performance Measures for Digital Systems

The estimation of performance measures for digital systems within the simulation context has many aspects that depend in part on the particulars of the modulation and coding

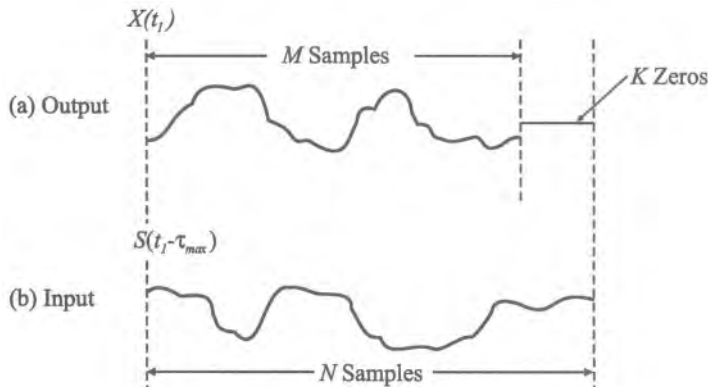


Figure 11.6. Selection of samples for frequency-domain calculation of SNR; the zeroth sample is $S(t_1 - \tau_{\max})$ for the input and $X(t_1)$ for the output; t_1 is arbitrary.

schemes that are used, the propagation channel and system characteristics, and the nature of the performance measure itself. In any given situation, one set of performance evaluation techniques (PETs) or another may be the most appropriate tool for extracting an estimate of the relevant performance measure. We have previously alluded (in Chapter 2, in particular) to the point that, in the present context, alternatives to the Monte Carlo methodology can be much more economical in terms of run time, and in certain cases may become indispensable.

In this section we shall discuss in some detail the various simulation alternatives to the Monte Carlo method, as well as Monte Carlo itself, for the purpose of estimating digital performance measures. First, in the next subsection, we will outline the nature of several such measures, and illustrate why alternatives may be needed. In the subsequent subsection, we will try to place all of the estimating techniques within a relatively unified context that will clarify their connection. The ensuing subsections will then consider in turn four general approaches to the problem. The last section of this chapter will then summarize and provide some guidance to the practitioner.

11.2.1. Performance Characterization for Digital Systems and Run-Time Implications

The performance of digital transmission systems is invariably described or specified in probabilistic terms related to the occurrence of errors. The specific form of the performance measure that is used, or may be appropriate, depends on a number of factors, most importantly the nature of the application and the standards that may have been adopted in specific types of services. Thus, depending on context, one will encounter such measures as bit error rate (BER),[†] symbol error probability (or symbol error rate, SER), word error rate, probability of an errored second (ES), probability of a severely errored second (SES), frame error rate, and outage probability. The meaning of many of these terms is self-evident, but will be explained more fully as needed.

As will be demonstrated shortly, in many cases it will be computationally prohibitive to estimate the above measures in a literal sense, that is, by actually generating, observing, and counting the error events of interest. This “actual” generation and counting of error events is the essence of the Monte Carlo method in this context. This computational burden has been the impetus for developing the alternatives to MC alluded to above, in order to estimate digital performance measures in a practical amount of simulation time. As we will see, these alternatives typically involve a combination of Monte Carlo methodology and analytical thinking. One particular methodology that employs such a combination was introduced in Chapter 2 as *quasianalytical* (QA; some authors use *semianalytic*) technique, although there is not merely one such technique. The applicable or best combination of simulation and analysis is generally problem dependent. QA techniques will be discussed in detail in Section 11.2.6. We should point out that in employing the analytical tools in the QA techniques, certain auxiliary assumptions may be required or implicit (such as independence of error occurrences), but such assumptions may be testable in the simulation itself or could also be possibly unverifiable because of computational limitations. These issues will also be discussed later.

In order to appreciate the run-time implications, it is necessary to provide a basis for calibrating the required run time in actual time units. Clearly, a proper basis for this purpose is

[†]Some authors prefer bit error probability, or use BER for bit error ratio. We consider these terms synonymous for the current discussion. For stylistic variation we will use one or the other term without intending any change in meaning.

the computation time per symbol C_s , since the total run time T_{run} is simply $T_{\text{run}} = N_s \times C_s$, where N_s is the number of symbols in the run. To be sure, C_s depends on the specific problem as well as on the specific machine running the simulation; and since machine capabilities are continually improving, we can only give a snapshot based on current technology. In any case, a rough estimate suffices for present purposes. The problem for which we have extensive run-time data is a two-hop satellite link, the simulation block diagram for which is shown in Figure 11.7. As discussed in Chapter 2, run time also depends on the complexity of models. Therefore, it is also appropriate to give a brief description of the nature of the models for the blocks of Figure 11.7.

- *Simulation sampling rate*: the sampling rate is 16 samples/bit.
- *Source*: The data source is a “random” bit generator (not a PN sequence) which is made up of a uniform RNG implemented from a linear congruential algorithm of period $2^{31} - 1$; these generators are discussed in Chapter 7.
- *Modulator*: The modulator is an MSK modulator, whose block diagram was given in Figure 8.32 of Chapter 8. The impairments discussed in Chapter 8 for quadrature modulators are incorporated here.

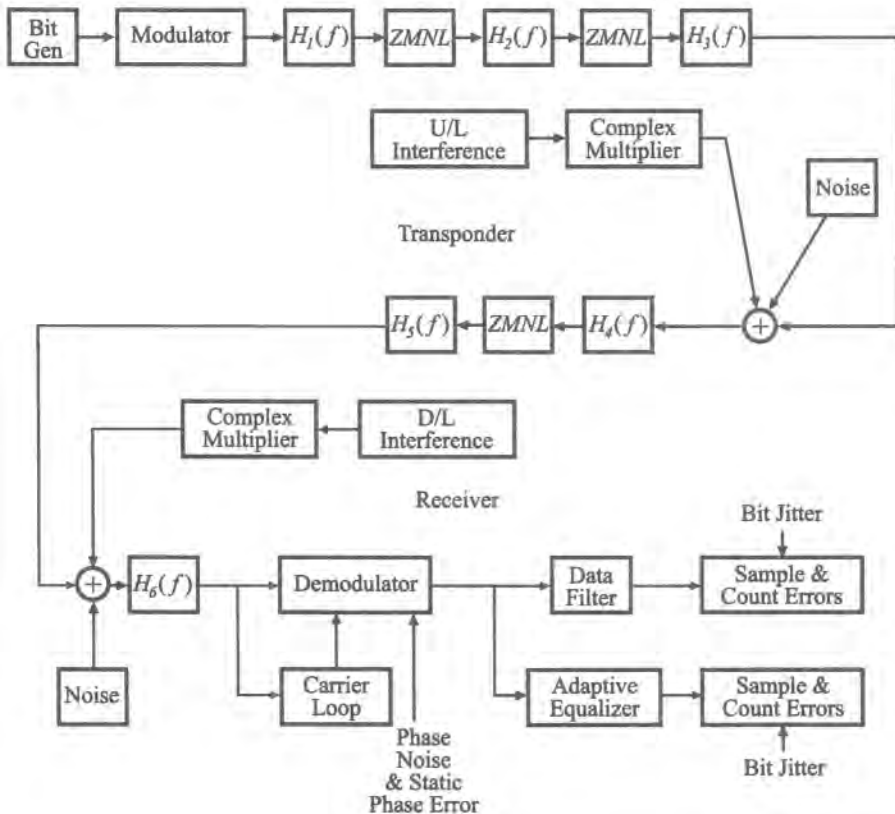


Figure 11.7. System block diagram for calibrating run time.

- *Transmitter model:* The transmitter, from the output of the modulator to the output of the antenna, is modeled as shown, by two nonlinearities sandwiched by three filters labeled $H_1(f)$, $H_2(f)$, and $H_3(f)$, respectively. This sequence of models represents a saturating upconverter and a power amplifier (PA) also operated at saturation, in accordance with one of the modeling paradigms outlined in Chapter 5. In addition, $H_2(f)$ includes any actual filtering that occurs after the upconverter but before the PA. Similarly, $H_3(f)$ incorporates any post-PA filtering. The three filters are implemented as tabular filters, as defined in Chapter 8. Each filter is defined by 409 measured amplitude/phase pairs which are then interpolated by a second-order formula to 8192 complex points for purposes of FFT processing. As just implied, the filter output is obtained via the FFT, and a seamless output sequence is produced via the overlap-save procedure. The filter impulse responses here, as well as those mentioned below, have different durations. A preliminary procedure sends an impulse through the system to determine the overall impulse response; this response is examined to assess an effective collective memory. In the problem at hand, this was determined to be about 75 symbols, and this was used as the value of overlap. Thus, the FFT processing efficiency is $(512 - 75)/512 \simeq 85\%$. The boxes labeled “nonlinearity” are described below.
- *Transponder model:* The transponder, from the input to the receiving antenna to the output of the transmitting antenna, is modeled as a memoryless nonlinearity sandwiched by two filters, $H_4(f)$ and $H_5(f)$. The nonlinearity model is described below. As with the transmitter, the two filters here are also tabular filters implemented in the same general way, but in this case the number of measured amplitude/phase pairs is 809 points. $H_4(f)$ also includes all filtering effects up to the power amplifier, which is a TWT, and $H_5(f)$ includes all filtering after the PA.
- *Nonlinearities:* The nonlinear behavior of the transmit amplifiers, both uplink and downlink, is implemented according to the “serial” memoryless model in Chapter 5. That is, the model consists of a pair of cascaded memoryless transfer characteristics: AM/PM followed by AM/AM. Each of these “curves” consists of a lookup table of 150 points derived from experimental data, which is accessed sample by sample during the simulation. A second-order interpolation in the power domain is used. Memory effects are incorporated in the filters on either side. The saturating upconverter is modeled similarly.
- *Receiver model:* The receiver model, from the input of the antenna to the input of the demodulator, is modeled as a single transfer function $H_6(f)$. It is of the same nature as the other filters, but the number of measured points here is 407. Other functions housed in the ground station are described below.
- *Atmospheric and antenna models:* The atmosphere is not an all-pass filter, as discussed in Chapter 9. This is also true of an antenna, the gain of which is proportional to f^2 . These effects are taken into consideration here, but do not add to the computational complexity because they are incorporated into the receive transfer functions on the uplink and downlink, i.e., $H_4(f)$ and $H_5(f)$, respectively.
- *Data filter:* The filter labeled such in the receiver acts as a matched filter. It is a two-pole Butterworth filter implemented as a recursive difference equation via the bilinear z -transform.
- *Initializing synchronization:* The carrier synchronizer as well as the symbol (bit) synchronizer are initialized according to the cross-correlation procedure described in Chapter 10.

- *Carrier synchronization*: The carrier synchronization loop is a form of the Costas loop (see Chapter 8). It is operated “on-line,” i.e., is in continuous operation during the simulation. For the particular timing benchmark given below, multirate processing was not utilized. Although downsampling here would be appropriate, it is not clear what the run-time savings would be.
- *Bit synchronization*: For the case discussed, an explicit bit synchronizer structure was not used. Instead, as discussed in Chapter 8, a sampling comb having 20 times, with 0.025-bit separation, was centered on the initializing value. Sampling jitter is added via a random noise generator.
- *Equalizer implementation*: A seven-tap “complex” equalizer (see the discussion in Section 8.9, Chapter 8) driven by a zero-forcing algorithm is explicitly simulated.
- *Interference*: As seen from Figure 11.7, the presence of interference is taken into account. In order to lend reality to the model, both the uplink and downlink interferences (one in each direction) are processed through similar chains as the wanted signal under study. Of course, this adds a not insignificant burden to the computation.
- *BER estimation*: The simulation is run Monte Carlo fashion. Two BERs are obtained, as indicated in Figure 11.7, one with and one without the equalizer, in order to test the equalizer’s effectiveness. Little penalty is paid for this information since most of the processing has already taken place.

The system and its models, as just described, require approximately 0.02 s/symbol (or, in this case, 0.01 s/bit, since two bits per symbol are sent) of processing time on a Sun Spare 10 workstation. To see what that implies in terms of T_{run} , we need to return briefly to the performance measures themselves. We shall not look at all the measures previously mentioned, as the point can be made with just two.

First we consider SER (which for the binary case is BER). Let N be the number of transmitted symbols and $n(N)$ be the corresponding number of observed errors. Then the simulation’s SER estimate is $\hat{p} = n(N)/N$. It will be shown in Section 11.2.3 that the number of symbols to be observed in Monte Carlo simulation, i.e., N , can be phrased as K/p , where p is the true symbol error probability and K is a constant, typically in the range 10–100, that determines the reliability of the estimate. Suppose that $p = 10^{-5}$. Then N is in the range 10^6 – 10^7 . At the computational rate of 0.02 s/symbols, such a MC simulation would take 5–50 hr. It is clear, therefore, that direct estimation of very low $\text{ER}^\dagger$ (say 10^{-9}) for systems of similar complexity is out of the question for virtually any computer (5000 hr in this case!). This provides one justification for seeking alternatives to Monte Carlo.

Another example of the limitations of MC simulation lies in the estimation of the measure p_{ES} , the probability of an errored second. An errored second is simply a second within which there is at least one error. Let R be the data rate on some link; the units of R of course are simply symbols/s, so R is exactly the number of symbols transmitted in 1 s. This means that to simulate 1 s of system behavior, the simulation must process R symbols. The observation rule mentioned above applies here also, where now one observation refers to 1 s. So, if we want p_{ES} to be 10^{-3} , say, we need to observe on the order of 10^4 – 10^5 s of real time. If R is in the multimegabit range, we can see from the previous example that the run time

[†]In order to avoid undue repetition as to the type of error that we may be dealing with, we will sometimes refer simply to the error rate (ER) or to the average error rate (AER). A relevant point about time per run is that in the normal course of system design, typically a large number of runs has to be made. Thus, many hours per run is tolerable if few runs have to be made, but much less so for many iterations.

required will also be impossibly long. Notice that here the computational burden is not directly related to the HER itself, but is essentially controlled by the number of bits per second in the real system. In such situations the alternatives to Monte Carlo are fewer than in the BER estimation case, and are of a different nature.

11.2.2. A Conceptual Framework for Performance Estimation

It is useful in the overall understanding of performance evaluation to try to place the various techniques within a common conceptual framework. Although this framework is not the most general, it does provide helpful insight. To begin with, let us describe the “experiment,” which we have stated a Monte Carlo simulation to be, within the classical probabilistic context. This will be useful in placing the techniques to be discussed in the proper perspective. We begin by considering the standard case of symbol-by-symbol detection (hard-decision). These symbols may be uncoded or may be the output of an encoder.

We visualize an ensemble of identical systems, the input to each of which is a distinct joint realization of signal and noise sources. The signal sequences are driven by the same clock, that is, zero crossings “line up,” but the sequences are otherwise independently generated. For stationary noise sources, this means the ensemble is cyclostationary. In each system there is a waveform input to the decision device/decoder. This set of waveforms also constitutes an ensemble. The output of each decision device/decoder is a digital sequence which, in general, will contain some errors. This set of output sequences constitutes yet another ensemble. Assume the ensemble is indexed by a continuous parameter ω . For the ω th member of the ensemble, we define the error rate as $P_\omega = \lim_{N \rightarrow \infty} n(\omega, N)/N$, where a particular sampling epoch τ is implicit. Now, let $e(\tau, \omega)$ be the error indicator for the ω th member at sampling epoch τ . That is, if the symbol at τ is in error, $e(\tau, \omega) = 1$, and if correct, $e(\tau, \omega) = 0$. We define the ensemble error rate as $p = E_\omega[e(\tau, \omega)]$, which for a cyclostationary ensemble will depend on τ only and not on t . If, for almost all ω , $P_\omega = p$, the ensemble has a cycloergodic property with respect to error rate. Clearly, not all P_ω will be identical since there are realizations of the input sequence that are not “typical,” e.g., the sequence of all ones. It is important to keep the distinction between P_ω and p in mind because in the Monte Carlo simulation context we do deal with (a portion of) a particular realization of the ensemble and the performance derived therefrom can be assumed to be representative of the ensemble if some care is taken to avoid “pathological” sequences. In practice, this is not difficult to achieve. On the other hand, the quasianalytical techniques discussed later are more properly thought of as estimates of ensemble properties.

The setting within which p is defined is identical to the classical Bernoulli trials with probability of “success” p , or the drawing of balls of one of two colors from an urn with respective proportions p and $1 - p$. The experiment in time on any member of the ensemble can also be placed in this classical setting if the errors are independently produced. In fact, the equivalence of P_ω and p does not depend on the independence of errors: this equivalence speaks only of averages. An MC simulation of a sample path can, of course, also extract information regarding the distribution of errors in time. However, the assumption of independence simplifies the calculation of other measures, as will be discussed later.

To place the preceding model in a somewhat more quantitative perspective, consider (without loss of generality) a binary baseband communication system whose receiver is shown in Figure 11.8a. The decision process can be described in terms of the pdf $f_0(v; \tau)$ and $f_1(v; \tau)$ of the voltage $V(t)$ input to the decision device at the sampling epoch τ , given that a zero or a one was sent, respectively. These densities are ensemble properties and are

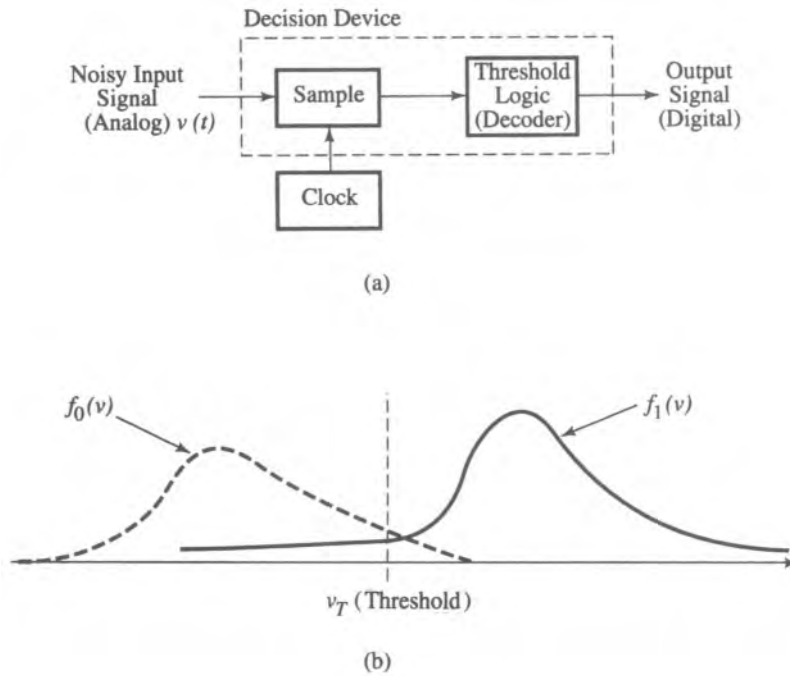


Figure 11.8. Illustration of some basic terms of reference in the discussion of BER estimation (from Ref. 19, © IEEE, 1984). (a) Typical decision mechanism in digital transmission. (b) Hypothetical probability density functions.

cyclostationary with period of one symbol duration. The corresponding densities measured on a “typical” sample path will be identical. Observe that, as illustrated in Figure 11.8b, the density for zeros need not be the same as for ones. For hard-decisioning, an error occurs when the threshold is crossed in the “wrong” direction; that is,

$$P[\text{error}|\text{one}] \triangleq p_1(\tau) = \int_{-\infty}^{V_T} f_1(v; \tau) dv = F_1(V_T; \tau) \quad (11.2.1a)$$

$$P[\text{error}|\text{zero}] \triangleq p_0(\tau) = \int_{V_T}^{\infty} f_0(v; \tau) dv = 1 - F_0(V_T; \tau) \quad (11.2.1b)$$

and the average probability of error is then

$$p(\tau) = \pi_1 p_1(\tau) + \pi_0 p_0(\tau) \quad (11.2.1c)$$

where π_1 and π_0 are the *a priori* probabilities of the two symbols. Observe that the error probabilities are functions of the sampling epoch, but for notational simplicity that dependence may sometimes be suppressed, though understood. The functions $F_1(\cdot)$ and $F_0(\cdot)$ are the CDFs corresponding to $f_1(\cdot)$ and $f_0(\cdot)$, respectively, and clearly contain all the necessary information. In fact, although we are here discussing hard decisions, it is clear that the densities or distributions in question also contain unquantized “soft” information. The density functions are evidently sufficient, but not necessary for every approach to estimation. However, in some cases that may be the simplest approach to the problem. Generally, one may

make assumptions on, or deal directly with, either the CDF or the pdf, depending upon the estimation technique. In either case, it is, of course, only the “tails” of these functions in which we are interested. In some cases we may for convenience assume a functional form *in toto*, even though only the tails are of interest, while in others it is only the tails for which the expressions apply.

As just implied, estimation techniques can be classified, at least in part, according to the properties of the CDF or the pdf, such properties becoming known in possibly different manners, e.g., measured or assumed. What are the fundamental implications, from the user’s point of view, behind the different alternatives available? Basically, the main implication is that each technique effectively offers a tradeoff between generality and the time reliability product ζ . It will be recalled that ζ itself embodies a tradeoff between the number of observations N and a measure of estimator reliability, e.g., the variance $\sigma^2(\hat{p})$. Since the Monte Carlo method makes no *a priori* assumption (or as little as possible), it is the most general of the techniques. In essence, it supplies an empirical determination of the distribution functions evaluated at a point. As was seen in the previous section, the cost of the generality is run time. The other techniques to be discussed are in some sense variance-reducing, each resting on some set of assumptions that make them less general or less easy to apply than MC. [Techniques that attempt to improve the efficiency of estimators are often called variance-reduction techniques (VRT) and are well known in the statistical literature.⁽⁷⁾ While the term VRT is apt in the present context, there is little overlap between the techniques considered here and those in Ref. 7.] A second implication of the differences among estimation techniques from a user’s point of view is that different simulation/software structures are necessary to implement them. It will usually be in the user’s interest to have a simulation package that has more than one of these techniques as options. We should also point out that BER estimation techniques have been developed in the context of real-time or on-line applications, where efficiency may or may not also be an issue (the term often used in this context is *BER monitoring*). Techniques for such a “live” environment and those for simulation can overlap, but some techniques may only be usable in one or the other situation; see, e.g., Refs. 8–11 for more discussion of BER monitoring.

The measures we have been discussing so far, SER or BER, are examples of one-dimensional measures, that is, they depend only on first-order distributions. Many of the measures listed previously are multidimensional in nature, that is, they involve joint distributions at multiple epochs. Again, under certain conditions of ergodicity, we can expect that the corresponding quantities will be replicated almost surely by making measurements on a single sample path. For example, if we wanted to find the joint probability of an error at some epoch τ and at another time removed by kT , given “ones” at both times, we would be interested in the second-order conditional density $f_{1,1}(v; \tau, kT + \tau)$ of the voltage at two such times. These higher order distributions become increasingly more demanding in terms of measured data as the order and the separation increase. For many applications, a less complete, but still very useful characterization is given by the temporal distribution of errors. Such information is also a fairly natural output of an MC simulation. We thus suppose that hard decisions are made and that at every sampling epoch $\tau + kT$ we find an error indicator $e(\tau + kT)$ which is 1 if an error is made and 0 if no error is made. The sequence $\{e_\tau\} \triangleq \{e(\tau + kT), k = 0, \pm 1, \pm 2, \dots\}$ will be called the *error sequence*, and contains all possible information regarding the temporal distribution of errors; for low error rates, it will consist mostly of zeros. We will discuss more fully the ways in which an error sequence can be described in Section 11.2.3.7, but for now let us look at one possibility, which is to segment the sequence into contiguous blocks of length n , say, and take as a characterization

the measure $P(m, n)$, the probability of m errors in a block of n symbols. Such a description is clearly of value if we are dealing with a block error-correcting code with codeword length n . If the code has a guaranteed error-correcting capability of t errors, we obviously wish to know $P(m, n)$ for $m > t$, or, equivalently, $P(m, n)$ for $m \leq t$. Thus, we are again interested in the “tail” of a distribution, as we were in the one-dimensional case. Of course, the nature of the variable is not the same, but here we wish only to draw a rough analogy.

11.2.3. The Monte Carlo Method

As we have said earlier, an ideal Monte Carlo simulation is an *as-is* emulation of the real system. Putting aside issues related to the fidelity of the models, the core idea in an MC simulation is that random processes, signals and noises, evolve in time bearing whatever statistical properties are ascribed to them. In relation to the BER or SER estimation problem, this means that Monte Carlo is merely a label for the implementation of a sequence of Bernoulli trials, except that in implementing the simulation the assumption of independent draws need not be made. The implication of such an experiment is that we count the number of “successes” (errors in this context) and divide by the number of trials,⁽¹²⁾ the result being an estimate of the average relative number of errors, or simply what we call the BER or SER. The method itself requires no assumptions about the system properties. In fact, in the implementation of the method, the system itself is bypassed (Figure 11.9) except, of course, for the output of the digital source and the decision device’s version of that source. Since the source output is known, comparing the two sequences at some relative delay provides the empirical basis for an error rate.

In order to implement the procedure, it is necessary to know the delay in question. This knowledge is, in effect, equivalent to symbol synchronization, and it should be kept in mind that different techniques will extract different sampling epochs. Thus, to the extent that the *delay* in Figure 11.9 may not be unique, the MC estimate will also not be unique. But, of course, this is saying nothing more than that the behavior of the system is dependent on all aspects of its implementation. From the point of view of the estimation technique itself, however, these details are generally not relevant.

The Monte Carlo method is essentially *defined* by the above procedure. However, it is instructive to “derive” the method in a way that explicitly displays the pdf of the decision variable. We noted previously that the pdf, hence the probability of error, could be different for different symbols. Hence, it is necessary in general to condition the BER upon a specific

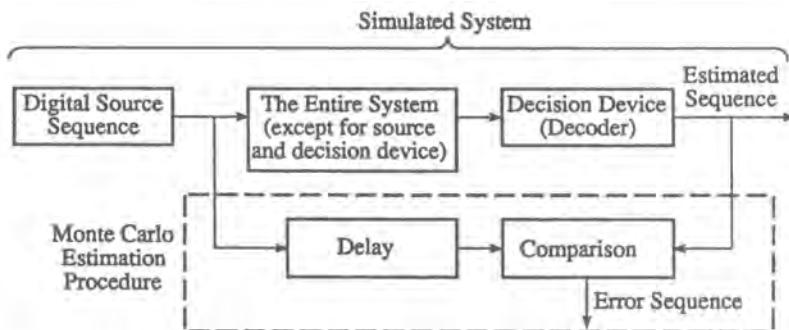


Figure 11.9. Schematic representation of the implementation of the Monte Carlo procedure.

type of transmitted symbol. Without loss of generality we shall concentrate on an arbitrary symbol, say symbol $\mathbf{a}_j$, $j \in \{0, 1, \dots, L-1\}$, and unless otherwise specified, all quantities that depend on a particular symbol may do so without explicit notation. It is important to understand, however, that the conditioning applies to the symbol upon which a decision is made, but does not constrain the symbol sequence preceding or following that symbol.

The formulation is identical *in form* for all symbols, whether or not their error probabilities are the same. Thus, the SER (BER) p_j , for the assumed symbol can be written as

$$p_j = \int_{v \in \mathcal{R}_j} f_{v_j}(v) dv \quad (11.2.2)$$

where f_{v_j} is the pdf of sampled values of symbol j at epoch τ (which we will leave implicit for now) and $\mathcal{R}_j$ is the region of v that corresponds to an error. By defining the error indicator function

$$I_{\mathcal{R}_j}(v) = \begin{cases} 1, & v \in \mathcal{R}_j \\ 0, & v \notin \mathcal{R}_j \end{cases} \quad (11.2.3)$$

we can recast (11.2.2) as

$$p_j = \int_{-\infty}^{\infty} I_{\mathcal{R}_j}(v) f_{v_j}(v) dv \quad (11.2.4)$$

The last equation is equivalent to the statement

$$p_j = E[I_{\mathcal{R}_j}(V)] \quad (11.2.5)$$

where E is the expectation operator, and since a natural estimator $\hat{p}_j$ of the expectation is the sample mean, we have

$$\hat{p}_j = \frac{1}{N_j} \sum_{i \in \mathcal{J}_j} I_{\mathcal{R}_j}(V_i) \quad (11.2.6)$$

where $V_i \triangleq V(t_i)$ is the sequence of symbol-spaced samples of the decision voltage, $\mathcal{J}_j$ is an integer set that contains all i such that t_i corresponds to sampling symbol j , and N_j is the number of elements in the set. We can see that $I_{\mathcal{R}_j}(V_i)$ acts as an error detector, the summation is an error counter, and $1/N_j$ is an averager. Thus, we have formally arrived at the desired result in a way that recognizes the role of the pdf of the observable, $V(t)$.

It is clear that (11.2.2) can be extended to the overall average SER by the equation

$$p = \sum_{j=0}^{L-1} \pi_j \int_{-\infty}^{\infty} I_{\mathcal{R}_j}(v) f_{V_j}(v) dv \quad (11.2.7)$$

which has the empirical counterpart

$$\begin{aligned}\hat{p} &= \frac{1}{N} \sum_{i=1}^N \sum_{j=0}^{L-1} I_{\mathbf{a}_j}(V_i) \mathbf{1}_{\mathbf{a}_j}(s_i) \\ &= n(N)/N\end{aligned}\quad (11.2.8)$$

where $s_1, s_2, \dots$, is the actual sequence of transmitted symbols, N is the total number of symbols processed, and n is the total number of errors observed. The symbol indicator $\mathbf{1}_{\mathbf{a}_j}(s_i)$ is equal to 1 if the i th transmitted symbol is $\mathbf{a}_j$ and zero otherwise. As $N \rightarrow \infty$, $\hat{p}$ converges to p almost surely by the strong law of large numbers (see, e.g., Ref. 13). Thus, the summand is 1 or 0, and that sequence has the same meaning as $\mathbf{e}_\tau$ defined earlier.

Note that in (11.2.8) the *a priori* probabilities of the symbols π_j are implicit, because in an MC simulation these probabilities manifest themselves naturally in the relative appearance of the symbols when the simulation is long enough. That is, they are well approximated by N_j/N .

We can also compute other related quantities of interest with a little more work. Let $\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{L-1}$ again denote the L symbols in the alphabet and let d_{jk} be the Hamming distance between $\mathbf{a}_j$ and $\mathbf{a}_k$. The estimate of the average bit error probability, i.e., the BER, is then simply

$$\hat{p}_b = \frac{1}{mN} \sum_{i=1}^N \sum_{j=0}^{L-1} I_{\mathbf{a}_j}(V_i) \mathbf{1}_{\mathbf{a}_j}(s_i) d_{jk} \quad (11.2.9)$$

where $\mathbf{a}_k$ is the detected symbol and m is the number of bits per symbol (assuming, of course, $L = 2^m$). Equation (11.2.9) simply says to weight each symbol error by the actual number of bits in error. We can go still further and calculate the specific bit error probability, which is of interest in certain block-coded modulation schemes (see Chapter 8 and Section 11.2.6). If $\mathbf{a}_k = (b_{k,m-1}, b_{k,m-2}, \dots, b_{k,0})$ is the binary representation of $\mathbf{a}_k$, we call $b_{k,l}$ the l th significant bit; in particular, $b_{k,0}$ is called the least significant bit and $b_{k,m-1}$ the most significant bit. Depending upon the bit-to-symbol mapping, the BER can be different in different bit positions. Let $d_{jk}^{(l)}$ be the Hamming distance between the l th significant bits of $\mathbf{s}_j$ and $\mathbf{s}_k$; of course, it is either 0 or 1. Then the specific BER is the obvious modification of (11.2.9)

$$\hat{p}_{b,l} = \frac{1}{N} \sum_{i=1}^N \sum_{j=0}^{L-1} I_{\mathbf{a}_j}(V_i) \mathbf{1}_{\mathbf{a}_j}(s_i) d_{jk}^{(l)} \quad (11.2.10)$$

In the next subsections we look into the reliability of these estimates.

■ *Remark.* So far, we have been speaking about *average* error rates, and while such a computation can be done irrespective of the system being studied, an average error rate is operationally most meaningful when the signal and noise power levels are time-invariant. For time-varying or fading channels, other measures, some of which were mentioned earlier, will be more meaningful. We will discuss these measures further in Section 11.2.3.7. ■

11.2.3.1. Confidence Interval: Binomial Distribution

The Monte Carlo method is one of the few cases where we can get an exact estimator distribution, assuming independence between error events. An error event can be an individual bit error, a symbol error, or any suitably defined event. The main point here is that observations can be divided into two classes, one which contains error events, and the other of

which does not. In other words, here we do not distinguish between errors on one type of symbol as opposed to another. With the independence assumption, the number of error events n is binomially distributed, $B(N, p)$, where p is the *a priori* probability of the event and therefore, for given N , the estimate $\hat{p} = n/N$ has a scaled binomial distribution. Recalling the definition of confidence interval from Section 10.1.3.3, it can be shown⁽¹⁴⁾ that for a $1 - \alpha$ confidence level, the upper and lower values of the symmetric two-sided interval, h_1 and h_2 , are the respective solutions of

$$\sum_{k=0}^n \binom{N}{k} h_1^k (1 - h_1)^{N-k} = 1 - F(h_1; n + 1, N - n) = \alpha/2 \quad (11.2.11a)$$

and

$$\sum_{k=n}^N \binom{N}{k} h_2^k (1 - h_2)^{N-k} = F(h_2; n, N - n + 1) = \alpha/2 \quad (11.2.11b)$$

where $F(x; a, b)$ is given by

$$F(x; a, b) = \frac{(a + b - 1)!}{(a - 1)!(b - 1)!} \int_0^x t^{a-1} (1 - t)^{b-1} dt \quad (11.2.12)$$

for a, b integers. The right-hand side of (11.2.12) represents $P(Y \leq x)$, where Y is a beta-distributed random variable, $Y \sim \text{beta}(a, b)$. This form is generally much easier to evaluate numerically than are the sums in (11.2.11).

For a one-sided interval, (11.2.11a) would be used with twice the value of α . The right-hand sides of (11.2.11) can also be different, say α_1 and α_2 such that $\alpha_1 + \alpha_2 = \alpha$, and this asymmetry can reduce slightly the length of the confidence interval. At one time, solving (11.2.11) would have been computationally tedious for the typical values of N and p encountered. However, progress in mathematical software now allows straightforward evaluation via the cumulative beta distribution in (11.2.12). Table 11.1 shows upper and lower multipliers on the observed $\hat{p}$ to construct a confidence interval at the indicated confidence level.[†] It should be noted from the above equations that the confidence interval depends explicitly upon N , but in the process of constructing Table 11.1, it was observed that the confidence intervals depended negligibly on N (i.e., differed only in the third or fourth decimal place) once N exceeded about 1000. Hence, a more compact table, as is shown, could be constructed.

Although the need for approximations to solving (11.2.11) is less compelling than it once was, they are still of interest for two reasons: if one is interested in a parameter range not covered by Table 11.1 and without the appropriate tool, these approximations will permit rapid and reasonably accurate results; second, these approximations, especially the normal one, provide an easier analytical visualization than do Equations (11.2.11).

If we were interested in constructing a confidence interval for the error rate of a particular symbol, it might seem that the preceding formulation would apply simply by replacing, say, p by p_j , $\hat{p}$ by $\hat{p}_j$, n by n_j , and N by N_j . In probably most cases of interest, namely, large N , the computation based on that assumption would turn out to be very close to the correct value.

[†]Table 11.1 was created by using Mathematica™ 3.0, in particular the “inverse regularized beta” function.

Table 11.1. Exact Confidence Intervals for BER Based on Binomial Distribution^a

Number of errors	90% Confidence	95% Confidence	99% Confidence
0	2.995	3.69	5.30
	0	0	0
1	4.74	5.57	7.43
	5.13×10^{-2}	2.532×10^{-2}	5.01×10^{-3}
2	3.147	3.61	4.64
	1.777×10^{-1}	1.211×10^{-1}	5.17×10^{-2}
3	2.584	2.922	3.66
	2.726×10^{-1}	2.062×10^{-1}	1.126×10^{-1}
4	2.288	2.560	3.147
	3.42×10^{-1}	2.725×10^{-1}	1.681×10^{-1}
5	2.102	2.333	2.829
	3.94×10^{-1}	3.25×10^{-1}	2.156×10^{-1}
10	1.696	1.838	2.139
	5.43×10^{-1}	4.80×10^{-1}	3.72×10^{-1}
20	1.452	1.544	1.732
	6.63×10^{-1}	6.11×10^{-1}	5.18×10^{-1}
30	1.356	1.427	1.572
	7.20×10^{-1}	6.75×10^{-1}	5.93×10^{-1}
40	1.301	1.361	1.483
	7.55×10^{-1}	7.15×10^{-1}	6.40×10^{-1}
50	1.265	1.317	1.424
	7.80×10^{-1}	7.43×10^{-1}	6.74×10^{-1}
60	1.239	1.286	1.382
	7.98×10^{-1}	7.64×10^{-1}	6.99×10^{-1}
70	1.219	1.262	1.349
	8.12×10^{-1}	7.80×10^{-1}	7.20×10^{-1}
80	1.203	1.243	1.324
	8.24×10^{-1}	7.94×10^{-1}	7.36×10^{-1}
90	1.191	1.228	1.303
	8.34×10^{-1}	8.05×10^{-1}	7.50×10^{-1}
100	1.180	1.215	1.286
	8.42×10^{-1}	8.14×10^{-1}	7.62×10^{-1}
500	1.075	1.089	1.118
	9.29×10^{-1}	9.16×10^{-1}	8.91×10^{-1}
1000	1.051	1.060	1.080
	9.51×10^{-1}	9.42×10^{-1}	9.24×10^{-1}

^a In the case of zero errors, the upper bound corresponds to a multiplicative factor of $1/N$. Results apply for a total number of observations in excess of 10^3 .

But, in fact, this procedure would not be quite correct because $\mathbf{n}_j$ is not binomially distributed. It is *conditionally* binomial, given a value of $\mathbf{N}_j$, while $\mathbf{N}_j$ itself is binomially distributed. It is not worthwhile pursuing this point here, but in Section 11.2.1.4 we shall obtain the mean and variance of $\hat{\mathbf{p}}_j$.

We should make an additional point about the randomness of $\mathbf{N}_j$. That randomness occurs because in an MC simulation, we are presumably imitating the system as it is, i.e., generating the symbols randomly. There is a variation of the MC methodology in which we can generate a fixed number of any given symbol, and this is when we can assume that the memory of the system is finite, say equal to m symbols. In that case there are L^{m-1} possible symbol patterns preceding each symbol and we can enforce that each of these patterns occurs

in conjunction with the chosen symbol the same number of times, for example by using a de Bruijn sequence, as discussed in Chapter 7.

11.2.3.2. Confidence Interval: Poisson Approximation

It is well known (e.g., Refs. 15 and 16) that the binomial distribution can be approximated by the Poisson distribution if $\hat{p} \rightarrow 0$ as $N \rightarrow \infty$ such that

$$\lim_{N \rightarrow \infty} N\hat{p} = \lambda$$

where $\lambda > 0$ is a constant. Under the limiting condition, the binomial can be replaced by the Poisson distribution, which leads to the relations

$$\sum_{k=0}^n e^{-\lambda_1} \frac{\lambda_1^k}{k!} = \frac{\alpha}{2}, \quad \sum_{k=n}^N e^{-\lambda_2} \frac{\lambda_2^k}{k!} = \frac{\alpha}{2} \quad (11.2.13)$$

as the equations to be solved for λ_1 and λ_2 from which we construct the confidence interval $(\lambda_1/N, \lambda_2/N)$ with confidence level $1 - \alpha$. The utility of (11.2.13) is that tables of the Poisson distribution already exist covering a usefully large range of values.^(17,18) Heuristically, the approximation is useful because a wide range of values of $\hat{p}$ and N can be collapsed to a relatively small range of the combining parameter $\lambda = N\hat{p}$.

A table of upper and lower values of λ for different confidence levels is given in Table 11.2 for $0 \leq n \leq 50$. Recall that n is the number of observed errors. To construct a confidence interval for p , enter the row at the observed n and divide the corresponding values of λ by N , the total number of observations. The values of λ in the table were obtained by using the same value $(\alpha/2)$ on the right-hand side of the sums in (11.2.13).

In general, as with the binomial, we can get somewhat tighter confidence intervals if we use different values for the two summations, say α_1 and α_2 , such that $\alpha_1 + \alpha_2 = \alpha$. Typically, this refinement will also not lead to significant reduction in the interval length. Table 11.2 covers a usefully large range of n . Comparing Tables 11.1 and 11.2, it can be seen (after a trivial computation) that the agreement is quite good for most purposes. In fact, for $n \gtrsim 10$, the results will be seen to essentially coincide as well with those from the normal approximation, discussed next.

11.2.3.3. Confidence Interval: Normal Approximation

It is also known that for large N the distribution of the estimator $\hat{p}$ is well approximated by a normal distribution with mean p and variance $p(1 - p)/N$. Hence, one can construct a confidence interval in the form^(13,14)

$$\begin{aligned} P \left\{ \frac{N}{N + d_\alpha^2} \left[\hat{p} + \frac{d_\alpha^2}{2N} - d_\alpha \left(\frac{\hat{p}(1 - \hat{p})}{N} + \left(\frac{d_\alpha}{2N} \right)^2 \right)^{1/2} \right] \right. \\ \left. \leq p \leq \frac{N}{N + d_\alpha^2} \left[\hat{p} + \frac{d_\alpha^2}{2N} + d_\alpha \left(\frac{\hat{p}(1 - \hat{p})}{N} + \left(\frac{d_\alpha}{2N} \right)^2 \right)^{1/2} \right] \right\} = 1 - \alpha \end{aligned} \quad (11.2.14)$$

Table 11.2 Confidence Intervals for BER Based on Poisson Approximation

n	$1 - \alpha = 0.90$		$1 - \alpha = 0.95$		$1 - \alpha = 0.99$	
	Lower	Upper	Lower	Upper	Lower	Upper
0	0.000	3.00	0.000	3.69	0.000	5.30
1	0.051	4.74	0.025	5.57	0.005	7.43
2	0.355	6.30	0.242	7.22	0.103	9.27
3	0.818	7.75	0.619	8.77	0.338	10.98
4	1.37	9.15	1.09	10.24	0.672	12.59
5	1.97	10.51	1.62	11.67	1.08	14.15
6	2.61	11.84	2.20	13.06	1.54	15.66
7	3.29	13.15	2.81	14.42	2.04	17.13
8	3.98	14.43	3.45	15.76	2.57	18.58
9	4.70	15.71	4.12	17.08	3.13	20.00
10	5.53	16.96	4.80	18.39	3.72	21.40
11	6.17	18.21	5.49	19.68	4.32	22.78
12	6.92	19.44	6.20	20.96	4.94	24.14
13	7.69	20.67	6.92	22.23	5.58	25.50
14	8.46	21.89	7.65	23.49	6.23	26.84
15	9.25	23.10	8.40	24.74	6.89	28.16
16	10.04	24.30	9.15	25.98	7.57	29.48
17	10.83	25.50	9.90	27.22	8.25	30.79
18	11.63	26.69	10.67	28.45	8.94	32.09
19	12.44	27.88	11.44	29.67	9.64	33.38
20	13.25	29.06	12.22	30.89	10.35	34.67
25	17.38	34.92	16.18	36.90	14.00	41.00
30	21.59	40.69	20.24	42.83	17.77	47.21
35	25.87	46.40	24.38	48.68	21.64	53.32
40	30.20	52.07	28.58	54.47	25.59	59.36
45	34.56	57.69	32.82	60.21	29.60	65.34
50	38.96	63.29	37.11	65.92	33.66	71.27

where p is the true value of the error rate and d_α is chosen so that

$$\frac{1}{(2\pi)^{1/2}} \int_{-d_\alpha}^{d_\alpha} e^{-t^2/2} dt = 1 - \alpha \quad (11.2.15)$$

Heuristically, we can see that the normal approximation will be reasonably good if at least $p \geq d_\alpha[p(1-p)/N]^{1/2}$, which means the standard deviation is smaller than p by a factor of d_α , a small integer. This translates to the inequality $N_p \geq d_\alpha^2$.

As pointed out in Chapter 10, the absolute value of the confidence interval, which is given by (11.2.14), is not as informative as the interval length normalized to the BER itself, as we have already done in Table 11.1. Perhaps the greatest utility of the normal approximation, apart from the universal accessibility of tables of the normal distribution, is that it lends itself to a general graphical representation⁽¹⁹⁾ which is inherently normalized to the error rate. To construct such a display, set $\hat{p} = 10^{-v}$ and $N = \eta 10^v$. (Note that $\hat{p}$ cannot be zero; for that case use either Table 11.1 or Table 11.2.) The approximations $N/(N + d_\alpha^2) \approx 1$ and $\hat{p}(1 - \hat{p}) \approx \hat{p}$ are usually more than amply satisfied. We can then express (11.2.14) as

$$P[y_- \leq p \leq y_+] = 1 - \alpha \quad (11.2.16)$$

where the confidence interval (y_-, y_+) is given by

$$y_{\pm} = 10^{-v} \{1 + (d_x^2/2\eta)[1 \pm (4\eta/d_x^2 + 1)^{1/2}]\} \quad (11.2.17)$$

This interval is plotted in Figure 11.10 for 90%, 95%, and 99% confidence intervals. [Observe that the preceding approximation can be applied to the BER for particular symbols or the average for all symbols. The conditions for the approximation to be valid must apply to whatever the case may be.]

We should note that the confidence interval, by whatever method obtained, applies to a single point on an SER or BER curve, for example, that corresponding to a fixed SNR. Typically, in a simulation of a time-invariant system, for example, a number of error rate estimates are made at several values of SNR (or E_b/N_0), and a smooth curve is fit to these points. The totality of the points provide more information about the error rate at a given value of SNR than does the single observation taken at that SNR. Therefore, the best-fit curve should provide somewhat better confidence than that predicted for any individual point (see Problem P11.5).

■ *Example 11.2.1.* From Figure 11.10, one sees the source of the popular rule of thumb that N should be on the order of $10/p$ to $100/p$. For the former, one can see, for example, that a vertical slice at $N = 10^{v+1}$ produces a 95% confidence interval of about $(1.8\hat{p}, 0.55\hat{p})$, i.e., a factor of about 2 on the error rate scale, which is generally considered a reasonable

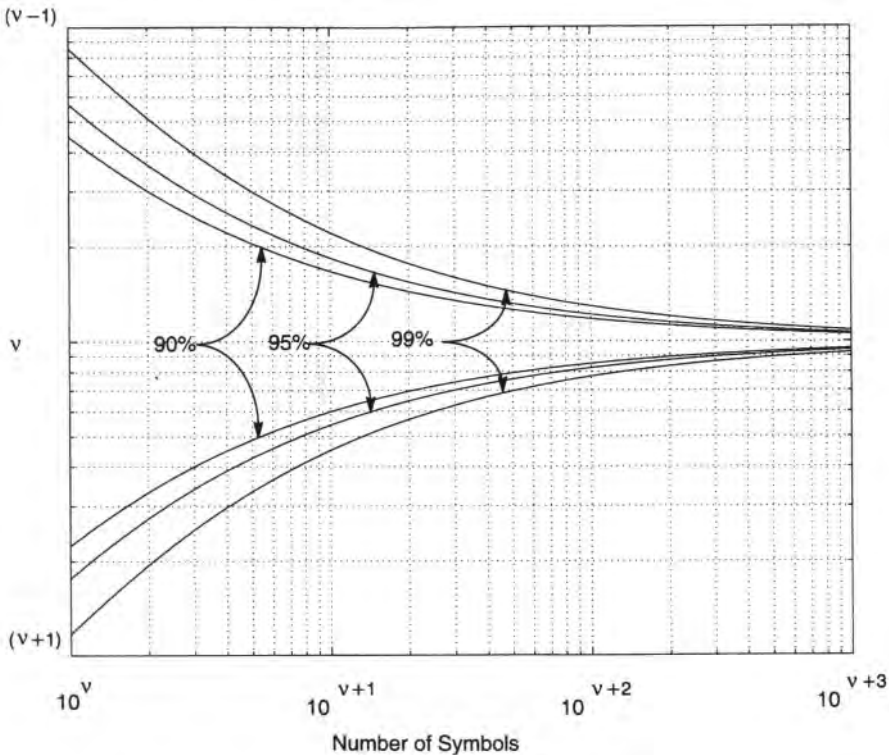


Figure 11.10. Confidence bands on BER when observed value is 10^{-v} for the Monte Carlo technique based on the normal approximation (from Ref. 19, © IEEE, 1984).

uncertainty. This corresponds to an uncertainty on the SNR scale amounting to perhaps a few tenths of a dB for mildly distorted systems, but possibly to larger amounts, depending of course on the actual system. If we increase N by a factor of 10, Figure 11.10 now gives a confidence interval of about $(1.25\hat{p}, 0.8\hat{p})$. As we expect, the confidence interval narrows relatively slowly since it decreases only as $1/\sqrt{N}$, as predicted by (11.2.14). Thus, the user is faced with a tradeoff between run time and statistical variability. ■

11.2.3.4. Mean and Variance of the Monte Carlo Estimator

It will be convenient here to change slightly the earlier notation for error sequence $\{\mathbf{e}_\tau\}$ to $\{\mathbf{e}_i\}$, where i now denotes the i th decision and the epoch τ is implicit. Thus, $\mathbf{e}_i = 1$ if an error occurs and $\mathbf{e}_i = 0$ otherwise. Then, the MC estimator takes the form

$$\hat{p} = \frac{1}{N} \sum_{i=1}^N \mathbf{e}_i \quad (11.2.18)$$

Since $E(\mathbf{e}_i) = p$, it is clear that $E(\hat{p}) = p$, hence the MC estimator is unbiased. The variance of $\hat{p}$ is

$$\sigma^2(\hat{p}) = E(\hat{p}^2) - p^2 \quad (11.2.19)$$

Since $E(\hat{p}) = p$, after some manipulation we can express $E(\hat{p}^2)$ as

$$E(\hat{p}^2) = p/N + (2/N^2) \sum_{i=1}^{N-1} \sum_{j=i+1}^N p \cdot p_{ij} \quad (11.2.20)$$

where

$$p_{ij} = P[\mathbf{e}_j = 1 | \mathbf{e}_i = 1, j > i] \quad (11.2.21)$$

For independent errors, $p_{ij} = p$ for all i, j . Since there are $(N^2 - N)/2$ terms in the double summation, they add to $(2/N^2)(N^2 - N)p^2/2 = (1 - 1/N)p^2$. Inserting this result into (11.2.20) and then (11.2.19) leads to (with $q = 1 - p$) the standard result

$$\sigma^2(\hat{p}) = \frac{p}{N} - \frac{p^2}{N} = \frac{1}{N} p(1 - p) = \frac{pq}{N} \quad (11.2.22)$$

If we wished to obtain the ER conditioned on a particular symbol, say symbol j , the estimator

$$\hat{p}_j = n_j/N_j \quad (11.2.23)$$

does not quite have the character of an average estimate because for a given N , N_j is random. It can be shown, however, that this estimator is still unbiased, i.e., $E(\hat{p}_j) = p_j$, and it is also straightforward to show (Problem P11.6) that the (conditional) variance is given by

$$\sigma^2(\hat{p}_j) = (p_j - p_j^2) E\left(\frac{1}{N_j}\right) \quad (11.2.24)$$

Actually, $E(1/N_j)$ does not exist unless we exclude $N_j = 0$, which is also sensible because otherwise we do not have an actual estimate (other than zero). Thus, if we do so, $E(1/N_j | N_j > 0)$ takes the appearance shown in Figure 11.11. It can be seen that even for moderate values of N , but greater than about 100, this function is well approximated by $1/\pi_j N$. This means that the estimator for p_j behaves essentially as an MC estimator.

The variance, or more precisely its square root, the standard deviation of the estimator, provides a compact way of judging the estimator reliability. As discussed in Chapter 10, a handy measure, in particular, is the normalized standard error, also called the relative precision

$$\epsilon(\hat{p}) \triangleq \sigma(\hat{p})/p = \sqrt{(1-p)/Np} \cong (Np)^{-1/2} \quad (11.2.25)$$

where the approximation $1-p \approx 1$ holds for almost any realistic case. We generally want ϵ to be a small number. Thus, for a given fractional error it can be seen that the observational requirement is given by

$$N = \frac{1}{\epsilon^2 p} \quad (11.2.26)$$

which gives an immediate sense of the number of symbols that have to be processed, and thus a corresponding sense of the computational demand.

■ *Remark.* The previous equations, such as (11.2.26), which depend on knowing the true error rate, obviously cannot be used in an actual case to estimate ϵ , since p is the quantity sought. In practice, an estimate of ϵ can be had by using the sample moments themselves. The crudest estimate would be to use $\hat{p}$ itself instead of p , or to estimate $\sigma^2(\hat{p})$ from the sample variance $s^2(\hat{p})$, as follows. Subdivide the error sequence into K subsequences $\{e_{jk}\}$, $j = 1, 2, \dots, J$ and $k = 1, 2, \dots, K$, such that $JK = N$. Define $\hat{p}_k = (1/J) \sum_{j=1}^J e_{jk}$ as the average error rate for the k th subsequence, and the overall average as $\bar{p} =$

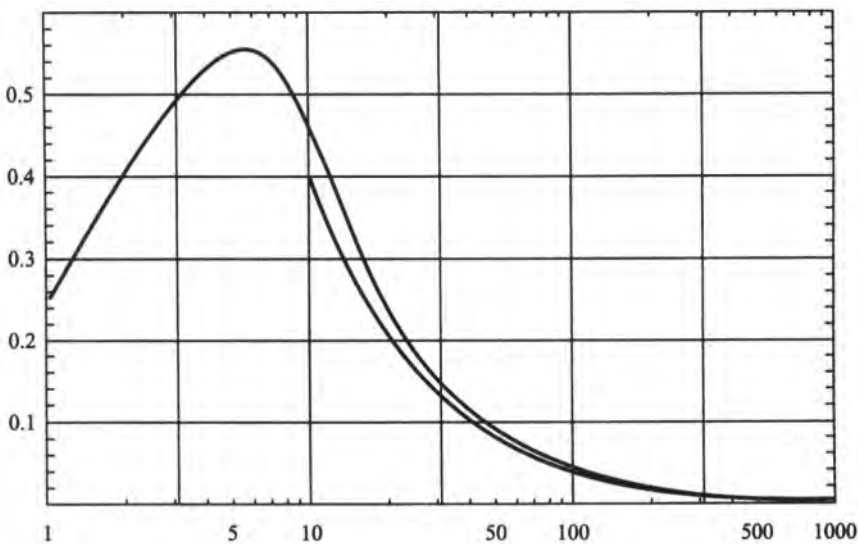


Figure 11.11. Values of $E(N_j^{-1} | N_j > 0)$ (upper curve) and $(\pi_j N)^{-1}$ (lower curve) as a function of N for $\pi_j = 0.25$.

$(1/K) \sum_{k=1}^K \hat{p}_k$. Then,

$$s^2(\hat{p}) = \frac{1}{K-1} \sum_{k=1}^K (\hat{p}_k - \bar{p})^2 \quad \blacksquare$$

11.2.3.5. Effect of Dependent Errors

It has been assumed thus far that errors are generated independently. This is a reasonable assumption in some cases, but not others. Dependence can be brought about by various causes, for example, filtering or fading. The form of the AER estimator $\hat{p}$ is still given by (11.2.8) or (11.2.18). Hence the estimator is always unbiased. The estimator distribution, however, is related to the manner in which the dependence manifests itself. Hence, one needs a specific model of the error dependence in order to compute a confidence interval or variance of the estimator. Such a model may simply be some hypothetical form for the correlation between errors^(19,20) derived from some form of finite-state machine.^(21,22) For present purposes we choose a simple model which gives the essence of the conclusion, namely, that for nonindependent errors more symbols need to be observed for a given confidence level. In the model⁽¹⁹⁾ below, we assume that an error event always consists of a burst of $m + 1$ errors (one simple example of such a case, for $m = 1$, occurs when the data are differentially encoded). Observe that if we subdivide the transmitted sequence into m -bit bytes, the preceding development applies where now each byte is an “observation” and byte errors are independent. Hence, for a given reliability, the simulation length measured in *bits* is now m times longer. Let us say now that we are interested in the *bit* error rate and seek to find the variance for the BER estimate. Equations (11.2.18)–(11.2.21) apply, and under the previous assumption we have

$$\begin{aligned} p_{ij} &= 1, & j &= i + 1, i + 2, \dots, i + m \\ p_{ij} &= p, & j &> i + m \quad (\text{independence}) \end{aligned} \quad (11.2.27)$$

In the double sum of (11.2.20) there are $Nm - m(m + 1)/2$ terms for which $j > i + m$. Using this fact and (11.2.27), we find for the variance of the estimator (Problem P 11.7)

$$\sigma^2(\hat{p}) = (pq/N)[1 + 2m - m(m + 1)/N] \quad (11.2.28)$$

and for the usual case $N \gg m$ this reduces to

$$\sigma^2(\hat{p}) \cong \frac{1}{N} pq(1 + 2m) \quad (11.2.29)$$

Comparing this last result to (11.2.22), we see that the variance is increased roughly by the factor $(1 + 2m)$, about as expected. The main message is that dependent errors provide less information, which is clear; hence, if one has a channel which is known or suspected of producing nonindependent errors, a given number of observations will not yield as tight a confidence interval. The simulation user should keep this in mind.

11.2.3.6. Sequential Estimation

In the preceding subsections, we have implicitly assumed that the simulation length is fixed, and equivalent to the observation of N symbols. The reliability of an estimator is not so much related to N itself as to the number of observed error events in the simulation run. Hence, an alternative simulation strategy would be to run the simulation for only as long as it takes to observe n errors, where n is predetermined. This is referred to as a *sequential* procedure,⁽²³⁾ which is governed by a *stopping rule*, the “rule” in this case being simply to stop only once the n th error has been seen. It is clear that n is no longer binomially distributed; indeed, it can only have the chosen value. Rather, it is N that is random. It can be shown⁽²³⁾ that an unbiased estimator of the ER, p , is given by

$$\hat{p} = \frac{n-1}{N-1} \quad (11.2.30)$$

and that the variance of that estimator is given by

$$\sigma^2(\hat{p}) = p^2 \sum_{j=1}^{\infty} \binom{n+j-1}{j} (1-p) \quad (11.2.31)$$

Generally, we are interested in relatively small error rates. A reasonable approximation should therefore result if we obtain the limiting value of (11.2.31) as $p \rightarrow 0$. However, as discussed previously, it is more meaningful to gauge variability relative to the true value. Thus, one can show (Problem P11.8) that (for $n > 2$)

$$\lim_{p \rightarrow 0} \sigma^2(\hat{p})/p^2 = 1/(n-2) \quad (11.2.32a)$$

or equivalently, in that limit, the standard error is

$$\varepsilon(\hat{p}) = 1/\sqrt{n-2} \quad (11.2.32b)$$

It can be seen that, for a large number of errors, ε is roughly equivalent in the sequential and MC cases, because in the latter instance, $\varepsilon = 1/\sqrt{Np}$ and for sufficiently large N , $Np \sim n$.

11.2.3.7. Estimation of Interval Measures

So far we have been discussing performance measures that are average event error rates. As we noted earlier, such average measures may not be very informative if the error rate measured over intervals that are operationally meaningful can vary greatly over different such intervals, as can be the case, for example, in a fading environment. Thus, for more general channels, descriptions of the error mechanisms over meaningful intervals are desirable. For obvious reasons, we shall refer to such measures as *interval* measures. The usual starting point for such measures is an error sequence $\{e_i\}$ that we already identified in Section 11.2.3.4, or more generally, an observation sequence (where the outcome is not necessarily two-valued), which we will denote by $\mathbf{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_N\}$, where each observation could be drawn from an outcome space $\mathbf{o} = (o_1, o_2, \dots, o_m)$. This space can be defined quite generally. It can refer, for example, to the error sequence already mentioned, to soft-decided detected voltages, to channel fade depths, and so on.

Given O , there are two broad approaches in using this information, labeled *generative* and *descriptive*.⁽²⁴⁾ The first of these, as might be inferred from the term, involves the construction of a model which can be used explicitly to generate sequences that are representative of the channel. There is generally not a unique correct model, but a multiplicity of possible such models that fit the observations more or less well in different ways. Most models are based on some sort of assumed Markov structure^(21,24,25) primarily because of the relative simplicity with which Markov models can represent dependence among events. Perhaps the most common such models are of the hidden Markov (HMM) type discussed in Chapter 9. Descriptive models are statistical in nature and involve constructing various types of statistics (distributions) from the observation sequence. The two approaches mentioned have some degree of overlap in that statistics can be derived in principle from a generative model, and a generative model can be extracted from the statistics. However, each tends to be better applied for different ends. We will now consider these approaches individually.

11.2.3.7.1. Using a Generative Model As discussed in Chapter 9, given an observation sequence, we can deduce a hidden Markov model that “best explains” what was seen. The usual application of such models is to generate “typical” error sequences that could be added to an information sequence and the result fed to downstream processing devices, for example, decoders. The utility of this modeling methodology is that we no longer have to deal with analog processes and the many samples per symbol thereby implied, nor with the burden of computing the effect of each device that processes these waveforms. (But in order to have such a model in hand, it may of course first be necessary to run a full-scale waveform simulation.) In principle, any interval statistic could then be derived since the HMM is putatively a complete description of the mechanism producing the observations. As an example, we show the connection between the HMM and $F(m, n)$, the probability of m errors or less in a block of n symbols. As indicated in Chapter 9, a K -state HMM is characterized in part by the vector of initial state **probabilities**[†]

$$\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_K)$$

and by a stationary state transition probability matrix $\mathbf{P}$,

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ \vdots & & & \vdots \\ p_{K1} & p_{K2} & \cdots & p_{KK} \end{pmatrix} \quad (11.2.33)$$

where p_{ij} is the probability of going to state j given that the the current state is state i . Each state will itself be described by an *outcome probability transition matrix* which represents probabilities associated with the outcomes in any given state. For present purposes it is convenient to introduce this matrix in a form which depends on the input symbol corresponding to the time of observation. Thus we define a matrix $\mathbf{B}_r(o_t)$ whose (k, l) element is defined by

$$b_{kr}(o_t) = \Pr(\mathcal{O}_t = o_t | X_t = k, A_t = a_r) \quad (11.2.34)$$

[†]The notation here is somewhat different, to fit the current context.

where X represents the state variable, $\mathcal{A} = \{a_1, \dots, a_L\}$ is the input symbol alphabet, and $\{A_t\}$ is the input sequence drawn from $\mathcal{A}$. We now construct a matrix $\mathbf{Q}_r(o_t)$ whose (i, j) element

$$\mathbf{Q}'_{ij}(o_t) = \Pr(X_{t+1} = j, \mathcal{O}_{t+1} = o_t | A_{t+1} = a_r, X_t = i) \quad (11.2.35)$$

is the probability that, given state i at time t , the process goes to state j at time $t + 1$, the input symbol is a_r , and the observation is o_t . From (11.2.34) and the definition of p_{ij} we have that

$$\mathbf{Q}_r(o_t) = \begin{pmatrix} p_{11}b_{1r}(o_t) & \cdots & p_{1K}b_{Kr}(o_t) \\ \vdots & & \vdots \\ p_{K1}b_{1r}(o_t) & \cdots & p_{KK}b_{Kr}(o_t) \end{pmatrix} \quad (11.2.36a)$$

or, equivalently,

$$\mathbf{Q}_r(o_t) = \sum_{i=1}^K \mathbf{P}\mathbf{E}_i b_i(o_t) \quad (11.2.36b)$$

where $\mathbf{E}_i = \xi_i' \xi_i$ and ξ_i is the i th basis vector of K components $(0, 0, \dots, 1, 0, \dots, 0)$ having 1 in the i th coordinate and 0's elsewhere. For a specific n -long observation sequence $\mathbf{O}$, we can write

$$\Pr(\mathbf{O} | A_1 = a_{r_1}, \dots, A_n = a_{r_n}) = \pi \prod_{i=1}^n \mathbf{Q}_{r_i}(\mathcal{O}_i) \mathbf{1}$$

where $\mathbf{1}$ is a column vector of K ones, and it is understood that $\mathcal{O}_i$ is a specified element of $\mathbf{O}$. As can be seen, the probability of a specified outcome sequence is in general dependent on the input sequence. For certain definitions of the outcome space, or under certain symmetry conditions, the input sequence dependence can vanish.

For example, consider $\mathbf{O}$ to be an error sequence for a binary symmetric channel (in each state) and define the weight of $\mathbf{O}$ as $|\mathbf{O}|$, namely, the number of errors observed. Then we have

$$F(m, n) = \sum_{|\mathbf{O}| \leq m} \pi \prod_{i=1}^n \mathbf{Q}(\mathcal{O}_i) \mathbf{1} \quad (11.2.37)$$

The practicability of evaluating (11.2.37) will depend on K , n , and m because $n + K$ matrix multiplications are indicated for each $\mathbf{O}$ satisfying the weight constraint and there are $\sum_{j=1}^m \binom{n}{j}$ such possibilities. For many cases of interest, direct evaluation of (11.2.37) will be prohibitively intensive, but for moderate values of n , say 100 or less, and for small values of t , say 3 or less, the computations need not be burdensome. It may be possible to derive more efficient forms for $P(m, n)$ or, equivalently, $F(m, n)$, at least in particular instances of HMMs. An example for the two-state Gilbert model will be given in Section 11.2.6. It should be pointed out, though, that it is difficult to quantify the accuracy of $F(m, n)$ (e.g., in terms of variance of the estimator) because the HMM is itself only an approximate description of the channel. Consequently, as mentioned above, it may be best to apply the HMM as a generative, rather than a descriptive tool.

11.2.3.7.2. Using a Descriptive Model A descriptive model involves, in essence, constructing empirical distributions or probability mass functions (pmf) from the given observation sequence. That, in itself, is straightforward, as it consists merely in assigning the observations to appropriate bins. Although the raw observations are made on the symbols themselves, the statistics to be collected are made on groupings of these. There are many ways to arrange this information, but here we shall confine ourselves to one quite general descriptor of the character of an observation sequence, namely, *multigap* distributions.⁽²⁴⁾ We will assume that the observation sequence is an error sequence. First we need to define an error gap. An error gap G of length r is an error sequence beginning with a 1 (error) followed by $r - 1$ consecutive 0's (no error) and then a 1. In other words, there are $r - 1$ consecutive 0's bracketed by two 1's. An observation sequence can be described as a concatenation of gaps. A multigap of order k , $\mathbf{G}^k$, is a sequence of k consecutive gaps. The corresponding probability mass function $g(r, K)$, $r \geq k$, is defined as the probability that the length of k consecutive gaps is equal to r . This pmf is deducible under some circumstances. For example, for a discrete memoryless channel with ER p , it is given by⁽²⁴⁾

$$g(r, k) = \binom{r-1}{r-k} p^k q^{r-k} \quad (11.2.38)$$

The mean and variance of this distribution are K/p and kq/p^2 , respectively. The single-gap pmf $g(r, 1) \equiv g(r)$ follows readily from (11.2.38), and is the well-known geometric distribution

$$g(r) = pq^{r-1}, \quad r \geq 1 \quad (11.2.39)$$

The single-gap distribution, if it were known, would give us, for example, p_{ES} , the probability of an errored second. That is, p_{ES} is simply $g(r)$ with $r = R + 1$, with R the data rate in bits/s.

The block error distribution $P(m, n)$ is related to the multigap distribution as follows⁽²⁴⁾:

$$P(m, n) = p \sum_{j=m-1}^{n-1} (n-j)[g(j, m-1) - 2g(j, m) + g(j, m+1)] \quad (11.2.40)$$

for $1 \leq m \leq n$, with the definition $g(j, 0) = \delta(j)$. Further, the block error probability for a t -error-correcting code, say, is given by

$$1 - F(t, n) = \sum_{m=t+1}^n P(m, n) = p \sum_{j=t}^{n-1} (n-j)[g(j, t) - g(j, t+1)] \quad (11.2.41)$$

In the two previous equations, p is the average error rate, which is also perhaps more fruitfully interpreted in this context as the reciprocal of the average single-error gap length.

Of course, as implied above, we can simply make a histogram of the number of errors in a block of n symbols from a long enough simulation, from which we could deduce the probability of exceeding t errors. Generally, it will be difficult, i.e., computationally demanding, to obtain a reliable estimate, the degree of difficulty depending on the target corrected error rate and the block length: the simulation length increases with block length and with decreasing error rate. Typically, coding is applied to produce relatively low error rate, say at least (no more than) somewhere in the range 10^{-5} – 10^{-6} . Following the

development in Section 11.2.3.4, for example, if we consider for the moment that block errors are independent (not true in perhaps most cases) and the block error probability is p_{block} , we have for the required number of errored blocks, in analogy with (11.2.26),

$$N_{\text{blocks}} = \frac{1}{\varepsilon^2 p_{\text{block}}} \quad (11.2.42)$$

and of course the number of processed *symbols* is n times longer than that. In many channels of interest, block errors cannot be assumed to occur independently. The effect on run time is analogous to that discussed in Section 11.2.3.5, that is, for a given reliability of the estimate, correlation among error events requires a greater number of observations. Thus, based on the timing example given earlier, it can be seen that the required run time can quickly become prohibitive.[†]

Because of the rather long simulation runs that would generally be required, it would be useful to infer analytically helpful conclusions from simulations of reasonable length. The most helpful conclusion would be that the channel behaves somewhat like an independent-error channel. For then any number of error statistics can be derived. Even if a channel is patently otherwise, some sense of the reasonableness of that assumption can be made by examination of the multigap distribution. In particular, the variance of G^k gives an indication of the degree to which we may be justified in assuming independence or a renewal property of the error sequence.⁽²⁴⁾ Another such indication would be given by the first-order gap pmf, which in the “ideal” case has a geometric distribution: a goodness-of-fit test (see Chapter 7 for a brief description of such tests in the context of RNGs) could also be used to ascertain how justified such an assumption might be.

11.2.3.7.3. Interval Simulation We mentioned above that a generative model could be used to produce a more or less typical observation sequence, which would be done simply by following the probabilistic prescription of the model for the state transitions and that for the outcomes. This method requires generating two random numbers per outcome. Descriptive statistics can also be used to generate typical sequences using a technique which we refer to as *interval simulation*, which is considerably more efficient than using the generative model. To illustrate the idea, let us begin with the simple case of a memoryless binary symmetric channel with crossover probability (BER) p . From (11.2.39), the single-gap distribution is geometrically distributed, namely,

$$g(r) = pq^{r-1}, \quad r \geq 1$$

From Chapter 7, we know that we can “simulate” a geometric random variable (in this case the gap r) from the formula

$$r = [1 + \log \xi / \log p] \quad (11.2.43)$$

where the square brackets indicate integer portion and ξ is a $[0, 1]$ uniformly distributed random number. (This method of generating r is an application of the inverse probability transform technique; see Chapter 7.) Since gaps can be quite long and one value of ξ

[†]It should not be inferred, however, that the case is always hopeless. An example of a practical application of this theory can be found in Ref. 26.

“generates” a gap, the potential for efficiency is apparent. We illustrate this point with an example.

■ *Example 11.2.2.* Consider a binary symmetric channel (BSC) with $p = 10^{-5}$ and suppose that the all-zero sequence $0, 0, \dots, 0$ has been sent. Suppose we draw the following four numbers from the uniform RNG:

$$0.96807, \quad 0.06673, \quad 0.47828, \quad 0.90953$$

Using (11.2.43), we obtain the following intervals between consecutive errors:

$$3246, \quad 270,709, \quad 73,756, \quad 9483$$

Therefore the channel output sequence is

$$0^{3245} \quad 1 \quad 0^{270708} \quad 1 \quad 0^{73755} \quad 1 \quad 0^{9482} \quad 1$$

where 0^x denotes x consecutive 0's. Hence, by simulating intervals, we generated effectively an error sequence 357,914 bits long, instead of 4, using only four random numbers. ■

The above technique can be extended to HMM models. In order to do so, we have to present the Markov chain as a particular case of a semi-Markov process, which is a Markov chain where the time between state changes is random and is dictated by some distribution.⁽²¹⁾ In the case at hand the state transition probabilities ρ_{ij} are given by

$$\rho_{ij} = a_{ij}/(1 - a_{ii}) \quad \text{for } i \neq j \quad \text{and} \quad \rho_{ii} = 0 \quad (11.2.44)$$

with geometric state-holding distributions (distributions of the time l that the chain spends in each state)

$$w_i(l) = (1 - a_{ii})a_{ii}^{l-1} \quad (11.2.45)$$

where $a_{ij} = P[S_{t+1} = j | S_t = i]$ is the state transition probability, as defined in Chapter 9. This representation allows us to generate error gaps by a double application of the interval technique: first, by generating state-holding intervals using the parameters in (11.2.44) in (11.2.43) and then using the same technique again to generate gaps while in a particular state. We illustrate this method with another simple example.

■ *Example 11.2.3.* Let us consider a Gilbert channel (see Figure 9.21); with state **1** $\equiv$ good state and state **2** $\equiv$ bad state, we assume the state transition probabilities $a_{11} = 0.997$, $a_{12} = 0.003$, $a_{22} = 0.966$, and $a_{21} = 0.034$. In state 1 the probability of error is 0, and in state 2 it is taken as 0.16. As described, we recast the model as a semi-Markov chain. Using (11.2.44), we obtain

$$\rho_{11} = 0, \quad \rho_{12} = 1, \quad \rho_{21} = 1, \quad \rho_{22} = 0$$

and the state-holding time distributions are

$$w_1(l) = 0.997^{l-1} 0.003, \quad w_2(l) = 0.966^{l-1} 0.034$$

We also assume the initial state distribution is stationary with respective probabilities for states 1 and 2⁽²¹⁾

$$\rho_1 = 0.9189, \quad \rho_2 = 0.0811$$

Suppose now that the following sequence of numbers has been drawn from a uniform RNG:

$$0.35169, 0.93253, 0.65444, 0.51220, 0.20202, 0.20408$$

The chain starts from the first state since $0.35169 < \rho_1$. It remains in that state for the duration $l_1 = [1 + \log 0.93253 / \log 0.997] = 24$ bits and transfers to the second state for the duration $l_2 = [1 + \log 0.65444 / \log 0.966] = 13$ bits and transfers back to the first state. Since errors in the second state occur independently with BER 0.16, we can use the interval technique within the 13-bit interval. Using the next two uniform numbers, we get $r_1 = [1 + \log 0.51220 / \log 0.84] = 4$ and $r_2 = [1 + \log 0.20202 / \log 0.84] = 10$. Thus, within 13 bits in the second state we have 3 correct bits, 1 error, and 9 correct bits: 0³ 1 0⁹. Next we generate the state-holding interval for the first state: $l_3 = [1 + \log 0.2048 / \log 0.997] = 538$. The results of this minisimulation can be summarized as 0²⁷ 1 0⁵³⁸.

Thus, using only six random numbers we “simulated” 566 bits, instead of only 6 bits with the conventional method. ■

The interval simulation technique can be used to generate subsequently an actual sequence, or can be used to advantage in obtaining interval statistics. The study of Markov models is a large discipline in itself. The reader is referred to the earlier discussion in Chapter 9 as well as to the cited references.

11.2.4. Tail Extrapolation

The notion of extrapolation refers to the making of inferences beyond the range seemingly justified by the data in hand. In the context of error-rate estimation, this implies collecting errors in a high-error regime, where it is relatively easy to make the estimates more reliable, and extending these estimates to make a “guess” at what the error rates would be at signal-to-noise conditions where errors would occur relatively rarely. An informal version of extrapolation has long been practiced using the “eyeball” method, which in this context consists in extending a BER curve beyond the range of measurement simply by what appears visually to be reasonable. This approach is perhaps too close to a guess; what we want, rather, is a systematic method, whose goodness can also be quantified. In this chapter, we will discuss two techniques that can be called extrapolative, the first, *tail extrapolation* (TE), to be discussed in this subsection, and the second, *importance sampling* (IS), to be discussed in the next. In both cases we will be discussing application of these methods to SER or BER evaluation, that is, not-interval-related measures. We should mention that both TE and IS are extensions of Monte Carlo methodology, and ideally would imply shortened versions of Monte Carlo runs on *as-is* systems. In practice, this is not quite the case, as will be discussed later.

The tail extrapolation technique is based on the assumption that the pdf in the *tail region* of the decision device input waveform is described by some member of a family of distributions that we shall refer to as the generalized exponential (GE) class,[†] namely,

$$f(x; \nu) = \frac{\nu}{2\sqrt{2}\sigma\Gamma(1/\nu)} \exp\left\{-\left|\frac{x-\mu}{\sqrt{2}\sigma}\right|^\nu\right\}, \quad -\infty < x < \infty \quad (11.2.46)$$

[†]It is clear that for $\nu = 2$, $f(x; \nu)$ reduces to the normal density.

where $\Gamma(\cdot)$ is the gamma function. The mean of the distribution is μ and the variance V_v is related to the parameter σ through

$$V_v = 2\sigma^2\Gamma(3/v)/\Gamma(1/v) \quad (11.2.47)$$

Although we define $f(x; v)$ over its entire range, this is only for convenience: it is mainly the “tail” region that is of interest here, that is, the relatively large values of x that can cause errors. If, in fact, the decision voltage pdf is governed by (11.2.46), one can deduce a procedure to estimate the BER specifically dependent on that assumption. This procedure was originally suggested by Weinstein⁽²⁷⁾ for binary signaling, which we shall assume here, but the idea is easily extended to more general schemes. The parameters v , σ , and μ may be different for ones than for zeros, so that the procedure outlined may have to be applied separately. For the moment we will concentrate on a particular symbol, say the zero symbol, which we take to be physically implemented by a negative voltage. For reasons of convenience, we will denote the probability of exceeding some value $t + \mu > 0$ by $p(t)$, so that

$$p(t) = \int_{t+\mu}^{\infty} f(x; v) dx$$

With the change of variable $y = (x - \mu)/(\sqrt{2}\sigma)$, we can express $p(t)$ as

$$p(t) = \int_{t/\sqrt{2}\sigma}^{\infty} \frac{v}{2\Gamma(1/v)} e^{-y^v} dy = \frac{1}{2\Gamma(1/v)} \Gamma\left(\frac{1}{v}, \xi\right) \quad (11.2.48)$$

where Γ is the gamma function, $\Gamma(\cdot, \cdot)$ is the incomplete gamma function, and $\xi = (t/\sqrt{2}\sigma)^v$. Since we are assuming the region of integration is in the tail, we can invoke the asymptotic expansion⁽¹⁷⁾ for $\Gamma(\cdot, \cdot)$; that is, for sufficiently large t one can approximate (11.2.48) as

$$P(t) \approx \exp\left\{-\left(\frac{t}{\sqrt{2}\sigma}\right)^v [1 - \epsilon(t)]\right\} \quad (11.2.49)$$

For $v = 1$, $\epsilon(t) = 0$, and for $v > 1$ and sufficiently large t , $\epsilon(t) \ll 1$. Assuming the latter to be true, we have that (11.2.49) is equivalent to

$$\ln[-\ln p(t)] \approx v \ln(t/\sqrt{2}\sigma) \quad (11.2.50)$$

which is to say the double log of the error probability is asymptotically linear in $\ln(t/\sqrt{2}\sigma)$.

The actual distribution of the decision voltage will generally not have the form (11.2.46) over its entire range, but will typically have a skewed appearance such as that shown in Figure 11.12a. Here, the actual threshold is shown at zero, and the BER p would be the hatched area. The sketch also indicates the mean of the actual distribution to be some value u , which it will be convenient to take as a reference. Thus, for the following discussion we will shift the density in Figure 11.12a to the right by u , as shown in Figure 11.12b; this obviously will have no effect if all quantities are correspondingly shifted along.

The basic idea behind tail extrapolation can now be explained as follows. Let us say that (11.2.49) is strictly true, with $\epsilon(t) = 0$. There is only one unknown to determine, namely v . So, in a sense, the purpose of the TE experiment is to extract v . If (11.2.49) is true, then it will apply to any value for t that is large enough, but smaller than the actual threshold, which is at

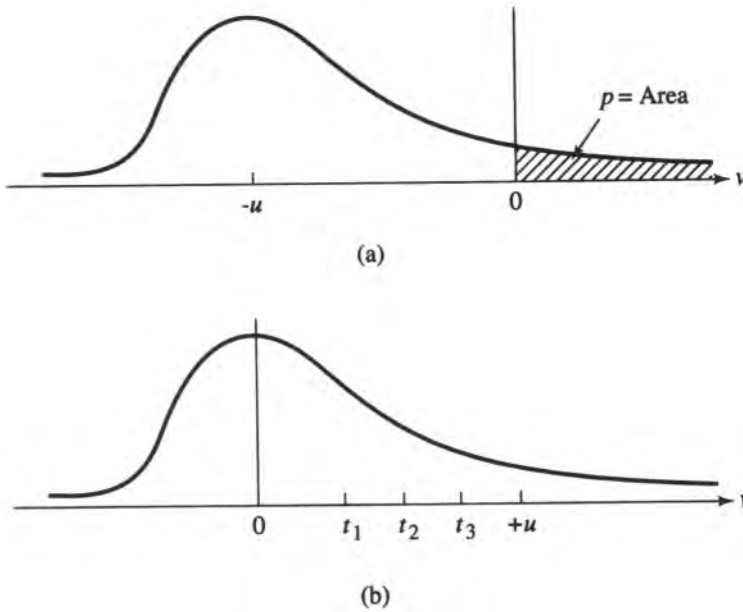


Figure 11.12. Illustrative amplitude distribution for tail extrapolation discussion (from Ref. 19, © IEEE, 1984). (a) Density function for transmitted zeros. (b) The same density function shifted.

u (Figure 11.12b). If we knew $p(t_1)$ exactly, we could infer v and thus calculate $p(t)$ for any value $t > t_1$. We will refer to t_1 as a *pseudothreshold* and to $p(t_1)$ as a *pseudo-BER*. In reality we have to estimate $p(t_1)$, but if t_1 is sufficiently lower than the threshold, we can obtain a reliable estimate in much less time than necessary to estimate the actual error probability. This point is also illustrated by the waveform sketched in Figure 11.13, which reemphasizes the fact that the crossings of levels increasingly removed from the mode have decreasing probabilities. However, to infer v , it is not recommended to rely on a single point, for two reasons: first, because the obtained point still has some statistical variability, and second, because the closest member of the GE family at t_1 may not be quite the same as that for t . Thus, a good procedure will use more than one pseudothreshold, and three seems to be about the best compromise between computational effort and stability. Figures 11.12b and 11.13 illustrate three such values, t_1, t_2, t_3 . Since in principle (11.2.50) applies to all three, we have

$$\ln[-\ln p(t_k)] \approx v \ln(t_k/\sqrt{2}\sigma), \quad k = 1, 2, 3 \quad (11.2.51)$$

If (11.2.51) were an exact equality, we would expect the points on the left side to fall on a straight line when plotted against $\ln(t_k/\sqrt{2}\sigma)$. In actuality, these points are used to obtain a “best” straight line, which is then extended to the true threshold. (From this step arises the term *tail extrapolation*.) Observe that $(t_k/\sqrt{2}\sigma)^2$ can also be termed a pseudo-SNR. As will be seen shortly, however, the actual value of σ need not play a part in the procedure. The numerical values of the t_k should be chosen carefully in accordance with the guidelines discussed shortly. Equation (11.2.51) should be interpreted as suggesting a linear form for $\ln(-\ln p)$ rather than strictly as an approximation, for we have omitted a possible constant which can move the line vertically and thus optimize the approximation. However, in practice we do not need to determine this constant explicitly, for it is automatically determined from

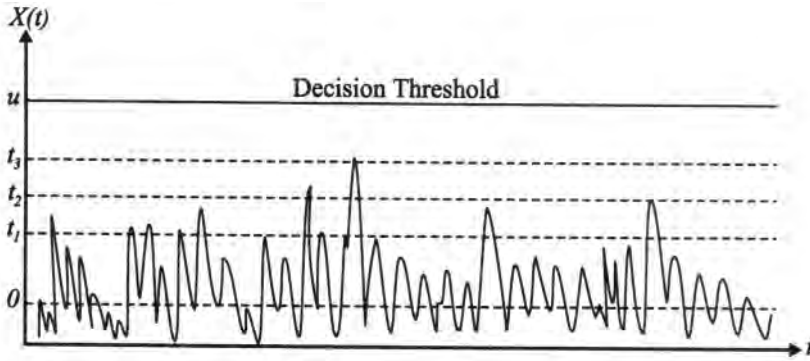


Figure 11.13. Illustration of pseudothresholds.

the procedure. That is, the correct pairs $(p(t_k), t_k)$ are known and place the straight line at the appropriate level.

11.2.4.1. Form of the Estimator

We assume that the procedure uses three pseudothresholds. The specific form of the estimator depends on this assumption. As hinted above, even if the pseudo-BERs were governed exactly by an equation like (11.2.48), experimentally they would fluctuate because they are random quantities. Thus, we use three points to determine a “best” straight line, which we take to be in the least-mean-square sense.

Let $y_i = \ln[-\ln \hat{p}(t_i)]$, where $\hat{p}(t_i)$ is the observed pseudo-BER at pseudo-threshold t_i , and let $x_i = \ln t_i$. Also, let $y = \ln(-\ln \hat{p})$ be the estimate of the true BER at the actual threshold x . [Notice that in the form (11.2.51) there is a factor $\sqrt{2}\sigma$ which divides t_i . Here, we can consider this factor absorbed in t_i , or, alternatively, we can actually ignore it since taking the log amounts to a constant shift in the horizontal scale.] We then have

$$y = \alpha + \beta x \quad (11.2.52)$$

where the coefficients α, β are chosen for a least-squares fit (See, e.g., Ref. 28 for the form of these coefficients.) We select pseudothresholds that are equally spaced on a logarithmic scale, which corresponds to equal separation in decibels. That is also a convenient choice because then α and β reduce to simple expressions. Thus, with $x_3 = x_2 + (x_2 - x_1) = 2x_2 - x_1$, (11.2.52) takes on the explicit form⁽²⁷⁾

$$y = \ln(-\ln \hat{p}) = \frac{1}{3}(y_1 + y_2 + y_3) + \frac{(x - x_2)(y_3 - y_1)}{2(x_2 - x_1)} \quad (11.2.53)$$

It is not required actually to calculate (11.2.53) if one obtains the estimate graphically.

■ **Example 11.2.4.** Figure 11.14 shows the result of an actual simulation run where tail extrapolation was applied. Three pseudo-error rates were measured and plotted as points 2, 3, and 4. A best straight-line fit of these points was made and the line extended to the actual value of the threshold. Note that the separation between points is the same, in dB. (The point labeled “bad” is discussed later.) At the actual threshold, a full MC run was made, and the

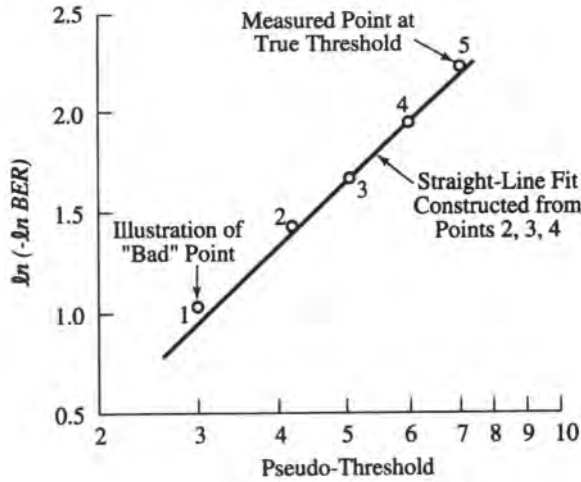


Figure 11.14. Tail extrapolation construction for Example 11.2.2 (from Ref. 19, © IEEE, 1984).

resulting BER (about 7.5×10^{-5}) is shown as point 5. It can be seen that the tail extrapolation estimate in this case is quite good. ■

11.2.4.2. Asymptotic Bias of the Estimator

If the distribution of $\hat{p}$ in (11.2.53) could be obtained, one would have a complete description of its statistical properties. Unfortunately, this seems quite difficult to do. However, we can obtain the expected value of $\hat{p}$, hence its bias, as $N \rightarrow \infty$. In this instance there is no statistical variability and y_1, y_2, y_3 are the true values at their respective thresholds. Hence y calculated from (11.2.53) yields in this case the irreducible bias or error inherent in the best-fit straight-line approximation.

Unfortunately, there is a nonzero bias in this technique, which for probably most cases of interest can be kept to an acceptable value. As discussed in Section 10.1.3.5, it is generally more meaningful to express errors in terms of the desired quantity. Here, we define the (asymptotic) relative bias $b(\hat{p}) = \hat{p}/p$, which is plotted in Figure 11.15 versus v for true BER values $p(t)$ of 10^{-6} , 10^{-7} , and 10^{-8} . To draw this figure we have assumed that for $v = 2$, the highest threshold corresponds to two, three, and four orders of magnitude extrapolation for $p = 10^{-6}$, 10^{-7} , and 10^{-8} , respectively. The pseudothresholds are separated by about 1 dB.

It should be noted that small errors will be magnified in the double exponentiation. This is why the curves do not converge identically to $b(\hat{p}) = 1$ at $v = 1$, which should have been the case. Thus, there is some (relatively small) error in Figure 11.15 due to roundoff followed by double exponentiation

11.2.4.3. Variance of the Estimator

An approximation for the variance of the estimator has been given in Ref. 27, namely,

$$\text{var}[\ln(-\ln \hat{p})] \approx \sum_{i=1}^3 \sum_{j=1}^3 a_i a_j \frac{1}{N} |p(t_i) p(t_j) \ln p(t_i) \ln p(t_j)|^{-1} p(t_k) \quad (11.2.54)$$

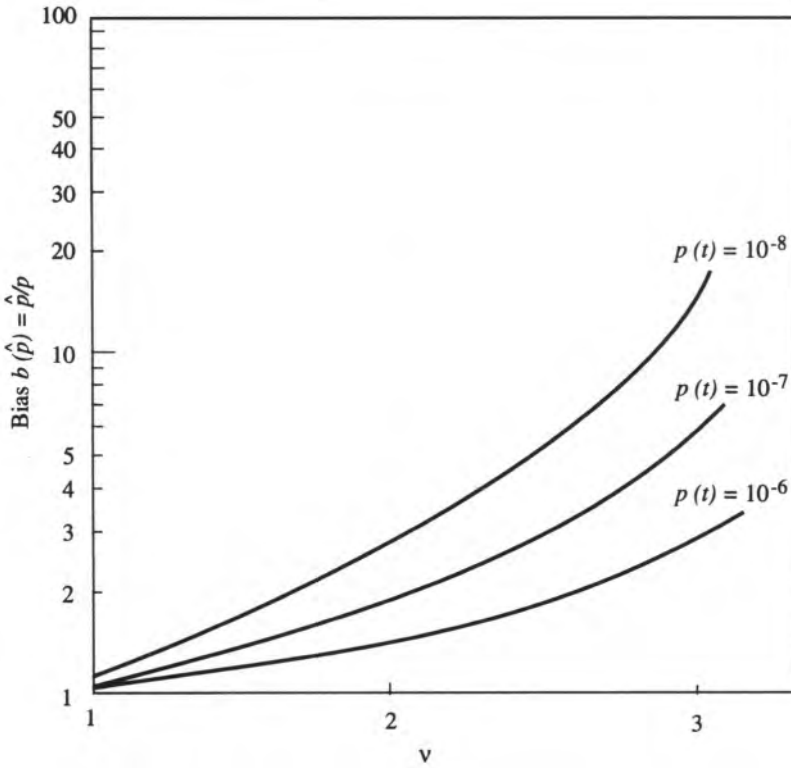


Figure 11.15. Bias in extrapolation estimate as a function of the v parameter for fixed values of the threshold and pseudothresholds; $p(t)$ stands for BER at some threshold t .

where $k = \max(i, j)$, $a_1 = 1/3 - c$, $a_2 = 1/3$, and $a_3 = 1/3 + c$, with $c = [\ln t - \ln t_2]/[\ln t_3 - \ln t_1]$. It can be seen that $\text{var}(y)$ depends on the threshold and pseudothreshold locations, as might be expected, as well as on v . As usual, we are not so much interested in $\text{var}(y)$ itself as in the variance relative to MC. In particular, the ratio of the time–reliability products

$$\eta = [p(\ln p)]^{-1} / \text{var}[\ln(-\ln \hat{p})] \quad (11.2.55)$$

is an indicator of the efficiency of the tail extrapolation estimate. Equation (11.2.55) has been calculated for the same set of conditions as those that led to Figure 11.15, and the results are shown in Figure 11.16.

■ *Remark on Bias and Variance of the Tail Extrapolation Estimator.* Figures 11.15 and 11.16 give a good picture of the tradeoff between bias and variance. The longer the extrapolation, the greater the improvement, but the larger the bias. Generally it is not acceptable that $b(\hat{p})$ be larger than about 2–3. This, therefore, sets the available limit on run-time improvement. The tradeoff is also a function of v . For systems with Gaussian noise, we can expect $v \leq 2$ at the output. The results displayed do depend on the values of the pseudothresholds t_1, t_2, t_3 , but for well-chosen values, this dependence should be minor. ■

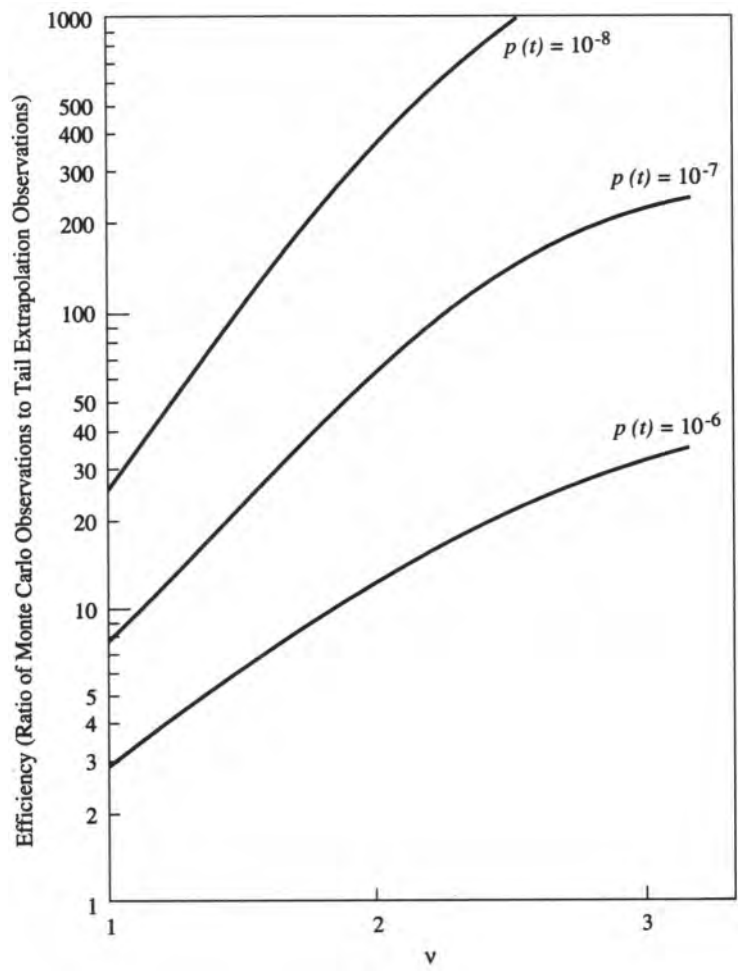


Figure 11.16. Efficiency of the extrapolation estimate as a function of the parameter v .

11.2.4.4. Summary of the Simulation Procedure for Implementing Tail Extrapolation

When implementing tail extrapolation, there are two main issues to consider: establishing the BER separately for ones and zeros, and selecting the pseudothresholds.

The previous discussion dealt with estimating the BER for zeros. If the tail distribution for ones is different from that for zeros, or at least sufficiently different, we would want separate estimates. In order to use tail extrapolation for ones it is necessary to cast the problem in the manner implied by Figure 11.12b. That is, if errors are associated with negative values of voltage, we must shift the pdf for ones down by the measured mean value for ones, then invert the sign. In other words, if t_a, t_b, t_c are the actual pseudothresholds for ones in increasing order ($t_c > t_b > t_a$ and $u > t_c$), then in order to use the graphical procedure illustrated in Figure 11.14, associate $p(t_a), p(t_b)$, and $p(t_c)$ with the pseudothresholds $(u - t_a), (u - t_b)$, and $(u - t_c)$, respectively.

If the error rate for ones and zeros can be assumed equal, a simpler procedure is to rectify the waveform at the decision device input, and then proceed as previously described.

If rectification is used, the area to the right of u (assuming zero is the actual threshold) is folded back to the left. The highest pseudothreshold should be chosen so that the foldover is small. Typically, for small BERs ($< 10^{-3}$) if $u - t_3$ is about one standard deviation of the noise, this should be adequate. In the example of Figure 11.14, the noise standard deviation (normalized to the mean value) was about 1.3.

The lowest pseudothreshold t_1 depends on how large a range of the pdf is describable by a particular exponential $\exp[-(x/\sigma)^V]$. A given exponential may not adequately describe both the tail and the mode, and we want to avoid t_1 being too close to the mode. A good rule of thumb is that $u - t_1 > 2\sigma_t$, where σ_t is the “total” standard deviation of the sampled waveform, which can be measured directly in the simulation. In Figure 11.14 a point labeled 1 is shown as a “bad” point and is meant to illustrate an injudicious choice of lowest threshold. This point is no longer in the tail (or at least the same tail as the other pseudothresholds) and too much toward the mode. Such a point would cause an error in the best-fit straight line that we would use for extrapolation. However, it can be seen that the graphical procedure is self-correcting to some degree because such a choice as just discussed for t_1 would become visually evident as inappropriate, and a new choice would then be made.

The (simplified) procedure for implementing tail extrapolation can thus be summarized as follows:

1. Process N symbols through a system.
2. At the decision device input, rectify the waveform (after having made sure that the system does not cause errors without noise).
3. Choose three pseudothresholds, in accordance with the preceding guidelines.
4. Establish experimentally (i.e., from the simulation) the corresponding pseudo-BERs.
5. Plot the pseudo-BERs versus the pseudothresholds [on $\log(-\log)$ versus $\log$ scales] and extend the best-fit straight line to the location of the true threshold. Double exponentiate that number to obtain the BER estimate.

■ *Remark.* While the generalized exponential family accommodates a large range of behavior through the parameter v , it is not necessarily appropriate for every situation. For example, Gooding⁽²⁹⁾ applies a tail extrapolation technique where it is the log of the pseudo-BERs that is proportional to the log of the pseudothreshold. Thus, the generalized exponential may not be suitable for all scenarios of interest, and the simulator should determine what is the appropriate family. Having done so, the simulator can attempt to apply “tail extrapolation,” where we can now interpret this term more generally as a linearization technique, the goal of which is to express some function of the pseudo-BER as a linear function of the pseudothreshold. ■

11.2.5. Importance Sampling

Importance sampling (IS) is potentially the most powerful run-time-reducing technique, in the best of circumstances able to reduce the time–reliability product by many orders of magnitude. The word “potentially” is not to be taken lightly, for two reasons. First, although IS implies a general underlying principle, its embodiment is not a unique procedure, independent of context, in the sense that Monte Carlo is. Indeed, an IS experiment has to be “designed” for the problem at hand and it is seldom obvious what the best scheme is for that problem. Second, for a given IS approach, there are usually parameters that have to be set to values that are not obvious, and less than the best choice can result in a tremendous loss of power, in fact to the point that a run-time *increase*, rather than a decrease, is the result. The “design” aspect just mentioned is often a nontrivial problem. Considerable manpower may

have to be expended to develop the proper scheme for a given situation, to write appropriate computer code, and to integrate it, if necessary, into a larger simulation. This is a cost that also has to be weighed in relation to the possible run-time savings. Thus, unless it is otherwise literally impossible to achieve one's objective, as might be the case with extremely low BER, IS may not be the method of choice. We should also add that IS is not suitable for all problems of interest. However, where it applies, its potential *is* so great that it merits a reasonably careful introduction.[†]

Before passing to the details, it may be instructive to begin with some qualitative preliminaries as to why IS can be so powerful. Let us first recall tail extrapolation and why it “works.” In TE, a standard MC experiment is performed. The system is *as-is*, but we modify our observational paradigm. We “cleverly” institute thresholds different from the actual one so as to induce a greater number of observed “important” events, i.e., pseudoerrors. In IS, our general aim is the same, that is, to induce more “important” events (this time true errors, rather than pseudoerrors) but the critical distinction is that in IS we *modify the underlying MC experiment itself*. Rather than being relatively passive observers, we alter the experiment to suit our purpose. This alteration is not meant in the sense that models of the operation of devices are changed, but rather that we modify the statistical properties of random processes input to the system. The alteration is artificial in the sense that the experiment may result in behavior that could not occur in the actual system, but since we (the simulator) are the agent of this artificiality, we can undo its effect on the observations.[‡] This idea is not unlike accelerated life testing, for example. Here, “abnormal” conditions of wear may be imposed on objects in order to determine their “life,” these abnormal conditions being related in a known or statistical way to “normal” conditions. A substantial literature exists on the subject of IS, only a small part of which we cite here for reference,^(30–46) but these works can be consulted for further citations; ref. 46, in particular, is a readable recent survey with many further references.

To motivate the subsequent development,[§] let us begin with the simple binary case, and recall Equation (11.2.1b) in slightly different form,

$$p_0 = \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(v) f_0(v) dv \quad (11.2.56)$$

which represents the conditional BER, conditioned on the zero symbol. Now define another probability density function f_0^* and consider the identity

$$p_0 = \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(v) \frac{f_0(v)}{f_0^*(v)} f_0^*(v) dv \quad (11.2.57a)$$

$$= \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(v) w(v) f_0^*(v) dv \quad (11.2.57b)$$

$$= E_*[I_{\mathcal{A}_0}(v) w(v)] \quad (11.2.57c)$$

[†]Importance sampling, or variations of it, have been very successfully applied to networking (queueing) problems, but appears to be considerably more difficult to apply usefully to link problems.

[‡]Changing the statistical nature of the experiment is just one way of trying to make the experiment more efficient. In the previous section, we changed one facet of the real system, namely, the distance to the threshold. In this section, we will discuss briefly still another strategy, the essence of which is to restart the experiment if it seems to be going in the “wrong” direction.

[§]The current discussion, till the end of this subsection, treats an artificial case that is more suitable initially to expose the main ideas of IS. The version applicable to the simulation context will be covered in the following subsection.

where f_0^* is called the *biasing density*.[†] $w = f_0/f_0^*$ is called the *weight*, E_* denotes expectation with respect to f_0^* , and we must have $f_0^* > 0$ whenever $f_0 > 0$. On the surface it may seem that we have changed nothing, but in fact this is not so if we interpret (11.2.54) to mean that when we run an MC experiment to estimate p_0 , we sample from f_0^* rather than from f_0 . The empirical (or MC) counterpart of (11.2.57) is

$$\hat{p}_{0,*} = \frac{1}{N_{0,*}} \sum_i I_{\mathcal{A}_0}(v_i) w(v_i) \quad (11.2.58)$$

where i indicates time or instance of a zero occurrence and $N_{0,*}$ is the number of such instances. (Generally, we shall append an asterisk to subscripts or superscripts to indicate a quantity associated with the use of f_0^* .) Equation (11.2.58) says that we still count errors, via the error counter $I_{\mathcal{A}_0}$, but that when an error occurs at instance i , we weigh it by the factor $w(v_i)$. Thus the summand is no longer an integer, as in the MC case, but a real number. The obvious question is what advantage does $\hat{p}_{0,*}$ have as an estimator. First, we can show that it is unbiased (Problem P11.11), as is the standard MC estimator. But its variance can be shown to be (Problem P11.12)

$$\sigma^2(\hat{p}_{0,*}) = \frac{1}{N_{0,*}} \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(v) f_0(v) w(v) dv - p_0^2 \quad (11.2.59a)$$

$$= \frac{1}{N_{0,*}} \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(v) f_0(v) [w(v) - p_0] dv \quad (11.2.59b)$$

which is to be contrasted with that of the MC estimator obtained simply by letting $w(v) = 1$, namely

$$\sigma^2(\hat{p}_0) = \frac{1}{N_0} \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(v) f_0(v) [1 - p_0] dv \quad (11.2.60)$$

Observe that in the MC case, the term $1 - p_0 \approx 1$ for most practical cases. Thus, it is clear that the term $w(v) - p_0$ offers the possibility of variance reduction, for if we can manage to make $w(v)$ smaller than 1, and hopefully relatively close to p_0 for all v in the error region, this term will reduce the integrand in (11.2.59) and thus the variance. The smaller we can make $w(v) - p_0$, the smaller we can make the estimator variance. In fact, if we were able to arrange that

$$f_0^*(v) = \frac{f_0(v) I_{\mathcal{A}_0}(v)}{p_0} \quad (11.2.61)$$

the integrand in (11.2.59b) would be precisely zero, and so would the estimator variance. Of course, this “perfect” biasing density is a tautology, since it requires knowledge of p_0 , the very quantity that we are looking for. But there is also another form of tautology in (11.2.61), perhaps not as obvious, namely the assumed knowledge of $f_0(v)$, the output voltage density given a zero. For if we knew f_0 , we could in principle evaluate p_0 directly from (11.2.56), thus

[†]This term is generally standard, although a bit unfortunate because of the possible confusion with *bias* in the usual statistical sense. The term *simulation density* has also been used,⁽³⁸⁾ which better implies its function.

negating the need for simulation in the first place. The preceding formulation of IS is called the “output” version because all quantities involved are those observable at the output.

In fact, it is the premise of simulation that closed-form results implied by this evaluation are not possible, or too difficult to obtain. Indeed, it is the function of simulation to *produce* the output waveform, and by implication its statistics, given a model for the system and a means of generating the input processes that are driving the system. It is only these input processes that are under the control of the simulator, and whose properties might be manipulated to some useful end. Hence, as implied in the previous footnote, the preceding discussion serves only to introduce the main idea behind IS with a minimum of formalism. The way to implement IS in a more realistic way will now be outlined.

11.2.5.1. Formulating IS for Simulation Implementation

As just discussed, it is the input processes that must be “biased” because it is these that we have direct control of. For this reason, the following formulation is often called the “input” version of IS. But the occurrences of errors still are events that are observable only at the output. Consequently, it is necessary to have a link between the inputs and the output. Accordingly, we define a system function g that embodies this connection (Figure 11.17). We will assume immediately that we are dealing with discrete time so that the output waveform to be sampled (or otherwise processed) can be expressed as

$$V_t = g(\Omega) \quad (11.2.62)$$

where t is an integer multiple of the sampling interval, and Ω is a collection of K vectors representing different input processes

$$\Omega = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_K)$$

where $\mathbf{z}_k = (z_t, z_{t-\Delta}, \dots, z_{t-M_k\Delta})_k$, Δ is identical to T_s , and M_k is the *memory* (in terms of simulation samples) of the system relative to the k th process. There is, of course, an equivalent memory in terms of number of symbols: if there are η samples per symbol,

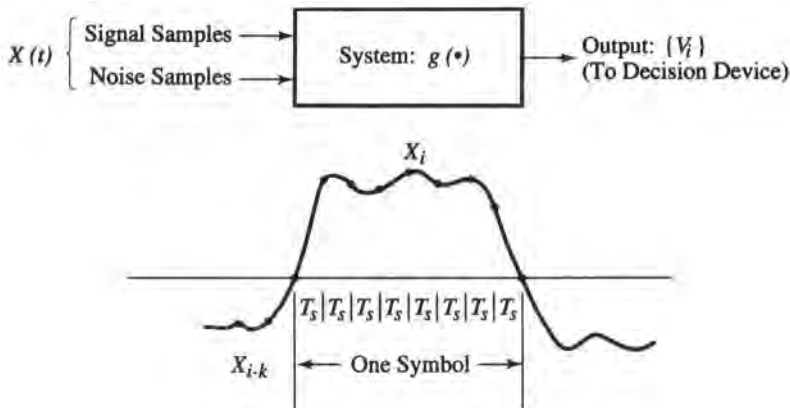


Figure 11.17. Illustration of some quantities used in importance sampling discussion (from Ref. 19, © IEEE, 1984). This illustration is a particular case of Equation (11.2.62), where $X(t)$ stands for the sum of input signal and noise.

$m_k = M_k/\eta$ is the memory in symbols. Naturally, one of these processes will be a signal; others may be one or more thermal noise sources, another may be interference, still another may be oscillator phase noise, and one of these may represent the state of a fading channel. Of course, in a physical system the memory will normally not be strictly time-limited, though for practical purposes we can usually define an effective finite memory. The reason that the memory may be different for different processes is to account for the possibility that these processes can enter the system at different points. For example, in a two-hop repeating satellite system, noise enters separately on the uplink and on the downlink. Furthermore, the memory for some processes may actually be randomly time-varying if part of the channel is a randomly time-varying medium.

It will be useful subsequently to explicitly name some of the processes. To begin with, we will identify $\mathbf{z}_1$ as the signal process: $\mathbf{z}_1 = (s_0, s_1, \dots, s_{M_1})$, where s_0 is the most recent sample. These signal samples have to be treated notationally somewhat differently than those of other processes because all samples belonging to the same symbol are deterministically related. Furthermore, when V_t is sampled for a decision, it will also be useful to isolate the symbol upon which a decision is being made; that is, at decision time t some subset of η samples corresponds to the observed symbol, which we will denote σ . The remainder of the samples will be denoted $\mathbf{s}$; this is simply the ISI pattern, which has L^{m_1-1} realizations.[†] Thus, $\mathbf{z}_1 = (\sigma, \mathbf{s})$. Again, for simplicity, and without losing too much generality, we will assume that all $\mathbf{z}_k$ are independent of one another and also that σ and $\mathbf{s}$ are independent of each other, and so we can write for the joint density [where the f are generally functionally distinct]

$$f(\Omega) = f(\sigma)f(\mathbf{s})f(\mathbf{z}_2) \cdots f(\mathbf{z}_K)$$

Consider now formulating[‡] the error probability conditioned on symbol $\mathbf{a}_j$:

$$\begin{aligned} p_j &= \int_{-\infty}^{\infty} h[g(\mathbf{a}_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K)] f(\mathbf{s}) f(\mathbf{z}_2) \cdots f(\mathbf{z}_K) d\mathbf{s} d\mathbf{z}_2 \cdots d\mathbf{z}_K \\ &= \int_{-\infty}^{\infty} I_{\mathcal{E}_j}(\mathbf{a}_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K) f(\mathbf{s}) f(\mathbf{z}_2) \cdots f(\mathbf{z}_K) d\mathbf{s} d\mathbf{z}_2 \cdots d\mathbf{z}_K \end{aligned} \quad (11.2.63)$$

where $h(\mathbf{u}) = 1$ if $\mathbf{u} \in \mathcal{R}_j$ and 0 otherwise, and the error set $\mathcal{E}_j$ is defined as

$$\mathcal{E}_j = \{(\mathbf{a}_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K) : g(\mathbf{a}_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K) \in \mathcal{R}_j\}$$

The overall SER is then

$$p = \sum_{j=0}^{L-1} p_j f(\mathbf{a}_j)$$

where $f(\mathbf{a}_j)$ is the probability that $\mathbf{a}_j$ is sent.

[†]In order not to overburden the notation we have made two implicit assumptions that are not limiting: first, that sampling for a decision does not require interpolation; second, that the number of symbols corresponding to the memory is an integer, which of course need not be so.

[‡]In this subsection, for simplicity we use a single integration sign where multiple integration is implied.

The formal analog of (11.2.57) is then

$$p_j = \int_{-\infty}^{\infty} I_{e_j}(a_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K) w(\mathbf{s}) w(\mathbf{z}_2) \cdots w(\mathbf{z}_K) \\ \times f^*(\mathbf{s}) f^*(\mathbf{z}_2) \cdots f^*(\mathbf{z}_K) d\mathbf{s} d\mathbf{z}_2 \cdots d\mathbf{z}_K \quad (11.2.64)$$

with the obvious identification of the weights $w(\mathbf{z}_k) = f(\mathbf{z}_k)/f^*(\mathbf{z}_k)$. In order not to overburden the notation, we have not indicated that the choice of biasing densities may depend on certain parameters. For example, they could depend on the conditioning symbol if the simulation structure were able to accommodate that flexibility. This point will be clarified shortly. The IS “design” problem that we mentioned earlier consists in determining the set of biasing densities $f^*(\mathbf{s}), f^*(\mathbf{z}_2), \dots, f^*(\mathbf{z}_K)$ so as to result in a substantial reduction in run time for estimating p . For the problem generally stated here, the joint determination of an optimal, or merely good, set of biasing densities is a very difficult task. A common approach to reduce the complexity of the problem is to bias only a subset of the input processes. This is referred to as conditioning because the IS experiment obtains, in effect, a conditional error rate. This approach will be outlined below.

Let us now obtain the form of the empirical estimator for p_j . Following the previous pattern, it can be seen that (11.2.64) is an expectation

$$p_j = E_*[I_{e_j}(a_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K) w(\mathbf{s}) w(\mathbf{z}_2) \cdots w(\mathbf{z}_K)]$$

the empirical counterpart of which is

$$\hat{p}_{j,*} = \frac{1}{N_j} \sum_{i=1}^{N_j} \{I_{e_j}(a_j, \mathbf{s}, \mathbf{z}_2, \dots, \mathbf{z}_K)\}_i w(\mathbf{s}_i) w(\mathbf{z}_{2,i}) \cdots w(\mathbf{z}_{K,i}) \quad (11.2.65)$$

where i indexes the instances of symbol j , of which there are N_j in number. The first term in braces is, as before, an error indicator. In actuality, of course, the occurrence of an error is determined by observing the output V_i . When an error occurs it is weighted by the products of the weights, which are evaluated for the particular values of their arguments that occur when an error does. Figure 11.18 illustrates the procedural aspects of implementing (11.2.65). For later use we will rewrite (11.2.65) in somewhat more compact form as follows:

$$\hat{p}_{j,*} = \frac{1}{N_j} \sum_{i=1}^{N_j} I_{e_j}(\Omega_i) w(\Omega_i) \quad (11.2.65a)$$

It is understood that (11.2.65a) is still conditioned on a_j , but it is well to reiterate that this “conditioning” is not a limitation. Symbol j is simply the symbol under observation, and occurs as it will in an *as-is* MC simulation, or can be forced to occur in a block simulation as long as the symbols preceding and following can also occur without restriction.

The incorporation of IS into a simulation has some methodological implications that are not apparent on the surface. In particular, certain choices for the biasing densities (and generally, this applies for “good” choices) do not permit a stream simulation[†] to be implemented: this will be illustrated below when we discuss particular forms for the biasing

[†]This term is defined in Chapter 2.

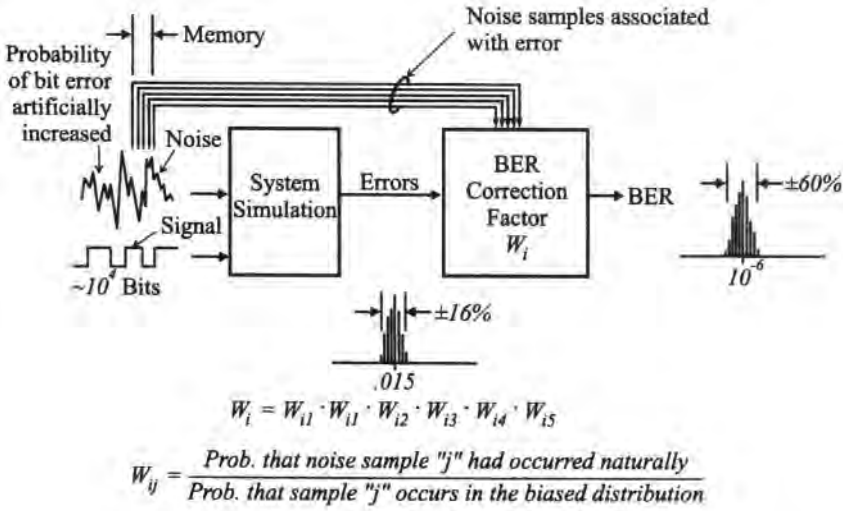


Figure 11.18. Illustration of the implementation structure in simulation for IS.

densities. The simulation, then, may not be *as-is*, and this is a potential limitation. Basically, the meaning of “instance i ” in (11.2.65) can be taken in two ways. If we were using a stream simulation, i would simply index the occurrences in time of symbol j . In that case, as discussed previously, N_j would be a random variable in a simulation of fixed total length if the input symbols were generated randomly. An alternative methodology, which we referred to in Chapter 2 as block simulation, consists, for each symbol $a_j, j = 0, \dots, L - 1$, in generating a set of random vectors in Ω independently a fixed number of times (controllable by the simulator). For suitably chosen biasing distributions, the number N_j of such replications would presumably be much smaller than for MC. In this approach, the ISI pattern $\mathbf{s}$ can in principle be handled differently than the continuous processes because there are only a finite number of realizations. We will illustrate this point presently.

In order to expose some of the previous ideas more concretely, let us consider the simplified scenario in which, besides the signal, another input to the system is noise, so that, say, $\mathbf{z}_2 = \mathbf{n}$, and there are no other inputs. Let us assume for now that only the noise is biased, so that Equation (11.2.64) takes the form

$$p_j = \int_{-\infty}^{\infty} I_{e_j}(a_j, \mathbf{s}, \mathbf{n}) f(\mathbf{s}) w(\mathbf{n}) f^*(\mathbf{n}) d\mathbf{s} d\mathbf{n} \quad (11.2.66)$$

A standard reformulation of (11.2.66) in terms of conditioning on $\mathbf{s}$ has a far-reaching consequence on methodology. Let us name the possible realizations of $\mathbf{s}$ as the set

$$\mathcal{A}^M = \{\mathbf{s} = (s_0, s_1, \dots, s_{M-1}) | s_i \in \mathcal{A}, 0 \leq i \leq M - 1\}$$

where $\mathcal{A}$ is the signal alphabet. Then (11.2.66) can be rewritten as

$$p_j = \sum_{\mathbf{s} \in \mathcal{A}^M} f(\mathbf{s}) d\mathbf{s} \int_{-\infty}^{\infty} I_{e_j}(a_j, \mathbf{s}, \mathbf{n}) w(\mathbf{n}; \mathbf{s}) f^*(\mathbf{n}; \mathbf{s}) d\mathbf{n} \quad (11.2.67)$$

In other words, (11.2.66) is interpreted as first conditioning on an ISI pattern and then averaging over the possible such patterns. The corresponding block methodology simulation would consist, for each $\mathbf{s}$, in running the simulation a sufficient number of times to estimate the conditional error rate represented by the integral. However, once we have allowed ourselves the possibility of this conditioning, there is no reason why we cannot consider tailoring the biasing density to each pattern. This is, in fact, the implication of writing w and f^* in (11.2.67) as (possibly) parametrically dependent on the ISI pattern. In principle, we have the flexibility of assigning a different biasing density for each $\mathbf{s}$ or perhaps assigning a biasing density to groupings of ISI patterns. The question is whether these possibilities are practically implementable. If the number of ISI patterns is large, this will, of course, pose a problem.

Now, however, we can amend the above strategy by considering biasing of the ISI patterns themselves. In particular, if we assume that only a small subset of the ISI patterns dominate the BER performance, then we only need to conduct simulations for that subset. This type of approach, suggested by Sadowsky,⁽³⁸⁾ is referred to as the error event simulation method. The corresponding extension of (11.2.67) is

$$p_j = \sum_{\mathbf{s} \in \mathcal{S}^M} w(\mathbf{s}) f^*(\mathbf{s}) \int_{-\infty}^{\infty} I_{e_j}(a_j, \mathbf{s}, \mathbf{n}) w(\mathbf{n}; \mathbf{s}) f^*(\mathbf{n}; \mathbf{s}) d\mathbf{n} \quad (11.2.68)$$

which we can also write as

$$p_j = \sum_{\mathbf{s} \in \mathcal{S}^M} w(\mathbf{s}) f^*(\mathbf{s}) p_{j|\mathbf{s}} \quad (11.2.68a)$$

There is a theoretical contradiction if we literally set $f^*(\mathbf{s})$ equal to zero for some $\mathbf{s}$. However, this does not lead to practical problems because the corresponding patterns never appear. The never-simulated patterns will necessarily lead to some estimator bias, but the selection of the retained patterns is presumably made precisely because they dominate. Hence, a properly biased ISI set should lead to acceptably small estimator bias. Of course, in order to implement this scheme, we need a practical way to rank order the ISI patterns, which may number in the thousands. For linear systems, at least, this can be straightforwardly done if we have the single typical pulse response of the system (see Problem P11.12.)

We can extend the strategy implied in (11.2.67) or (11.2.68) to the case where the output depends on any number of additional processes $\mathbf{z}_1, \dots, \mathbf{z}_K$ by successively conditioning on specific values and then averaging over the (perhaps biased) distribution of such values. In other words, we can reinterpret (11.2.64) operationally as a sequence of conditional experiments. As we discussed in conjunction with the ISI patterns, the removal of the conditions may be far from computationally trivial, for the $\mathbf{z}_k$ are multidimensional. In some instances, however, the quasistatic assumption (see Chapter 2) may hold, namely, that a particular process may vary sufficiently slowly that it can be considered constant over the ISI's memory span. In that case, the random vector can be replaced by a random variable, which makes the averaging considerably simpler.

11.2.5.2. Properties of the Importance Sampling Estimator

Here we will briefly discuss the bias and variance (more generally the time-reliability product) of the IS estimator. As might be surmised from (11.2.65) these properties are likely to depend on the choice of the biasing densities. Hence, pending a discussion on this choice,

which is given in the next subsections, we will present here only general expressions. We will take (11.2.65a) as our basis for discussion. It can be shown (see Problem P11.11) that

$$E(\hat{p}_{j,*}) = p_j \quad (11.2.69)$$

In other words, the IS estimator is unbiased. This is true if the dimensions of the involved random vectors are equal to the actual corresponding memories. If the actual memories are infinite, some degree of truncation is inevitable and there will necessarily be some bias. There will thus generally be some tradeoff between estimator bias and computational load. Some discussion of this issue is given in Refs. 31 and 32.

The expression for the estimator variance depends on the specific form of the estimator, for example, if there is a single biasing density for each random process, or if there is a set of biasing densities for one or another process tailored to each ISI pattern. For simplicity we look only at the case embodied in (11.2.65). From the definition of variance $\sigma^2(\hat{p}_{j,*}) = E(\hat{p}_{j,*})^2 - p_j^2$, we obtain (Problem P11.12)

$$\begin{aligned} \sigma^2(\hat{p}_{j,*}) &= \frac{1}{N_j} \int_{-\infty}^{\infty} I_{e_j}(\mathbf{\Omega}) f(\mathbf{\Omega}) w(\mathbf{\Omega}) d\mathbf{\Omega} \\ &+ \frac{1}{N_j^2} \sum_{m \neq n} \sum_{m, n \in \mathcal{J}_j} E_* \{ I_{e_j}(\mathbf{\Omega}_m) I_{e_j}(\mathbf{\Omega}_n) w(\mathbf{\Omega}_m) w(\mathbf{\Omega}_n) \} - p_j^2 \end{aligned} \quad (11.2.70)$$

where $f(\mathbf{\Omega})$ is the joint density of the components of $\mathbf{\Omega}$. The double sum exists only if we use stream simulation, for then there may be some correlation between the occurrence of an error at the m th and n th instances, although such correlation will generally be weak, at least for error rates not exceeding 10^{-2} , since such instances must be at least one symbol duration apart. In the block simulation method, independent replications of the experiment are carried out, so separate instances of errors cannot be correlated. We are thus justified in applying that assumption to (11.2.70), which then reduces to the more compact form

$$\sigma^2(\hat{p}_{j,*}) = \frac{1}{N_j} \int_{-\infty}^{\infty} I_{e_j}(\mathbf{\Omega}) f(\mathbf{\Omega}) w(\mathbf{\Omega}) d\mathbf{\Omega} - \frac{p_j^2}{N_j} \quad (11.2.71)$$

An equivalent form which is better for some purposes is

$$\sigma^2(\hat{p}_{j,*}) = \frac{1}{N_j} \int_{-\infty}^{\infty} I_{\mathcal{A}_j}(g(\mathbf{\Omega})) f(\mathbf{\Omega}) w(\mathbf{\Omega}) d\mathbf{\Omega} - \frac{p_j^2}{N_j} \quad (11.2.71a)$$

The notation in (11.2.71) and (11.2.71a) has been kept relatively simple for readability. But from the discussion above, we know that we can run a conditional IS experiment, in which case expressions like (11.2.71) and (11.2.71a) would apply, but only a subset of the processes in $\mathbf{\Omega}$ would be generated. We would then be computing a conditional variance.

Naturally, we want to make the variance as small as possible. We know that it cannot be negative, and ideally we would like it to be zero. Generally, this is an unattainable goal, but what might be a reasonable one? Recall that in MC simulation, for a given relative error the number of observations increases inversely proportional to the error probability. This is the reason MC is so time-consuming. The practical ideal, therefore, would be to make the number

of observations *independent* of error probability as $n \rightarrow \infty$, or perhaps only a slowly increasing function of p^{-1} . This is the “design” problem mentioned earlier. Biasing densities which produce this desideratum are said to be *asymptotically efficient*, but are not generally easy to derive for all problems of interest. Further discussion of this subject is provided below. While a seemingly desirable property, asymptotic efficiency is not a prerequisite for the utility of a biasing density.

Clearly, the variance will depend on the choice of biasing density (embedded in w) for the processes involved. But the expressions (11.2.71) cannot be further simplified or evaluated, except for particular cases. It is worthwhile pointing out, however, that the form of this expression is rather similar to that for p_j itself, e.g., (11.2.64). This implies that the difficulty of solving (11.2.71) is of the same order as that for calculating p_j , a task which we agree is not doable for a complex system, for otherwise we would not need the simulation. Therefore, as we will see, it will be necessary to make simplifying assumptions simply for the sake of performing illustrative calculations.

11.2.5.3. Choosing Biasing Densities

So far we have not discussed how to choose the biasing densities, other than the guidance which may be provided by the unrealizable ideal of (11.2.61). This form suggests the reasonable idea that in a good biasing density the bulk of the probability mass should be such as to increase the occurrence of errors as much as possible. This statement applies “multidimensionally” to all input processes. It is not difficult to imagine certain ways in which this can be done. It is more difficult, however, to deduce biasing densities that satisfy more formal definitions of optimality. In this section we shall first take a heuristic approach to this problem which does lead to potentially useful methodologies, depending on the system at hand. Formal or systematic approaches are technically more demanding and generally more difficult to apply to practical cases. Nevertheless, a brief introduction is worthwhile and is given subsequently.

11.2.5.3.1. A Heuristic Approach To lay the groundwork for the following discussion, let us look at a simple system for which the output voltage at discrete time t , given that $\mathbf{a}_j$ is the symbol sampled, is given by

$$V_t = g(\mathbf{a}_j, \mathbf{s}_t) + \mathbf{h} \cdot \mathbf{n}_t \quad (11.2.72)$$

where $\mathbf{s}_t$ is the symbol sequence preceding or following $\mathbf{a}_j$ that contributes to the noiseless voltage at time t (i.e., the ISI pattern); $\mathbf{h}$ is the linear impulse response affecting the noise, and is assumed time-limited to M samples; $\mathbf{n}_t$ is the noise vector contributing to the output at time t , $\mathbf{n}_t = (\mathbf{n}_t, \mathbf{n}_{t-\Delta}, \dots, \mathbf{n}_{t-(M-1)\Delta})$; and g is the transfer characteristic for the signal. This model is simply a slightly generalized version of a linear system where the noise is processed linearly, but the signal may not be. The simplest case where $\mathbf{g} = \mathbf{h}$ is used in the example below. If we denote $\boldsymbol{\eta}_t = \mathbf{h} \cdot \mathbf{n}_t$ and $\mathbf{n}_t$ is a Gaussian process, then $\boldsymbol{\eta}_t$ will be normal.

To simplify matters a little more, let us assume a binary transmission system and take $\mathbf{a}_j$ to be a zero, which corresponds to a negative voltage. Conditioned on a zero, Figure 11.19a shows a hypothetical distribution of the noiseless voltage $\mathbf{g}(\cdot)$. Let us now further condition on a particular ISI sequence $\mathbf{s}$ and label $\mathbf{v}(\mathbf{0}, \mathbf{s})$ the corresponding sampled noiseless voltage, as indicated in Figure 11.19b. A hypothetical distribution for the net noise $\boldsymbol{\eta}$ at that time is also shown, as well as that for the sum $\mathbf{v}(\mathbf{0}, \mathbf{s}) + \boldsymbol{\eta}$. Since the probability of error is the area under the tail to the right of the threshold, it seems intuitive that increasing the noise variance

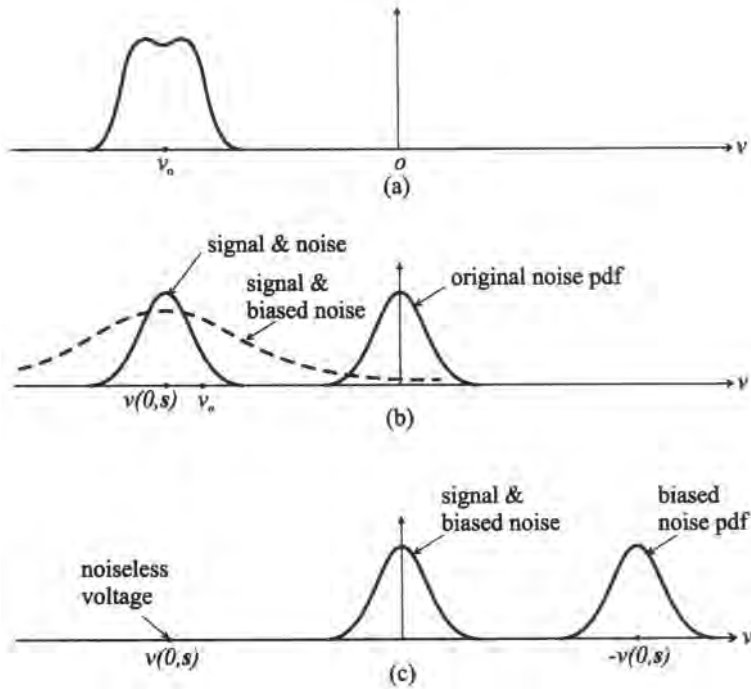


Figure 11.19. Illustration of some biasing ideas. (a) Distribution of output voltages given a 0 is sent; (b) distribution of signal plus noise, both biased (by variance scaling) and unbiased, given an ISI pattern $\mathbf{s}$; (c) distribution of signal plus noise biased by mean translation.

as shown will have the desired effect of increasing the number of observed errors. If $\mathbf{n}$ is a stationary Gaussian sequence with zero mean and operational variance σ^2 , and if we assume without loss of generality that $\|\mathbf{h}\|^2 = 1$, then $\eta \sim \mathcal{N}(0, \sigma^2)$. Then it would seem that a legitimate choice for a biasing density would be $\mathcal{N}(0, \sigma_*^2)$ with $\sigma_*^2 > \sigma^2$. This was in fact the first strategy proposed in this context_(30,31) and is variously referred to in the literature as variance scaling (VS) or conventional importance sampling (CIS). It turns out that the value of $\gamma = \sigma_*^2/\sigma^2$ that optimizes the improvement in run time depends on the memory M and on the degree of ISI. We will show this dependence in the example below. We can also observe that an optimal choice for γ will depend on $\mathbf{s}$ and so a simulation strategy that uses a single value for γ must be a compromise if we use a “stream” simulation methodology.

Continuing with the same case, where $\mathbf{v}(0, \mathbf{s})$ is received, consider now biasing $\mathbf{n}$ in such a way that its components are given a nonzero mean such that the distribution of η has a mean equal to $-\mathbf{v}(0, \mathbf{s})$. The distribution of the sum $\mathbf{v}(0, \mathbf{s}) + \eta$ now has zero mean and a variance still equal to σ^2 if the components of the biased $\mathbf{n}$ retain their original variance, which it turns out is about the best thing to do. Looking at Figure 11.19c, it seems clear that this biasing strategy is optimum in the sense that it produces the largest error rate possible (0.5), which was our objective. How do we choose the mean for $\mathbf{n}$, so as to achieve the desired translation? In principle, it makes no difference so long as $E(\eta) = \sum_{i=0}^{M-1} E(\eta_i) = -\mathbf{v}(0, \mathbf{s})$. So, for example, all the means except one could be zero, the nonzero one being $-\mathbf{v}(0, \mathbf{s})/\mathbf{h}_k$, where the nonzero entry is in position $k + 1$. It is clear that in this case the simulation distribution is not stationary. One interesting choice (illustrated in Figure 11.20) that we shall revisit below is to choose $\boldsymbol{\mu}$, the mean of the biased noise vector, as $\boldsymbol{\mu} = \mathbf{c}\mathbf{h}$, so that $E(\eta) = \mathbf{c}\|\mathbf{h}\|^2 = \mathbf{c}$ and

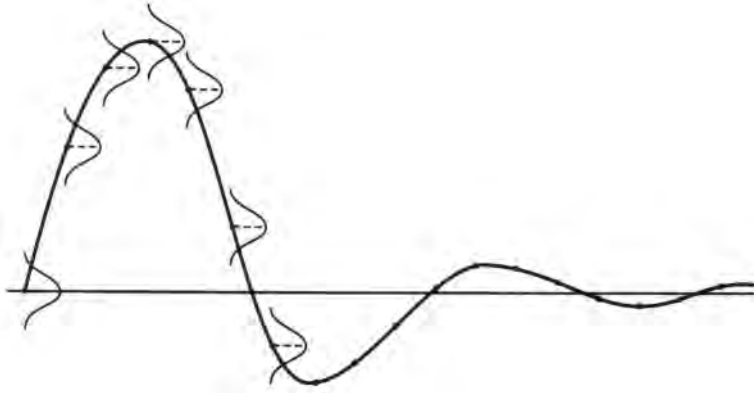


Figure 11.20. Illustration of one scheme for biasing by mean translation: the mean is time-varying and proportional to the impulse response.

$c = -v(\mathbf{0}, \mathbf{s})$ will be the right value.[†] Again, in principle, one could customize c to the particular ISI pattern, e.g., $\boldsymbol{\mu} = c, \mathbf{h}$, so that the resulting error rate for *every* ISI pattern would be 0.5. Although the degree of detailed knowledge implied by such a procedure would exceed our expectations, and indeed would obviate the need for simulation, it does demonstrate the utility of biasing by translation. This approach was initially suggested in this context by Lu and Yao⁽³⁴⁾ and is often referred to as *improved* importance sampling (IIS). A less demanding version of this idea will be addressed in the example below. However, any such procedure (except for one case) necessitates a “block” simulation methodology, for if $\boldsymbol{\mu}$ has been optimized for a particular ISI pattern, it cannot be simultaneously optimum for two overlapping patterns. Hence, nonoverlapping or independent blocks must be separately simulated for every ISI pattern (or every pattern chosen to be simulated) and by the nature of this method, only a single symbol per block can be the observed symbol.

Figure 11.21 summarizes the two basic biasing options that we have just discussed. It will be noticed that in these two approaches, only the additive Gaussian noise source is biased: this has been the most common application of IS. However, in certain problems, it may be useful to also bias other (generally non-Gaussian) processes, possibly in addition to an additive noise source. For example, in a multipath channel, it would be advantageous to bias the channel fading process in such a way that deep fades occur more often. In optical applications the receiver signal level, which is the output of a photodiode, is itself a non-Gaussian random process. It has been shown by Balaban⁽³⁰⁾ that appropriately biasing the photodiode current distribution, differently for ones than for zeros leads to large run-time improvements.

■ **Example 11.2.5.** Here we provide some quantitative feel for the possible run-time improvement due to IS. We assume the simple linear system.[‡]

$$V_i = \mathbf{h} \cdot \mathbf{A} + \mathbf{h} \cdot \mathbf{n}$$

where $\mathbf{A}$ is the sampled signal sequence, again conditioned on a zero being sampled, and $\mathbf{h}$ is M -dimensional and again normalized to unit energy. (Since $\mathbf{h}$ is linear, it is understood that $\mathbf{A}$

[†]This choice is intuitively appealing because in M dimensions, $\boldsymbol{\mu} = c\mathbf{h}$ is in the “direction” of $\mathbf{h}$.

[‡]As implied previously, examples inevitably require simplifying assumptions if they are to be reasonably contained. This does not mean that IS is applicable only to such cases, as evidenced in actual applications (e.g., Refs. 35 and 43), but these applications do reveal the complexity of employing this tool in a real environment.

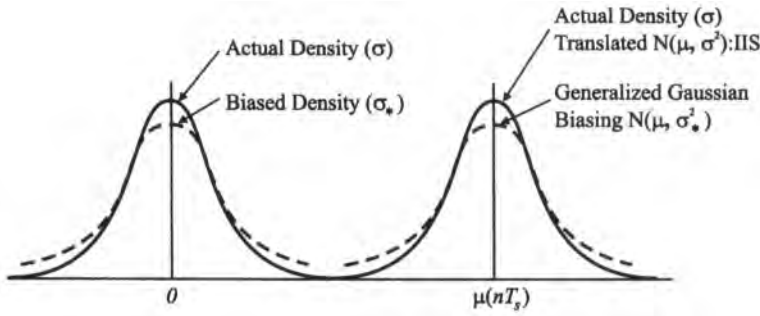


Figure 11.21. Illustration of biasing options for Gaussian distributions: variance scaling, mean translation, or a combination.

and $\mathbf{n}$ are actually the time-reversed sequences.) We will also assume that only the noise is biased. Using (11.2.71a), we then obtain

$$N_0 \sigma^2(\hat{p}_{0,*}) = \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(\mathbf{h} \cdot \mathbf{A} + \mathbf{h} \cdot \mathbf{n}) f_1(\mathbf{s}) f(\mathbf{n}) w(\mathbf{n}) d\mathbf{s} d\mathbf{n} - p^2$$

We will assume that $\mathbf{n}$ is Gaussian with independent components, each $\mathcal{N}(0, \sigma^2)$, and look at biasing using the CIS and IIS methods. However, we consider an approach which is not ISI sequence-dependent, even though such an approach will be theoretically less powerful. The advantage is that it does not demand the detailed knowledge otherwise required, and results in a simpler IS experiment. In other words, we only need to find one best value for the biasing parameter, γ or $\boldsymbol{\mu}$ for CIS or IIS, respectively. To do so implies averaging out the effect of the ISI pattern. To this end, we write the previous equation as

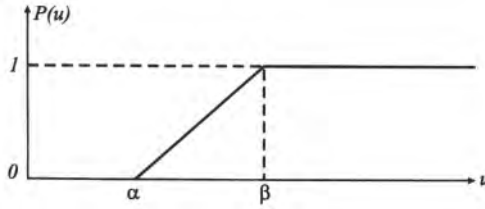
$$\begin{aligned} N_0 \sigma^2(\hat{p}_{0,*}) &= \int_{-\infty}^{\infty} f(\mathbf{n}) w(\mathbf{n}) d\mathbf{n} \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(\mathbf{h} \cdot \mathbf{A} + \mathbf{h} \cdot \mathbf{n}) f_1(\mathbf{s}) d\mathbf{s} - p^2 \\ &= \int_{-\infty}^{\infty} \mathcal{P}(\mathbf{n}) f(\mathbf{n}) w(\mathbf{n}) d\mathbf{n} - p^2 \end{aligned}$$

where

$$\mathcal{P}(\mathbf{n}) = \int_{-\infty}^{\infty} I_{\mathcal{A}_0}(\mathbf{h} \cdot \mathbf{A} + \mathbf{h} \cdot \mathbf{n}) f_1(\mathbf{s}) d\mathbf{s}$$

can be interpreted as a conditional probability of error, averaged over the ISI patterns and conditioned on a particular noise vector $\mathbf{n}$. We will now assume that $\mathbf{f}^*$ is biased simultaneously in variance, $\sigma_*^2 = \gamma \sigma^2$, and its mean is $\boldsymbol{\mu}$. By a suitable change of variables (Problem P11.14) we can obtain

$$\begin{aligned} N_0 \sigma^2(\hat{p}_{0,*}) &= \left(\frac{r^2}{\sqrt{2r^2 - 1}} \right)^M \exp \frac{\|\boldsymbol{\mu}\|^2 (2r^2 - 1)}{\sigma^2} \\ &\quad \times \int_{-\infty}^{\infty} \frac{P(u)}{\sqrt{2\pi}\sigma r} \exp \frac{-|u - (1 - 2r^2)(\mathbf{h} \cdot \boldsymbol{\mu})|^2}{2\sigma^2 r^2} du - p^2 \end{aligned} \quad (11.2.73a)$$


 Figure 11.22. Illustration of the function $P(u)$ assumed for Example 11.2.5.

where because of linearity we can set $\mathbf{h} \cdot \mathbf{n} = u$ and $\mathcal{P}(\mathbf{n}) \Rightarrow P(u)$ with u still normal, and $r^2 = \gamma^2/(2\gamma^2 - 1)$.

Without specifically evaluating the last equation, we see the interesting fact that if $r = 1$ and $\mathbf{h} \cdot \boldsymbol{\mu}$ is independent of M , then the variance expression itself becomes independent of M , which is “pure” IIS, while in CIS, for which $\boldsymbol{\mu} = \mathbf{0}$, that is not the case. In order to evaluate the variance numerically, we need a specific form for $P(u)$. For computational ease we will assume the simple form

$$P(u) = \begin{cases} 0, & u \leq \alpha \\ \frac{u - \alpha}{u - \beta}, & \alpha < u < \beta \\ 1, & u \geq \beta \end{cases}$$

which is illustrated in Figure 11.22. For arbitrary $\boldsymbol{\mu} \in \mathbb{R}^M$ we now have the following result:

$$N_0 \sigma^2 (\hat{p}_{0,*}) = \left(\frac{r^2}{\sqrt{2r^2 - 1}} \right)^M \frac{\exp[\|\boldsymbol{\mu}\|^2 (2r^2 - 1)/\sigma^2]}{B - A} \\ \times [BQ(B) - AQ(A) + \phi(A) - \phi(B)] - p^2$$

where

$$A = \frac{\sqrt{\rho}}{r} \left[1 - \delta + \frac{(\mathbf{h} \cdot \boldsymbol{\mu})(2r^2 - 1)}{\sigma \sqrt{\rho}} \right] \\ B = \frac{\sqrt{\rho}}{r} \left[1 + \delta + \frac{(\mathbf{h} \cdot \boldsymbol{\mu})(2r^2 - 1)}{\sigma \sqrt{\rho}} \right] \\ \delta = \frac{\beta - \alpha}{\beta + \alpha} \\ \rho = \frac{[0.5(\beta + \alpha)]^2}{\sigma^2} \equiv \text{SNR} \\ Q(x) = \int_x^\infty \phi(y) dy \\ \phi(y) = (\sqrt{2\pi})^{-1} e^{-y^2/2}$$

While this result is now straightforward to evaluate, we reduce it still further for purposes of this example. In particular, we assume $\alpha = \beta$, which makes $P(u)$ a step function.[†] The following result is easy to establish from the original equation (rather than as a limit of the

[†]This is equivalent to saying there is no ISI.

previous one):

$$N_0 \sigma^2(\hat{p}_{0,*}) = \left(\frac{r^2}{\sqrt{2r^2 - 1}} \right)^M \left[\exp \frac{\|C\|^2(2r^2 - 1)}{\sigma^2} \right] Q(A') - p^2 \quad (11.2.73b)$$

with

$$A' = \frac{\sqrt{\rho}}{r} \left[1 + \frac{(\mathbf{h} \cdot \mathbf{C})(2r^2 - 1)}{\sigma \sqrt{\rho}} \right]$$

We will specifically evaluate now “pure” CIS, for which $\boldsymbol{\mu} = \mathbf{0}$:

$$N_0 \sigma^2(\hat{p}_{0,*}) = \left(\frac{r^2}{\sqrt{2r^2 - 1}} \right)^M Q\left(\frac{\sqrt{\rho}}{r}\right) - p^2$$

and “pure” IIS, for which $r = 1$, and the specific choice $\boldsymbol{\mu} = c\mathbf{h}$ with $|\mathbf{h}|^2 = 1$:

$$N_0 \sigma^2(\hat{p}_{0,*}) = e^{c^2/\sigma^2} Q[\sqrt{\rho}(1 + (c/\sigma)/\sqrt{\rho})]$$

Our interest is to compare these results to the MC case, for which the variance $\sigma_{MC}^2 = p(1 - p)/N_{MC}$. We can thus compute the ratio of time–reliability products

$$\mathcal{F} = \frac{N_{MC} \sigma_{MC}^2}{N_0 \sigma^2(\hat{p}_{0,*})}$$

which gives the improvement due to IS relative to MC. In either CIS or IIS, there is a parameter to optimize, namely r in the former and c in the latter. Rather than present the actual values of $\mathcal{F}$, it may be more instructive to show a measure of the actual amount of work to estimate the BER using one or another form of IS. Recall that in pure MC, the number of symbols to be observed N is of the form K/p , where the value of K is determined by the relative precision. Hence, the actual number of symbols that would have to be run with IS would simply be

$$N_* = \frac{K}{p\mathcal{F}}$$

In Figure 11.23 for $K = 1$ we plot N_* as a function of p for IIS and for CIS, the latter with M as a parameter. ■

11.2.5.3.2. A Formal Approach The two biasing strategies discussed above were based on intuitive notions of how to increase the production of errors. Example 11.5 makes it clear that one of these approaches is superior in terms of run-time improvement.[†] The difference between the two becomes increasingly larger as the error rate becomes smaller. The question arises whether it is possible to formulate a more formal or systematic framework that would allow the identification of biasing densities that are “optimal” in the sense that the computational work involved remains relatively fixed as p tends toward zero, as appears to be the case for IIS in the example above, or perhaps does not increase too rapidly over a range of p which corresponds to realistic requirements. While different possibilities have been suggested, it appears that the most suitable approach for this context is one based on the

[†]While run-time improvement is clearly the objective, once the run time has been made small enough, other considerations can make one technique more preferable than another.

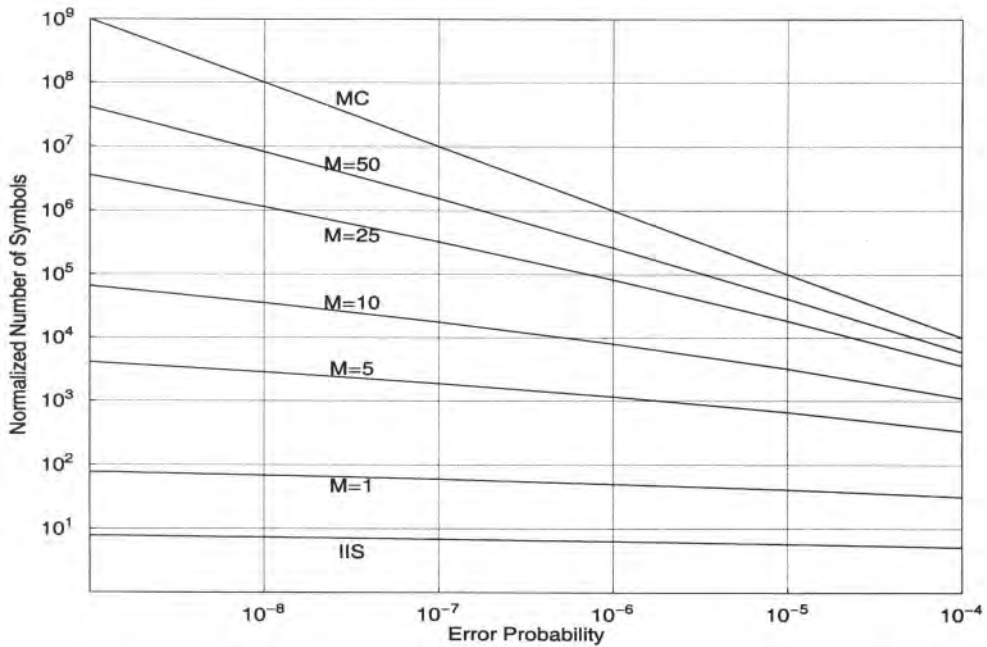


Figure 11.23. Results of Example 11.2.5. Normalized computational requirements for MC, IIS, and CIS with various amounts of memory M measured in number of simulation samples.

theory of large deviations.⁽³⁶⁾ Large-deviations theory (LDT) requires a great deal of technical apparatus to expose, which is beyond our intended scope. Our objective here is to merely to provide a broad-brush appreciation of the general ideas with a minimum of technical details, for which the reader is referred to, e.g., Refs. 47 and 48.

The formulation is typically set up as follows: Consider a sequence of random vectors $\{\mathbf{Y}_n; n = 1, 2, \dots\}$ in $\mathbb{R}^M$ for which we wish to compute the probability

$$p_n = P(\mathbf{Y}_n \in \mathcal{E}) = \int_{\mathbb{R}^M} I_{\mathcal{E}}(\mathbf{y}) f_n(\mathbf{y}) d\mathbf{y}$$

In our case, $\mathcal{E}$ represents an error region, say for symbol j , and $\mathbf{Y}_n$ stands for whatever processes we wish to simulate: so, $\mathbf{Y}_n$ here takes the place of $\mathbf{\Omega}$ in our earlier discussions, f_n is the density of $\mathbf{Y}_n$. Consequently, patterning ourselves after (11.2.71), we can write for the variance of the estimator of p_n

$$\sigma^2(\hat{p}_{n,*}) = \frac{1}{N} \int_{\mathbb{R}^M} I_{\mathcal{E}}(\mathbf{y}) \frac{f_n^2(\mathbf{y})}{f_n^*(\mathbf{y})} d\mathbf{y} - \frac{p_n^2}{N} \quad (11.2.74)$$

which is typically written in the form

$$\sigma^2(\hat{p}_{n,*}) = \frac{1}{N} [\eta_n(f^*) - p_n^2] \quad (11.2.75)$$

where

$$\eta_n(f_n^*) = \int_{\mathbb{R}^M} I_\varepsilon(y) f_n^2(y) / f_n^*(y) dy$$

in order to make the dependence on the choice of the biasing density f_n^* explicit; the density f_n is given. Recalling the definition of relative precision

$$\varepsilon = \frac{\sigma(\hat{p}_{n,*})}{p_n}$$

and making use of (11.2.75), one obtains

$$N_n = \frac{\eta_n(f_n^*) - p_n^2}{\varepsilon^2 p_n^2} \quad (11.2.76)$$

where N has now been subscripted with n to indicate that for a given value of ε , its value will generally depend on n . Indeed, this is the crux of the matter. Let us rewrite (11.2.76) as

$$N_n = \frac{[\eta_n(f_n^*)/p_n^2] - 1}{\varepsilon^2} \quad (11.2.76a)$$

If $p_n \rightarrow 0$ as $n \rightarrow \infty$, it is clear that the work required to estimate p_n will increase without limit unless $\eta_n(f_n^*) \sim p_n^2$. (The symbol $\sim$ stands for asymptotically equal to.) In the LDT context, conditions are established on f_n that enable a “large-deviations principle” to hold, namely, that p_n tends to zero exponentially fast as $n \rightarrow \infty$. Under these conditions we can write

$$p_n \sim c_n e^{-I(\mathcal{E})n} \quad (11.2.77a)$$

or, in its more usual form,

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log(p_n) = -I(\mathcal{E}) \quad (11.2.77b)$$

The function $I(\cdot)$ is called the large-deviations rate function, and $I(\mathcal{E})$ is its smallest value in $\mathcal{E}$.

We have just alluded to a “large-deviations principle,” and for completeness we give a more formal definition. A sequence $\mathbf{Y}_1, \mathbf{Y}_2, \dots$ satisfies a large-deviations principle with rate function I if for every set $\mathcal{E}$ in C_I

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log P(\mathbf{Y}_n \in \mathcal{E}) = -\inf\{I(y) : y \in \mathcal{E}\}$$

where C_I is a particular class of subsets[†] of $\mathbb{R}^M$.

If $\eta_n(f_n^*) \sim p_n^2$, this means we can write $\eta_n(f_n^*) = k_n^2 e^{-2I(\mathcal{E})n}$ as $n \rightarrow \infty$, hence $N_n \rightarrow [(k_n/c_n)^2 - 1]/\varepsilon^2$, which means that N_n does not increase exponentially as $p_n \rightarrow 0$ if

[†]The sets in C_I are called I -continuity sets (see, e.g., Ref. 48). In all cases of interest here $\mathcal{E}$ is an I -continuity set.

k_n/c_n tends to a finite limit. A sequence of simulation densities $\{f_n^*$ is said to be asymptotically efficient (a.e.) if

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log[\eta_n(f_n^*)] = -2I(\mathcal{E}) \quad (11.2.78)$$

Roughly speaking, asymptotic efficiency means that the work to estimate p_n does not “explode” as the latter tends to zero. While this is a satisfying property, in practice the error probability is always far from zero, and it is not clear how well a simulation density satisfying (11.2.78) will perform in such a case.

■ *Example 11.2.6.* The simple case of binary signaling is often used as an example to show that the corresponding error probability as a function of signal-to-noise ratio obeys a large-deviations principle. Assume we transmit a zero represented by the voltage m_0 and let the noise be $\mathcal{N}(0, \sigma_n^2)$, i.e., zero-mean Gaussian with variance $\sigma_n^2 = \sigma_0^2/n$. Interpreting Y_n as signal plus noise, we have $Y_n \in \mathcal{N}(m_0, \sigma_n^2)$ and the error region is $\mathcal{E} = \{y : y \geq 0\}$, for which we know that $p_n = Q(\sqrt{\rho_n})$, where $\rho_n = (m_0/\sigma_n)^2 = nm_0^2/\sigma_0^2$. As $\rho_n \rightarrow \infty$, we have the well-known result

$$p_n \sim \frac{1}{\sqrt{2\pi\rho_n}} e^{-\rho_n/2}$$

or

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log(p_n) = -\frac{1}{2}$$

which shows that for this case $I(\mathcal{E}) = -1/2$. ■

For the sequel, we need to define a key concept called the rate function, introduced above. For the type of applications that we have in mind, the sequence of random vectors $\{\mathbf{Y}_n\}$ will satisfy the following conditions. First, the normalized (or scaled) log-moment generating function, defined as

$$\mu_n(\mathbf{s}) = \frac{1}{n} \log[E(\exp(n\mathbf{s} \cdot \mathbf{Y}_n))] \quad (11.2.79)$$

exists for every n , and second, the limit

$$\mu(\mathbf{s}) = \lim_{n \rightarrow \infty} \mu_n(\mathbf{s})$$

exists for every $\mathbf{s} \in \mathbb{R}^M$. Then, we define the rate function as

$$I(\mathbf{y}) = \sup_{\mathbf{s} \in \mathbb{R}^M} \{\mathbf{s} \cdot \mathbf{y} - \mu(\mathbf{s})\} \quad (11.2.80)$$

The function $E(\exp(\mathbf{s} \cdot \mathbf{Y}_n)) \triangleq M_n(\mathbf{s})$ will be recognized as the moment generating function (mgf) of $\mathbf{Y}_n$. A couple of examples will help to develop a sense of the nature of the rate function.

■ *Example 11.2.7.* Consider a sequence of i.i.d. Bernoulli random variables $\{X_i\}$ with

$P(X_i = 1) = p$ and $P(X_i = 0) = 1 - p$, and define

$$\bar{S}_n = \frac{S_n}{n} \quad \text{with} \quad S_n = X_1 + \cdots + X_n$$

From these definitions, we know that $E(X_i) = p$. Let $a > p$ and consider the probability

$$P(\bar{S}_n \geq a) = P(X_1 + \cdots + X_n \geq na) \quad (11.2.81)$$

By the weak law of large numbers

$$P(|\bar{S}_n - p| \geq a) \rightarrow 0 \quad \text{as} \quad n \rightarrow \infty$$

so the probability in (11.2.81) $\rightarrow 0$ as $n \rightarrow \infty$. We are interested in how *fast* this probability tends to zero with n .

Because the $\{X_i\}$ are Bernoulli, the sum $X_1 + \cdots + X_n$ is binomially distributed, so we have

$$P(S_n \geq na) = \sum_{k=\lceil na \rceil}^n \binom{n}{k} p^k (1-p)^{n-k} \quad (11.2.82)$$

The reader is invited to perform the following “computer experiment.” Calculate (11.2.82) for a set of values for n , then plot $\log P(S_n \geq na)$ as a function of n . Performing this experiment, one would find (as shown in Figure 11.24 for $a = 0.6, p = 0.35$, and $n = 1, 2, \dots, 100$) that $\log P(S_n \geq na) \sim c + dn$ as $n \rightarrow \infty$. The irregularity of the curve in the figure is due to the fact that n is relatively still quite small. But the trend can definitely be seen

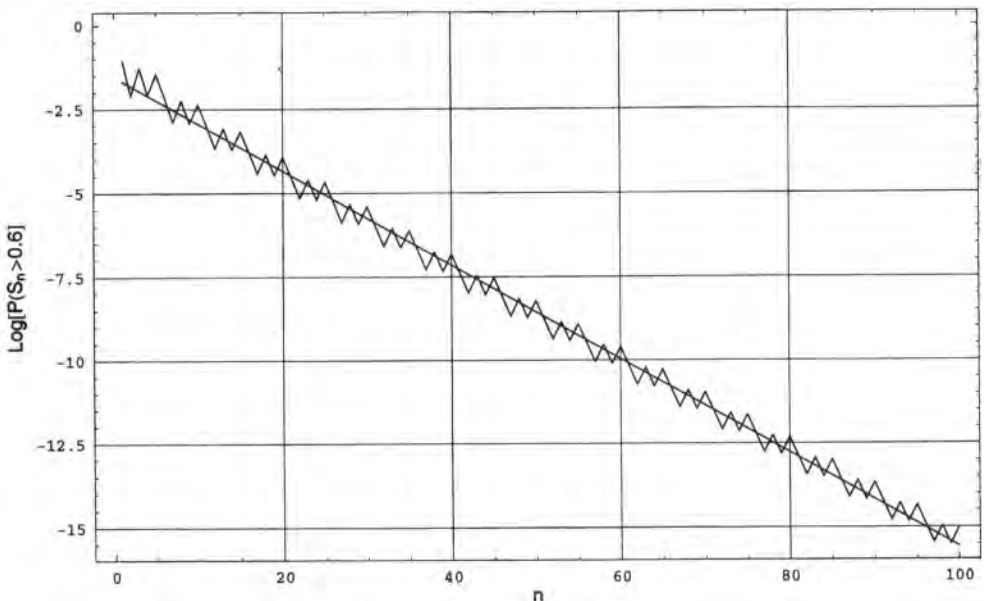


Figure 11.24. Result of the “computer experiment” in Example 11.2.7; the best-fit slope is -0.140 , while the theoretical slope for the parameters of the example is -0.129 .

by the best straight-line fit indicated on the figure. Thus, we would expect that

$$\lim_{n \rightarrow \infty} \frac{1}{n} \log P(X_1 + \cdots + X_n) \geq na = \lim_{n \rightarrow \infty} \left(\frac{c}{n} + d \right) = d$$

Now, if we repeated this experiment for a different value of a (other things unchanged), we would expect the constants c and d to change also. But in the limit, only the value of d would matter. So we could express this idea as $d = I(a)$, some function of a that we call the rate function.

In this case, the rate function is actually known⁽⁴⁸⁾ to be

$$I(a) = a \log \left(\frac{a}{p} \right) + (1 - a) \log \left(\frac{1 - a}{1 - p} \right)$$

which the reader can compare to the best-fit line mentioned above. ■

A second illustration of the computation of the rate function is worthwhile for a more typical problem in communications.

■ *Example 11.2.8.* Consider first, as in Example 11.2.6, that $Y_n \in \mathcal{N}(m_0, \sigma_n^2)$. It is shown in many texts that the mgf for a normal distribution is given by (in this case s is a scalar s)

$$M_n(s) = e^{sm_0} e^{s^2 \sigma_n^2 / 2}$$

so that

$$\mu_n(s) = \frac{1}{n} \log M_n(ns) = sm_0 + \frac{s^2 \sigma_0^2}{2}$$

which in this case is independent of n . Hence

$$\mu(s) = \lim_{n \rightarrow \infty} \mu_n(s) = sm_0 + \frac{s^2 \sigma_0^2}{2}$$

To obtain the rate function, consider y fixed and set

$$g(s) = ys - \mu(s) = ys - sm_0 + \frac{s^2 \sigma_0^2}{2}$$

Differentiating the last expression with respect to s and setting to zero

$$g'(s) = y - m_0 - s\sigma_0^2 = 0$$

we find the value of s that maximizes $g(s)$, namely $s = (y - m_0)/\sigma_0^2$. Substituting this result into the definition (11.2.79) produces

$$I(y) = \frac{(y - m_0)^2}{2\sigma_0^2}$$

In order to provide more insight into this computation, let us now consider a simple two-dimensional case, in which we assume $\mathbf{Y}_n \in \mathcal{N}(\mathbf{m}, \sigma_n^2 I)$ with independent components, I is

the 2×2 identity matrix, and $\mathbf{m} = (m_1, m_2)$. Then,

$$\begin{aligned} M_n(n\mathbf{s}) &= M_n(ns_1, ns_2) = M_n(ns_1)M_n(ns_2) \\ &= e^{nm_1s_1} e^{nm_2s_2} e^{n\sigma_0^2(s_1^2+s_2^2)/2} \end{aligned}$$

from which

$$\mu_n(s) = \frac{1}{n} \log M_n(n\mathbf{s}) = m_1s_1 + m_2s_2 + \frac{\sigma_0^2}{2} \|\mathbf{s}\|^2$$

which is again independent on n . Hence

$$\mu(\mathbf{s}) = \lim_{n \rightarrow \infty} \mu_n(\mathbf{s}) = v_1s_1 + v_2s_2 + \frac{\sigma_0^2}{2} \|\mathbf{s}\|^2$$

Now, the rate function is given by

$$I(\mathbf{y}) = \sup_{s_1, s_2} \left[s_1y_1 + s_2y_2 - \left(m_1s_1 + m_2s_2 + \frac{\sigma_0^2}{2} \|\mathbf{s}\|^2 \right) \right]$$

In analogy with the one-dimensional case, set $\mathbf{g}(s_1, s_2) = s_1(y_1 - m_1) + s_2(y_2 - m_2) - \frac{1}{2} \sigma_0^2(s_1^2 + s_2^2)$; we seek the values (s_1, s_2) for which $\nabla \mathbf{g} = \mathbf{0}$, namely

$$\begin{aligned} y_1 - m_1 - \sigma_0^2 s_1 &= 0 \\ y_2 - m_2 - \sigma_0^2 s_2 &= 0 \end{aligned}$$

whose solutions are $s_1 = (y_1 - m_1)/\sigma_0^2$ and $s_2 = (y_2 - m_2)/\sigma_0^2$, so that, introducing these values into $I(\mathbf{y})$ above, we finally obtain $I(\mathbf{y}) = (1/2\sigma_0^2) \|\mathbf{y} - \mathbf{m}\|^2$. ■

The main result arising from LDT is the formulation of the sequence $\{f_n^*\}$ which is asymptotically efficient. If we can synthesize a random number generator that produces sequences of numbers governed by f_n^* , for any n , we will have a putatively very good, if not optimal (for any particular n), simulation distribution. Actually, the form of $\{f_n^*\}$ depends on the problem at hand, specifically the geometry of the set $\mathcal{E}$. We will confine ourselves here to the relatively simple case when $\mathcal{E}$ has a *dominating point*.

A dominating point is a point $\mathbf{v} \in \mathbb{R}^M$ with the following properties:

- $\mathbf{v}$ is on the boundary of $\mathcal{E}$
- $\mathbf{v}$ is a minimum rate point, i.e., $I(\mathbf{v}) = \min\{I(\mathbf{y}): \mathbf{y} \in \mathcal{E}\}$
- The tangent to the boundary of $\mathcal{E}$ at $\mathbf{v}$ defines a half-space containing $\mathcal{E}$
- There exists a unique $\mathbf{s}_v \in \mathbb{R}^M$ such that $\nabla \mu(\mathbf{s}_v) = \mathbf{v}$

By its definition, if there exists a dominating point, it is unique. Heuristically, such a point “dominates” the error probability, for if there is such a point, it must correspond to the point yielding $I(\mathcal{E})$ in (11.2.72b), and this equation does indeed predict (asymptotically) the error probability. If there is more than one point in $\mathcal{E}$ for which the error probability is the same, but not exceeded elsewhere, such points are called minimum rate points, but they are not necessarily dominating points. In the latter instance, a more complicated formulation is

necessary for the a.e. sequence of distributions. In the case where a dominating point exists, the *unique* form of $\{f_n^*\}$ which is asymptotically efficient is given by⁽³⁶⁾

$$f_n^*(\mathbf{y}) = \exp\{n[\mathbf{s}_v \cdot \mathbf{y} - \mu_n(\mathbf{s}_v)]\} f_n(\mathbf{y}) \quad (11.2.83)$$

which is often referred to as an exponentially twisted density. We note that there is not always a dominating point, and some very common signaling schemes do not possess one, e.g., QPSK. In such cases, as alluded to just above, more elaborate constructions are necessary, and the reader is referred to Ref. 36. In this instance, the optimal $\{f_n^*\}$ is not given by (11.2.83).

■ *Example 11.2.9.* As our final example related to LDT, let us look at the implications of (11.2.83) when f_n is Gaussian. We will take the two-dimensional case $\mathbf{Y}_n \in \mathcal{N}(\mathbf{m}, \sigma_n^2 I)$, as the single-dimensional case follows trivially. The condition $\nabla \mu(\mathbf{s}_v) = \mathbf{v}$ defines $\mathbf{s}_v$, which must first be solved for. From the preceding example

$$\mu(\mathbf{s}) = m_1 s_1 + m_2 s_2 + \frac{\sigma_0^2}{2} \|\mathbf{s}\|^2$$

Hence, letting $\mathbf{s}_v = (s_{1,v}, s_{2,v})$ and $\mathbf{v} = (v_1, v_2)$, we have

$$\begin{aligned} \frac{\partial \mu_n(\mathbf{s}_v)}{\partial s_{1,v}} &= m_1 + \sigma_0^2 s_{1,v} = v_1 \\ \frac{\partial \mu_n(\mathbf{s}_v)}{\partial s_{2,v}} &= m_2 + \sigma_0^2 s_{2,v} = v_2 \end{aligned}$$

so that

$$\begin{aligned} s_{1,v} &= (v_1 - m_1)/\sigma_0^2 \\ s_{2,v} &= (v_2 - m_2)/\sigma_0^2 \end{aligned}$$

Substituting the last equations for $\mathbf{s}_v$ into the “twisting” exponent of (11.2.83), $n[\mathbf{s}_v \cdot \mathbf{y} - \mu_n(\mathbf{s}_v)]$, yields for that term, with $\mathbf{y} = (y_1, y_2)$,

$$\begin{aligned} & y_1(v_1 - m_1)/\sigma_n^2 + y_2(v_2 - m_2)/\sigma_n^2 - m_1(v_1 - m_1)/\sigma_n^2 - m_2(v_2 - m_2)/\sigma_n^2 \\ & - (v_1 - m_1)^2/2\sigma_n^2 - (v_2 - m_2)^2/2\sigma_n^2 \end{aligned} \quad (11.2.84)$$

Now,

$$f_n(\mathbf{y}) = \frac{1}{2\pi\sigma_n^2} \exp\left[\frac{-(y_1 - m_1)^2}{2\sigma_n^2}\right] \exp\left[\frac{-(y_2 - m_2)^2}{2\sigma_n^2}\right]$$

so that, combining the exponents with (11.2.84), we obtain after some manipulations

$$f_n^*(\mathbf{y}) = \frac{1}{2\pi\sigma_n^2} \exp\left[\frac{-(y_1 - v_1)^2}{2\sigma_n^2}\right] \exp\left[\frac{-(y_2 - v_2)^2}{2\sigma_n^2}\right] \quad (11.2.85a)$$

which is observed to be equivalent to

$$= f_n(\mathbf{y} + \mathbf{m} - \mathbf{v}) \quad (11.2.85b)$$

Hence we see that, in this particular case, $\{f_n(\mathbf{y})\}$ is merely a mean-shifted version of the original in each dimension, so that the density is now centered on $\mathbf{v}$. ■

The reader will have noticed that this is exactly the mean-shifted density proposed as an intuitive choice in the last section. However, this result arises specifically from the Gaussian assumption on f_n , probably the most important special case in the context of communications, but it may not apply to all cases of interest, or to geometries where a dominating point does not exist. In any case, although the subject is very “technical,” the preceding exposition should show its promise, and for problems dealing with estimation of very low error probability, the work in developing an LDT formulation may be quite cost-effective.

11.2.5.4. Stochastic Importance Sampling

We have seen that it is generally not possible, or at least very difficult, to analytically determine a biasing scheme in its entirety. For example, in Example 11.2.5 we saw that in either form of IS discussed, there is at least one parameter to optimize. The determination of such parameters embodies one of the difficulties just alluded to. To further illustrate this point, refer to Figure 11.25, which shows the parameter $\mathcal{F}$ for the IIS case from Example 11.2.5 for $p = 10^{-8}$. It can be seen that there is a considerable range in the values of $\mathcal{F}$ as a function of the translation parameter c . Furthermore, the optimum value of c is different for different BERs. Thus, we cannot know the optimum value of c unless we knew the BER in the

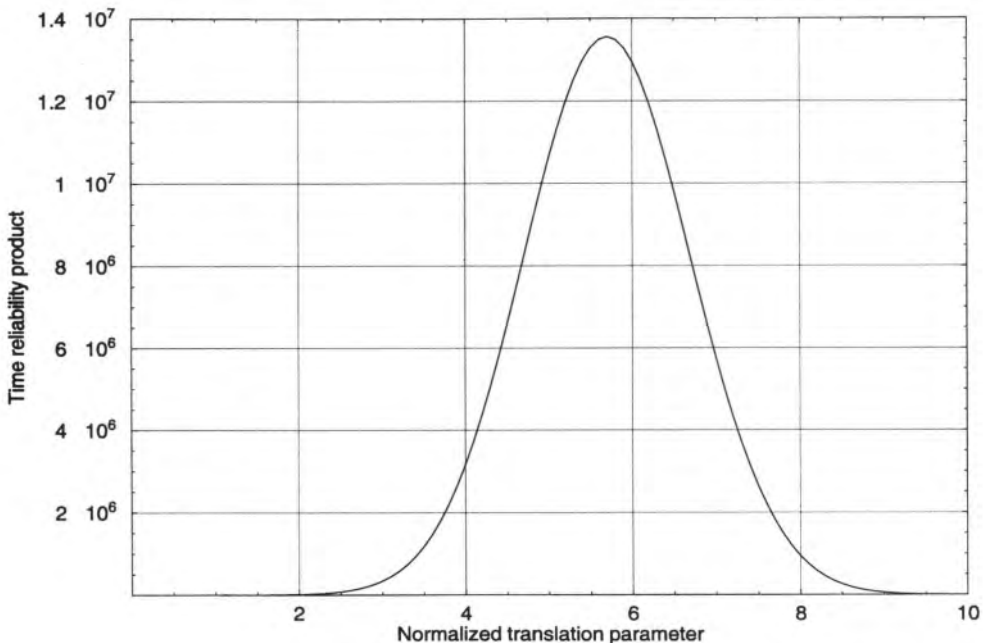


Figure 11.25. Improvement factor for IIS in Example 11.2.5 for $p = 10^{-8}$ as a function of the normalized translation parameter.

first place, which is exactly what we are looking for! Means of circumventing this type of difficulty are necessary in order to put IS to practical use. A class of approaches for dealing with this issue is referred to as *stochastic importance sampling* (SIS). These techniques are most beneficial when it is known that a parametric biasing technique would be effective, although the complexity of the problem does not allow us to obtain the optimum biasing parameters directly. Near-optimal values for the biasing parameters are determined by a stochastic optimization routine, such as mean-field annealing or stochastic gradient descent. The optimization routine minimizes a cost function, which is the estimator variance. Thus, the optimization routine controls Monte Carlo simulation runs of the system being modeled to obtain estimates of the estimator variance. Once the optimization phase is completed, the near-optimal parameter values are then used in the simulation to obtain estimates of the desired probabilities.

Stochastic importance sampling techniques have been applied to a number of communication links ranging from simple systems with additive white Gaussian noise to systems with adaptive equalization and Rayleigh fading.^(45,49,50) We outline here one of these approaches,⁽⁴⁵⁾ which is based on a succession of small trial simulations which can home in to a narrow range of values of the translation parameter in the neighborhood of the optimal value, if not the optimal value itself. The procedure may be thought of as “adaptive” in a rough sort of way.

On the surface, the idea of successive simulations might appear to be self-defeating. The key, however, to the utility of the approach is that each of these trial simulations must be very short. The question naturally arises, then, as to how we can learn much from short simulations if the error rate is on the order of 10^{-8} , say. The answer lies in Figure 11.26. The underlying character of this figure can be formally explained,⁽⁴⁵⁾ but here we describe in heuristic terms the gist of the idea as applied to IIS (mean translation). As implied previously, in IIS a simulation is structured in the following way. For a particular ISI pattern, we send through the system a block of bits whose length equals the memory of the system. We shall call one such block transmission a run. A run results in one decision. A simulation is a set of N runs. For

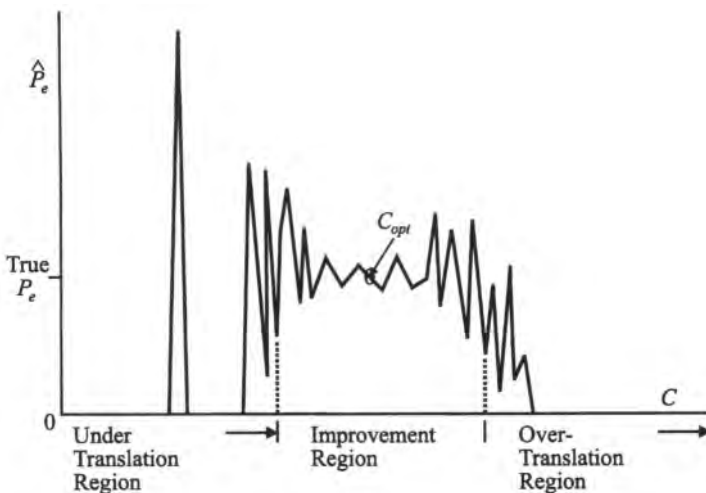


Figure 11.26. A typical curve of estimated BER from simulation, versus the translation parameter C . The labeled regions are explained in the text (from Ref. 43, © IEEE, 1993).

discussion purposes, let us suppose that a simulation consists of 100 runs. With $c = 0$, i.e., an unbiased simulation, and $p = 10^{-8}$, we can expect that with $N = 100$ runs we will almost never see an error. Therefore, the IS BER estimator given in (11.2.62), for example, would almost always be zero. We now increment c by some amount, and we can reasonably expect that for relatively small amounts of translation we would likewise rarely see any errors. Thus, if we were to plot estimated BER versus the translation parameter for $N = 100$, we would almost always see zero for that estimate if c were sufficiently below the optimal value, namely in the range labeled “undertranslation.” This is indicated in Figure 11.26, which also shows the possibility of a BER “blip” if perchance there happens to be an error in that short a run: in that case the “estimated” BER, $\hat{p}_*$, would be $1/100$, much higher than p , the expected BER neighborhood. Let us now look at the opposite case of “overtranslation,” that is, use a value of c much larger than optimal. This means almost all of the biased density lies in the error region. Thus, virtually every run will yield an error. But each error is now multiplied by the weight $w(\mathbf{\Omega}_i)$, which will be very small. To illustrate this point, suppose that $\mathbf{\Omega}$ is a single-dimensional Gaussian random variable n so that $w = f(n_i)/f^*(n_i)$. With f a normal distribution and n_i drawn from f^* , the translated distribution, it can be seen that for almost all n_i the weight will be very small. Hence, in the overtranslation region, the estimated BER will also be very small, as indicated in the corresponding region of Figure 11.26.

As c approaches the optimal value, we can expect the BER estimate to tend to become better behaved because it will now assume approximately correct values. We can deduce this from the fact that at or near the optimal c the improvement becomes large enough to produce a legitimate estimate. (This range of c is called the “improvement region.”) For example, we see from Figure 11.23 that the number of samples required for the optimal c is less than 10 for practically any BER with the relative precision constant $K = 1$. Acceptable precision would dictate $K \gtrsim 10$, so that with 100 runs we can indeed expect that for some range about the optimal c we will obtain estimates with a reasonably small variance, as indicated in the figure. In fact, rather than merely obtaining a range of c to focus on, as just discussed, we may well obtain a good enough estimate of the BER itself where we see the least fluctuation in $\hat{p}_*$. Of course, this succession of trials reduces the overall improvement that we might have had. But if c is reasonably discretized, because the original improvement is so large we can still get a very substantial reduction in run time.

11.2.6. Efficient Simulation Using Importance Splitting

11.2.6.1. Introduction

We have seen that the difficulty in applying importance sampling is obtaining the proper biasing density. In the tail extrapolation method discussed earlier, we obtained a lesser, though useful, improvement only by changing the threshold. This can be more fruitfully interpreted as a modification of our observational paradigm. We now briefly introduce a variant of this general idea, that is, a method which does not alter the experiment in the sense that the models of equipment and random processes are unchanged, but institutes a more “clever” observational procedure. Like importance sampling, this procedure is not passive, but “proactive” in inducing the rare events of interest. This technique is called importance splitting,^{51–54} though other terminology is also found. It has successfully been applied to rare-event simulation of communication *networks*, but holds promise for communication link simulation. We provide here only a very brief introduction.

The rare-event simulation methodology is called trajectory splitting. The fundamental idea of splitting is based on the assumption that there exist some well-identifiable intermediate system states that are visited much more often than the target states of interest. This idea is most easily understood in the context of a queueing system, which is a common modeling abstraction in the performance analysis of communication networks. For example, if the target states represent a full queue in a queueing system, the states that correspond to the situation when the queue is at least half full can be regarded as intermediate states.

In a communications link, these states could be defined based on the relative distance between the receiver output sample at each unit of simulation time and the decision threshold. The system trajectory would therefore be defined based on this distance. Target states in this case would correspond to distances that are zero or negative.

A very important feature of splitting is that the step-by-step evolution of the system follows the original probability measure, i.e., no biasing of the underlying random processes is required. Entering the intermediate states during the simulation—which is usually characterized by the crossing of a *threshold* by a *control* parameter—triggers the splitting of the trajectory. The current system state is saved and a number of independent *subtrajectories* are simulated from that state. Achieving considerable variance improvement usually requires a hierarchical system of splitting conditions (a set of thresholds). Any events counted during these artificial subtrajectories are weighted by the reciprocal of the number of subtrajectories to obtain an unbiased estimate.

Prior knowledge about the system is used to determine subset boundaries. For example, in a queueing system, rare subsets of states would correspond to a longer queue, and would likely be a prerequisite to the rare events of interest.

Existing splitting techniques assume different restrictions on how the splitting conditions are defined, apply different rules to terminate subtrajectories, and use different approaches to obtain the unbiased parameter estimates. To explain the above key features, we use the splitting technique based on *direct probability redistribution* (DPR) as an example.⁽⁵¹⁾ DPR partitions the state space $\mathcal{S}$ of a Markovian system into m subsets $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m$ by a mapping function $\Gamma(s_i) \in [1, m]$ and assigns oversampling factors $\mu = \{\mu_1, \dots, \mu_m\}$ ($\mu_1 \leq \mu_2 \leq \dots \leq \mu_m$) to each subset. $\Gamma(s_i)$ is the subset indicator, i.e., it assigns each state its subset index.

Oversampling factors are determined in advance of the simulation using a relatively fast iterative technique which chooses these factors so all subsets are visited with nearly equal probability.

Splitting occurs whenever the system enters a given subset from a lower indexed subset. The number of subtrajectories and their terminating conditions are defined⁽⁵¹⁾ such that every state in subset $\mathcal{S}_i$ is visited (relatively and in the steady-state sense) μ_i times more often than in a Monte Carlo simulation. By assigning high oversampling factors to low-probability subsets, we can ensure that they are visited much more often during the DPR simulation. If the partitioning is chosen properly, this should also result in more samples in the important region. Since each state s_i is oversampled by the factor $\mu_{\Gamma(s_i)}$, unbiased estimates can be obtained by weighting a subset-dependent factor $1/\mu_{\Gamma(s_i)}$; see Figure 11.27.⁽⁵²⁾

In Ref. 51, it is pointed out that DPR simulation has an equivalent Markov chain representation, and that the resulting Markov chain can be derived from the original Markov chain by manipulating the transition probabilities in a systematic way. Thus, splitting can be regarded as an indirect IS technique, which instead of modifying individual stochastic sources, changes the entire transition probability structure of the system.

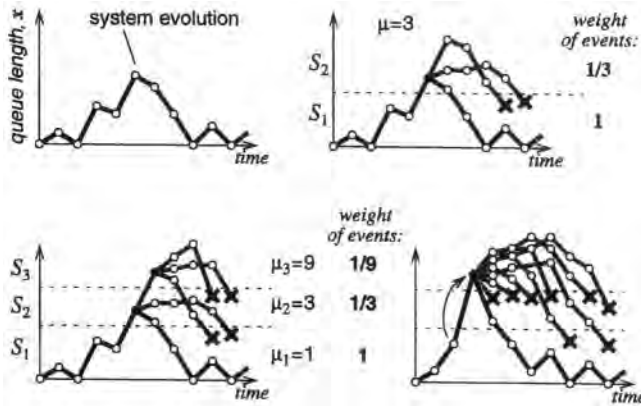


Figure 11.27. Example of importance splitting. Splitting increases the probability of reaching rare states by splitting the system trajectory upon entering well-defined subsets of the state space. Upper left: a sample path of the original single-queue system where queue length is plotted versus discrete time. Upper right: queue length partitioned into two subsets. Three independent subtrajectories are generated when the subset S_2 is entered (since $\mu = 3$, as determined from an independent exploration process). Each important event counted while the system is in subset S_2 must be weighted by a factor of $1/3$ to yield a correct estimate. Lower left: an example illustrating subtrajectories for the same system with three subsets. Lower right: the same example as in the lower-left plot, except here the system evolution skips a subset. (From Ref. 51, © IEEE, 1998.)

11.2.6.2. Application of DPR-Based Splitting Simulation

Systems with internal control loops cannot be handled easily with input-biased IS techniques. To demonstrate one of the main advantages of DPR-based trajectory splitting, here we consider a simple system with a control loop: a queueing system with feedback⁽⁵²⁾ (see Figure 11.28). The model can be regarded as a generic building block for asynchronous transfer mode (ATM) switches with internal traffic control. Incoming cells from the independent ON/OFF sources are multiplexed to the outgoing main buffer via a nonblocking shared bus. The buffer can store up to K cells. This model could represent the connection of N different computers to a multiplexer in a data network. In this case, each ON/OFF source would model the output data traffic of a computer. To minimize the occupancy of the main

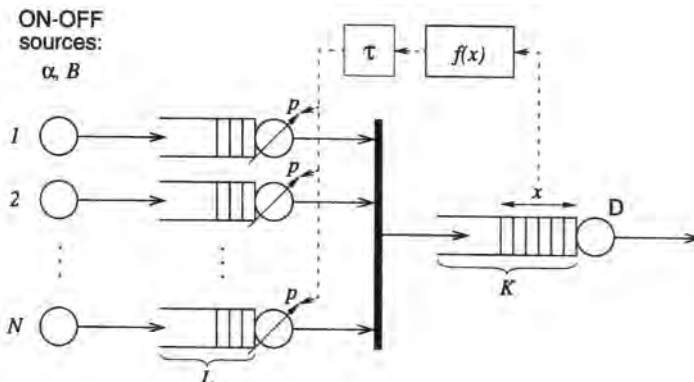


Figure 11.28. Model of an ATM multiplexer stage with internal traffic control, for the discussion of Section 11.2.6.1 (from Ref. 51, © IEEE, 1998).

buffer and the probability of losing cells, a feedback signal is broadcast by the main buffer to the input ports. The main buffer is required to smooth the bursty nature of the data traffic arriving from the N input ports. The randomly generated data from the N data sources results in a nonzero probability of cells overflowing the main buffer and being lost. In our generic model the feedback signal is a probability value p with which input ports are allowed to pass cells to the bus. Cells that are held back are stored at the input ports in buffers with capacity L cells. This feedback mechanism throttles back the data coming into the main buffer and reduces the probability of cell loss. The value of p is derived from the actual queue length x via the function $f(x)$, and the feedback latency is modeled by a constant delay factor τ .

To demonstrate the speedup factor in execution time when compared to conventional Monte Carlo techniques, we obtained the queue length probability mass function (pmf) using DPR-based splitting for the case $k = 84$ cells, $L = 500$ cells, and $\tau = 3$. The number of sources used was $N = 16$ and the mean burst length was $B = 3$ cells. We used a two-threshold feedback function where $f(x) = 1$ if $x < 44$, $f(x) = 0.5$ if $44 < x \leq 74$, and $f(x) = 0$ otherwise.

The total simulation time required to generate a complete pmf with probabilities down to 10^{-18} (including the iterative subset balancing procedure) was less than 40 min using a conventional desktop workstation. The corresponding factors of speedup over conventional Monte Carlo techniques varied between 10^4 ($\alpha = 0.6$) and 10^{13} ($\alpha = 0.2$).

11.2.7. Quasianalytical (Semianalytic) Estimation

In the broadest terms, quasianalytical (QA)[†] techniques are combinations of some elements of simulation and mathematical analysis. There is not a unique QA procedure. The particular combination of simulation and analytical tools to be used depends on the problem at hand. Under this definition, any simulation which is not *as-is* Monte Carlo is some form of QA. The “analytical” part of QA implies some *a priori* knowledge (or assumptions) which allows the construction of an abstraction that can be dealt with “analytically.” It is this prior information that holds the potential for run-time reduction since, in a manner of speaking, any such information does not have to be extracted from a simulation run—conversely, the absence of *a priori* knowledge or assumptions implies that we need to run a Monte Carlo simulation, which is the most time-consuming. QA techniques are not derived from standard statistical estimation procedures, and are perhaps best viewed as variations of simulation methodology. Hence, the associated estimator cannot be easily interpreted as an estimator with a distribution. Rather, it is best to look upon QA techniques as computational methods which provide results with varying degrees of approximation, depending on the case at hand.

Most applications of QA techniques have the following common theme. The system, whatever it is, is assumed to be linear at least from the point of entry of all Gaussian noise sources till the input to the decision device (see Figure 11.29). Thus, prior to the first noise source, the system could be nonlinear. (Variations on this theme, called *mixed* QA, will be considered below.) The utility of this assumption is twofold: first, as is well known, a linearly filtered Gaussian process remains Gaussian; and second, because the system is linear from the point of the noise insertion, superposition holds. This has two important implications. Superposition implies that the signal and the noise can be dealt with separately and the corresponding responses added at the end. Second, the fact that the noise (if Gaussian)

[†]The term *semianalytic* is also used synonymously by many workers.

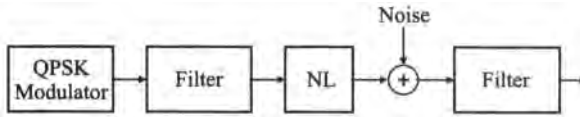


Figure 11.29. Example of system layout where QA techniques can be applied.

remains Gaussian suggests the possibility of omitting its explicit generation, since this very fact tells us that at the decision sampling instant a normally distributed random variable can be added to the noiseless sample to form the decision variable. Thus, the general idea is to separate the problem into two parts, one dealing with the generation of the noiseless signal (the simulation part) and the other with the noise contribution to the sampled waveform (the analytical part). Conceptually, the process consists in replacing the actual noise sources by an “equivalent” noise source (ENS) just prior to the decision device (Figure 11.30). This methodology can be generalized, at least in principle, to permit an arbitrary distribution for the ENS. Once we admit that possibility, we can also (again in principle) remove the linear assumption and postulate that the ENS might account for all noise-induced effects. Thus, the general setting for the theme under discussion can be summarized as follows:

- All of the effects of noise can be considered additive at the input to the decision device, irrespective of how many noise sources there are or where they are located. This artificial noise source is the ENS.
- The probability density function of the ENS is of a known form specified by the user.
- The simulation is conducted by shutting off all noise sources and generating only signal; this will produce all the ISI patterns under that condition.

The efficiency of the QA method arises in part because noise does not have to be explicitly generated, although this advantage is partly offset by another computation, as will be indicated. However, by far the bulk of the run-time improvement is due to the ability to run, in many cases, a sequence of symbols whose length is independent of the BER of interest and, again in most cases of interest, far shorter than that which would be required in a Monte Carlo simulation. This circumstance comes about if we can assume that the system has finite memory, which we have argued before is a reasonable assumption. Even for IIR systems, a finite “effective” memory can always be defined. If we label by m the effective memory measured in symbols, then there are L^m possible symbol patterns to be dealt with, and, as we saw in Chapter 7, a pseudo-random shift-register (de Bruijn) sequence of length precisely L^m is the shortest sequence that will replicate each pattern exactly once. For a binary system ($L = 2$) and $m = 7$, say, the corresponding sequence length is 128. A Monte Carlo run for a

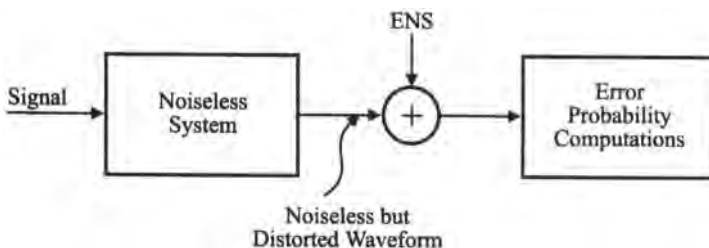


Figure 11.30. Illustration of generic simulation structure for applying QA techniques.

BER of, say, 10^{-5} would require running at least 10^6 bits. So in this case a run-time advantage of $10^6/128 \approx 8000$ would result. On the other hand, for $L = 16$ and the same m we have $16^7 \approx 2.7 \times 10^8$, which gives quite a different picture. (We will have more to say later about the $L > 2$ case.) Evidently, we can get an improvement even here for low enough BER, but the absolute run time is still prohibitive. It should now also be clear that the size of m will have a great impact on the sequence length. Therefore, circuits which have long memory, such as synchronization subsystems, cannot be explicitly simulated if we are to take full advantage of the QA technique. Here, to take account of such “slow” effects, we have to conduct the conditional experiments that we have previously discussed.

In the following we will describe a number of applications and variations of the QA method.

11.2.7.1. General Scheme for the QA Method

The underlying methodology for one class of applications of the QA method can be explained with respect to Figure 11.31. Two of the possible signals $\mathbf{s}_a$ and $\mathbf{s}_b$ in an L -ary two-dimensional constellation are shown, with their respective decision regions $\mathcal{D}_a$ and $\mathcal{D}_b$. Assume $\mathbf{s}_a$ is sent, and because of ISI and distortion it is received as $\mathbf{v}_a$. We will assume that the distortion is not so severe as to place $\mathbf{v}_a$ outside of the proper decision region. This is the usual assumption for the QA method and is obviously a reasonable one. In any case, it is easy to verify if there are noiseless errors, and this should probably be done in any case as a trouble-shooting check. Noise is added “analytically” to $\mathbf{v}_a$ so that the decision voltage $\mathbf{w}_a$ is

$$w_a = v_a + n$$

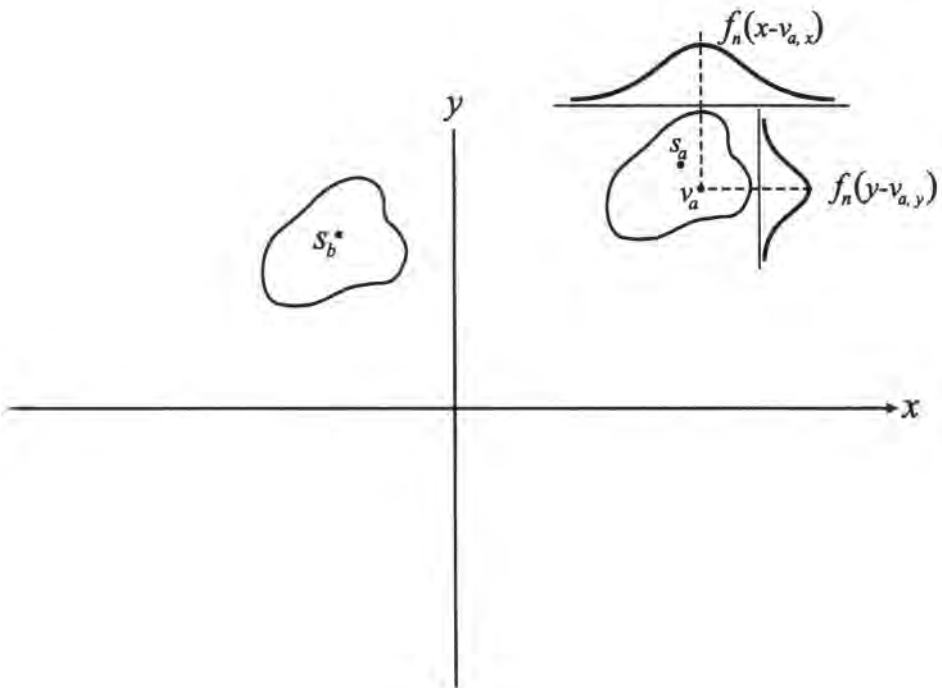


Figure 11.31. Illustration of general setup for QA computation of error rates for two-dimensional signaling schemes.

The probability of error, conditioned on s_a being sent, is

$$p_a = \int_{(x,y) \notin \mathcal{D}_a} f_{w_a}(x,y) dx dy \quad (11.2.86)$$

where f_{w_a} is the (two-dimensional) pdf of w_a . (For notational simplicity, we label here the I and Q channels by x and y , respectively.) We may also sometimes want the specific error probability that s_a is sent and received as s_b , namely

$$p_{ab} = \int_{(x,y) \in \mathcal{D}_b} f_{w_a}(x,y) dx dy \quad (11.2.87)$$

The pdf of w_a is simply

$$f_{w_a}(x,y) = f_n(x - v_{a,x}, y - v_{a,y}) \quad (11.2.88)$$

where f_n is the pdf of the noise and $v_{a,x}$ and $v_{a,y}$ are the I- and Q-channel components of v_a . We will assume in what follows that the x and y components of n are independent and that the corresponding pdf's belong to the generalized exponential family (11.2.46). The value v_a is obtained from the noiseless simulation, while the evaluation of p_a or p_{ab} is the analytical portion.

The QA technique is implemented generally by sending a PN sequence as mentioned, though this is not a requirement. Then, to obtain the average error rate, for example, Equation (11.2.86) is evaluated for every sample and averaged. The ease with which (11.2.86) or (11.2.87) can be evaluated depends on the shape of the decision region. Specific cases will be illustrated below.

As stated, because the QA method is not MC-based, it does not provide an estimator in the classical sense. If we can speak of bias and variance at all, it is in the following sense. If we had a linear system with strictly limited memory, then a distribution with $\nu = 2$ would be correct and a time-limited sequence would exhibit all possible realizations of intersymbol interference. In that case, aside from aliasing error, we could say that the estimator is perfect, hence has zero bias and zero variance. In general, efficient use of this technique does require impulse response truncation; and the QA estimate would then contain some approximation error due to that fact as well as due to aliasing.

11.2.7.2. QA Method for Binary Systems

To set the stage for more involved cases, we begin with a simple binary system. Figure 11.32 shows the ideal received points at $\pm A$. Distortion will manifest itself in received amplitudes around these points, as illustrated by the sample shown at $v_k = (1 - \epsilon_k)A$, where ϵ_k is the fractional error for the k th sample.[†] For that sample, the (conditional) probability of error is given by

$$p_k = \int_{-\infty}^{-v_k} f_n(\eta) d\eta = F_n(-v_k) \quad (11.2.89)$$

[†]In the following, it will be convenient to change notation to index the sample in time, rather than its symbol value.

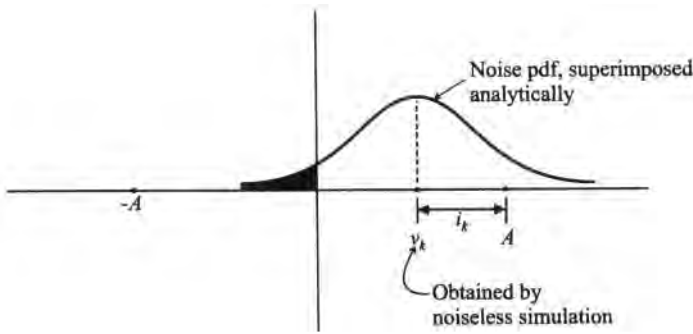


Figure 11.32. Illustration of the QA method for a binary system: idealized receiver voltages at $\pm A$; received voltage at time k is v_k .

where f_n here is one-dimensional and F_n is the CDF. For a Gaussian ENS, $p_k = \frac{1}{2} \text{erfc}(v_k / \sqrt{2}\sigma)$, where $v_k = (1 - \varepsilon_k)A$ and σ^2 is the noise power. If a sequence of N bits is transmitted, the total (average) BER is just the average of the p_k ,

$$p = \frac{1}{N} \sum_{k=1}^N p_k \quad (11.2.90)$$

All illustration of the implication of (11.2.89) is given in Figure 11.33, which shows a slice of a distorted waveform as it appears at the input to the decision device. The actual shape of the waveform will of course depend on the system at hand. The shaded “tails” in the figure correspond to the integral (11.2.89) for different instances k and n .

In general, when we observe a particular noiseless voltage v_k we would not know if that was due to a one or a zero. This affects which tail of the noise density we integrate under. However, because we have assumed that there are no noiseless errors, we know that the sign of v_k agrees with that of the transmitted bit. This knowledge will simplify the procedure because we can then rectify the waveform and not have to check the sign of the received voltage. Thus, we define $u_k = |v_k|$. For f_n in the generalized exponential family, it follows that

$$\begin{aligned} p_k &= \int_0^\infty \frac{v}{2\sqrt{2}\sigma\Gamma(1/v)} e^{-(x+u_k)/\sqrt{2}\sigma} v dx \\ &= \frac{1}{2\Gamma(1/v)} \int_\xi^\infty e^{-z} z^{(1/v)-1} dz \\ &= \frac{1}{2\Gamma(1/v)} \Gamma\left(\frac{1}{v}, \xi\right) \end{aligned} \quad (11.2.91)$$

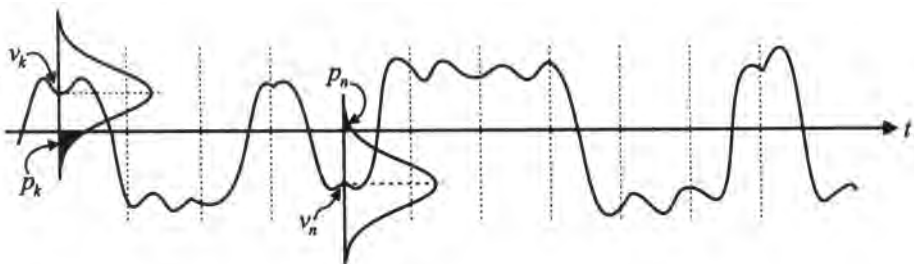


Figure 11.33. Illustration of the QA method: received distorted waveform (input to decision device). The shaded tails represent the error probability at two sampling times k and n .

where $\xi = (\mathbf{u}_k/\sqrt{2}\sigma)^\nu$ and $\Gamma(\cdot, \cdot)$ is the incomplete gamma function. As we know, the Gaussian case obtains when $\nu = 2$. The BER estimate depends of course on the indicated parameters. The values $\mathbf{u}_k$ are implicitly dependent on the sampling epoch τ as well as on any phase error θ in the case of passband transmission. The noise power depends on the input noise density and on the filtering characteristics, and the parameter σ is dependent on ν and the noise power: only in the Gaussian case is the noise power σ^2 . The implied calibrations must be carefully carried out, as discussed in Section 8.13.

Figure 11.34 gives a capsule description of the QA procedure. In the most streamlined version of the procedure, phase and timing synchronization can be assumed known, for example, using the correlation-based estimates of Section 10.6. In that case, we have one incomplete gamma function evaluation or one erfc evaluation per bit; computationally, this effectively replaces the generation of many random numbers per bit needed in the MC approach. However, the QA procedure can be extended to yield additional information at relatively little cost, as implied in Figure 11.34. This extension involves obtaining the BER as a function of θ and τ , which is useful information in any case, and is required in order to account for the jitter in the synchronization processes, as will be shown later. Figure 11.34 also indicates that we can vary ν , the exponent of the GE distribution. This, too, can yield useful information on the sensitivity of BER to noise statistics and can be used in conjunction with short MC runs as a form of extrapolation procedure.⁽¹⁹⁾

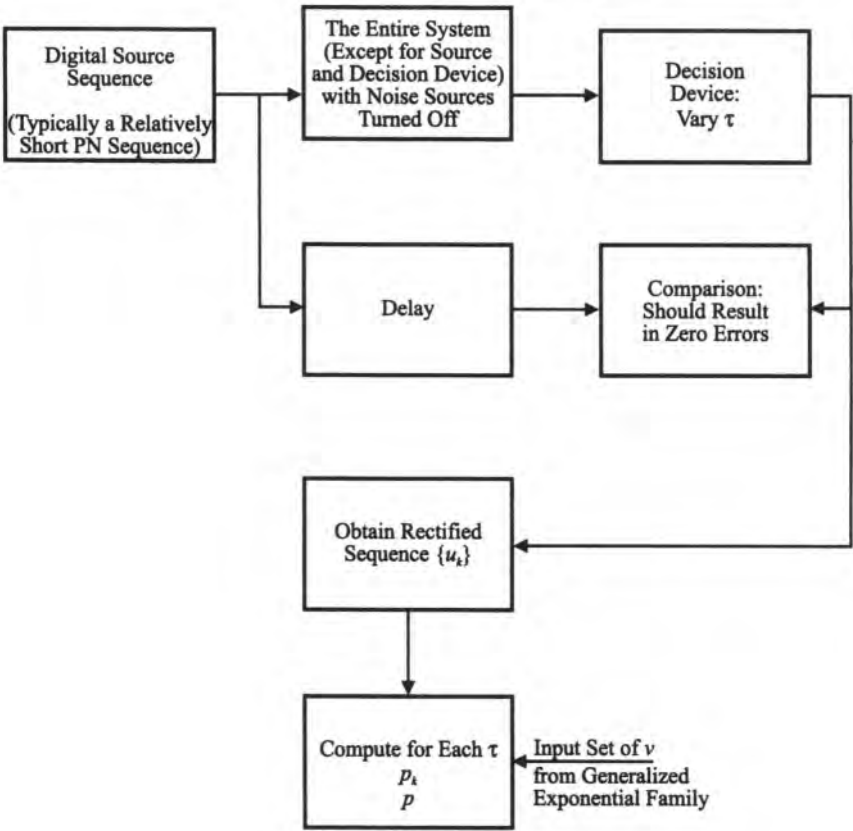


Figure 11.34. Block diagram of QA method implementation.

The described procedure effectively assumes independence of the noise at each sampling instant. In many cases this is a reasonably good assumption since these samples are at bit intervals, the reciprocal of which is on the order of the system bandwidth. In any case, the result (11.2.90) is the correct ensemble-average BER, but the method is incapable of providing information on the temporal distribution of errors. This is doubly so because the sequence duration N is typically much shorter than gap lengths of interest.

11.2.7.3. QA Method for Single-Dimensional Multiamplitude Modulation

The basic method above can be easily extended to the case of single-dimensional multiamplitude modulation, that is, baseband PAM or L -ary amplitude-shift-keying in the case of bandpass systems. The constellation is shown in Figure 11.35, where the vertical dashed lines indicate the decision region boundaries. Due to ISI or other distortions, the k th sample for decisioning might again be received as the point labeled v_k , assumed to be displaced by $\epsilon_k A$ from the ideal point. The figure also shows the ENS pdf superimposed on the received sample, and the probability of symbol error is the shaded area under the two tails. Defining as before $u_k = |v_k|$, we can also write

$$u_k - I_k A = \epsilon_k A$$

where $I_k A$ is the odd integer multiple of A nearest to u_k . The *symbol* error probability corresponding to the k th sample for an interior point (as illustrated) is then given by

$$p_k = \int_{-\infty}^{-A(1+\epsilon_k)} f_n(\eta) d\eta + \int_{A(1-\epsilon_k)}^{\infty} f_n(\eta) d\eta \quad (11.2.92)$$

which, for the generalized exponential distribution, is

$$p_k = \frac{1}{2\Gamma(1/\nu)} \left[\Gamma\left(\frac{1}{\nu}, \xi_+\right) + \Gamma\left(\frac{1}{\nu}, \xi_-\right) \right] \quad (11.2.93)$$

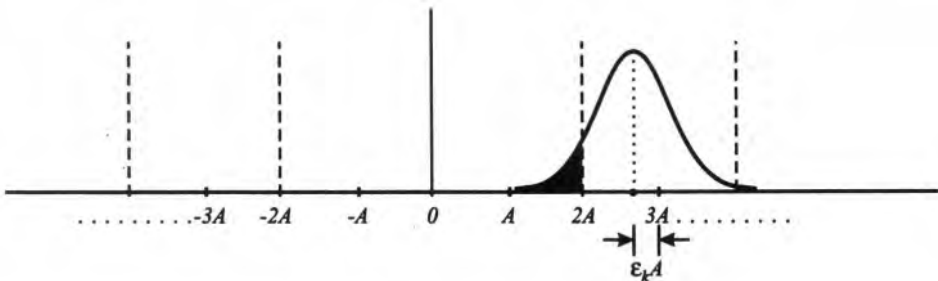


Figure 11.35. Illustrating the QA method for single-dimensional multiamplitude system (PAM): idealized received voltages at $\pm A, \pm 3A, \dots$; received voltage at time k is in error by $\epsilon_k A$.

where $\xi_{\pm} = [A(1 \pm \epsilon_k)/\sqrt{2}\sigma]^v$. For an exterior symbol, nominally $\pm(M-1)A$, only one tail applies and the corresponding result is

$$p_k = \frac{1}{2\Gamma(1/v)} \Gamma\left(\frac{1}{v}, \xi_{\pm}\right) \quad (11.2.94)$$

In the case at hand we may be interested in the bit error probability, which depends on the bit-to-symbol mapping used. Reasonable approximations can be made without computation. For example, a Gray code will likely induce one bit error per symbol error. But the QA method will permit an exact calculation for any mapping since we can calculate specific error probabilities, as discussed in conjunction with (11.2.87); see also Problem P11.18.

The average error rate is given by an expression identical to (11.2.90) with appropriate values for p_k and N . With respect to the latter, as we pointed out previously, the sequence length may be considerably longer than for the binary case. Still, even a sequence on the order of about 10^6 is reasonably efficient compared to MC, though not so dramatic. However, there are still more efficient ways to implement the QA method, which are described in conjunction with Case Study I in Chapter 12.

11.2.7.4. QA Method for QAM Modulation

The QA method presented just above is easily extended to QAM modulation. Figure 11.36 shows the ideal constellation with signal points separated by $2A$ in both dimensions. The figure shows a square constellation of $L = q^2$ elements, but the following methodology is easily applied to a constellation on any rectangular lattice. Let us consider an interior point which has been received, due to distortion and ISI, as $v_k = v_{k,x} + jv_{k,y}$, and, similar to the previous case, define $|v_{k,x}| = u_{k,x} = I_{k,x}A + \epsilon_{k,x}A$ and $|v_{k,y}| = u_{k,y} = I_{k,y}A + \epsilon_{k,y}A$. Let the noise pdf be identical in the two dimensions and of the generalized exponential type. The probability of symbol error is the probability that the received point plus noise lies outside the correct decision region. The probability of incorrect decision in either the x or y dimension is given in form by (11.2.93) with ϵ_k replaced by $\epsilon_{k,x}$ and $\epsilon_{k,y}$, respectively. Thus, the probability of symbol error on the k th symbol, assuming it is an interior symbol, is given by

$$p_k = p_{k,x} + p_{k,y} - p_{k,x}p_{k,y} \quad (11.2.95)$$

where

$$p_{k,x} = \frac{1}{2\Gamma(1/v)} \left[\Gamma\left(\frac{1}{v}, \xi_{+}^{(x)}\right) + \Gamma\left(\frac{1}{v}, \xi_{-}^{(x)}\right) \right] \quad (11.2.96a)$$

and

$$p_{k,y} = \frac{1}{2\Gamma(1/v)} \left[\Gamma\left(\frac{1}{v}, \xi_{+}^{(y)}\right) + \Gamma\left(\frac{1}{v}, \xi_{-}^{(y)}\right) \right] \quad (11.2.96b)$$

where independence of the x and y noise components has been assumed. The probability of symbol error for symbols on the periphery of the constellation or on the corners is a simple extension of the above result (Problem P11.19). As before, then, we have for the average SER

$$P = \frac{1}{N} \sum_{k=1}^N p_k$$

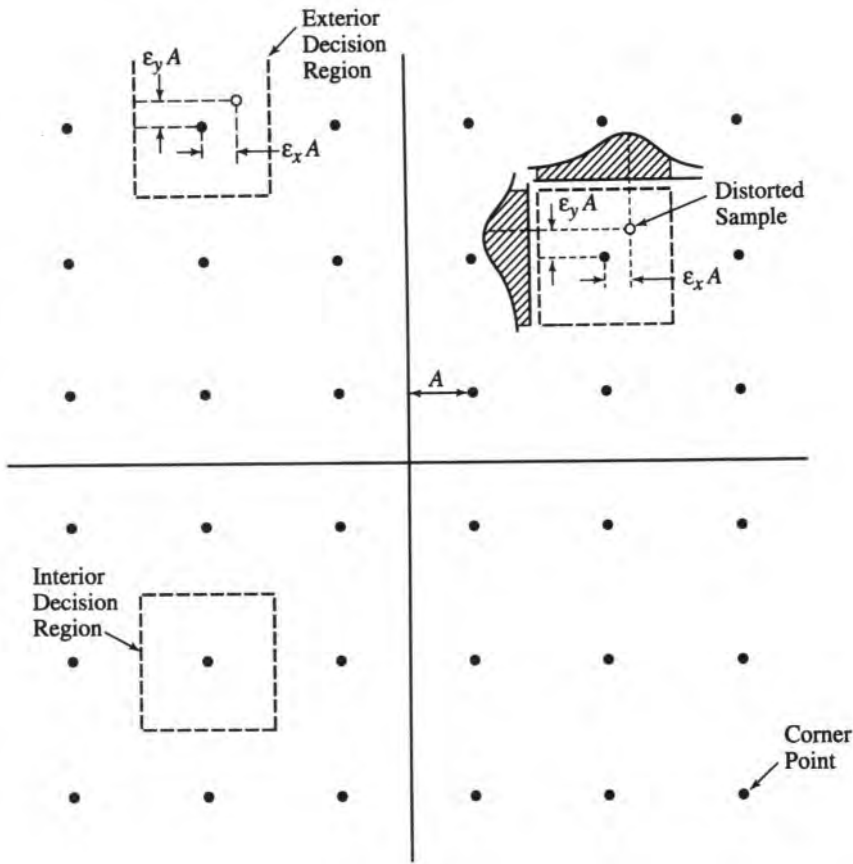


Figure 11.36. Illustration of rectangular constellation and terms used in the QA method for QAM signals.

The comment made earlier about the possibly very long sequence length applies all the more because L will tend to be even larger. In many quadrature systems, even though there is some degree of channel cross-talk, the strength of interaction may be sufficiently small that the channels can be considered independent. In that case, instead of using a sequence of length L^m , one may use a separate input sequence in each channel that is of length $\sqrt{L^m}$, which will be considerably shorter. Doing so, of course, will not change the nature of the interaction between the two channels; it simply means that not all possible symbol pairs on the two channels will occur.

11.2.7.5. QA Method for PSK Modulation

Here, the evaluation of (11.2.86) or (11.2.87) is not as straightforward as for the rectangular constellations. This is because the decision region boundaries are not parallel to the x and y noise components. Refer to Figure 11.37, which shows an L -ary PSK constellation whose ideal points are spaced around the circle $\theta_L = 360/L$ deg apart. Once again, because of ISI and distortion, a point sent as $(A, 0)$ is received as (A', ϕ) . We will assume zero phase is

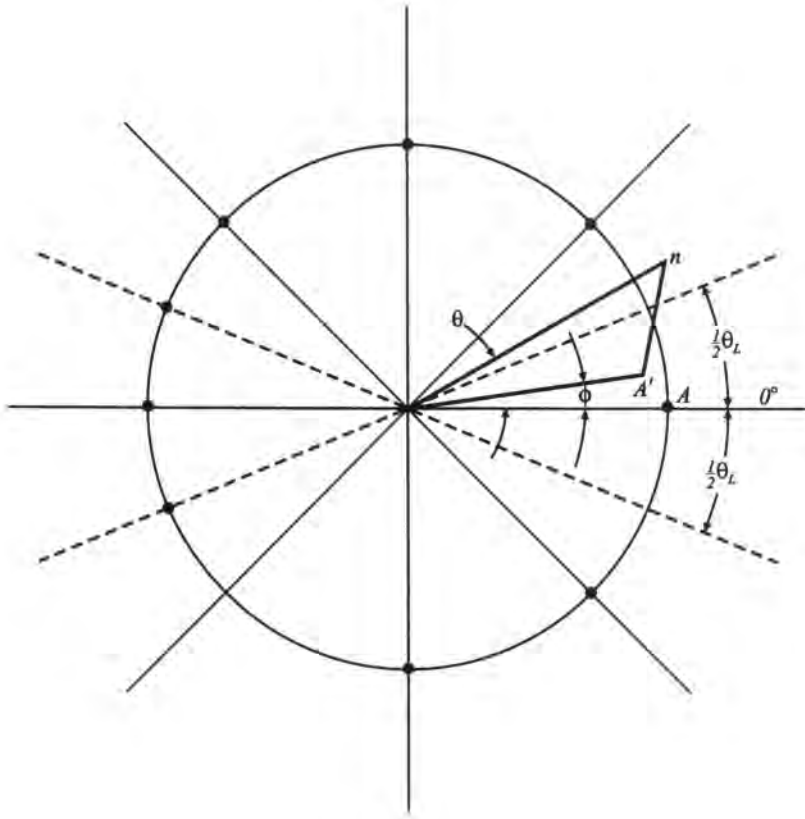


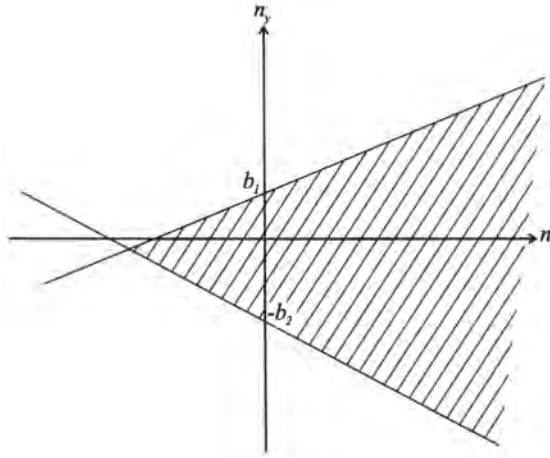
Figure 11.37. Illustration of the QA method for L -ary PSK.

sent, without loss of generality, since whatever phase is sent, we know what the proper sector is, and can rotate the received point to the first sector. Defining $x = A' \cos \phi$, $y = A' \sin \phi$, and n_x and n_y the corresponding noise components, we have that the angle of the vector $(A' \cos \phi + n_x, A' \sin \phi + n_y)$ is given by

$$\tan \theta = \frac{A' \sin \phi + n_y}{A' \cos \phi + n_x} \quad (11.2.97)$$

with $-\pi/2 \leq \theta \leq \pi/2$. A correct decision will be made if $-\frac{1}{2}\theta_L < \theta < \frac{1}{2}\theta_L$, and an error otherwise. We will consider several ways of evaluating the error probability, as they are instructive. The preferred technique will depend on their computational efficiencies and the length of the sequence used. To begin with, let us formulate the probability of correct reception P_c , namely,

$$P_c = \Pr[-\kappa \leq \tan \theta \leq \kappa]$$


 Figure 11.38. Illustration of the region of correct reception for L -ary PSK.

where $\kappa = \tan \frac{1}{2} \theta_L$. Substituting (11.2.97), we find the following conditions for n_x and n_y to be satisfied:

$$-\kappa n_x - \kappa A' \cos \phi - A' \sin \phi \leq n_y \leq \kappa n_x + \kappa A' \cos \phi - A' \sin \phi$$

which is the wedge shown in Figure 11.38. Labeling the upper and lower intercepts $b_1 = A'(u \cos \phi - \sin \phi)$ and $-b_2 = -A'(u \cos \phi + \sin \phi)$, respectively, we obtain

$$P_c = \int_{-A' \cos \phi}^{\infty} f_n(n_x) dn_x \int_{-\kappa n_x - b_2}^{\kappa n_x + b_1} f_n(n_y) dn_y$$

The integral over x can be expressed in terms of incomplete gamma functions, but that appears to be as much of a simplification as is possible. Generally, numerical integration will be necessary, which makes this approach suitable only for relatively short sequences.

If we assume that f_n is Gaussian, we can obtain more useful results. The pdf of θ is known to be (see, e.g., Refs. 55 and 56)

$$g(\theta) = \frac{1}{2\pi} e^{-\rho} + \sqrt{\frac{\rho}{\pi}} \cos(\theta - \phi) e^{-\rho \sin^2(\theta - \phi)} \left\{ 1 - Q\left(\sqrt{2\rho} \cos(\theta - \phi)\right) \right\} \quad (11.2.98)$$

where ρ is the signal-to-noise ratio. Then, the probability of error (given zero phase is sent) is given by

$$p = 1 - \int_{-\theta_L/2}^{\theta_L/2} g(\theta) d\theta \quad (11.2.99)$$

and the *specific* error probability p_{0j} that symbol a_0 corresponding to zero phase is detected as symbol a_j is

$$p_{0j} = \int_{\theta \in \mathcal{D}_j} g(\theta) d\theta \quad (11.2.100)$$

The latter is particularly of interest in block-coding schemes using unequal error protection since the final error rate depends on the specific bit errors as corrected by whatever error correction is applied to that bit position.

Equations (11.2.99) and (11.2.100) can be evaluated by numerically integrating (11.2.98). But it is also possible to use an easy-to-compute approximation that is quite good in most cases. Refer to Figure 11.37; the approximation consists in calculating the probability that the vector terminus lies in the half-plane above the line defined by $\theta = \frac{1}{2}\theta_L$ or in the half-plane below the line defined by $\theta = -\frac{1}{2}\theta_L$. Approximations with varying degrees of accuracy can be derived based on these half-plane probabilities (see, e.g., Refs. 57 and 58 and Problem P11.20). The corresponding expression for p is

$$p \leq \frac{1}{2} \operatorname{erfc}\left(\frac{A'}{\sqrt{2}\sigma} \sin(\theta - \phi)\right) + \frac{1}{2} \operatorname{erfc}\left(\frac{A'}{\sqrt{2}\sigma} \sin(\theta + \phi)\right) \quad (11.2.101)$$

Extensions of this half-plane probability approach can be used to compute specific error probabilities exactly⁽⁵⁹⁾ or approximately.^(60,61)

11.2.7.6. QA Techniques for Coded Systems with Hard-Decision Decoding

The estimation of ER for coded systems presents a computational problem not so much because of simulating the encoder/decoder (codec) as such, although there would obviously be an associated computational burden (discussed in Chapter 8), but rather because the target ER at the decoder output is typically too low to be estimated in reasonable time by MC. Of course, if the target ER is not too low, say on the order of 10^{-5} or higher, then error counting at the decoder output may be perfectly feasible. Otherwise, some form of extrapolation will be needed. Extrapolation methods such as the ones discussed earlier, TE and IS, could be brought to the task, but are not straightforward to apply. Here we consider QA techniques where the raw channel error rate or error production behavior is obtained via MC and extended to the decoder output by analytical means. In this subsection we consider only the case of hard-decision decoding (HDD), which would be used primarily in conjunction with block codes. However, as will be discussed, it also serves a simulation purpose when used with convolutional codes.

11.2.7.6.1. Independent-Error Channel We assume for now that errors are produced independently. For practical purposes we can say that this assumption is satisfied if an interleaver of appropriate depth is used.

For HDD systems, the analytical part of the technique is contained in the BER-transfer characteristic, which relates the BER at the decoder output, which we label p_o , to that at the input, the *channel* BER, p_c . Many reasonably good approximations of this function exist for

the case at hand. Specifically, for binary block-coded systems with independent errors, one such approximation is given by⁽⁶²⁾

$$p_o \approx \frac{d}{n} \sum_{i=t+1}^d \binom{n}{i} p_c^i (1-p_c)^{n-i} + \frac{1}{n} \sum_{i=d+1}^n i \binom{n}{i} p_c^i (1-p_c)^{n-i} \quad (11.2.102)$$

where n is the block length, d is the minimum (Hamming) distance, and $t = [(d-1)/2]$, where the square brackets indicate integer part. Note that p_o here is the BER for *information* bits. As we mentioned in Section 11.2.3.1, partial sums of binomial terms can be much more efficiently evaluated using the equivalence with the beta distribution, as indicated in (11.2.12). With a little manipulation, the sums on the right-hand side of (11.2.102) can be expressed in such a form. The net result is

$$p_o \approx \frac{d}{n} \{P[\text{beta}(d+1, n-t) \leq p_c] - P[\text{beta}(d+1, n-t)] \leq p_c\} \\ + \frac{p_c}{n(1-p_c)} P[\text{beta}(d, n-d) \leq p_c] \quad (11.2.103)$$

An equation identical in form to (11.2.102) applies to nonbinary codes if we replace p_o by P_o , the output error probability for the *information symbols*, and replace p_c by P_c , the raw channel *symbol* error probability. A nonbinary code of particular interest is the Reed-Solomon (RS) code, which is used in many applications. In this case, letting N represent the codeword length in symbols, one can obtain⁽⁶²⁾ the following approximation for the *bit* error probability:

$$p_o \approx \frac{N+1}{2N^2} \left\{ d \sum_{i=t+1}^d \binom{N}{i} P_c^i (1-P_c)^{N-i} + \sum_{i=d+1}^N i \binom{N}{i} P_c^i (1-P_c)^{N-i} \right\} \quad (11.2.104)$$

and d and t have the same interpretation (with respect to symbols). Again, numerical evaluation is greatly facilitated by using the cumulative beta distribution. The result follows straightforwardly from (11.2.103):

$$p_o \approx \frac{d(N+1)}{2N^2} \{P[\text{beta}(d+1, N-t) \leq P_c] - P[\text{beta}(d+1, N-t)] \leq P_c\} \\ + \frac{(N+1)P_c}{2N^2(1-P_c)} P[\text{beta}(d, N-d) \leq P_c] \quad (11.2.105)$$

For binary convolutional codes the BER-transfer characteristic can be bounded from above by the well-known transfer function bound (see, e.g., Refs. 63 and 64)

$$p_o < \sum_{d=d_f}^{\infty} \beta_d P_2(d) \quad (11.2.106)$$

where $\{\beta_d\}$ are the coefficients in the expansion of the derivative of $T(D, N)$, the transfer function (or generating function) of the code evaluated at $N = 1$, d_f is the free distance of the code, and $P_2(d)$ can be approximated (bounded) by

$$P_2(d) = [4p_c(1 - p_c)]^{d/2} \quad (11.2.107)$$

Even though soft decisions are normal for convolutional codes, the relative simplicity of (11.2.106) and (11.2.107) is computationally attractive. The rough rule of thumb that soft decisions are about 2 dB more efficient can be applied after the fact.

■ **Example 11.2.10.** An illustration of one procedure for implementing the technique under discussion is illustrated in Figure 11.39.⁽⁶⁵⁾ The left half of the figure shows four BER curves. The curve labeled theoretical is for reference and is the ideal curve for BPSK or QPSK modulation. The curve next to the right is the actual system's performance with no coding. The label *BEF* on this curve stands for bandwidth expansion factor and is the reciprocal of the code rate, assuming no change in modulation. Thus, the bandwidth expansion induces additional degradation, which is seen in the curve labeled *BEF*=1.1, that is, about 10% bandwidth increase. The abscissa in the figure is E_b/N_0 , where E_b is the energy per information bit. Using (11.2.102), we have that the right half of the figure shows the BER-transfer characteristic for a BCH(255, 231, 3) binary code, i.e., *BEF* = 255/231 $\approx$ 1.1. The procedure starts at point A, which represents some value of p_c obtained by simulation. Proceeding along the dashed line to the right, one obtains the corresponding value of p_o , which is projected to the ordinate on the left part of the figure, so that this axis shows both p_c and p_o on the appropriate curve. The p_o which corresponds to the p_c at point A exists, of course, for the same value of E_b/N_0 as at point A. Therefore, the p_o in question is

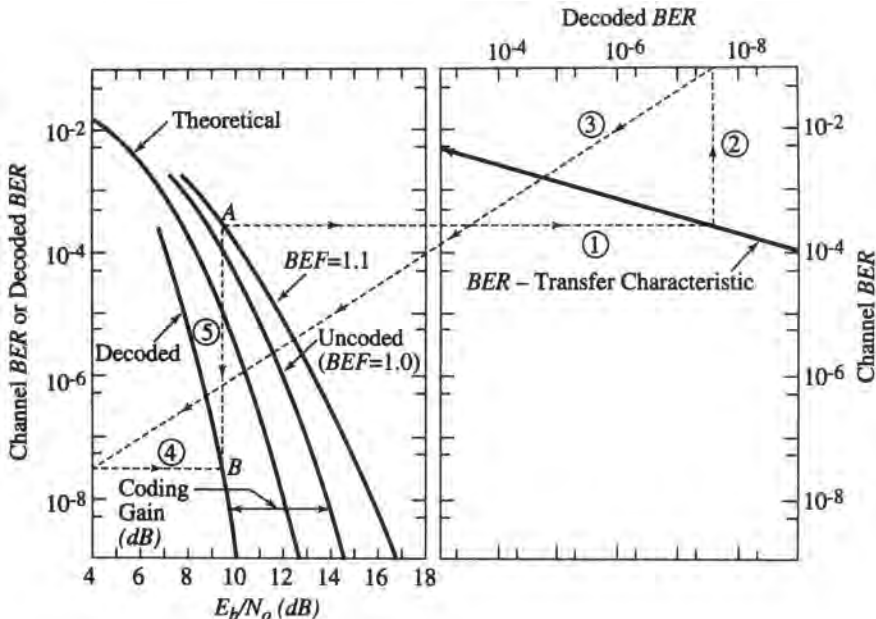


Figure 11.39. Illustration of simulation-based process for determining BER for hard-decision decoding of a block code (from Ref. 64, © IEEE, 1986).

located at point B , which is the intersection of the dashed line dropping vertically from point A and the horizontal line at p_o . By repeating this procedure, one can draw out an entire decoded BER curve, which is the leftmost curve on the figure. ■

It should be mentioned, since the decoder is not explicitly simulated, that the encoder also does not need to be explicitly simulated. The bandwidth expansion is simulated by simply increasing the chip rate or, equivalently, decreasing the bandwidth of all filters by the code rate. The actual simulated curve could be obtained through any of the techniques discussed earlier. But it should be noted that whatever uncertainties may be in this curve will be magnified in the estimate of the decoded BER. Such uncertainties will be present particularly if the uncoded curve is obtained by MC means. To illustrate the previous point, first observe that the BER-transfer characteristic shown in Figure 11.39 is well approximated by

$$\log p_o = G \log p_c + \log \alpha$$

or

$$p_o = \alpha p_c^G$$

where G can be thought of as a form of coding gain. The uncertainty in the simulation can be described by a band $\eta_u \hat{p}_c$, $\eta_l \hat{p}_c$ above and below the observed estimate $\hat{p}_c$; the values of $\eta_u > 1$ and $\eta_l < 1$ depend on the confidence level. The corresponding band of uncertainty for decoded BER can then be expressed as $\alpha(\eta_u \hat{p}_c)^G$, $\alpha(\eta_l \hat{p}_c)^G$. It can be seen, then, that the band of uncertainty has been magnified from (η_u, η_l) to (η_u^G, η_l^G) . As an example, for the BER-transfer characteristic of Figure 11.39, $G = 3.875$, and for $\eta_u = 1.1$, say, $\eta_u^G = (1.1)^{3.875} = 1.447$. Thus, a 10% uncertainty is magnified to one of about 45%.

The QA method discussed in this section is easily extended to unequal error protection schemes once we have the specific error probabilities (i.e., those associated with specific bits in a symbol). Applications of this technique for 8PSK with LSB coding can be found, e.g., in Refs. 60 and 61.

11.2.7.6.2. Dependent-Error Channel As mentioned, the BER-transfer characteristic above is based on the independent error assumption. In the physical channel, errors within a block of bits equal to the codeword length are not generally produced independently (at the average error rate) because of the presence of various mechanisms, for example, slow processes such as phase noise. Before evaluating the effect of a code, one should first establish whether or not the independent error assumption is a reasonable one. There are different ways to test this assumption. We previously mentioned doing a goodness-of-fit test on the error gap distribution. Another way is to gather a histogram of the number of channel errors in blocks equal to the codeword length. Such a histogram obtained from a simulation is shown in Figure 11.40. Since the distribution of errors for an independent-error channel is known to be binomial, a goodness-of-fit test on the histogram will indicate whether or not independence is warranted. Generally, a statistically significant number of errors will be seen in reasonable time only for relatively small signal-to-noise ratio, which may not be the one of interest. Otherwise there would be no need for extrapolation. Thus, a determination of nonindependence should then lead to a choice of an appropriate channel model that could be used to extrapolate analytically.

As an example, suppose that it is determined that a Gilbert model is an appropriate one. The state diagram shown in Figure 9.32 applies here as well. The probability of error in states B

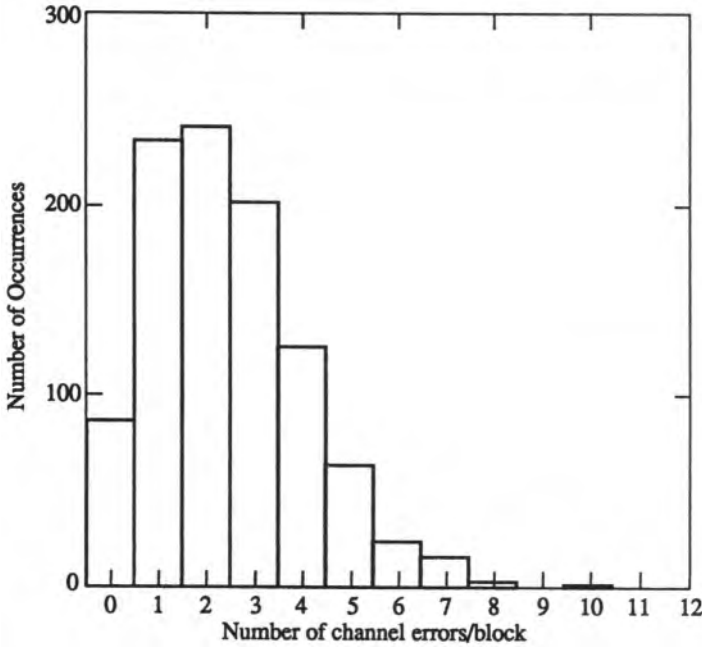


Figure 11.40. Example of histogram of number of errors per code block ($E_b/N_0 = 6\text{dB}$) obtained from simulation.

and G will be denoted P_B and P_G , respectively. Then, it can be shown⁽⁶⁶⁾ that

$$P(m, n) = P_G(m, n) + P_B(m, n) \quad (11.2.108)$$

where $P_G(m, n)$ is the probability of m errors in n transmissions with the channel ending in state G , and $P_B(m, n)$ is the probability of m errors in n transmissions with the channel ending in state B . For $m = 1, 2, 3, \dots$ and $n = 1, 2, 3, \dots$, the terms on the right side of (11.2.108) can be computed recursively as follows⁽⁶⁶⁾:

$$\begin{aligned} P_G(m, n) = & P_G(m, n-1)a_{11}(1 - P_G) \\ & + P_B(m, n-1)a_{21}(1 - P_G) \\ & + P_G(m-1, n-1)a_{11}P_G \\ & + P_B(m-1, n-1)a_{21}P_G \end{aligned} \quad (11.2.109)$$

and

$$\begin{aligned} P_B(m, n) = & P_B(m, n-1)a_{22}(1 - P_B) \\ & + P_G(m, n-1)a_{12}(1 - P_B) \\ & + P_B(m-1, n-1)a_{22}P_B \\ & + P_G(m-1, n-1)a_{12}P_B \end{aligned} \quad (11.2.110)$$

Equations (11.2.108)–(11-2-110) are examples of the use of a simple enough generative model to construct an efficient analytical extrapolation. Further results along these lines for the Gilbert model can be found in Refs. 66 and 67.

In the foregoing discussion, a high level of abstraction (in the form of an HMM) is used to describe the channel. The description of the physical channel essentially disappears in the process. Another example of the application of the QA method to block codes in a channel whose characterization is closer to a physical description is given in Ref. 68. There, the channels are described in terms of frequency nonselective Rayleigh, Ricean, and Nakagami distributions. The system uses binary noncoherent frequency-shift-keying (NCFSK) along with diversity combining. An error rate expression can be formulated for this case based on a nonfading channel or, equivalently, conditioned on a specific fade. The error rate averaged over the fading distribution is then obtained by simulation, i.e., drawing from the fading distribution using a random number generator.

11.2.7.7. QA Method for Convolutionally Coded Systems with Soft-Decision Decoding

For soft-decision convolutional codes, a QA approach similar to the one above is available. In this case, however, the simulation is used to provide the channel transition probability (CTP) matrix associated with soft decisions, rather than simply the hard decisions. A soft-decision scheme creates a discrete channel with L input symbols $\mathbf{x}_i, i = 1, 2, \dots, L$; Q output symbols (the soft decisions) $y_j, j = 1, 2, \dots, Q$; and an associated CTP matrix $[p_{ij}] = [p(y_j|x_i)]$. For given input signal and noise levels, the simulation is run to obtain $[p_{ij}]$.

There seem to be relatively few analytical results for the situation at hand, but one that is available applies to a binary input ($\mathbf{x}_1 = \mathbf{0}, \mathbf{x}_2 = \mathbf{1}$) memoryless discrete channel. For such a channel, the BER can be bounded from above as follows⁽⁶³⁾:

$$p_o = \left. \frac{\partial T(D, N)}{\partial N} \right|_{N=1, D=D_o} \quad (11.2.111)$$

where, as before, $T(D, N)$ is the transfer function of the code, and

$$D_o = \sum_{j=1}^Q [p(y_j|x_1)p(y_j|x_2)]^{1/2} \quad (11.2.112)$$

Thus, this technique is QA in that the BER is calculated “analytically” from (11.2.111), while D_o is obtained from simulated generated transition probabilities. The run-time advantage depends on the nature of the simulation from which D_o is obtained. If a QA simulation is itself used to find the CTP, then the procedure as a whole will be quite efficient. If D_o is obtained from an MC simulation, there can still be substantial savings since that simulation’s run time will generally be only as long as necessary to estimate the channel BER. However, just as was discussed in conjunction with hard decoding, any uncertainty in the result of the MC simulation will be magnified by the extrapolation (see Problem P11.25).

11.2.7.8. Incorporating Jitter in the QA Technique

We mentioned earlier that because the sequence length in QA is often relatively short, it is not appropriate explicitly to simulate slow processes like phase and timing jitter. They can be taken into account statistically by the following procedure. As mentioned in Section

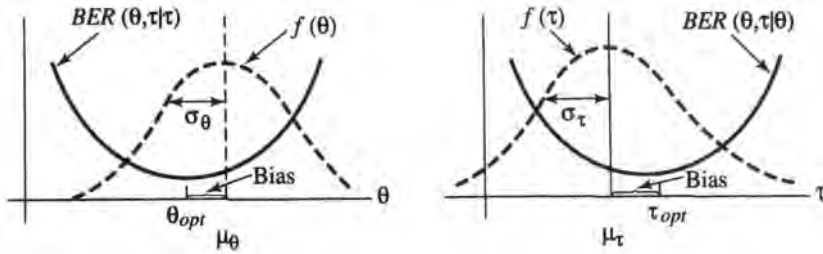


Figure 11.41. Illustration of calculating BER by averaging over phase and timing distributions.

11.2.6.2, a single QA run is performed to produce a conditional result, conditioned on a value of phase reference θ and sampling epoch τ . Therefore, the first step is to run a sequence of QA runs for a set of $\{\theta\}$ and of $\{\tau\}$ to obtain the functional dependence of error rate on these variables. Note that this does not imply redoing entire runs: the file of samples up to the demodulator can be saved and reprocessed. Thus, we can express the BER as a function of θ and τ , namely $p(\theta, \tau)$. Two cross sections of this functional surface are illustrated in Figure 11.41. The average BER is then given by

$$p = \int_{\mathcal{S}} p(\theta, \tau) f(\theta, \tau) d\theta d\tau \quad (11.2.113)$$

where $f(\theta, \tau)$ is the pdf of (θ, τ) , $\mathcal{S} = \{(\theta, \tau) | -\pi \leq \theta \leq \pi, 0 \leq \tau \leq T\}$, and T is the symbol duration. Since $p(\theta, \tau)$ can only be obtained for a finite set of argument pairs, (11.2.113) would be implemented as a sum, unless the discrete set of values had been fitted to a continuous function. When the phase and timing are estimated by different circuits it is reasonable to assume independence, so that $f(\theta, \tau) = f_1(\theta)f_2(\tau)$. The form of $f(\theta, \tau)$, $f_1(\theta)$, or $f_2(\tau)$ can replicate some theoretical distribution or approximation thereto, or can be deduced either from hardware experiments or from stand-alone simulations of synchronization circuits (see Chapter 8). Figure 11.39 shows hypothetical pdf's superimposed on the error rate characteristics. This analytical approach is flexible and efficient, but, as is the case with QA in general, can provide only average quantities, not temporal information on the distribution of errors.

11.2.7.9. Mixed QA Technique

In the usual conception of QA techniques, random (noise) processes are not explicitly simulated. As noted, however, the QA technique is valid, or at least straightforward to apply only when the noise is processed linearly. In systems where there is more than one noise source, at least one of which encounters a nonlinearity and at least one of which is processed linearly, a variation on the QA technique called *mixed* QA (MQA) can be applied. MQA is a combination of MC and QA methodologies. Again, there is not a unique such technique. One version of MQA can be found in Ref. 69. Another will be presented below, where a noise source prior to a nonlinearity is explicitly generated, while a noise source after a nonlinearity is treated quasianalytically. The kind of system to which this technique might be applied is shown in Figure 11.42, where g_1, g_2, g_3 are system functions. $N_1(t)$ and $N_2(t)$ are assumed to be Gaussian noise sources. If g_2 is nonlinear (possibly with memory), but g_3 is linear, $N_2(t)$

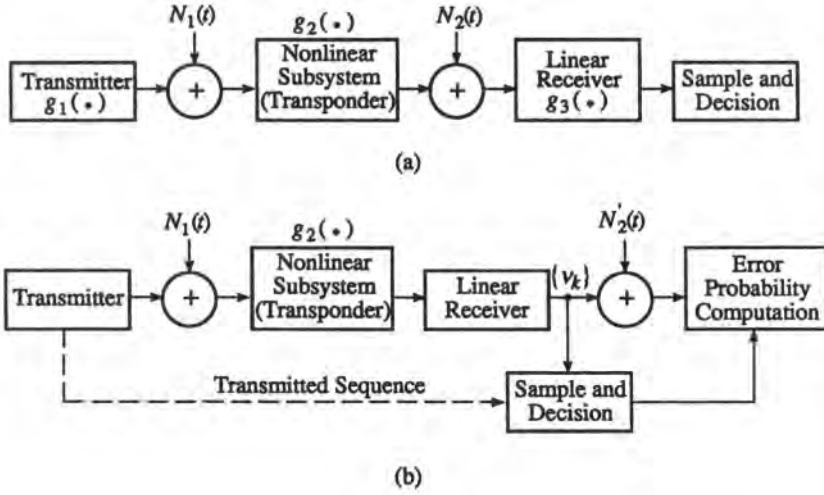


Figure 11.42. Block diagram of system and modifications for application of mixed QA. (a) Simplified block diagram of two-hop nonlinear system. (b) Equivalent block diagram for application of MQA: $N'_2(t) = N_2(t) * g_3(t)$.

can be treated by ordinary QA. That is, it can be treated analytically as an equivalent noise source $N'_2(t)$ which is a filtered version of $N_2(t)$ and is still Gaussian.

An estimator for the BER p is easily derived. Let the sequence of symbol-spaced samples of the waveform at the point indicated in the figure be $\{v_k\}$. This sequence includes the effect of $N_1(t)$, but not yet $N_2(t)$. Denote by $\{s_k\}$ the corresponding sequence of input symbols, which for simplicity we shall assume binary. Then, an error occurs if

$$v_k + n_{2k} < 0, \quad s_k > 0 \quad (11.2.114a)$$

$$v_k + n_{2k} > 0, \quad s_k < 0 \quad (11.2.114b)$$

where $\{n_{2k}\}$ is the sequence of samples of $N'_2(t)$ and, by implication, the decision variable is $v_k + n_{2k}$. Assuming $N_2(t)$ is Gaussian, we can calculate the probability of the events in (11.2.114):

$$P[v_k + n_{2k} < 0 | s_k > 0] = Q(v_k / \sigma_2) \quad (11.2.115a)$$

$$P[v_k + n_{2k} > 0 | s_k < 0] = Q(-v_k / \sigma_2) \quad (11.2.115b)$$

where σ_2 is the standard deviation of $N'_2(t)$, and, as usual,

$$Q(x) = \frac{1}{(2\pi)^{1/2}} \int_x^\infty e^{-u^2/2} du$$

Note that v_k in (11.2.115a) is not necessarily positive and v_k in (11.2.115b) is not necessarily negative. Hence the BER estimator can be written as

$$\hat{p} = \frac{1}{N} \sum_{k=1}^N Q\left(\frac{v_k}{\sigma_2}\right) I_+ + Q\left(\frac{-v_k}{\sigma_2}\right) I_- \quad (11.2.116)$$

where

$$I_+ (I_-) = \begin{cases} 1 (0), & s_k > 0 \\ 0 (1), & s_k < 0 \end{cases} \quad (11.2.117)$$

Note that (11.2.117) is equivalent to making a decision, which is symbolically indicated in Figure 11.42b. It is clear that

$$E(\hat{p}) = p$$

After a little manipulation it can also be shown that

$$\begin{aligned} \sigma^2(\hat{p}) &= E(\hat{p}^2) - p^2 \\ &= \frac{1}{N} E \left\{ Q^2 \left(\frac{v_k}{\sigma_2} \right) I_+ + Q^2 \left(\frac{-v_k}{\sigma_2} \right) I_- \right\} - \frac{1}{N} p^2 \end{aligned} \quad (11.2.118)$$

if v_k and v_j are independent for $k \neq j$. The expectation in (11.2.118) is carried out with respect to v_k , a generally difficult procedure for a nonlinear system with memory. However, it is instructive to evaluate (11.2.118) for the simplest case possible just for the purpose of establishing a sense of the improvement available with this technique.

The simplest case is an additive Gaussian noise channel; here, $v_k = s_k + n_{1k}$, and $s_k = s$ or $s_k = -s$ for a one or a zero, respectively. Since $\text{sgn}(n_{1k})$ has equal probability of being positive or negative, for the current case we can rewrite (11.2.118) as

$$\sigma^2(\hat{p}) = \frac{1}{N} \left\{ E \left[Q^2 \left(s + \frac{n_{1k}}{\sigma_2} \right) \right] - p^2 \right\} \quad (11.2.119)$$

It can be shown that, as expected,

$$\begin{aligned} \sigma^2(\hat{p}) &\rightarrow 0 \quad \text{as } \sigma_1 \rightarrow 0 \quad (\sigma_2 \neq 0) \\ \sigma^2(\hat{p}) &\rightarrow p(1-p)/N \quad \text{as } \sigma_2 \rightarrow 0 \quad (\sigma_1 \neq 0) \end{aligned}$$

What is of particular interest is the ratio η of Monte Carlo variance to that in (11.2.119), namely,

$$\eta = \frac{p(1-p)}{E[Q^2(s + n_{1k}/\sigma_2)] - p^2} \quad (11.2.120)$$

Equation (11.2.120), or rather the expectation in the denominator, is surprisingly difficult to evaluate analytically. In fact, it was evaluated numerically through Gaussian quadrature integration, a powerful technique which is generally very useful in QA-type problems. (See also Sections 6.6.5 and 12.1.) The result of evaluating (11.2.120) is shown in Figure 11.43. It can be seen, perhaps as expected, that the smaller σ_1 , the larger the improvement. In satellite communications this corresponds to the usual case, where the uplink CNR is appreciably larger than the downlink CNR. The run-time improvement, although appreciable, is perhaps less than expected, and for rather low BER (**say** $<10^{-7}$) may still not be sufficient except on a

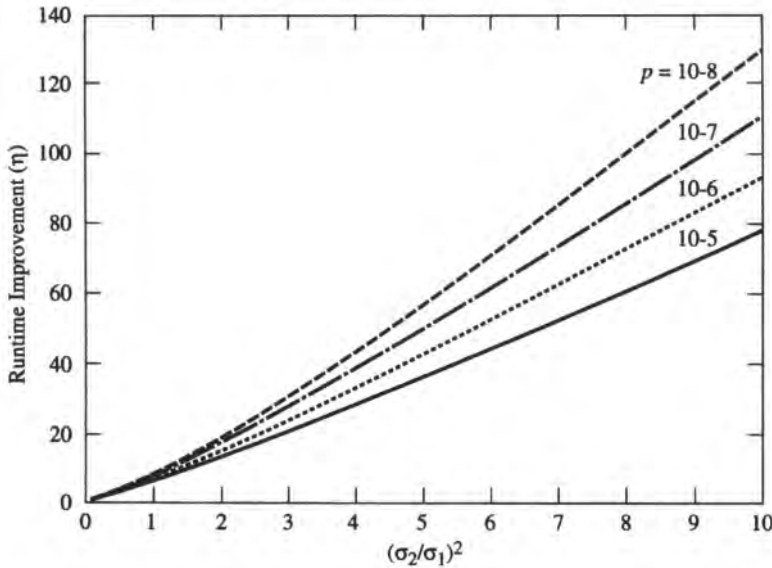


Figure 11.43. Improvement factor for mixed QA (under stated conditions) for various values of BER.

fast machine. We remark that this same technique can also be used in conjunction with importance sampling applied to $N_1(t)$, and the resulting improvement could then be even larger.⁽⁷⁰⁾

11.3. Summary

In this chapter we have considered means of estimating within simulation the two main types of criteria used to assess the quality of signals: the signal-to-noise ratio (SNR), which is the appropriate measure for analog signals (but also useful for digital signals), and various aspects of error production which are appropriate for digital signals.

In the first instance, we showed that if “signal” and “noise” are defined by a mean-squared criterion, which is reasonable, then a cross-correlation procedure between the input and output signals could be used to extract the relative amounts of signal and noise, hence compute the SNR. This computation can be achieved either through a time-domain procedure or a frequency-domain procedure, the particulars of which were indicated. It was also seen that the computational requirements for SNR estimation are relatively modest.

Estimation of error-rate (ER) measures in Monte Carlo simulation was shown to be very demanding of computer run time for very small error rates (say, $< 10^{-6}$) or for measures based on error production in fixed intervals within which large numbers of symbols could occur. The computational demand is exacerbated for nonconstant channels since, in effect, we have to develop ER estimates for all channel “states.” We outlined several approaches to estimating the reliability of ER estimates.

Because of the computational demands of MC simulation, various alternatives have been proposed for mitigating this disadvantage, with varying degrees of run-time improvement and varying degrees of generality. First we considered tail extrapolation (TE), a technique which artificially increases the production of errors by the use of “pseudothresholds,” which allows

us to extrapolate to the (more or less) correct value. The method is relatively intuitive, fairly easy to apply, and produces moderate, but useful gains.

We then considered importance sampling (IS), a conceptual cousin of TE in the sense that it is based on increasing the production of errors artificially, but uses a more rigorous route. It is thus capable of much larger run-time improvements, but using it presents a number of practical difficulties that may limit its application to all but cases that would otherwise be impossible. But the potential power of IS is such that it is worthwhile presenting, and since it is a relatively active area of research, the difficulties mentioned may well be overcome. The issues involved have been outlined. It is also worthwhile mentioning that IS and related ideas have been applied to great effect in discrete-event and queueing problems, which appear to lend themselves more naturally to the IS construct than do link-related problems.

We then developed various aspects of a general approach that we called quasianalytical (QA), also commonly referred to as semianalytic simulation. The QA approach can be strictly justified for linear systems, but can be applied (less usefully) in a nonlinear context. In the latter situation, a variation of QA called mixed QA (MQA) can be useful. The underlying idea behind QA is that in a linear system Gaussian noise remains Gaussian. Hence we do not have to generate noise sequences explicitly. We only have to simulate the distortion and analytically superimpose the noise that would be present. This yields a potentially very large improvement, as discussed. The idea of combining simulation and analysis can be extended in many directions, and for many problems this combination of tools may be the most expedient approach to obtaining a solution. Several examples of this combination were provided.

We also considered the problem of estimating error-rate measures in channels where the average error production in operationally meaningful intervals tends not to remain constant, as in fading channels. As mentioned above, this situation increases the demands on run time because we have to estimate the error rate in a statistically representative number of channel conditions. We discussed some alternatives for dealing with this problem. One useful approach is to derive from a waveform simulation a finite-state model, one common version of which is a hidden Markov model (HMM). Assuming the model is reasonably accurate, we can then use it to generate typical error sequences or use analytical means to obtain certain measures of interest. This procedure is explored in Case Study IV, Chapter 12.

There is not a “best” way of approaching the BER estimation problem, or a best universal technique. In practice there is a tradeoff among engineering manpower that may be required to set up the problem, the accuracy of a technique, and the computational demands. The latter in turn will depend on the machine that is used and the nature of the particular investigation, e.g., whether only a few runs are necessary or hundreds of them. Thus, techniques might be well used in combination. For example, the tradeoff space might be narrowed using QA simulation, even though it may be somewhat inaccurate, and the final down-selection might be based on Monte Carlo simulation.

References

1. M. C. Jeruchim and R. J. Wolfe, Estimation of the signal-to-noise ratio (SNR) in communication simulation, in *Conference Record, IEEE Global Telecommunications Conference (GlobeCom)*, Dallas, Texas (November 1989).
2. W. H. Tranter and M. D. Turner, Signal-to-noise ratio estimation in digital computer simulation of low-pass and bandpass systems with applications to analog and digital communications, Final Report—Vol. III, Contract NAS 9-14848, University of Missouri-Rolla Technical Report CSR-77-6 (July 1977).

3. M. D. Turner, W. H. Tranter, and T. W. Eggleston, The estimation of signal-to-noise ratios in digital computer simulation of low-pass and bandpass systems, presented at 1977 IEEE Midcon Conference (November 1977).
4. N. L. Johnson and S. Kotz, *Continuous Univariate Distributions*, Vol. 2, Houghton-Mifflin, Boston (1970).
5. R. M. Gagliardi and C. M. Thomas, PCM data reliability monitoring through estimation of signal-to-noise ratio, *IEEE Trans. Commun. Tech.* **COM-16**(3), 479–486 (1968).
6. A. Papoulis, *Signal Analysis*, McGraw-Hill, New York (1977).
7. J. P. C. Kleijnen, *Statistical Techniques in Simulation*, Part I, Dekker, New York (1974).
8. A. E. Newcombe and S. Pasupathy, Error rate monitoring for digital communications, *Proc. IEEE* **70**(8), 805–828 (1982).
9. J. B. Scholz, Error performance monitoring of digital communications systems, *Aust. Telecomm. Res. J.* **25**(2), 1–26 (1991).
10. J. B. Scholz and B. R. Davis, Error probability estimator structures based on analysis of the receiver decision variable, *IEEE Trans. Commun.* **43**(8), 2311–2315 (1995).
11. N. Celandroni, E. Ferro, and F. Potorti, Quality estimation of PSK modulated signals, *IEEE Commun. Mag.* **35**(7), 50–55 (1997).
12. J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*, Chapman and Hall, New York (1964).
13. M. Fisz, *Probability and Mathematical Statistics*, 3rd ed., Wiley, New York (1963).
14. A. M. Mood, *Introduction to the Theory of Statistics*, McGraw-Hill, New York (1950).
15. V. K. Rohatgi, *Statistical Inference*, Wiley, New York (1984).
16. H. Cramer, *Mathematical Methods of Statistics*, Princeton University Press, Princeton, New Jersey (1946).
17. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, U.S. Department of Commerce, National Bureau of Standards, Washington, D.C. (1964).
18. E. S. Pearson and H. O. Hartley, *Biometrika Tables for Statisticians*, Vol. I, Cambridge University Press, Cambridge (1954).
19. M. C. Jeruchim, Techniques for estimating the bit error rate in the simulation of digital communication systems, *IEEE J. Select. Areas Commun.* **SAC-2**(1), 153–170 (1984).
20. B. R. Davis, The effect of correlation on probability of error estimates in Monte Carlo simulations, *J. Elect. Electron. Eng. Aust.* **8**(4), 222–230 (1988).
21. W. Turin, *Digital Transmission Systems: Performance Analysis and Modeling*, McGraw-Hill, New York (1998).
22. M. D. Knowles and A. I. Drukarev, Bit error rate estimation for channels with memory, *IEEE Trans. Commun.* **36**(6), 767–769 (1988).
23. M. Kendall and A. Stuart, *The Advanced Theory of Statistics*, Vol. 2, 4th ed., Charles Griffin, London (1979).
24. L. N. Kanal and A. R. K. Sastry, Models for channels with memory and their applications to error control, *Proc. IEEE* **66**(7), 724–744 (1978).
25. M. Zorzi, Outage and error events in bursty channels, *IEEE Trans. Commun.* **46**(3), 349–356 (1998).
26. P. M. Crespo, R. M. Pelz, J. P. Cosmas, and J. Garcia-Frias, Results of channel profiles for DECT, *IEEE Trans. Commun.* **44**(8), 913–917 (1996).
27. S. B. Weinstein, Estimation of small probabilities by linearization of the tail of a probability distribution function, *IEEE Trans. Commun. Technol.* **COM-19**(6), 1149–1155 (1971).
28. M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, 3rd ed., Vol. 2, Hafner, New York (1961).
29. D. J. Gooding, Performance monitoring techniques for digital receivers based on extrapolation of error rate, *IEEE Trans. Commun. Technol.* **COM-16**(3), 380–387 (1968).
30. P. Balaban, Statistical evaluation of the error rate of the fiberoptic repeater using importance sampling, *Bell Syst. Tech. J.* **55**(6), 745–766 (1976).
31. K. S. Shanmugan and P. Balaban, A modified Monte Carlo simulation technique for the evaluation of error rate in digital communication systems, *IEEE Trans. Commun.* **28**(11), 1916–1924 (1980).
32. B. R. Davis, An improved importance sampling method for digital communication system simulations, *IEEE Trans. Commun.* **COM-34**(7), 715–719 (1986).
33. P. M. Hahn and M. C. Jeruchim, Developments in the theory and application of importance sampling, *IEEE Trans. Commun.* **COM-35**(7), 706–714 (1987).
34. D. Lu and K. Yao, Improved importance sampling technique for efficient simulation of digital communication systems, *IEEE J. Select. Areas Commun.* **6**(1), 67–75 (1988).
35. M. C. Jeruchim, P. M. Hahn, K. P. Smyntek, and R. T. Ray, An experimental investigation of conventional and efficient importance sampling, *IEEE Trans. Commun.* **37**(6), 578–587 (1989).
36. J. S. Sadowsky and J. A. Bucklew, On large deviations theory and asymptotically efficient Monte Carlo estimation, *IEEE Trans. Inform. Theory* **36**(3), 579–588 (1990).

37. R. J. Wolfe, M. C. Jeruchim, and P. M. Hahn, On optimum and suboptimum biasing procedued for importance sampling in communication simulation, *IEEE Trans. Commun.* **38**(5), 639–647 (1990).
38. J. S. Sadowsky, A new method for Viterbi decoder simulation using importance sampling, *IEEE Trans. Commun.* **38**(9), 1341–1351 (1990).
39. J. S. Sadowsky and R. K. Bahr, Direct-sequence spread spectrum multiple access communication with random signature sequences: A large deviation analysis, *IEEE Trans. Inform. Theory* **37**(3), 514–527 (1991).
40. K. Ben Letaief and J. S. Sadowsky, Computing bit-error probabilities for avalanche photodiode receivers by large deviations theory, *IEEE Trans. Inform. Theory* **38**(3), 514–527 (1992).
41. H.-J. Schlegbusch, On the asymptotic efficiency of importance sampling techniques, *IEEE Trans. Inform. Theory* **39**(2), 710–715 (1993).
42. J.-C. Chen, D. Lu, J. S. Sadowsky, and K. Yao, On importance sampling in digital communications—Part I: Fundamentals, *IEEE J. Select. Areas Commun.* **11**(3), 286–299 (1993).
43. J.-C. Chen and J. S. Sadowsky, On importance sampling in digital communications—Part II: Trellis-coded modulation, *IEEE J. Select. Areas Commun.* **11**(3), 300–308 (1993).
44. W. A. Al-Qaq, M. Devetsikiotis, and J. K. Townsend, Importance sampling methodologies for simulation of communication systems with time-varying channels and adaptive equalizers, *IEEE J. Select. Areas Commun.* **11**(3), 317–327 (1993).
45. M. Devetsikiotis and J. K. Townsend, An algorithmic approach to the optimization of importance sampling parameters in digital communication simulation, *IEEE Trans. Commun.* **41**(10), 1464–1473 (1993).
46. P. J. Smith, M. Shafi, and H. Gao, Quick simulation: A review of importance sampling techniques in communications systems, *IEEE J. Select. Areas Commun.* **15**(4), 597–613 (1997).
47. J. A. Bucklew, *Large Deviations Techniques in Decision, Simulation, and Estimation*, Wiley, New York (1990).
48. A. Schwartz and A. Weiss, *Large Deviations for Performance Analysis: Queues, Communications, and Computing*, Chapman & Hall, London (1995).
49. W. A. Al-Qaq, M. Devetsikiotis, and J. K. Townsend, Stochastic gradient optimization of importance sampling for the efficient simulation of digital communication systems, *IEEE Trans. Commun.* **43**(12), 2975–2985 (1995).
50. W. A. Al-Qaq and J. K. Townsend, A stochastic importance sampling methodology for the efficient simulation of adaptive systems in frequency nonselective Rayleigh fading, *IEEE J. Select. Areas Commun.* **15**(4), 614–625 (1997).
51. Z. Haraszti and J. K. Townsend, The theory of direct probability redistribution and its application to rare event simulation, in *Proc. International Conference on Communications ICC'98*, IEEE, Piscataway, New Jersey (1998).
52. J. K. Townsend, Z. Haraszti, J. A. Freebersyser, and M. Devetsikiotis, Simulation of rare events in communications networks, *IEEE Commun. Mag.* **36**(8), 36–41 (1998).
53. M. Villen-Altamirano *et al.*, Enhancement of the accelerated simulation method RESTART by considering multiple thresholds, in *Proc 14th International Teletraffic Congress, ITC 14*, Juan-les-Pains, France (1994), Vol. 1a, pp. 797–810.
54. P. Shahabuddin, P. Glasserman, and P. Heidelberger, Splitting for rare event simulation: Analysis of simple cases. In *1996 Winter Simulation Conference*, Coronado, California (1996), pp. 302–306.
55. V. K. Prabhu, Error rate considerations for digital phase-modulation systems, *IEEE Trans. Commun. Technol.* **COM-17**(1), 33–42, (1969).
56. I. Korn, *Digital Communications*, Van Nostrand Reinhold, New York (1985).
57. V. K. Prabhu, Error rate considerations for coherent phase-shift keyed systems with co-channel interference, *Bell Syst. Tech. J.* **48**(3), 743–767 (1969).
58. M. C. Jeruchim and F. E. Lilley, Spacing limitations of geostationary satellites using multilevel coherent PSK signals, *IEEE Trans. Commun.* **COM-20**(5), 1021–1026 (1972).
59. M. I. Irshid and I. S. Salous, Bit error probability for coherent M -ary PSK systems, *IEEE Trans. Commun.* **39**(3), 349–352 (1991).
60. M. Vanderaar, J. Budinger, and P. Wagner, Least reliable bits coding (LRBC) for high data rate satellite communications, in *AIAA 14th International Communications Satellite Systems Conference*, Washington, D.C. (1992) pp. 507–514.
61. H. L. Berger and T. J. Kolze, Bit error rate performance of coded 8PSK, in *AIAA 16th International Communications Satellite Systems Conference*, Washington, D.C. (1996).
62. D. J. Torrieri, The information-bit error rate for block codes, *IEEE Trans. Commun.* **COM-32**(4), 474–476 (1984).
63. A. J. Viterbi, Convolutional codes and their performance in communications systems, *IEEE Trans. Commun. Technol.* **COM-19**(5), 751–772 (1971).

64. A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding*, McGraw-Hill, New York (1979).
65. M. C. Jeruchim, On the coding gain for degraded channels, *IEEE Trans. Commun.* **COM-34**(5), 492–496 (1986).
66. J. R. Yee and E. J. Weldon, Evaluation of the performance of error-correcting codes on a Gilbert channel, *IEEE Trans. Commun.* **43**(8), 2316–2323 (1995).
67. M. Zorzi and R. R. Rao, On the statistics of block errors in bursty channels, *IEEE Trans. Commun.* **45**(6), 651–659 (1997).
68. A. Seyoum and N. C. Beaulieu, Semianalytical simulation for estimation of block-error rates on fading channels, *IEEE Trans. Commun.* **46**(7), 916–920 (1998).
69. M. Pent, L. LoPresti, G. D’Aria, and G. DeLuca, Semianalytic BER estimation by simulation for nonlinear bandpass channels, *IEEE J. Select. Areas Commun.* **6**(1), 34–41 (1988).
70. M. L. Steinberger, P. Balaban, and K. S. Shanmugan, On the effects of uplink noise, in *Conference Record: International Conference on Communications (ICC ’81)*, IEEE, New York (1981).

This page intentionally left blank

Four Case Studies

The application of specific simulation techniques, and particularly groups of techniques, is typically best illustrated by examples. Case studies are simply extended examples. In this chapter we present four case studies that have been chosen to collectively illustrate as broad a range as possible of ideas and methods and their synthesis into a methodology for solving particular problems. The specific subset of techniques invoked in each case study will be outlined in the introduction to each one.

12.1. Case Study I: 64-QAM Equalized Line-of-Sight Digital Radio Link in a Fading Environment

12.1.1. Introduction

The objective of this case study is to evaluate the performance of a 90-Mb/s 64-QAM line-of-sight (LOS) digital radio system in a multipath fading environment in the 4-GHz frequency band. For this case study the performance in question is defined to be the *outage probability*, which is the fraction of the time that the error probability exceeds a given value (10^{-3} in this case), for some (unfaded) signal-to-noise ratio. In order to speak meaningfully about outage probability as defined, there is an implicit assumption that the channel is in a more or less fixed state for a large enough number of symbols for the notion of error probability to be meaningful. This is one of two major assumptions underlying this case study, which we now formalize.

■ *Assumption 1:* Quasistatic channel. The multipath channel on a typical route and under typical atmospheric conditions is very slowly varying in relation to the data rate (90 Mbps) or to the system response. Consequently, for purposes of performance evaluation we can treat the channel as *quasistatic*, meaning that the channel can be considered in a fixed state for a large number of symbols. The behavior of the system over a long duration can therefore be visualized as a sequence of time segments, each with locally constant fading parameters. This assumption effectively means that during each time segment we have replaced a time-varying system by a time-invariant one. This has a significant consequence, namely, that a cascade connection of simulation models is now commutative. This allows us to rearrange modules as needed, in order to expedite simulation. ■

For a given channel condition, the error probability either will or will not meet the requirement. Thus, it can be seen that in order to assess the outage probability, we will have to

simulate the system performance under a large number of channel conditions (this statement will be made more precise later). It is therefore imperative to have efficient methods of error probability evaluation for this purpose. Particular methods will be discussed later, but all such methods require linearity in order to be applicable. This is the second fundamental assumption that we invoke, and again formalize below.

■ **Assumption 2: Linearity.** The system, including the channel, is assumed linear. This leads to two significant consequences. First, it means that the output of the system can be viewed as the superposition of a single basic pulse. This idea is used to derive an improved version of the quasianalytical method. The second major implication, which is the crux of the QA method, is that the (assumed) Gaussian noise at the receiver does not have to be explicitly generated, but can be incorporated analytically. ■

Aside from the channel characteristics, the error probability will itself depend on various aspects of the system, most notably the ability of the receiver equalizer to compensate for the fades. An understanding of the system behavior in any particular state is served by examining in detail the behavior of individual system elements, which can be done, for example, by observing waveforms in and out of system elements, plotting eye diagrams, measuring power spectra, and measuring settling time for the equalizer. While it is desirable to have an understanding of the system behavior in all channel states, the large number of states to be examined makes this an impractical goal. For this more detailed look, a few well-chosen channel conditions will suffice to give the system engineer a satisfactory assessment of the system design.

We can see from the foregoing that two different methodologies are called for to deal with the two modes of study discussed. These methodologies are now summarized.

1. *Selected Channel Snapshot Methodology.* We use the term *snapshot* to denote the situation where the fading channel is in some (relatively) fixed state. As we mentioned, insight into the system performance can be obtained by selecting one or more specific fading channel conditions. Any one such condition leads to the *selected* channel snapshot analysis. This type of analysis is performed in order to study the effects of fading on various subsystems such as filters, equalizers, etc. The sensitivity of system performance to channel fading parameters and the optimization of the system to these variations is also done this way. The observed performance measures for this type of analysis are as follows:

- a. Signal waveforms
- b. Power spectra
- c. Eye diagrams and scatter plots
- d. Error rates for the selected channels

This methodology (which we will sometimes abbreviate as SCS1) will be examined in more detail in Section 12.1.3.

2. *Stochastic Channel Sequence Methodology.* In the selected snapshot methodology, only a few channels are typically chosen, hence that approach does not lead to sufficient statistical information regarding system behavior over the possible range of channel conditions. To obtain such information, we resort to a Monte Carlo approach that we call a stochastic channel sequence methodology. Here, what is randomly generated are the various channel conditions. This is done by drawing the channel parameters in accordance with their probabilistic descriptions. For each set of drawn parameters we evaluate the BER. As mentioned, we need to do this evaluation efficiently because of the large number of channel

conditions examined. Once we obtain a sufficiently large number of BERs we can construct an empirical distribution and estimate the outage probability.

This methodology (which we will sometimes abbreviate as SCS2) will be examined in more detail in Section 12.1.4.

This case study brings together a number of previously discussed simulation tools, techniques, and methodological ideas, as well as some new ones, as listed here:

- Generation of M -ary maximum-length digital sequences
- Generation of random variables with specified pdfs
- FIR filtering and FFT processing
- Synchronization via cross-correlation
- Methods of incorporating equalization
- New methods of quasianalytical estimation
- An additional form of QA via Gaussian quadrature integration
- Methodological ideas: complexity reduction, partitioning of a problem, quasistatic approximation.

The application of these various items will be seen as we go along.

12.1.2. The System Model

The simulation block diagram of the 64-QAM digital radio system is shown in Figure 12.1, with a somewhat expanded version of the actual modem layout given in Figure 12.2. Note the slight difference between the two with respect to the location of the receiver filter and the synchronization subsystems. In Figure 12.1 the synchronization functions are assumed lumped together in the “Demodulator” block. There is essentially no loss in verisimilitude by doing so, and we gain computational efficiency because of our chosen method for simulating the synchronization functions, which performs the carrier phase and symbol timing recovery in one operation. We now briefly describe each of the blocks in the system diagram, in the order shown.

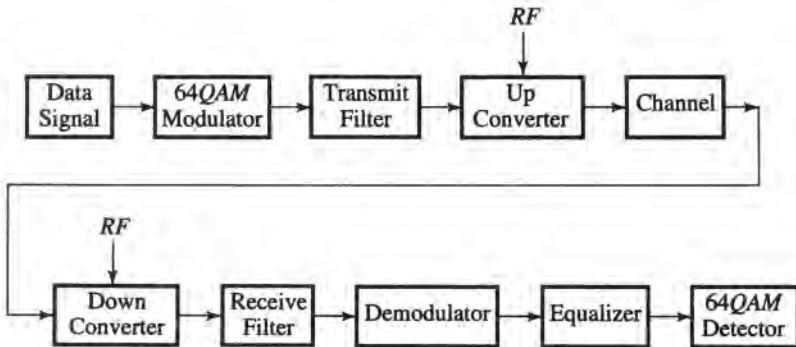


Figure 12.1. Simulation block diagram for Case Study I.

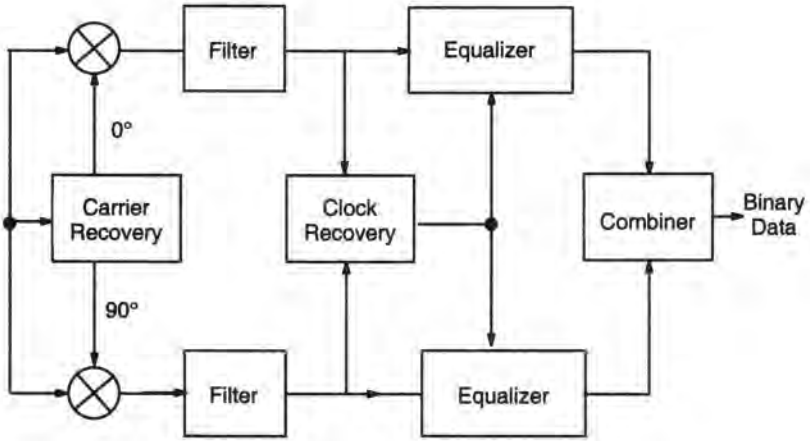
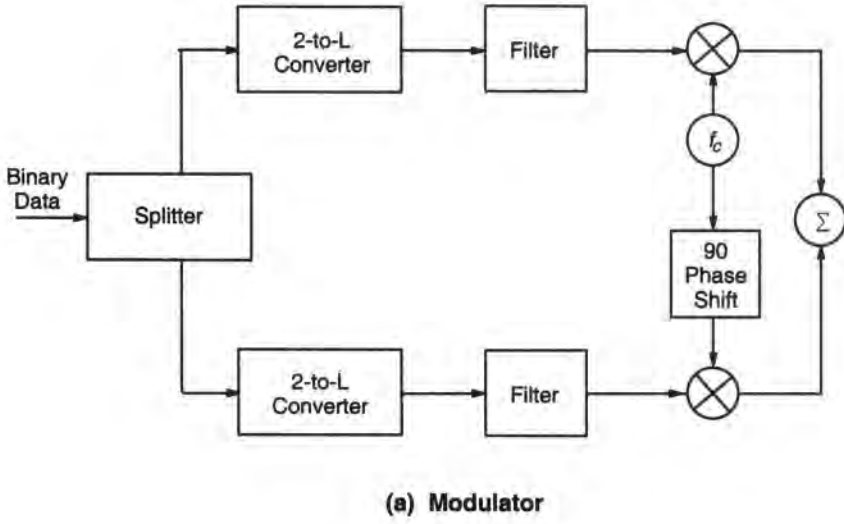


Figure 12.2. More detailed block diagram of part of the system. (a) Modulator; (b) demodulator.

12.1.2.1. The Source

As seen from Figure 12.1.2, original binary data are formatted into two L -ary sequences, with $L = 8$. We label these data sequences $\{A_n\}$ and $\{B_n\}$ for the in-phase and quadrature channels, where A_n and B_n can be expressed as

$$A_n = a_n A, \quad B_n = b_n A \quad (12.1.1a)$$

where A is a scaling parameter (used to set the SNR), and

$$a_n \in \{\pm 1, \pm 3, \pm 5, \pm 7\}, \quad b_n \in \{\pm 1, \pm 3, \pm 5, \pm 7\} \quad (12.1.1b)$$

The conversion of these logical sequences into waveforms is done by associating a “pulse” with each symbol. The simple choice of a fixed pulse shape is made, modulated with the symbol value, i.e., conventional PAM in each channel:

$$A_n \rightarrow A_n p_s(t); \quad B_n \rightarrow B_n p_s(t) \quad (12.1.2)$$

It is often attempted to implement $p_s(t)$ as $p_{T/2}(t)$, the unit rectangular pulse with duration T .

It is customary in this type of problem to generate the symbol sequences using PN (maximum-length shift register) sequences. Initially, we have to ask whether we should generate 64-ary sequences for the 64-ary symbol $S = \{A, B\}$ or simply two 8-ary sequences for $\{A\}$ and $\{B\}$, respectively. This is a tradeoff between accuracy and run time. In principle, it should be the first, but the coupling between the two channels is not so strong as to invalidate the use of the two 8-ary sequences. The run time, on the other hand, is proportional to L^m , where m is the system memory in symbols, and L here is either 64 or 8. It is plain to see that in the former case, the computational load will be enormous. For example, with $m = 7$ (assumed later), 64^7 is on the order of 10^{12} , while 8^7 is on the order of 10^6 . While the latter is quite an improvement, it still requires a significantly long sequence. Consequently, we also tried using much shorter ordinary random sequences, produced by random number generators. The justification for this approach will be given later.

12.1.2.2. Modulator

The modulator output is of the form

$$s(t) = \text{Re} \sum_n (A_n + jB_n) p_s(t - nT) \exp(j\omega_{\text{IF}} t)$$

and since the system is simulated entirely in the lowpass-equivalent domain, its lowpass equivalent is simply modeled as

$$\tilde{s}(t) = \sum_n (A_n + jB_n) p_s(t - nT) \quad (12.1.3)$$

For the selected channel snapshot methodology, we explicitly simulated the filtering following the modulator. For the stochastic channel sequence methodology, we used a not quite equivalent, but more efficient approach based on the idea that we do not need to have an explicit model for $p_s(t)$ if we have one directly for the transmitted pulse $p(t)$, which we will describe shortly. In other words, if we assume the *intended* $p(t)$ is a sufficiently accurate model, we need not bother generate the intermediate waveforms.

12.1.2.3. Filters

We assume the *raised-cosine* shape $H_{\text{RC}}(f)$ for the overall transfer function (Section 8.9.2), with the rolloff parameter indicated below. The optimum partitioning of this transfer function is to implement the square-root filter $\sqrt{H_{\text{RC}}(f)}$ both at the transmitter and the receiver (Section 8.9.2). In order to generate a pulse whose Fourier transform is $\sqrt{H_{\text{RC}}(f)}$, the input to the filter must be an impulse (delta function). Hence, for any modulator pulse $p_s(t)$ the transmitter filter would have to be preceded by a pulse-to-impulse (PTI) filter to produce (ideally) an impulse at the input to the square-root raised-cosine filter there. The PTI filter

transfer function is simply the inverse of the Fourier transform of the input pulse. For the rectangular pulse $p_{T/2}(t)$ the PTI transfer function is (to within a constant) $\pi f T / \sin(\pi f T)$. In this case, therefore, the transfer functions for ISI-free transmission are given by[†]

$$H(f) = \begin{cases} \frac{\pi f T}{\sin(\pi f T)}; & |f| \leq \frac{1-\beta}{2T} \\ \frac{\pi f T}{\sin(\pi f T)} \cos^2 \left\{ \frac{\pi T}{2\beta} \left(f - \frac{1-\beta}{2T} \right) \right\}; & \frac{1-\beta}{2T} \leq |f| \leq \frac{1+\beta}{2T} \\ 0; & |f| > \frac{1+\beta}{2T} \end{cases} \quad (12.1.4)$$

$$H_T(f) = \begin{cases} \frac{\pi f T}{\sin(\pi f T)} \\ \frac{\pi f T}{\sin(\pi f T)} \cos \left\{ \frac{\pi T}{2\beta} \left(f - \frac{1-\beta}{2T} \right) \right\} \\ 0 \end{cases} \quad (12.1.5)$$

$$H_R(f) = \begin{cases} 1 \\ \cos \left\{ \frac{\pi T}{2\beta} \left(f - \frac{1-\beta}{2T} \right) \right\} \\ 0 \end{cases} \quad (12.1.6)$$

where $H(f)$ is the overall transfer function, $H_T(f)$ is the transmitter filter, and $H_R(f)$ is the receiver filter. If the PTI filter is assumed to do its job perfectly, we do not need to include it; all we need to do is generate discrete impulses as the source output (properly weighted) instead of pulses.

In the instances when we emulated the evolution of a waveform (which was not done for all methodological approaches), both the transmit and receive filters were simulated in the frequency domain as FIR filters, using the FFT with the overlap-and-add method of splicing. We assumed that an impulse response duration of eight symbol times is adequate, even for small β . Assuming an overlap-and-add ratio of 2 (see Section 4.1.2.2.3), we find for the FFT-FIR filter an impulse response duration $T_h = 16T = 1.067 \mu\text{s}$. Thus, the resultant FFT frequency-domain array for the FIR filter has a frequency increment $\Delta F = 1/T_h = 0.9375 \text{ MHz}$ and a total frequency span $F_h = N\Delta F = 1/T_s = 240 \text{ MHz}$. The number of samples in the FFT array is then $N = 256$.

The frequency responses of the FFT filters were generated by following the procedure described in Section 4.1.5. The filter frequency response was read into an array with $N/2$ samples. This array was transformed into the time domain using the IFFT. The $N/2$ samples of the impulse response were shaped by a Hamming window. The time domain array was then augmented with zeros to N samples. The frequency response that was obtained from the impulse response via an N -sample FFT was used for “frequency-domain” filtering.

[†]We use here a differently normalized version of the raised-cosine family than we did in Chapter 8, as it suits present purposes better. The “excess bandwidth” parameter β here is the parameter with the same symbol in (8.9.5) divided by $2T$.

12.1.2.4. The Transmitted Signal

The transmitted signal (the output of the upconverter) is now given by

$$x(t) = \text{Re} \sum_n (A_n + jB_n) p(t - nT) \exp(j\omega_c t)$$

where $p(t)$ is a square-root raised-cosine pulse and f_c is the carrier frequency. Given the descriptions for A_n and B_n in (12.1.1), the transmitted signal is a rectangular constellation of the form shown in Chapter 11. The lowpass-equivalent transmitted signal is

$$\tilde{x}(t) = \sum_n (A_n + jB_n) p(t - nT) \quad (12.1.7a)$$

$$= \tilde{x}_p(t) + j\tilde{x}_q(t) \quad (12.1.7b)$$

As mentioned above, the design objective is to produce a pulse $p(t)$ whose Fourier transform is a square-root raised-cosine shape.

12.1.2.5. The Channel

The channel model used for this case study is Rummmler's model, which is described in Chapter 9. For convenience we repeat here the form of the lowpass-equivalent transfer function for the minimum-phase case:

$$\tilde{H}_c(f) = a[1 - be^{-j2\pi(f-f_0)\tau}] \quad (12.1.8)$$

where the parameter $\tau = 6.3$ ns. Typical amplitude and delay responses for this channel were given in Chapter 9. For later purposes it will also be useful to have the channel impulse response, the Fourier transform of $\tilde{H}_c(f)$:

$$\tilde{h}_c(t) = a[\delta(t) - be^{j2\pi f_0 \tau} \delta(t - \tau)] \quad (12.1.9a)$$

$$= \tilde{h}_{c,p}(t) + j\tilde{h}_{c,q}(t) \quad (12.1.9b)$$

By equating real and imaginary parts of (12.1.9a) and (12.1.9b), it can be seen that

$$\tilde{h}_{c,p}(t) = a\delta(t) - ab(\cos \omega_0 \tau) \delta(t - \tau) \quad (12.1.10a)$$

$$\tilde{h}_{c,q}(t) = -ab(\sin \omega_0 \tau) \delta(t - \tau) \quad (12.1.10b)$$

For the selected channel snapshot methodology we simply choose values for (a, b, f_0) ; the basis for the selection might be, for example, to obtain a fade of a given depth. For the stochastic channel sequence methodology, a sequence of triplets (a, b, f_0) is generated from the channel parameter distributions that are described in Section 9.1.3.7. It should be reiterated that in the latter case, any choice of (a, b, f_0) is considered to remain fixed for an indefinitely long period of time for purposes of BER evaluation.

The channel output $c(t)$, which becomes the input to the receiver, is simply the transmitted signal filtered by the channel. Thus,

$$\tilde{c}(t) = \tilde{x}(t) * \tilde{h}_c(t) \quad (12.1.11a)$$

$$= \tilde{c}_p(t) + j\tilde{c}_q(t) \quad (12.1.11b)$$

where

$$\tilde{c}_p(t) = \tilde{x}_p(t) * \tilde{h}_{c,p}(t) - \tilde{x}_q(t) * \tilde{h}_{c,q}(t) \quad (12.1.12a)$$

$$\tilde{c}_q(t) = \tilde{x}_q(t) * \tilde{h}_{c,p}(t) + \tilde{x}_p(t) * \tilde{h}_{c,q}(t) \quad (12.1.12b)$$

Observe the skew symmetry in the two components of $\tilde{c}(t)$, which is characteristic of filtering in quadrature systems. In the SCSI methodology, we actually simulate $\tilde{c}(t)$ because we want to be able to probe the system at that point as well as other points. But the channel filtering can actually be performed analytically and directly because of the form of $\tilde{h}_c(t)$. That is, the **δ -functions** allow us to do the filtering by inspection. As an example, take the first term on the right-hand side of (12.1.12a),

$$\tilde{x}_p(t) * \tilde{h}_{c,p}(t) = a\tilde{x}_p(t) - ab(\sin \omega_0 \tau)\tilde{x}_p(t - \tau) \quad (12.1.13)$$

As discussed in Chapter 8, to evaluate the second term in the preceding equation may require interpolation and resampling, or if τ is close to a multiple of T_s , we could make it equal to that (but we need not alter it in the term $\sin \omega_0 \tau$).

In the SCSI2 methodology there is no actual need for performing these operations because of our arrangement of the simulation block diagram. Because of the quasistatic assumption, filtering will be commutative, so we may as well evaluate the waveform directly at the output of the receiver filter, as we will see.

The input to the receiver also consists of an additive noise source, which can be considered part of the channel. It is assumed to be a white Gaussian process with one-sided noise spectral density N_0 . Because the receiver is linear, its processing can be considered separately, at least up to the equalizer. It will be notationally clearer to do so; we will take up receiver noise in Section 12.1.2.8.

12.1.2.6. Receiver Filtering

The actual location of the receiver shaping filter is at baseband, though its location in the block diagram of Figure 12.1 implies it is at IF. This is, of course, immaterial for present purposes since we are simulating at complex baseband. (In practice there will also be an IF filter, which for this study we can assume is distortionless.) The receiver filter's transfer function is simply the square-root raised-cosine function, as discussed earlier. Denoting the

[†]Although we speak of the receiver filter, there are two as shown in Figure 12.2. Note that both cases are subsumed by (12.1.14) since $z(t)$ is complex.

corresponding impulse response as $h_{\sqrt{RC}}(t)$, which is purely real, we have for the (noiseless) filter output[†]

$$\tilde{y}(t) = \tilde{c}(t) * h_{\sqrt{RC}}(t) \quad (12.1.14)$$

From (12.1.11a) this is equal to

$$\begin{aligned} \tilde{y}(t) &= \tilde{x}(t) * \tilde{h}_c(t) * h_{\sqrt{RC}}(t) \\ &= \tilde{x}(t) * h_{\sqrt{RC}}(t) * \tilde{h}_c(t) \end{aligned}$$

Define $\tilde{z}(t) = \tilde{x}(t) * h_{\sqrt{RC}}(t)$; using (12.1.7a), this is

$$\tilde{z}(t) = \sum_n (A_n + jB_n)p(t - nT) * h_{\sqrt{RC}}(t)$$

and since, by construction, $p(t) * h_{\sqrt{RC}}(t) = p_{RC}(t)$, the designed raised-cosine pulse, we have

$$\tilde{z}(t) = \sum_n (A_n + jB_n)p_{RC}(t - nT) \quad (12.1.15)$$

and

$$\tilde{y}(t) = \tilde{z}(t) * \tilde{h}_c(t) \quad (12.1.16)$$

Evaluating (12.1.16) directly is straightforward and can be done by inspection since $p_{RC}(t)$ is known, and we have seen that convolving with the channel transfer function is immediate.

12.1.2.7. Demodulator

In the actual system, the demodulator consists essentially of the carrier recovery subsystem followed by downconversion to baseband using the recovered carrier. Since we are operating at (complex) baseband, downconversion to baseband is implicit, as is the RF-to-IF downconversion shown in the block diagram. The effect of phase error, however, must be explicitly incorporated. As we hinted earlier, for simulation purposes we select a “virtual” synchronizer which performs carrier and symbol synchronization simultaneously. First, we will discuss the synchronization aspect, and then the translation to baseband.

(a) *Synchronization.* We chose the cross-correlation method described in Chapter 10. In terms of the present notation, this technique consists in the evaluation of

$$\mathcal{R}(\tau) = \int_{t_1}^{t_2} \tilde{s}(t - \tau)\tilde{y}(t) dt = |\mathcal{R}(\tau)| \exp[j\phi(\tau)] \quad (12.1.17)$$

and declaring symbol synchronization to be established for the $\tau = \hat{\tau}$ that maximizes $|\mathcal{R}(\tau)|$. If sampling occurs nominally mid-symbol, then the actual sampling epoch is at $\hat{\tau} + T/2$. The recovered carrier phase is then given by $-\phi(\hat{\tau})$.

The cross-correlation was carried out in the frequency domain, using 30–40 symbols. Note that the obtained delay applies up to the input to the equalizer (see Figure 12.2). However, the optimality of this delay, such as it may be, is altered by the action of the

equalizer, the output of which is sampled for decisions. Thus, the delay $\hat{\tau} + T/2$ was used as a starting point for a search of optimal sampling time for the synchronous equalizer. The $T/2$ -spaced equalizer, however, is virtually insensitive to the initial delay, as will be seen. Similarly, the estimate of carrier phase $-\phi(\hat{\tau})$ applies to the signal as seen by the demodulator, and may not be optimal. But the equalizer itself will compensate for static phase error, so the precise value of reference phase should not be critical.

(b) *Demodulated Waveform.* Demodulation takes place by multiplying the incoming signal $y(t)$ by $2k_d \cos[\omega_{IF}t - \phi(\hat{\tau})]$ to obtain the in-phase baseband channel (rail) $d_p(t)$, and by $2k_d \sin[\omega_{IF}t - \phi(\hat{\tau})]$ to obtain the quadrature channel $d_q(t)$, where $2k_d$ is a demodulator constant. Expressing

$$\tilde{y}(t) = \tilde{y}_p(t) + j\tilde{y}_q(t)$$

and

$$d(t) = d_p(t) + jd_q(t)$$

it is straightforward to show that

$$d(t) = \tilde{y}(t) \exp[j\phi(\hat{\tau})] \quad (12.1.18)$$

It will be useful for subsequent use to expand $d(t)$ in terms of earlier quantities. Using (12.1.15), (12.1.16), and (12.1.18), one can show after straightforward manipulations that (for $k_d = 1$)

$$\begin{aligned} d_p(t) = & \sum_n A_n p_{RC}(t - nT) * [\tilde{h}_{c,p}(t) \cos \phi(\hat{\tau}) - \tilde{h}_{c,q}(t) \sin \phi(\hat{\tau})] \\ & - \sum_n B_n p_{RC}(t - nT) * [\tilde{h}_{c,p}(t) \sin \phi(\hat{\tau}) + \tilde{h}_{c,q}(t) \cos \phi(\hat{\tau})] \end{aligned} \quad (12.1.19a)$$

and

$$\begin{aligned} d_q(t) = & \sum_n B_n p_{RC}(t - nT) * [\tilde{h}_{c,p}(t) \cos \phi(\hat{\tau}) - \tilde{h}_{c,q}(t) \sin \phi(\hat{\tau})] \\ & + \sum_n A_n p_{RC}(t - nT) * [\tilde{h}_{c,p}(t) \sin \phi(\hat{\tau}) + \tilde{h}_{c,q}(t) \cos \phi(\hat{\tau})] \end{aligned} \quad (12.1.19b)$$

It can be seen that we can define a basic (complex) pulse that is the *unequalized* system response to a single pulse in either channel. That is,

$$A_I p_{RC}(t - lT) \rightarrow A_I [u_p(t - lT) + ju_q(t - lT)] \quad (12.1.20a)$$

where $u_p(t)$ is the response in the I rail and $u_q(t)$ is the response in the Q rail. Similarly, a single pulse in the Q rail generates a response

$$B_I p_{RC}(t - lT) \rightarrow B_I [u_p(t - lT) - ju_q(t - lT)] \quad (12.1.20b)$$

and it is clear that

$$u_p(t) = p_{RC}(t) * [\tilde{h}_{c,p}(t) \cos \phi(\hat{\tau}) - \tilde{h}_{c,q}(t) \sin \phi(\hat{\tau})] \quad (12.1.21a)$$

$$u_q(t) = p_{RC}(t) * [\tilde{h}_{c,p}(t) \sin \phi(\hat{\tau}) + \tilde{h}_{c,q}(t) \cos \phi(\hat{\tau})] \quad (12.1.21b)$$

Therefore we can write for the I-channel demodulated output

$$d_p(t) = \sum_n [A_n u_p(t - nT) - B_n u_q(t - nT)] \quad (12.1.22a)$$

$$d_q(t) = \sum_n [B_n u_p(t - nT) + A_n u_q(t - nT)] \quad (12.1.22b)$$

which we can also express succinctly as

$$d(t) = \sum_n S_n u(t - nT) \quad (12.1.23)$$

where $S_n = A_n + jB_n$ and $u(t) = u_p(t) + ju_q(t)$. Each rail of $d(t)$ is now fed to an equalizer, which we consider next.

12.1.2.8. Receiver Noise

Depending upon the methodology, we will generally not generate noise explicitly: we do so only in conjunction with one of the equalizer training options described below. Nevertheless, it will be useful to look at the properties of this noise as it would actually present itself, particularly since its autocorrelation properties enter directly in the covariance matrix inversion method of setting the equalizer tap gains. Further related discussion is provided in connection with calibration (Section 12.1.3.2). As mentioned before, the receiver noise source is assumed to be additive white Gaussian with PSD N_0 . As long as $\phi(\hat{\tau})$ is constant, or at least very slowly varying, the Gaussianness of the demodulated noise still holds. (See Ref. 1; the same principle will also be used in the case study following.) The specific value of $\phi(\hat{\tau})$ will not affect the noise power or PSD in either rail as long as we assume the passband in-phase and quadrature components of the noise are uncorrelated. This assumption is consistent with the original AWGN assumption. Thus, with the demodulator constant $2k_d$ (assuming as before $k_d = 1$), the noise two-sided PSD at the equalizer input in the I or Q rail is given by $N_0 |H(f)|^2$, where $H(f)$ is the filtering transfer function between the receiver input and the equalizer input. In the case at hand, we have $H(f) = H_{\sqrt{RC}}(f)$, the square-root raised-cosine filter's Fourier transform (Section 12.1.3.2). This PSD defines the correlation properties of the noise at the equalizer input, which was labeled $\mathbf{p}$ in Section 8.9.6. Thus, labeling $N_p(t)$ and $N_q(t)$ the I- and Q-rail noise processes, respectively, we can define a complex noise source

$$N(t) = N_p(t) + jN_q(t)$$

so that the (complex) waveform at the equalizer input can be stated as

$$V(t) = d(t) + N(t)$$

12.1.2.9. Equalization

The initial design for the equalizer was a seven-tap linear transversal filter (i.e., tapped delay line) driven by the LMS algorithm. Both a synchronous (T -spaced) and $T/2$ -spaced equalizer were studied. Depending on methodology, the steady-state (or converged) tap gains were obtained in one of two ways, which we will describe below. First, we will obtain the form of the equalizer output for later use.

(a) *The Equalizer Output Waveform.* Since we are dealing with the lowpass equivalent, the equalizer can be represented as a “complex” structure, meaning the tap gains are complex quantities: $C_k = \alpha_k + j\beta_k$. (In a physical implementation at baseband, there will be four separate tapped delay lines, and the actual adaptation algorithm may independently adjust each set of tap gains.) The equalizer output can be written as

$$R(t) = \sum_{k=-K}^K C_k V(t - kT') = \sum_{k=-K}^K C_k [d(t - kT') + N(t - kT')] \quad (12.1.24)$$

Again, it will be simpler to proceed by considering only the signal portion of (12.1.24); we will reinsert the noise at the end of the subsection. Thus, the noiseless $R(t)$ has the I- and Q-channel components

$$\begin{aligned} R_p(t) &= \sum_{k=-K}^K \alpha_k d_p(t - kT') - \beta_k d_q(t - kT') \\ R_q(t) &= \sum_{k=-K}^K \alpha_k d_q(t - kT') + \beta_k d_p(t - kT') \end{aligned}$$

Rewriting the last equations in terms of (12.1.22), we get in the I rail

$$\begin{aligned} R_p(t) &= \sum_{k=-K}^K \alpha_k \sum_n [A_n u_p(t - nT - kT') - B_n u_q(t - nT - kT')] \\ &\quad - \sum_{k=-K}^K \beta_k \sum_n [B_n u_p(t - nT - kT') + A_n u_q(t - nT - kT')] \end{aligned}$$

which can be rearranged as

$$\begin{aligned} R_p(t) &= \sum_n A_n \sum_{k=-K}^K [\alpha_k u_p(t - nT - kT') - \beta_k u_q(t - nT - kT')] \\ &\quad - \sum_n B_n \sum_{k=-K}^K [\beta_k u_p(t - nT - kT') + \alpha_k u_q(t - nT - kT')] \end{aligned}$$

with a similar (skew-symmetric) expression for $R_q(t)$. Now define

$$r_{pp}(t) = \sum_{k=-K}^K [\alpha_k u_p(t - kT') - \beta_k u_q(t - kT')] \quad (12.1.25a)$$

$$r_{qp}(t) = \sum_{k=-K}^K [\beta_k u_p(t - kT') + \alpha_k u_q(t - kT')] \quad (12.1.25b)$$

In other words, $r_{pp}(t)$ is the equalized pulse in the I rail due to a single pulse input in the same rail, and $r_{qp}(t)$ is the single pulse in the I rail due to a single pulse input in the Q rail. Therefore, the equalizer output waveforms can be stated as

$$R_p(t) = \sum_n A_n r_{pp}(t - nT) - \sum_n B_n r_{qp}(t - nT) \quad (12.1.26a)$$

$$R_q(t) = \sum_n B_n r_{pp}(t - nT) + \sum_n A_n r_{qp}(t - nT) \quad (12.1.26b)$$

where again skew symmetry holds, i.e., $r_{qq}(t) = r_{pp}(t)$ and $r_{pq}(t) = -r_{qp}(t)$.

For expository purposes it is convenient to assume, as has been done implicitly, that the delay lines are analog. When they are sampled at particular instances, the sums become sums of numbers rather than waveforms, which could have been obtained equally well by sampling before the equalizer. Mathematically, there is no difference, but the same is not true for the implementations. We will use (12.1.26) later for performance evaluation. Below we will turn briefly to the methods of obtaining the equalizer tap gains. Before doing so, however, we will establish the noise power at the equalizer output.

The noise waveform at the equalizer output is given by

$$\begin{aligned} N_e(t) &= \sum_{k=-K}^K C_k N(t - kT') \\ &= \sum_{k=-K}^K \alpha_k N_p(t - kT') - \beta_k N_q(t - kT') + j \sum_{k=-K}^K \beta_k N_p(t - kT') + \alpha_k N_q(t - kT') \end{aligned}$$

Let us compute the noise power separately in the I and Q rails. Assuming independence between N_p and N_q , we find for the noise power in either rail

$$\sigma^2 = P_N \sum_k (\alpha_k^2 + \beta_k^2) + P_N \sum_{k \neq l} (\alpha_k \alpha_l + \beta_k \beta_l) \rho(k, l) \quad (12.1.27)$$

where $P_N = E(N_p^2(t)) = E(N_q^2(t))$ [see (12.1.34)] and

$$P_N \rho(k, l) = E(N_p(t - kT') N_p(t - lT')) = E(N_q(t - kT') N_q(t - lT'))$$

Given the $\{C_k\}$, the noise power can be computed directly for BER evaluation using the QA methods described later. An alternative method of calculating the noise power is given in Section 12.1.3.2.

(b) *The Stochastic Gradient Algorithm.* This algorithm was used in the selected snapshot methodology, when it was desired to study the temporal properties of the adaptations. The stochastic gradient algorithm was described in Section 8.9.6. For present purposes, the algorithm can be rephrased in discrete-time form as

$$\mathbf{C}(n+1) = \mathbf{C}(n) - \Delta \mathbf{G}(n) \quad (12.1.28)$$

where $\mathbf{C}(n)$ is the vector of tap weights at tune $t_0 + nT$, where t_0 is the sampling epoch:

$$\mathbf{C}(n) = (C_{-K}(n), \dots, C_0(n), \dots, C_K(n)) \quad (12.1.29a)$$

$\mathbf{G}(n)$ is the gradient at time n ,

$$\mathbf{G}(n) = -E[\varepsilon(n)\mathbf{V}^*(n)] \quad (12.1.29b)$$

where

$$\varepsilon(n) = d(n) - R(n) \quad (12.1.29c)$$

and

$$\mathbf{V}(n) = (V(nT + KT'), \dots, V(nT), \dots, V(nT - KT')) \quad (12.1.29d)$$

Here $\Delta = 0.02$ is a heuristically adjusted step size (see Section 8.9.6).

Many different versions of the gradient algorithm have been devised (see, e.g., Refs. 6 and 11 in Chapter 8). Commonly, the expectation in (12.1.29b) is simply replaced with the “instantaneous” value of the gradient $-\varepsilon(n)\mathbf{V}^*(n)$. Then, because the true values $\{d(n)\}$ will not be known (except for a training sequence) the form (12.1.28) cannot literally be implemented. So, an estimate $\hat{d}(n)$ (usually taken to be the decision) will replace $d(n)$. Another variation is to use instead of the gradient a weighted version of past values. This effectively approximates the expectation in (12.1.29b). For example, instead of $\mathbf{G}(n)$ as defined above, use a “filtered” version $\bar{\mathbf{G}}(n) = \sum_{k=n-M}^n \mu_k \mathbf{G}(k)$, where M is the memory of the filter. For this case study the simpler version given by (12.1.28)–(12.1.29) was implemented, using $\mathbf{G}(n) = -\varepsilon(n)\mathbf{V}^*(n)$; and in simulation, of course, we do know the sequence $\{d(n)\}$, so that estimated values need not be used.

The MSE was monitored to ensure convergence. The equalizer was assumed converged when no significant change (less than 0.1%) was achieved in five consecutive iterations. Since optimum performance of the equalizer is highly dependent on the sampling time, the adaptation algorithm included a search for the optimum sampling time (minimum MSE). In order for the linear MSE equalizer to converge to the correct tap weights, Gaussian noise has to be injected at the input to the receive filter. Failing to do that will result in tap settings for a zero-forcing equalizer. The noise power density N_0 is computed from the specified unfaded signal-to-noise ratio.

(c) *Covariance Matrix Inversion.* The covariance matrix inversion (CMI) approach for computing the optimum equalizer tap settings was used for the stochastic sequence simulations and for the selected snapshot simulations when behavior during convergence was not of interest. The CMI method could also be used to verify that the steady-state tap gains reached by the gradient algorithm were close to the correct values. The CMI approach is described in Section 8.9.6. It is equivalent to the training algorithm used in many communication systems. By this procedure, the optimum tap gains are computed in a single step from the overall single pulse response of the system $u(t)$, the autocorrelation function of the noise $N(t)$ defined in Section 12.1.2.8, and the signal-to-noise ratio. Because it is a one-step process, it is generally much faster than explicit equalizer adaptation.

The optimum tap gains $\mathbf{C}_{\text{opt}}$ are computed from (8.9.36) of Chapter 8, namely

$$\mathbf{C}_{\text{opt}} = \mathbf{\Gamma}^{-1} \mathbf{r}^*$$

where $\mathbf{\Gamma}$ and $\mathbf{r}$ are defined in Section 8.9.6. [In Chapter 8, $p(t)$ was used for what we call here $u(t)$.] The matrix $\mathbf{\Gamma}$ is sometimes called the channel covariance matrix, which accounts for the label that we have given this method.

In this particular case study we have assumed that the normalized autocorrelation function of the noise is given as $\rho(\tau) = 0$ for $\tau \neq 0$; this assumption is satisfied when the equalizer tap spacing is T and the receiver filter is the square-root raised-cosine filter. In the covariance matrix inversion approach the sampling time is also a parameter (t_0 in Section 8.9.6) and this parameter was optimized by tracking the minimum MSE given by (8.9.37) as a function of t_0 . Since $\mathbf{\Gamma}$ and $\mathbf{r}$ are in principle infinite in extent, they evidently have to be truncated to produce a finite-length equalizer.

12.1.2.10. The Detector

The outputs of the equalizer in the I and Q rails are fed to separate detectors, or decision devices, which make hard decisions on the symbols in the respective channels. Eight-level quantizers are implemented for this function.

12.1.3. The Selected Channel Snapshot Simulation

In this simulation methodology the cascade connection of the individual subsystems follows the same order as the actual system (except for the slight modifications noted), as shown in Figure 12.1, in order to make internal points of the system observable. Since we want to observe eye diagrams, spectra, etc., it is necessary to generate an evolving waveform in time. This we do by generating the source symbol sequences in one of two ways.

1. *PN Sequence Signal.* It is customary to use PN sequences or de Bruijn sequences (Chapter 7) to represent signals for simulation of systems using the QA method to estimate the BER. These sequences have the property that for a given system memory of m symbols, they generate all possible combinations of subsequences of m symbols and each combination appears only once per period. However, for higher level modulation and larger memory, the length of the sequence can be excessively long. For example, for an octal signal with a system memory $m = 6$, the sequence is $8^6 = 256 \times 10^3$ symbols long. The implications of a long symbol sequence for run time for error-rate calculations are evident.

2. *Random Signals.* The alternative to PN signal generation is to use a random sequence. In this case the data $\{A_n\}$ and $\{B_n\}$ are selected from uniformly distributed random variables. The possible advantage of this approach will be discussed later.

12.1.3.1. Simulation Procedure

The system was first initialized by establishing the appropriate signal and noise levels, using the calibration procedure described below. If the stochastic adaptation algorithm was used, the simulation was run for a few hundred symbols with the receiver noise source actually emitting noise samples using a random number generator. When the equalizer converged to the steady-state tap gain values, the noise source was disconnected and the tap coefficients fixed to the previously found “optimum” values. The simulation was then ready to be run in QA mode for BER evaluation. If the covariance matrix inversion method was used, the pulse response was computed first, then the covariance matrix established and the optimum tap weights evaluated. In either tap-gain setting approach, random or PN sequences would be generated to produce waveforms (in discrete time, of course) that would eventually be sampled at the input to the decision device for error-rate calculation.

12.1.3.2. Calibration Procedure

In digital communication it is common to express the probability of error as a function of either $\rho = E_b/N_0$, where E_b is the energy per bit and N_0 is the noise power spectral density, or the signal-to-noise ratio (SNR) $\zeta = S/N$, where S is the signal power and $N = N_0 B$ is the noise power in some reference bandwidth B . These quantities are normally defined at the input to the receiver, and sometimes at points in the baseband, for example, the input to the decision device. We will assume the former here, but in order to apply the QA method, whatever values of ρ or ζ are selected, it will be necessary to relate them to the value of noise power σ^2 that results at the input to the decision device. In the case at hand, it is also convenient to establish these quantities in the unfaded channel, so that they can provide a common reference for all channel conditions.

Starting with the definition of ρ , we have

$$N_0 = \frac{E_b}{\rho} = \frac{PT}{k\rho} \quad (12.1.30)$$

where P is the average signal power and k is the number of bits per symbols, which is 6 in this case. Now, ρ , k , and T are known parameters. In general, the power P can either be measured in the simulation or perhaps determined analytically. In the first case, we simply compute its estimate from the complex envelope of the signal $\tilde{c}(t)$ at the input to the receiver:

$$P \cong \frac{1}{2Q} \sum_{i=1}^Q |\tilde{c}(iT_s)|^2$$

for a sufficiently large number of simulation samples Q . In the case at hand, recalling that this is the unfaded signal, we can calculate P exactly from (see Section 8.13)

$$\begin{aligned} P &= \langle |\tilde{c}(t)|^2 \rangle \\ &= \frac{E(A_n^2)}{T} \int_{-\infty}^{\infty} |S(f)|^2 df \end{aligned} \quad (12.1.31)$$

where $S(f)$ is the Fourier transform of $p(t)$, the square-root raised-cosine pulse, which is precisely (8.9.8). It is simple to show that $\int_{-\infty}^{\infty} |S(f)|^2 df = 1$ (refer to Problem P8.28). Since $E(A_n^2) = A^2(L^2 - 1)/3$, which reduces to $21A^2$ in this case, we have

$$P = 21A^2/T$$

Note that this result depends specifically on the assumed form for $H_T(f)$. Now we have all the ingredients for computing N_0 in (12.1.30). The next step for QA calculation is to find σ^2 , the noise power at the input to the decision device in either the I rail or the Q rail. This is given by

$$\sigma^2 = k_d^2 N_0 B_N \quad (12.1.32)$$

where $2k_d$ is the demodulator constant and B_N is the noise bandwidth between the receiver input and the decision device input. We saw earlier that we can find σ^2 analytically. In general, if the analysis is difficult, it may be simpler to obtain B_N using a time-domain simulation procedure, which is simply to insert a discrete impulse at the receiver input. The resulting impulse response $h(t)$ of the chain of equipment between the receiver input and the equalizer output yields B_N by invoking Parseval's theorem:

$$B_N = \frac{\int_{-\infty}^{\infty} |H(f)|^2 df}{|H(0)|^2} = \frac{\int_{-\infty}^{\infty} |h(t)|^2 dt}{|\int_{-\infty}^{\infty} h(t) dt|^2} \quad (12.1.33)$$

Again, in this particular case it is also straightforward to calculate B_N directly and explicitly. First consider the noise power P_N at the receiver filter output in either channel, but before the equalizer. This can easily be shown to be

$$P_N = k_d^2 N_0 \int_{-\infty}^{\infty} |S(f)|^2 df \quad (12.1.34a)$$

$$= k_d^2 N_0 \quad (12.1.34b)$$

because the integral is unity as pointed out above. But, by definition of noise bandwidth,

$$P_N = k_d^2 N_0 |S(0)|^2 B_N \quad (12.1.35)$$

Hence, equating (12.1.35) to (12.1.34b), and recalling that $S(0) = \sqrt{T}$, we obtain

$$B_N = T^{-1} \quad (12.1.36)$$

Finally, the noise is enhanced to some degree by the equalizer. It is left as an exercise for the reader to show that in the case of the synchronous equalizer and for a square-root raised-cosine pulse, the noise bandwidth at the equalizer output is given by

$$B_N = T^{-1} \sum_k |C_k|^2 \quad (12.1.37)$$

If the calibration cannot be done analytically, it would be done during the initialization stage of the simulation.

12.1.1.3.3. Estimation of Error Probability

In the selected snapshot methodology, we used the conventional QA method (Chapter 11) to calculate the probability of error. The 64-QAM system has a square signal constellation, and the decision regions are square boxes for interior points or semiinfinite rectangles for exterior points. The distortion induced in the system (basically, only by the channel in this study) alters the original pattern. An error will occur if the noise is large enough to move the distorted point outside its decision region. The probability of error is computed from (11.2.95) and (11.2.96) in Chapter 11 for the interior points of the constellation. (For points on the periphery of the constellation, refer to Problem P11.18.) These equations yield P_e , the symbol error probability. An upper bound on bit error probability is simply $\text{BER} = P_e/k$.

As hinted above, both PN and random sequences were used to evaluate the error probability. The complete PN sequence gives a “complete” performance prediction since it generates all possible m -tuples of symbols for a system with an m -symbol memory. However, we saw earlier that such sequences can be very long. It has been demonstrated⁽²⁾ that good convergence to the actual error rate can be achieved by sequences as short as $0.1L^{m/2}$. This result was achieved for both truncated PN sequences and random sequences. The results were only slightly sensitive to the SNR operating point.

For shorter sequences it is preferable to use the random sequence rather than a PN sequence because with the latter the dominant m -tuples may appear in clusters. It is advisable

[†]In this subsection we will give simply a number triplet to represent the selected channel parameters (a, b, f_0) , in that order.

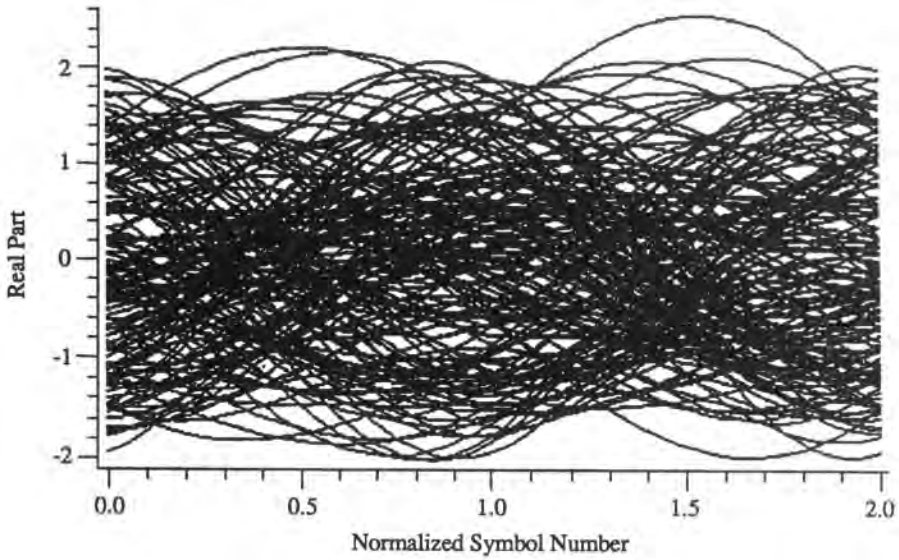


Figure 12.3. Eye diagram of faded signal.

to compute confidence intervals whenever random sequences or shorter than maximum-length PN sequences are used.

It is significant to note the following. The error probability computation for different values of E_b/N_0 is not mechanized simply by varying the E_b/N_0 parameter in an expression. This is because, as we saw, the equalizer tap gains are themselves functions of E_b/N_0 . Thus, the entire procedure must be replicated for *each* value of E_b/N_0 for which the error rate is calculated.

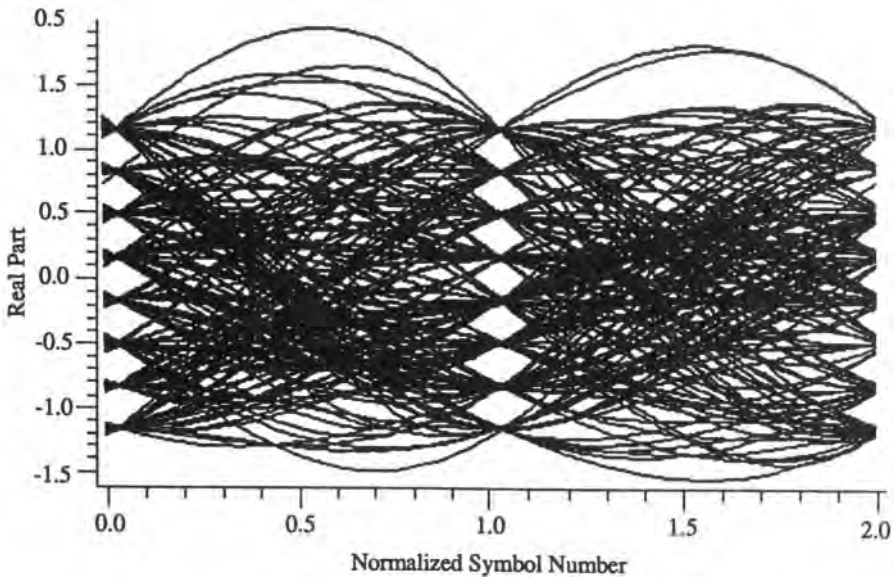


Figure 12.4. Eye diagram of equalized signal.

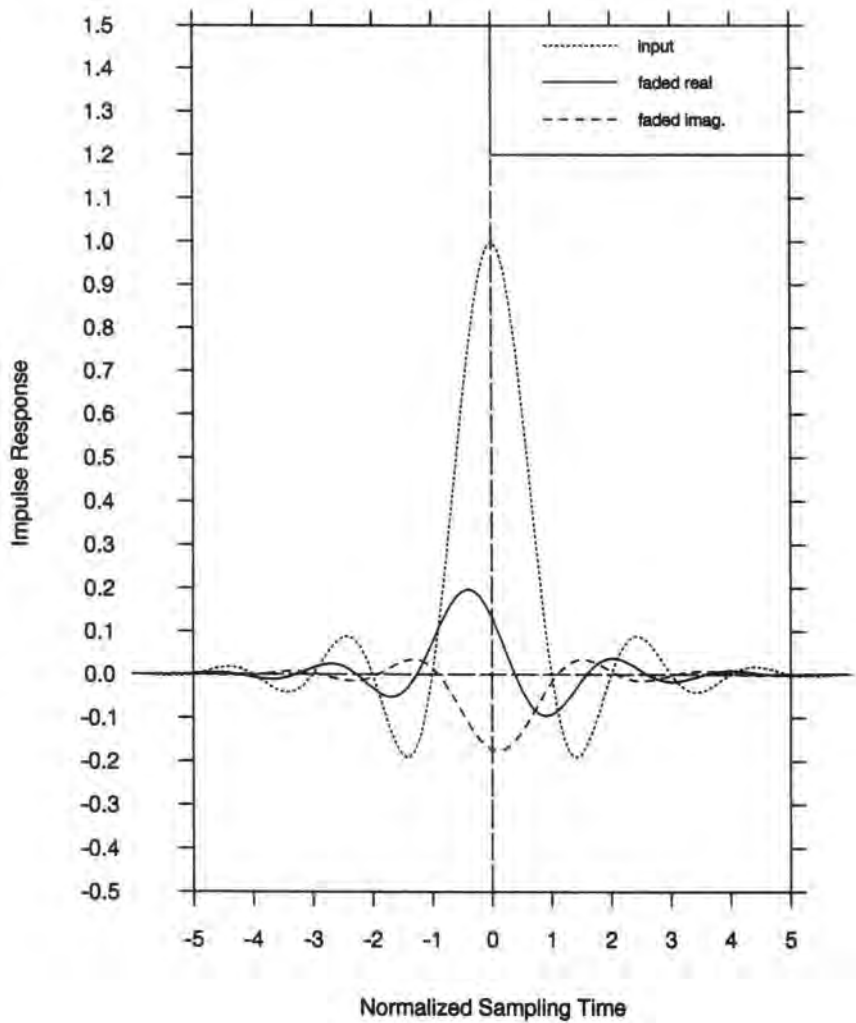


Figure 12.5. Unequalized impulse response for the Rummier channel with $A = 0$ dB, $B = 20$ dB, $f_0 = 5$ MHz, and filter rolloff = 0.25.

12.1.3.4. Selected Simulation Results

We present a few examples of the output of the selected channel snapshot simulation. For a channel with parameters[†] (1.0, 0.92, 0.0), the unequalized eye diagram is shown in Figure 12.3, while Figure 12.4 shows the eye diagram for the equalized signal using a seven-tap synchronous equalizer (SE). As another example, Figure 12.5 shows the unequalized single pulse response of the channel for $(1.0, 0.9, 5 \times 10^6)$, which can be compared in the same figure to the unfaded response (the raised-cosine pulse). Figures 12.6 and 12.7 show, respectively, the equalized pulse response for a seven-tap SE and a seven-tap fractionally spaced equalizer (FSE). It can be seen in this instance that the FSE “cleans up” the distortion considerably better than the SE. Finally, Figure 12.8 displays several error rate curves for the selected channels with parameters as shown on the figure.

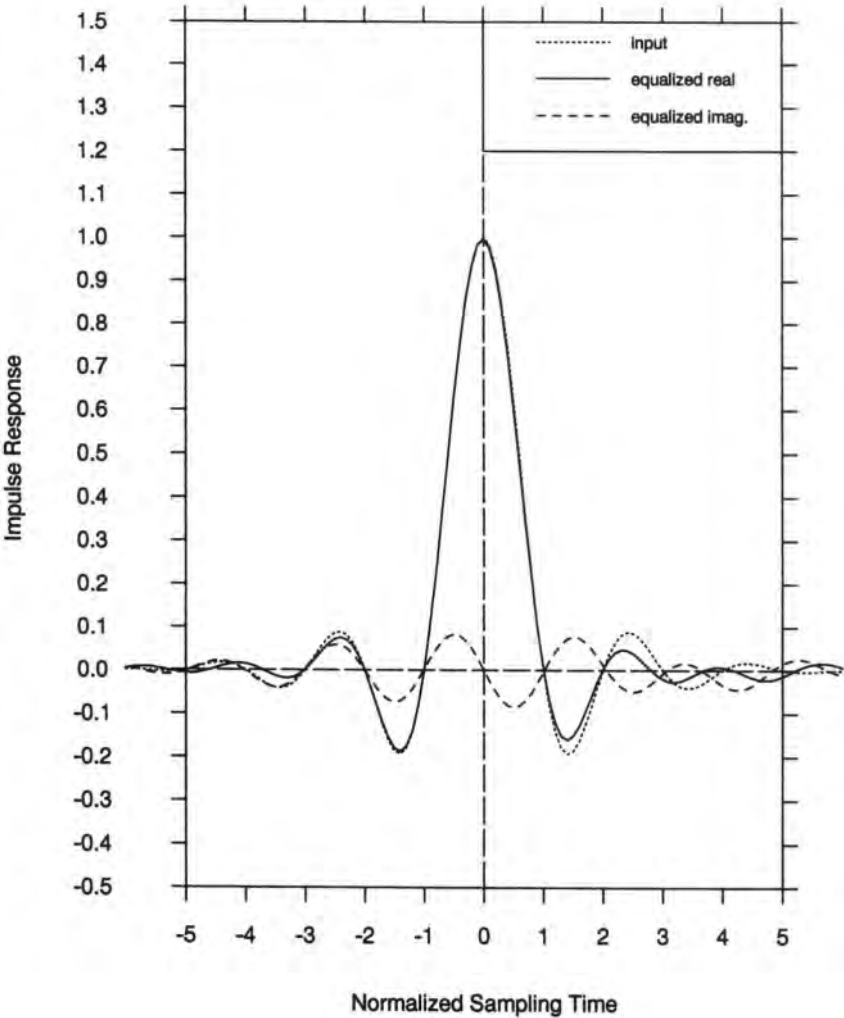


Figure 12.6. Equalized impulse response for the Rummler channel with $A = 0$ dB, $B = 20$ dB, $f_0 = 5$ MHz, and filter rolloff = 0.25; seven-tap T -spaced equalizer; unfaded SNR = 60 dB. For this case $MSE = 3.46 \times 10^{-3}$.

12.1.4. The Stochastic Channel Sequence Simulation

As mentioned in the introduction, the principal objective of this case study was to obtain the system's outage probability. In principle, this could be obtained from the very same simulation methodology described above, except that, instead of choosing particular channel conditions, we let these channel conditions be naturally generated by their statistical distributions. The only difficulty is that such a statistical approach requires a fairly large number of runs (30,000 were used in this case study) and hence the individual simulation run has to be computationally very efficient. This requirement, and the fact that for BER evaluation purposes *the internal nodes of the system do not have to be observable*, leads to a considerable restructuring and simplification of the system topology for simulation purposes. The evolution of this simplified system is shown in Figure 12.9. Initially, the topology must be as-

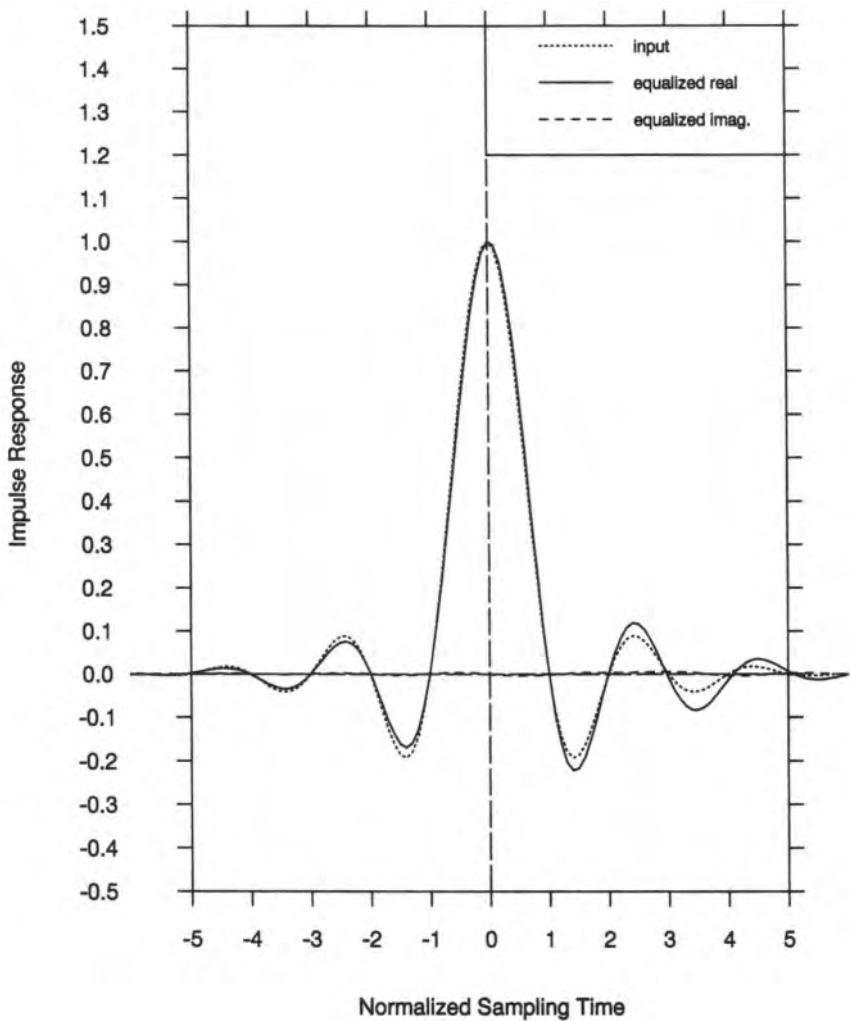


Figure 12.7. Equalized impulse response for the Rummler channel with $A = 0$ dB, $B = 20$ dB, $f_0 = 5$ MHz, and filter rolloff = 0.25; seven-tap $T/2$ -spaced equalizer; unfaded SNR = 60 dB. For this case MSE = 7.17×10^{-4} .

is in order to train the equalizer. The single channel-distorted pulse is first obtained and used in the covariance matrix inversion method to obtain the steady-state equalizer tap gains. Since the system is linear and time-invariant (because of the quasistatic assumption) the cascade connection of the models is commutative. We can therefore combine the transmit and receive filters and use as input a discrete impulse. The output of that combination is simply the unfaded raised-cosine pulse. This is then followed by the channel model and the equalizer. The equalizer can also be regarded as linear and time-invariant under the fundamental assumptions we have made. Hence, the output of the equalizer is the faded and equalized response of the system. But that response, as we have seen in Section 12.1.2.8, is simply a superposition of a single basic (complex) equalized pulse. That single-pulse response contains all the necessary information to evaluate the system BER and leads to alternative and more efficient methods of performance evaluation. These alternatives are summarized in the

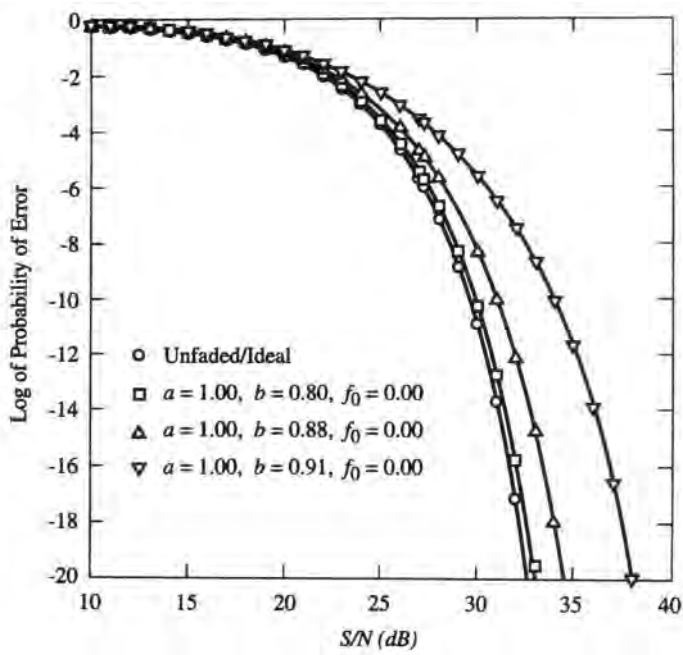


Figure 12.8. Error performance of 64-QAM in different fading environments.

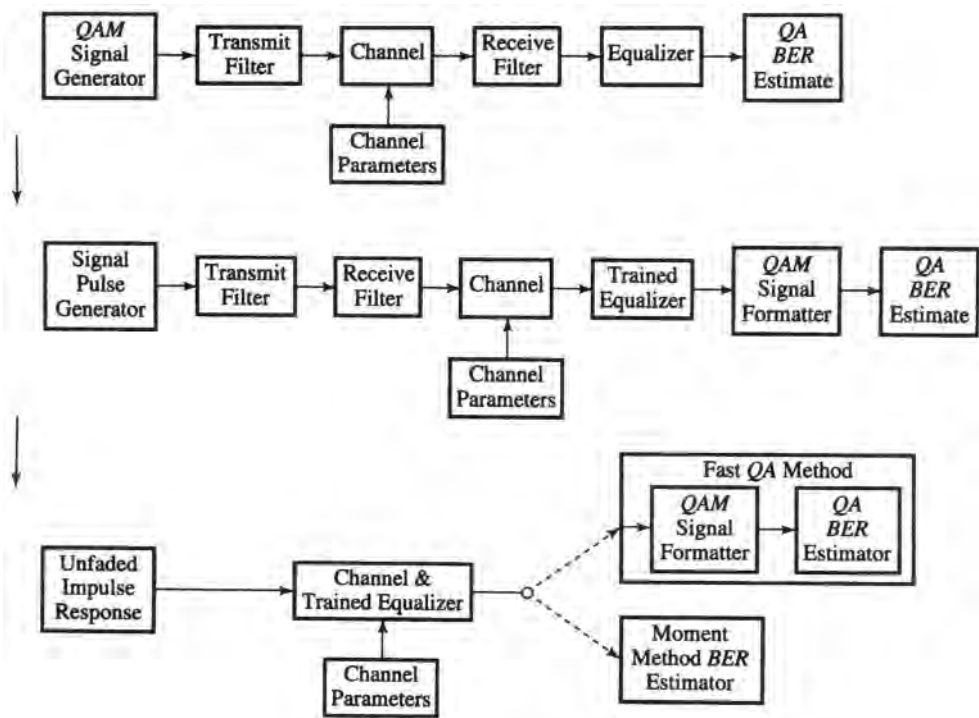


Figure 12.9. Evolution of Monte Carlo simulation methodology.

bottom part of Figure 12.9 as the “fast QA” method (of which there are actually two variants) and the moment method, both of which will be described below.

12.1.4.1. Stochastic Channel Sequence Generation

Before describing the performance evaluation methods, we should briefly describe the procedure that gives the name “stochastic” to this methodology, which is simply the random generation of the channel triplets (a, b, f_0) . In Chapter 9 we provided the distributions of these

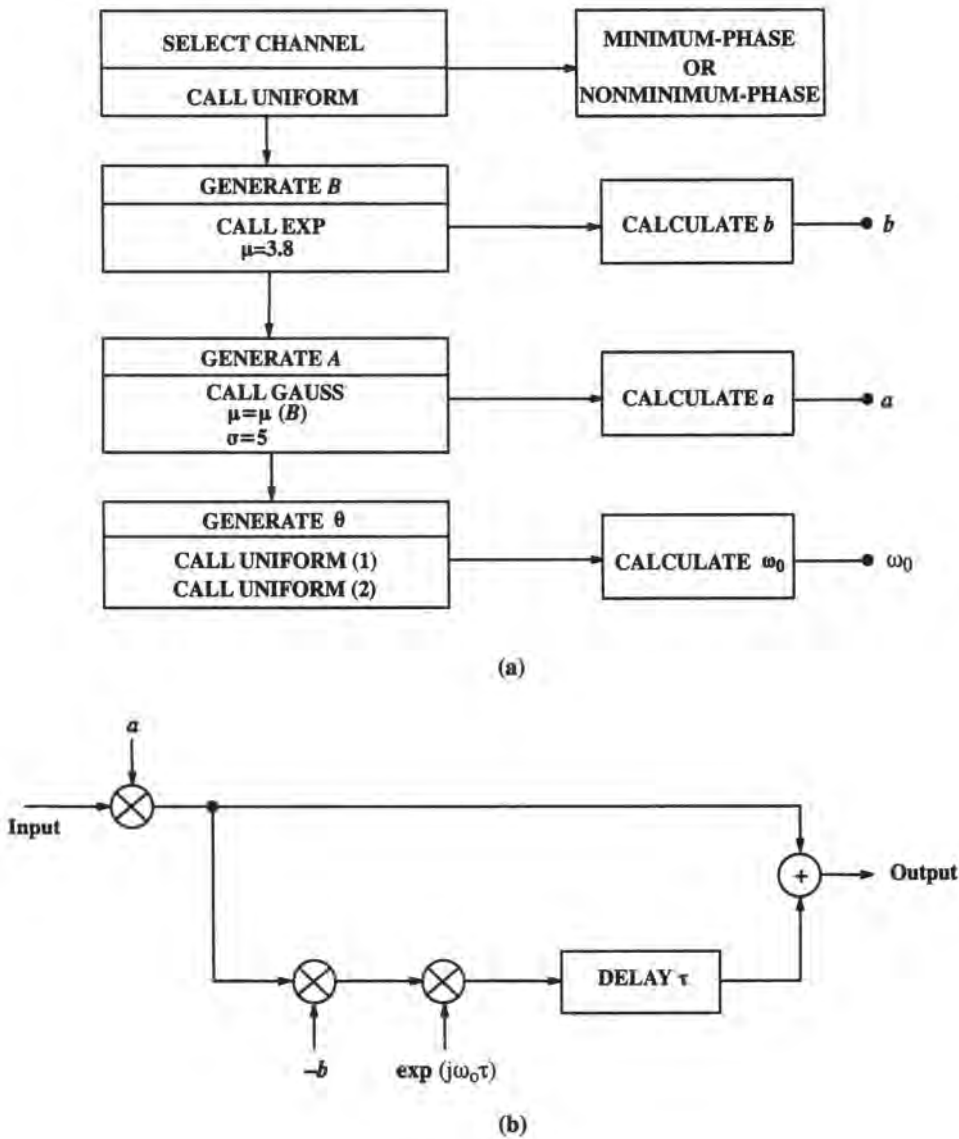


Figure 12.10. (a) Random number generation for the stochastic channel sequence methodology. The distributions of the parameters are described in Chapter 9; (b) effect of the channel impulse response on the signal, for the minimum-phase case.

parameters (in dB form for a and b). These parameters can be randomly generated by drawing from exponential, Gaussian, and uniform random number generators, all of which we know how to do (see Chapter 7). A visual summary of the channel generation procedure is given in Figure 12.10. Figure 12.10a shows the random number generators that have to be called, and it is worthwhile pointing out that aside from the parameter triplet (a, b, f_0) , there is another implied random quantity, namely whether the channel is in a minimum-phase or nonminimum-phase state. Each has 0.5 probability, hence the uniform RNG at the top of the figure. Figure 12.10b indicates diagrammatically the transformation of the signal by the channel impulse response in the minimum-phase case, in accordance with (12.1.9a).

12.1.4.2. Evaluation of Error Probability: Fast Quasianalytical Method 1 (FQA-1)

In the present approach, a waveform unfolding in time is *not* generated. As we just indicated, what we do produce is the response of the system to a single symbol input (in either the I or the Q rail). Because of linearity, the output in either channel can be conceived as a sum of scaled and delayed replicas of this single “unit” (complex) pulse response, e.g., $r_p(t) = r_{pp}(t) + jr_{qp}(t)$. This is implicit in (12.1.26), which we repeat here for convenience, but slightly modified by showing explicit limits on the summations:

$$R_p(t) = \sum_{n=-m/2}^{m/2} A_n r_{pp}(t - nT) - \sum_{n=-m/2}^{m/2} B_n r_{qp}(t - nT) \quad (12.1.38a)$$

$$R_q(t) = \sum_{n=-m/2}^{m/2} B_n r_{pp}(t - nT) + \sum_{n=-m/2}^{m/2} A_n r_{qp}(t - nT) \quad (12.1.38b)$$

where we must assume, in practice, that $R_p(t)$ and $R_q(t)$ have an effectively limited duration of $m + 1$ symbols. We can now apply the usual QA technique to determine the error rate. If we sample $R_p(t)$ and $R_q(t)$ at $t_0 + nT$, $n = 0, 1, \pm 2, \dots$, where $t_0 \equiv \hat{t} + T/2 + \delta$, and δ is a possible timing adjustment, the only waveform generation involved is to produce $r_{pp}(t)$ and $r_{qp}(t)$. We then have the noiseless component of the decision voltage, to which we add noise “analytically” in accordance with (12.1.32)–(12.1.37), and then call the erfc function to complete the process.

The process of evaluating (12.1.38) is illustrated in Figure 12.11. For brevity we will label $r_{pp}(t_0)$ as $r_{pp}(0)$, and $r_{pp}(t_0 + nT)$ as $r_{pp}(n)$, and similarly with r_{qp} . Thus, having first found $r_p(t)$, we can visualize the constants $r_{pp}(0), r_{pp}(1), r_{pp}(-1), \dots, r_{pp}(m/2)$, and $r_{pp}(-m/2)$ as tap gains of a shift register through which passes the sequence $\{A_n\}$, and similarly with r_{qp} . In this figure, $m = 6$ has been assumed. This diagram makes it clear that obtaining $r_p(t)$ is the only waveform generation involved. We might thus call this approach a *single-waveform-event* technique. Obtaining this single waveform takes very little computational effort. The bulk of that effort is in the repeated calculation of the symbol-spaced sampled values of $R_p(t)$ and $R_q(t)$ and the associated error function call. Note, however, the significant improvement in run time with respect to the conventional QA method, in which the waveform unfolds in time at T_s -spaced intervals. In FQA-1, the simulation in effect unfolds at T -spaced intervals. The improvement due to this fact alone is on the order (T/T_s) , which is typically at least one order of magnitude, and is further magnified by the number of filtering elements in the system.

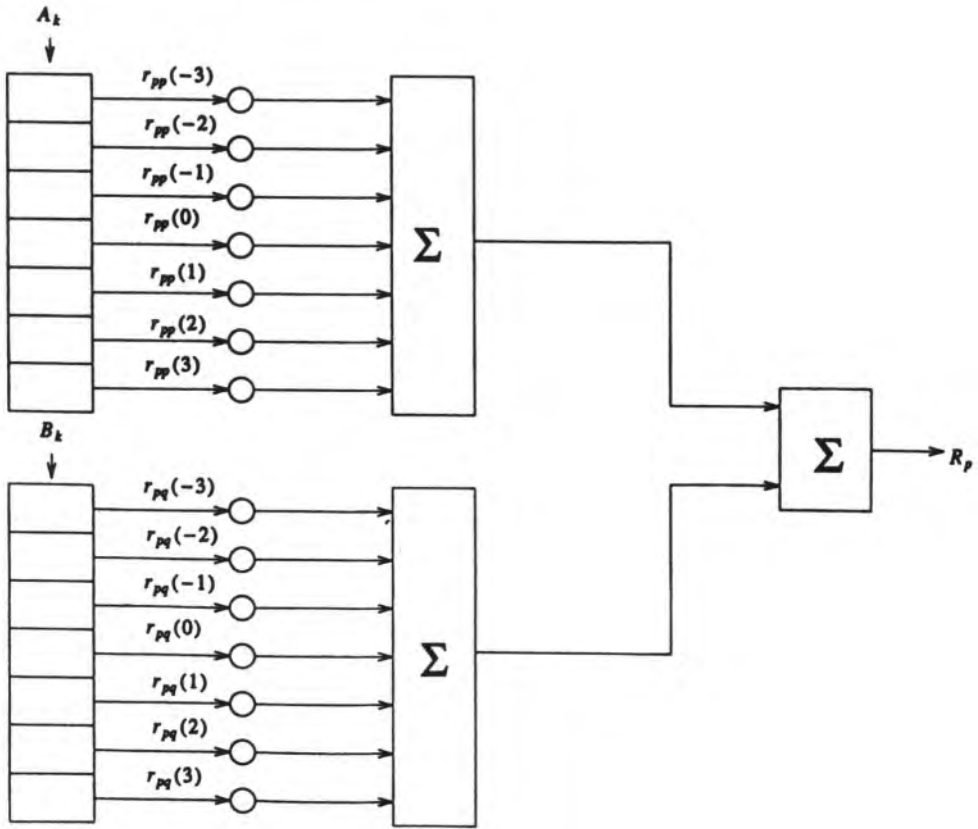


Figure 12.11. Illustration of FQA-1 method, with hypothetical “unit” equalized baseband pulse $r_p(t) = r_{pp}(t) + jr_{qp}(t)$.

12.1.4.3. Evaluation of Error Probability: Fast Quasianalytical Method 2 (FQA-2)

A significant further improvement to the computational efficiency of the previous technique can be obtained by a simple stratagem based on the observation that the products

$$A_n r_{pp}(t_0 \pm nT); \quad B_n r_{qp}(t_0 \pm nT)$$

have only a limited number of values and can be *precomputed and stored*, rather than be repeatedly recomputed as the symbol sequences evolve. Since A_n and B_n each has only eight possible values, and (as an illustration) assuming $m = 6$, two sets of 56 products are precomputed and stored in arrays, as shown in Figure 12.12. Shift registers for the A and B sequences then act as pointers to select the appropriate number from each column, and the numbers thus selected are summed. Thus, except for the initial creation of the arrays, there are no multiplications in this technique.

12.1.4.4. Evaluation of Error Probability: The Moment Method (Gaussian Quadrature)

Although reasonably fast, the above two methods may still be relatively time-consuming if we elect to use PN symbol sequences and the number of points in the constellation and/or

$\{A_k\} \rightarrow$	A_3	A_2	A_1	A_0	A_{-1}	A_{-2}	A_{-3}	$\rightarrow$
	$7S_3$	$7S_2$	$7S_1$	$7S_0$	$7S_{-1}$	$7S_{-2}$	$7S_{-3}$	
	$5S_3$			$5S_0$			$5S_{-3}$	
	$3S_3$			$3S_0$			$3S_{-3}$	
	S_3			S_0			S_{-3}	
	$-S_3$			$-S_0$			$-S_{-3}$	
	$-3S_3$			$-3S_0$			$-3S_{-3}$	
	$-5S_3$			$-5S_0$			$-5S_{-3}$	
	$-7S_3$			$-7S_0$			$-7S_{-3}$	

$\{B_k\} \rightarrow$	B_3	B_2	B_1	B_0	B_{-1}	B_{-2}	B_{-3}	$\rightarrow$
	$7Q_3$	$7Q_2$	$7Q_1$	$7Q_0$	$7Q_{-1}$	$7Q_{-2}$	$7Q_{-3}$	
	$5Q_3$			$5Q_0$			$5Q_{-3}$	
	$3Q_3$			$3Q_0$			$3Q_{-3}$	
	Q_3			Q_0			Q_{-3}	
	$-Q_3$			$-Q_0$			$-Q_{-3}$	
	$-3Q_3$			$-3Q_0$			$-3Q_{-3}$	
	$-5Q_3$			$-5Q_0$			$-5Q_{-3}$	
	$-7Q_3$			$-7Q_0$			$-7Q_{-3}$	

Figure 12.12. Illustration of FQA-2 method, based on table lookup.

the system's effective memory is relatively large. An alternative which is essentially independent of these parameters is based on the use of Gaussian quadrature⁽³⁻⁵⁾ (see also Chapter 7), a method which makes use of the moments of a random variable, as we will see shortly, and hence which we refer to as the moment method.

Gaussian quadrature (GQ) is a numerical integration technique that can be used to approximate an integral. Applied to the current context, the integral in which we are interested is an expectation

$$E[G(\xi)] = \int_a^b G(\xi)f(\xi) d\xi \approx \sum_{i=1}^N w_i G(\xi_i) \quad (12.1.39)$$

for $-\infty < a < b < \infty$; f is the probability density function of the random variable ξ , and G is a given function. In other words, the integral is approximated by a weighted sum of function values. The *weights* w_i and the *abscissas* ξ_i , $i = 1, 2, \dots, N$, are called a quadrature rule. The quadrature rule is a best approximation of the integral in a certain sense, and the approx-

imation improves as N becomes larger. A very useful property of quadrature rules is that they depend *only* on f , and for a variety of f the rules (w_i, ξ_i) have been tabulated for different N .^(3,4) Hence, if we know f , the problem is “solved.” In the problem at hand, ξ represents the ISI variable, and in fact we do not know *a priori* its distribution. However, the quadrature rule can be obtained from the knowledge of the first $2N$ moments of ξ , μ_k , $k = 0, \dots, 2N - 1$:

$$\mu_k = \int_a^b \xi^k f(\xi) d\xi$$

The procedure for obtaining the quadrature rule from the moments can be found in the cited references, and is not specific to this particular problem. What is specific to this problem is the computation of the moments, the procedure for which we will outline below.

First, we need to formulate the problem in integral form, as in (12.1.39). Without loss of generality, consider deciding on the symbol at t_0 and with $r_p(n)$ defined as $r_p(t_0 + nT)$, $r_{pp}(n)$ as $r_{pp}(t_0 + nT)$, and similarly for the Q rail, we write

$$R_p(0) = A_0 r_{pp}(0) + \xi + n_p$$

$$R_q(0) = B_0 r_{qq}(0) + \eta + n_q$$

where n_p and n_q are the noise processes in the I and Q rails, respectively, at the input to the decision device, and ξ and η represent the ISI in the I and Q rails, respectively:

$$\xi = \sum_{n \neq 0} A_n r_{pp}(n) - \sum_n B_n r_{pq}(n) \quad (12.1.40a)$$

$$\eta = \sum_{n \neq 0} B_n r_{qq}(n) + \sum_n A_n r_{pq}(n) \quad (12.1.40b)$$

With a the standard deviation of n_p (and n_q), one can show that the conditional symbol error probability in the I channel (conditioned on ξ) is given by

$$P_p(\xi) = \frac{1}{2} \left(1 - \frac{1}{L} \right) \left\{ \operatorname{erfc} \left[\frac{r_{pp}(0) + \xi}{\sqrt{2}\sigma} \right] + \operatorname{erfc} \left[\frac{r_{pp}(0) - \xi}{\sqrt{2}\sigma} \right] \right\} \quad (12.1.41)$$

and the expression for the conditional error probability in the Q channel, $P_q(\eta)$, is identical, but with η replacing ξ . With $M = L^2$, the M -ary symbol error probability, P_M is given by

$$\begin{aligned} P_M(\xi, \eta) &= 1 - [1 - P_p(\xi)][1 - P_q(\eta)] \\ &= P_p(\xi) + P_q(\eta) - P_p(\xi)P_q(\eta) \end{aligned}$$

which for even moderately small P_p and P_q is well approximated by

$$P_M(\xi, \eta) = P_p(\xi) + P_q(\eta)$$

Hence the unconditioned symbol error probability is given by

$$P_M = \int P_p(\xi) f(\xi) d\xi + \int P_q(\eta) f(\eta) d\eta \quad (12.1.42)$$

which is in the desired form for solution by Gaussian quadrature.

We now give just a brief indication of a recursive algorithm for obtaining the moments.⁽⁶⁾ Let us write

$$\xi = \sum_{i=1}^{2m+1} \gamma_i$$

with an obvious identification of terms, from (12.1.40a), Now define

$$\xi_k = \sum_{i=1}^k \gamma_i$$

so that $\xi_{2m+1} \equiv \xi$. The v th moment $\mu_{k,v}$ of the partial sum ξ_k can be written as

$$\mu_{k,v} = E[\xi_k^v] = E[(\xi_{k-1} + \gamma_k)^v]$$

which suggests a recursion. Applying the binomial expansion to the second expectation, one can arrive at the recursion

$$\mu_{k,2v} = \sum_{i=0}^{2v} \binom{2v}{2j} \mu_{k-1,2v-2j} \alpha_{k,2j}$$

with $\mu_{k,0} = 1$, $\mu_{k,2v+1} = 0$, and

$$\alpha_{k,l} = \frac{2r_k^l}{L} [1 + 3^l + \dots + (L-1)^l]$$

where $\alpha_{k,l} = 0$ for l odd, and $r_k \equiv r_{pp}(k)$. One can thus evaluate the necessary moments very efficiently. This completes our description of the moment method. For systems with a large constellation and/or large memory, it is computationally much more efficient than any other approach. It does, however, require the initial manpower expenditure to set up the procedure for calculating the moments and the abscissas and weights of the quadrature rule.

12.1.4.5. Simulation Procedure

An array of 30,000 sets of channel parameters drawn from the channel parameter distributions was obtained as part of the initialization phase and set up as an input file. During this same phase, the impulse response of the unfaded system was also computed and stored. This response, of course, is simply the intended raised-cosine pulse.

The simulation procedure consisted in 30,000 repetitions of the following steps. First, a channel parameter triplet was taken from the channel parameter array. The faded impulse response was calculated next, and passed to the covariance matrix inversion routine to obtain the equalizer tap settings. An input to this last computation is the unfaded signal-to-noise ratio. The equalized impulse response is obtained next and used in one of the three error probability evaluation methods described. All three were tried in order to obtain a true assessment of their relative efficiencies.

12.1.4.6. Evaluation of the Outage Probability

The outage probability P_o during fading is defined as the probability that a signal will experience an error probability P_M equal to or higher than a given threshold P_t :

$$P_o = \text{Prob}(P_M \geq P_t | \text{fading})$$

In order to evaluate P_o , we first construct an estimate of the probability density function of the error rate by constructing a histogram of the error rates computed for all the generated channels. The outage probability is easily determined from the cumulative distribution, which is simply obtained from the histogram.

The reliability of the estimate depends on the number of channels simulated. The situation is conceptually similar to the estimation of BER as discussed in Chapter 11. For a given unfaded SNR a chosen channel will have a P_M that is either greater than or less than P_t . Thus, the (infinite) set of possible channels can be placed in one of two categories that either does or does not satisfy the threshold. The drawn set of channels is thus an estimate of the proportion of channels that does (or does not) satisfy that threshold. Thus, the reliability of the estimate of P_o is inversely proportional to the number of channels simulated, just as the reliability of the estimate of BER is inversely proportional to the number of bits observed.

In order to estimate the outage probability for a given period of time T , we have to know the probability $p_f(T)$ of the channel being in a faded state during that period. The outage probability of the digital radio during a period T is then

$$P_o(T) = P_o p_f(T)$$

$p_f(T)$ is generally a function of geographic location, operating frequency, and the time of year.⁽⁷⁾ A typical value for the probability of fading per year ($T = Y$) is about 3×10^{-3} .

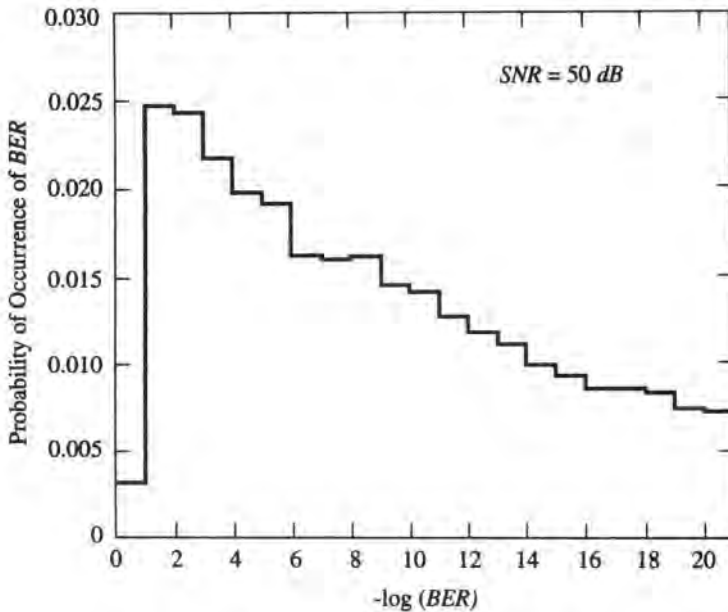


Figure 12.13. Histogram of BER of 64-QAM digital radio.

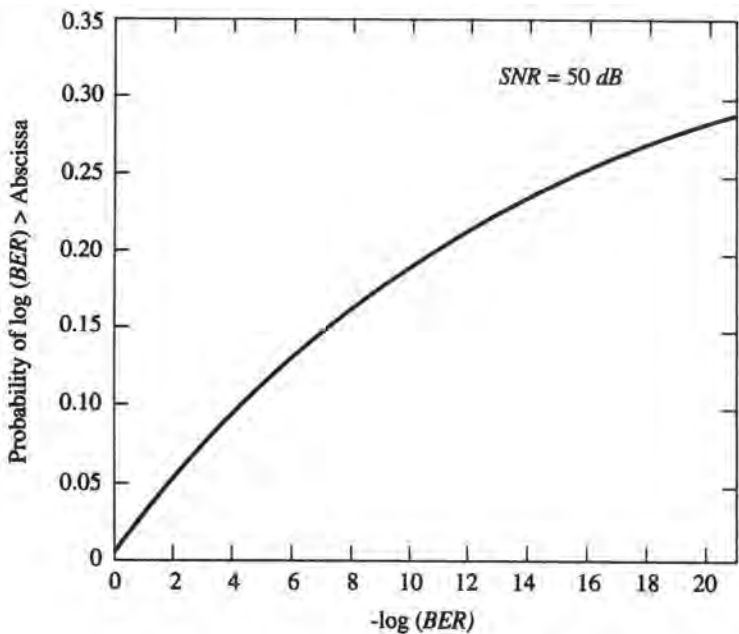


Figure 12.14. Probability of outage of 64-QAM digital radio.

12.1.4.7. Simulation Results

Figure 12.13 shows a histogram for the error rate for a signal-to-noise ratio of 50 dB, and Figure 12.14 shows the outage probability P_o for the same system.

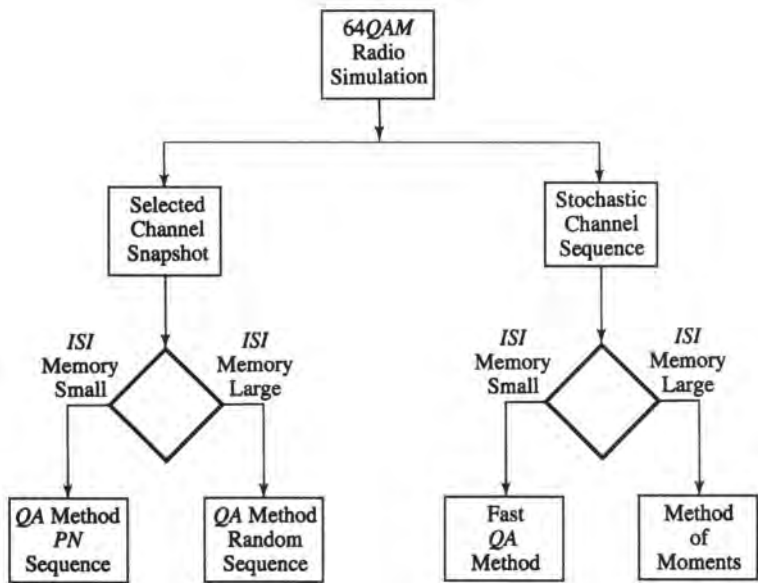


Figure 12.15. Flowchart for selecting a simulation method and a BER estimation method.

12.1.5. Conclusions

In this case study we had two main objectives: the determination of outage probability and the study of system behavior under particular fading events. Although, in principle, both objectives could be reached by the same simulation process, efficiency considerations led us to devise two distinct methodologies, each tailored for the aspect treated. Furthermore, we found that within each methodology there are further branching points in terms of possible ways to proceed. For example, in the selected channel snapshot method the use of a PN sequence may be inadvisable because of excessive period, which in turn depends on the size of the signaling alphabet and the memory of the system. In the stochastic channel sequence methodology we saw that there are different computational options, the preferable one again depending on the same parameter values. The various simulation methodologies are encapsulated in the flow diagram of Figure 12.15. Apart from specific techniques, an important general lesson imparted by this case study is that any new problem faced by the simulator should be examined with a fresh eye for efficient means of solution, and that “standard” or “canned” approaches should not be automatically accepted, especially if they lead to large computational burdens.

12.2. Case Study II: Phase Noise and Its Effect on Block-Coded Systems

12.2.1. Introduction

The problem to be addressed is as follows: Given a digital satellite communication system as illustrated in Figure 12.16, what should be the interleaver depth to approach the coded performance of an independent error channel? In this system, the main factor which produces correlation among errors is phase noise, generated both by thermal noise and oscillator frequency instability. The deleterious effect of phase noise was initially studied by Lindsey⁽⁸⁾ for uncoded systems, and subsequently by many others for both coded and uncoded systems.^(9–13) It is known that correlated errors on the channel degrade the performance of both convolutional or trellis-coded signals^(9,10) and block-coded signals.^(11,12) Hence an appropriate interleaver is beneficial, but using a larger than necessary interleaver incurs needless satellite weight and power, expense, and decoding delay.

Let us assume that we have adequate models for all of the boxes. Then, from the simulation standpoint, the main issue is how to deal with the extremely long run times necessary to obtain a reliable estimate of BER using Monte Carlo simulation if the target BER is very small. Indeed, if the BER objective is moderate, say $\geq 10^{-5}$, Monte Carlo simulation is feasible. The methodological challenge lies in the case where the BER of interest is very small, say on the order of 10^{-9} . Such low error rates typically require long codewords if the code rate is to remain reasonably high. There are different ways to deal with the problem at hand. The approach developed here is one possibility.

Thus, we can state the objective of this case study as follows: Develop a simulation-based tool[†] to evaluate the steady-state BER performance of the block-coded system in Figure 12.16 for very low error rates, as a function of interleaver depth. Toward this end, we will apply several ideas discussed in earlier parts of the book, as outlined in the following list.

[†]A different approach to this problem was taken in Ref. 14. This approach was purely analytical in nature, though it was labeled “computer-aided” because numerical results required evaluation on a computer.

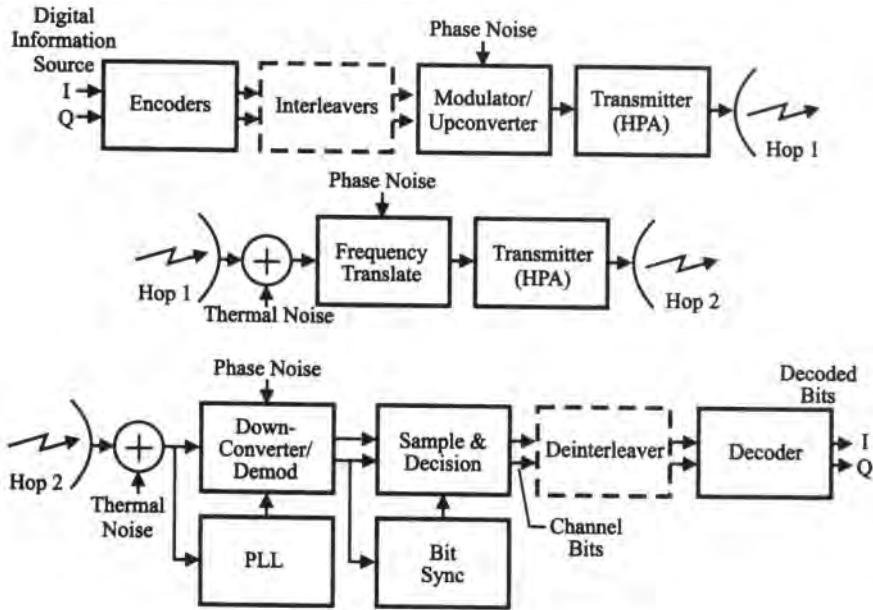


Figure 12.16. Block diagram of system under study for Case Study II.

- *Methodology:* The major methodological idea to be applied is that of complexity reduction.
- *Modeling Construct:* As a corollary to complexity reduction, we seek computational simplification in otherwise complex models. We achieve this by using an equivalent random process (erp) model (Section 8.12.3) for the phase noise.
- *Multirate Techniques:* As discussed in Chapter 3, such techniques can be helpful when processes of different bandwidths coexist. We apply this idea here to the phase noise process, which is much slower than the information process.
- *Performance Evaluation:* The complexity reduction approach taken permits the detection process to be handled in part by analytical techniques. The performance evaluation approach can be regarded as a type of *mixed* quasianalytical technique (MQA).
- *Postprocessing:* The combined application of the previous techniques permits a good deal of the work to be done on a postprocessing basis.

We now elaborate a bit on the specific ways in which these ideas are applied. Some of these will be further expanded in the subsequent sections.

As mentioned, the main methodological principle that we want to apply is complexity reduction. That is, we want to simplify the system as much as possible, while retaining the essential nature of the problem as much as possible. There are, of course, judgments applied here that lead to a choice of some set of simulation techniques. But the validity of these techniques is not directly connected to the soundness of those judgments. As we discussed in Chapter 2, some independent means must eventually be used to validate the simulation results.

[†]We are assuming here a block code, with hard-decision decoding, so that a discrete channel model which produces an error sequence is appropriate.

Because of the very low BER of interest, only modest complexity decrease would be insufficient to make the run time practical. It would be necessary to reduce the problem to its most basic form. One possible approach has been mentioned earlier in conjunction with coding, and this is to use a discrete channel model. In this case, the entire system between the encoder and the decoder (the codec) is represented by an error generator.[†] Of course, it is first necessary to synthesize such a channel model, which is very time-consuming in itself. It may be a reasonable tactic when the channel BER of interest is not too low and when the operating conditions of interest are relatively fixed. One of the main goals of Case Study IV is, in fact, to expose the steps leading to a discrete channel model.

In this particular problem, we chose a different way primarily because it was desired to investigate a range of channel conditions, for each of which one would have to develop a discrete model if that approach were taken. The reduced complexity model is shown in Figure 12.17. The fundamental assumption made is that the system is distortionless. Only additive noise and phase noise degrade the signal. While this assumption may seem to be an oversimplification, it is justified by the following reasoning. The relative effect of different depths of interleaving depends primarily on the correlation properties of the detected waveform. The correlatedness of the waveform depends on the memory of different mechanisms that exist. For systems of this type, memory induced by distortion, e.g., filtering, is typically on the order of a small number of symbols, perhaps 10 or less. Since we will be dealing with block lengths at least on the order of 10^3 , these effects will tend to average out. The more persistent correlatedness, having time constants on the order of codeword duration, will be due to phase noise.

The simplifying assumption leads to a great reduction in computation for several reasons. One, of course, is that detailed models of subsystems do not have to be simulated, since they are idealized. But probably the more significant effect is to allow the simulation to proceed at the symbol rate, rather than at the simulation sampling rate. For example, if we would have used 16 samples/symbol, this amounts immediately to a 16 to 1 reduction in run time. A concomitant of the simplifying assumption is that the bit streams, assumed rectangular at the transmitter, are received undistorted. The matched filter for such waveforms is an integrate-and-dump (I&D) filter. In conjunction with the assumed properties of the phase noise, this will allow a quasianalytic formulation for the detected waveform.

Concerning the phase noise, we will develop a model (erp model) that accounts for its effect without explicitly simulating all the sources or equipment leading up to it. This, too, is a great time saver, but it is not directly related to the system simplification described. The key assumption leading to the erp model is that we are interested in the steady-state BER

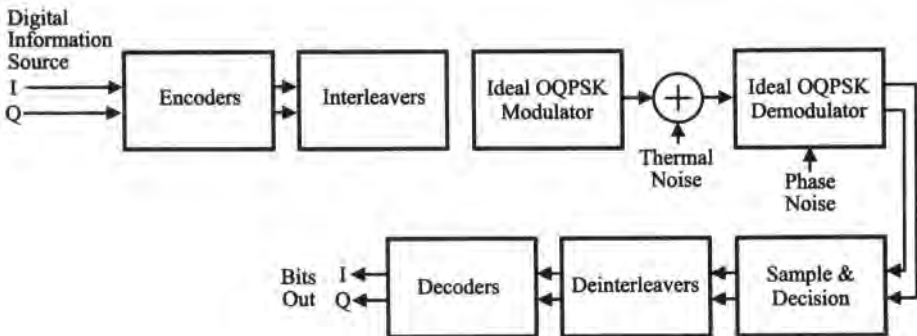


Figure 12.17. Simplified system block diagram for purposes of Case Study II.

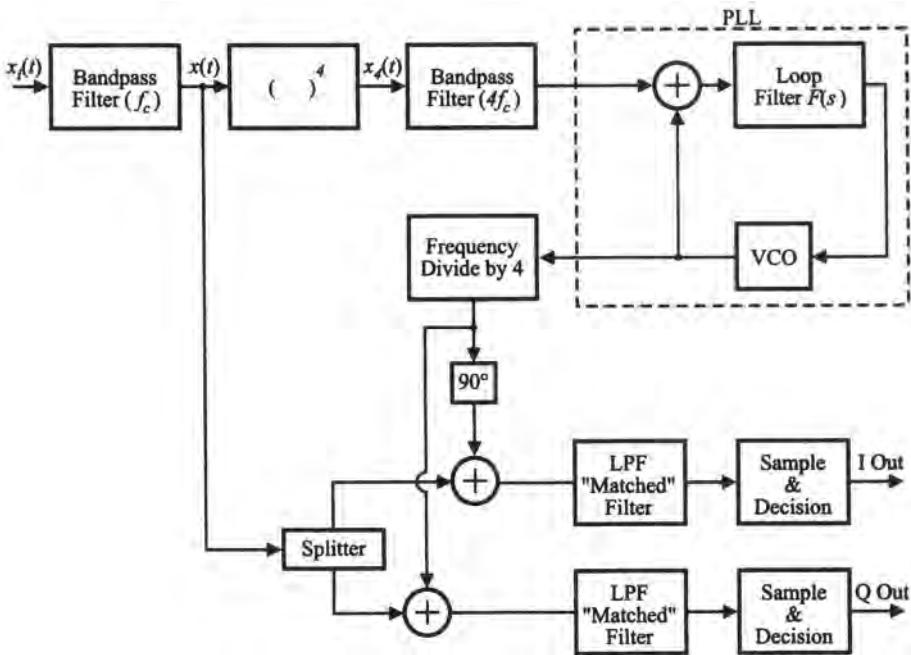


Figure 12.18. Expanded block diagram of the receiver.

performance. This means the receiver tracking loops can be assumed to be in their acquired state. Even here, as we will see, further simplifying assumptions (which can be reasonably justified) will have to be made on the way to the final model.

12.2.2. Analytical Formulation: Demodulated Signal

We first develop the set of equations that lead to the form of the demodulated signal. Part of the idealizing assumption is that the HPAs are linear (or operated in the linear region, or have ideal predistorters). Thus, uplink and downlink additive noise sources can be combined into one equivalent receiver input noise source $n_i(t)$, and oscillator noise from all segments of the system can also be combined into one oscillator noise source $\delta(t)$. Referring to Figure 12.18, we can write the input to the receiver as

$$x_i(t) = A_c I(t) \cos[\omega_c t + \delta(t) + \theta_c] + A_c Q(t) \sin[\omega_c t + \delta(t) + \theta_c] + n_i(t) \quad (12.2.1)$$

where $I(t)$ and $Q(t)$ are independent random binary waveforms with bit durations T_b , offset by $0.5T_b$. The bandpass filter is conventionally assumed to bandlimit the noise, but have no effect on the signal. The trivial extension[†] of (12.2.1) is then

$$x(t) = A_c I(t) \cos[\omega_c t + \delta(t) + \theta_c] + A_c Q(t) \sin[\omega_c t + \delta(t) + \theta_c] + n(t) \quad (12.2.2)$$

[†]We make the obviously trivial distinction between (12.2.1) and (12.2.2) only to reiterate the point that assumptions often made for tractability are in fact approximations.

where now $n(t)$ is bandlimited “white” Gaussian noise, which we can represent in the usual in-phase and quadrature terms

$$n(t) = n_c(t) \cos \omega_c t + n_s(t) \sin \omega_c t$$

Different types of tracking structures could be used. We will assume here a “times-four” or “quadrupling” loop (see Section 8.12). We defer a discussion of the loop operation till the next section. The carrier reference provided to the I channel of the demodulator can be written in the form[†]

$$C_1(t) = k_d \cos \left\{ \frac{1}{4} \left[\omega_4 t + \hat{\theta}_4(t) \right] \right\} \quad (12.2.3)$$

where $\omega_4 = 4\omega_c$, $\hat{\theta}_4(t)$ is an estimate of $\theta_4(t) = 4[\delta(t) + \theta_c]$, and k_d is a demodulator constant, which we will take equal to 2, without loss of generality. Applying (12.2.3) to (12.2.2), we obtain the demodulated I-channel waveform:

$$D_1(t) = \{x(t) \times C_1(t)\}_{\text{lowpass}} = A_c I(t) \cos[\epsilon(t)] + A_c Q(t) \sin[\epsilon(t)] + n_1(t) \quad (12.2.4)$$

where

$$n_1(t) = n_c(t) \cos \left[\frac{1}{4} \hat{\theta}_4(t) \right] - n_s(t) \sin \left[\frac{1}{4} \hat{\theta}_4(t) \right] \quad (12.2.5)$$

and

$$\epsilon(t) = \frac{1}{4} \left[\theta_4(t) - \hat{\theta}_4(t) \right] \quad (12.2.6)$$

which is recognized as one fourth the phase error in the loop. We will call $\epsilon(t)$ the *residual* phase noise.

The waveform $D_1(t)$ is processed through the I&D filter and detected. That sequence of decisions forms the output to be post processed. We shall formulate this part of the problem later. We note first that the demodulated noise $n_1(t)$ is not Gaussian, but it has been argued⁽¹⁾ that when $\hat{\theta}_4(t)$ is a slow enough process, then $n_1(t)$ can be reasonably approximated as a Gaussian process. We shall make that assumption here, actually the first of three “Gaussianizing” assumptions that we shall need. If $n_1(t)$ is Gaussian, and we take $I(t)$ and $Q(t)$ to be random binary waveforms, the problem is then well formulated if we can describe the nature of $\epsilon(t)$. We consider this in the next two sections.

[†]We will consider only the I channel since identical considerations apply to the Q channel.

12.2.3. Quadrupling Loop Operation

The input to the phase-locked loop is the filtered version of the fourth-power device. Defining $\Phi = \omega_c t + \delta(t) + \theta_c$, we obtain the relevant output of the fourth-power device $x_4(t)$ as

$$\begin{aligned} x_4(t) = & A_c^4 \sin^2 2\Phi + 4A_c^3 n(t)I(t)Q(t)(\sin 2\Phi)(\cos \Phi + \sin \Phi) \\ & + 2A_c^2 I(t)Q(t)n^2(t)(\sin 2\Phi) + 4A_c n^3(t)(\cos \Phi + \sin \Phi) \\ & + 4A_c^2 n^2(t)(\cos \Phi + \sin \Phi) + n^4(t) \end{aligned} \quad (12.2.7)$$

This expression contains only terms which include components around $4f_c$; other terms in $x^A(t)$ for which this is not the case have already been discarded. Also implicit in this equation is that $I(t)$ and $Q(t)$ are perfectly rectangular waveforms. The first term is clearly the desired one, and the others are unwanted. Thus, we can take the input to the loop (output of the bandpass filter) as

$$y(t) = \sqrt{2}A_4 \cos(\omega_4 t + \theta_4) + n_4(t) \quad (12.2.8)$$

where $n_4(t)$ is the filtered version of the unwanted terms. Here, the second Gaussiannizing assumption is made, namely, that if the bandpass filter preceding the loop is sufficiently narrowband, the filtered version of these unwanted terms is approximately Gaussian; thus, we will take $n_4(t)$ to be Gaussian.

The VCO output is of the form

$$v(t) = \sqrt{2}K_1 \sin(\omega_4 t + \hat{\theta}_4) \quad (12.2.9)$$

so that the lowpass portion of the mixer output is given by

$$e(t) = A_4 K_1 K_m \sin[\theta_4(t) - \hat{\theta}_4(t)] + K_1 K_m N(t) \quad (12.2.10)$$

where

$$N(t) = \left\{ \sqrt{2}n_4(t) \sin \omega_4 t + \hat{\theta}_4 \right\}_{\text{lowpass}} \quad (12.2.11)$$

Here is the third Gaussiannizing assumption: $N(t)$ is approximately Gaussian under the same conditions that were previously applied to $n_1(t)$. The lowpass-equivalent representation of the loop is shown in Figure 12.19. We assume here that the VCO is tuned to $4f_c$ and the term $\delta_r(t)$ represents its internal phase noise. Because we are interested in the steady-state performance, we will use the linearized version of the loop. Hence we have (in operational form)

$$\begin{aligned} \phi_4(t) &= \theta_4(t) - \hat{\theta}_4(t) \\ &= \theta_4(t)[1 - H(s)] - H(s) \frac{N(t)}{A_4} - \delta_r(t)[1 - H(s)] \end{aligned} \quad (12.2.12)$$

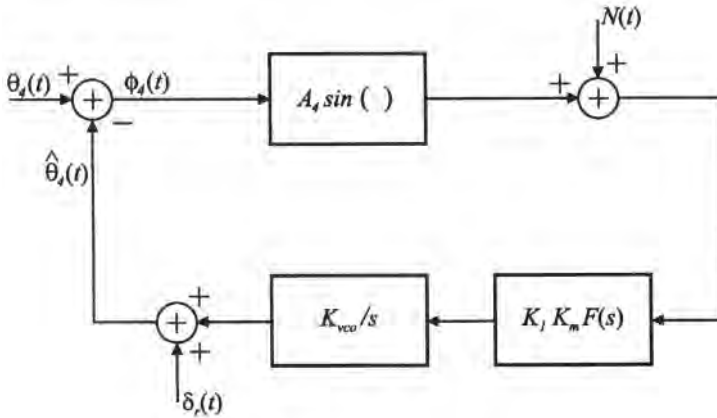


Figure 12.19. Equivalent block diagram of the second-order phase-locked loop at “phase level.” $N(t)$ is the equivalent additive noise source and $\delta_r(t)$ is the reference (local) oscillator’s own phase noise.

where

$$H(s) = \frac{A_4 K F(s)}{s + A_4 K F(s)} \quad (12.2.13)$$

is the closed-loop transfer function and $K = K_1 K_m K_{vco}$. Since the residual phase noise is one fourth of $\phi_4(t)$, its properties are embedded in (12.2.12). We discuss this next.

12.2.4. Residual Phase Noise Model

12.2.4.1. Introduction

Equation (12.2.12) tells us that the residual phase noise is a filtered version of three processes: $\theta_4(t)$, $N(t)$, and $\delta_r(t)$. We have already argued that $N(t)$ can be approximated as Gaussian, and we now argue that the same can be said for the filtered versions of $\theta_4(t)$ and $\delta_r(t)$. The argument is similar to before, namely, that sufficiently narrowband filtering Gaussiannizes a process. There is perhaps little other choice here because the nature of the oscillator jitter random process is incompletely understood. Rather, it is typically described by second-order measures.^(15–18) In order to synthesize a random number generator, it is of course necessary to have a random process description. On the other hand, the Gaussian assumption for $\epsilon(t)$ would lead to a practical random number generator. Thus, partly by reasoning and partly by exigency we will make that assumption. In this case, all we need, in addition, is the power spectral density of $\epsilon(t)$. From (12.2.12) it is given by

$$S_\epsilon(f) = \left[S_\delta(f) + \frac{1}{16} S_\delta^{(r)}(f) \right] |1 - H(j2\pi f)|^2 + \frac{1}{16A_4^2} S_N(f) |H(j2\pi f)|^2 \quad (12.2.14)$$

where $S_\delta(f)$, $S_\delta^{(r)}(f)$, and $S_N(f)$ are the PSDs of $\delta(t)$, $\delta_r(t)$, and $N(t)$, respectively, and for a second-order loop with (see Section 8.12.8)

$$F(s) = \frac{\tau_2 s + 1}{\tau_1 s + \alpha}$$

we have

$$|1 - H(j2\pi f)|^2 = \frac{\lambda^2 [\lambda^2 (\alpha \omega_n / AK)^2]}{\lambda^4 + 2(2\zeta^2 - 1)\lambda + 1} \quad (12.2.15)$$

and

$$|H(j2\pi f)|^2 = \frac{1 + (\lambda \tau_2 \omega_n)}{1 + \lambda^4} \quad (12.1.16)$$

where

$$\alpha = 1 \text{ (passive) or } 0 \text{ (active)}$$

$$\lambda = f/f_n$$

$$\omega_n^2 = AK/\tau_1$$

$$\zeta = \frac{1}{2} \tau_2 \omega_n (1 + \alpha/\tau_2 AK)$$

$\omega_n = 2\pi f_n$ is the natural frequency of the loop, and ζ is the damping factor. The first term on the right-hand side of (12.2.14) is often referred to as *untracked* oscillator noise, and the second term as *tracked* thermal noise.

In order to obtain a specific form for $S_e(f)$, we need one for $S_\delta(f)$, $S_\delta^{(r)}(f)$, and $S_N(f)$. The latter, by previous assumption, is a constant, assuming the input noise is white. Oscillator phase noise spectra can assume a variety of shapes, depending on the technology, as illustrated in Figure 12.20.⁽¹⁹⁾ However, as suggested from this figure and the literature cited, a fairly general model for (the one-sided) phase noise spectrum can be taken to be of the form

$$S_\delta(f) = \sum_{k=0}^4 S_k(f) \quad (12.2.17)$$

where

$$S_k(f) = \sum_{i=1}^{r_k} a_i^{(k)} f^{-k} I_{J_i^{(k)}}(f) \quad (12.2.18)$$

and $J_i^{(k)} = [\alpha_i^{(k)}, \beta_i^{(k)}]$ is a subinterval on which $a_i^{(k)}$ is not zero, so that each term $S_k(f)$ is **nonzero over** $\cup_{i=1}^{r_k} J_i^{(k)}$. An example of a measured spectrum is shown in Figure 12.21, where the particular frequencies f_1 , f_2 , and f_n are design-specific. For this spectrum, $S_\delta(f)$ has two terms, one for $k=0$ and one for $k=2$. For $k=0$, $S_k(f)$ has a single subinterval, and for $k=2$, the corresponding $S_k(f)$ has two subintervals. The coefficients in these terms are constrained by the requirement of continuity at the interval boundaries. As an illustration, assuming $\alpha=0$ and $\zeta=\sqrt{2}/2$, (12.2.15) applied to this phase noise spectrum yields the untracked phase noise spectrum shown in Figure 12.22, and (12.1.16) applied to $S_N(f) = \text{const}$ produces the tracked thermal noise spectrum in Figure 12.23.

We note that the oscillator phase noise spectrum is traditionally problematic as f nears zero. Any term in (12.2.17), other than the one for $k=0$, leads to an ever-increasing PSD if it applies as $f \rightarrow 0$. Measurements taken very close to zero seem to indicate such an increase,

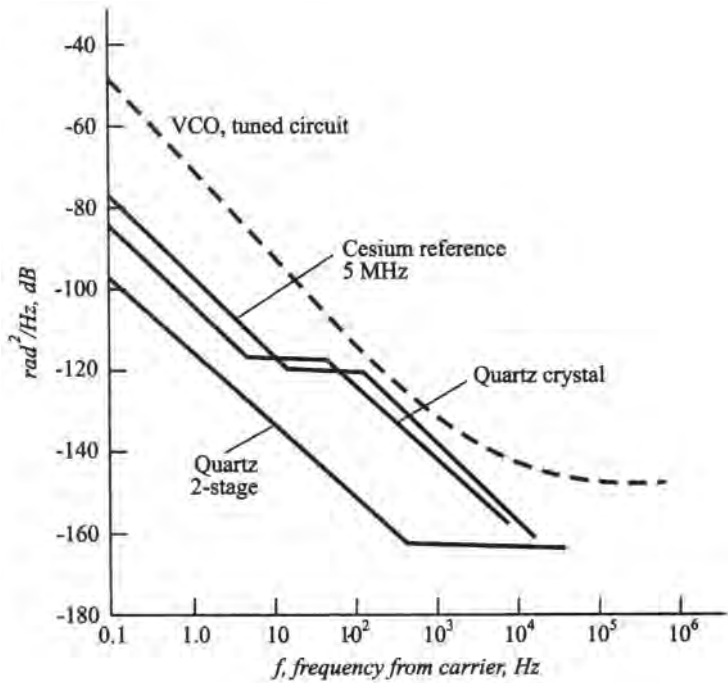


Figure 12.20. Power spectral densities for the phase noise of various oscillator sources (from R. M. Gagliardi *Introduction to Communications Engineering*, 2nd Ed., © 1988, John Wiley & Sons, Inc., reproduced by permission).

but clearly the PSD cannot become infinite. This behavior near $f = 0$ poses difficulties in the modeling and direct generation of the phase noise process. Some of the related considerations and ways of dealing with them are discussed in Ref. 20, under the assumption that the process is Gaussian. Reference 21 also considers the generation of random vectors drawn from a non-Gaussian process with spectrum of the form $f^{-\alpha}$. The generation procedure used in this study bypasses to a large extent the dilemma alluded to because we do not attempt to generate the phase noise directly, but rather, the untracked phase noise. Since the actual phase noise PSD

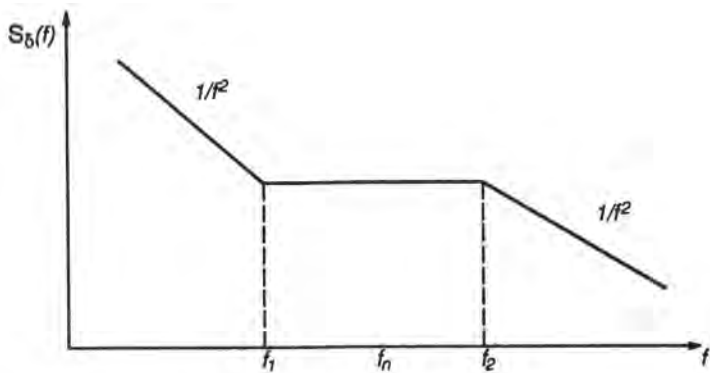


Figure 12.21. Phase power spectral density assumed for the case study.

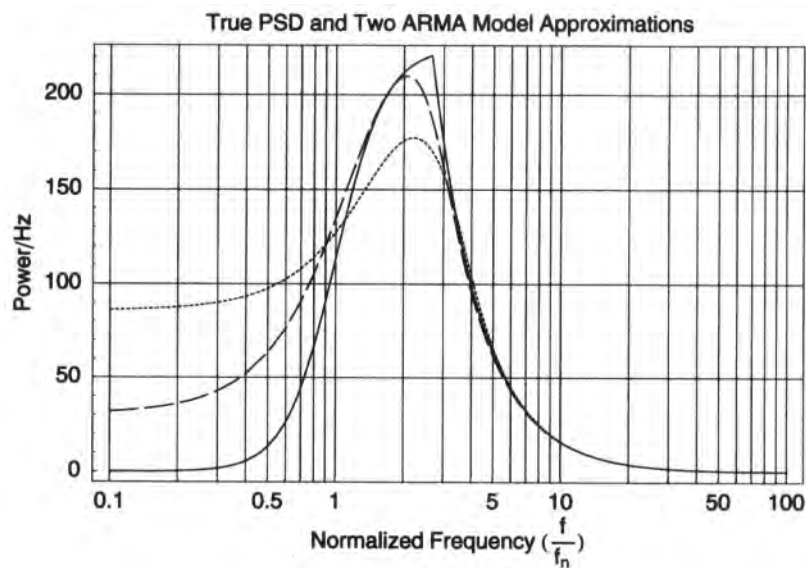


Figure 12.22. The untracked phase noise spectrum and two ARMA approximations. Actual untracked phase noise PSD is the solid curve; the ARMA(20, 20) model is the dotted curve; the ARMA(40,40) model is the dashed curve.

cannot be infinite at $f = 0$, while the closed-loop transfer function is zero at $f = 0$, the untracked phase noise PSD must be zero at $f = 0$.

12.2.4.2. *Calibrating the Model*

The ordinate scales in Figures 12.22 and 12.23 are arbitrary. In any particular instance, we have to calibrate the model so that the untracked phase noise variance is equal to some

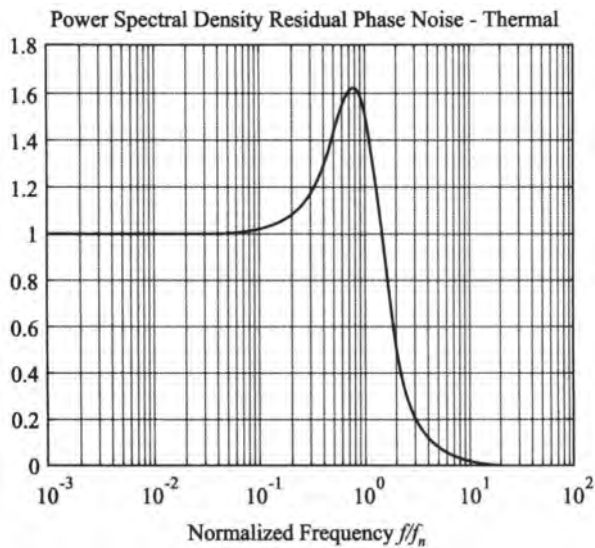


Figure 12.23. Power spectral density of tracked thermal noise.

specified value or to a measured value, and similarly for the tracked thermal phase noise. Thus, for the general phase noise spectrum of (12.2.17)–(12.2.18) and a second-order loop, and defining $\lambda = f/f_n$, the untracked phase noise variance is given by

$$\sigma_{\varepsilon 1}^2 = \sum_{k=0}^4 \sum_{i=1}^{r_k} a_i^{(k)} f_n^{(1-k)} \int_{\mathcal{J}_i^{(k)}} I_{\mathcal{J}_i^{(k)}}(\lambda) \frac{1}{\lambda^k} \frac{\lambda^2(\lambda^2 + (\alpha\omega_n/AK)^2)}{\lambda^4 + 2(2\zeta^2 - 1) + 1} d\lambda \quad (12.2.19)$$

where $\mathcal{J}_i^{(k)} = J_i^{(k)}/f_n$. In the particular case that $\alpha = 0$ and $\zeta = \sqrt{2}/2$, (12.2.18) simplifies to

$$\sigma_{\varepsilon 1}^2 = \sum_{k=0}^4 \sum_{i=1}^{r_k} a_i^{(k)} f_n^{(1-k)} \int_{\mathcal{J}_i^{(k)}} I_{\mathcal{J}_i^{(k)}}(\lambda) \frac{1}{\lambda^k} \frac{\lambda^4}{\lambda^4 + 1} d\lambda \quad (12.2.20)$$

In general, these last two integrals have to be evaluated numerically.

An analysis of the nature of $\mathcal{S}_N(f)$ (see, e.g., Ref. 19) shows that the PSD of the tracked thermal noise can be written as

$$|H(j2\pi f)|^2 N_0 \frac{\gamma_4}{2A_c^2}$$

where N_0 is the original one-sided noise spectral density and the term $\gamma_4 \geq 1$ is the so-called quadrupling loss factor, which takes account of the degradation due to the fourth-power nonlinearity. Hence, the portion of phase noise variance due to the presence of thermal noise is

$$\begin{aligned} \sigma_{\varepsilon 2}^2 &= \int_{-\infty}^{\infty} N_0 \frac{\gamma_4}{2A_c^2} |H(j2\pi f)|^2 df \\ &= \frac{N_0 B_L \gamma_4}{A_c^2} \end{aligned} \quad (12.2.21)$$

where B_L is the one-sided loop bandwidth

$$B_L = \int_0^{\infty} |H(j2\pi f)|^2 df$$

The total residual phase noise variance is then

$$\sigma_{\varepsilon}^2 = \sigma_{\varepsilon 1}^2 + \sigma_{\varepsilon 2}^2$$

In general, there may of course be several untracked oscillator noise terms.

12.2.5. Residual Phase Noise Random Number Generator

In our methodology there will be a need to produce a sequence of values of $\varepsilon(t)$. We thus need to synthesize a random number generator (RNG) that possesses its properties, namely, a Gaussian process with the spectral density given by (12.2.14). Although it is possible to construct a single RNG based on $\mathcal{S}_{\varepsilon}(f)$, it is probably more practical to sum the outputs of more than one RNG, each one generating the components of $\varepsilon(t)$, which is proper since they

are independent. In this fashion, the simulator can deal more easily with changes in the characteristics or parameters of the individual contributors.

We choose to synthesize an RNG using an ARMA model, or equivalently by passing white noise through an IIR filter. For convenience, we will change notation and call the generator output sequence $\{y(n)\}$. The ARMA model is of the form

$$y(n) = \sum_{k=1}^p a(k) y(n-k) + \sum_{k=0}^q b(k) x(n-k) \quad (12.2.22)$$

where $a(k)$ and $b(k)$ are coefficients to be determined, and the $x(n)$ form an i.i.d., zero-mean, white Gaussian sequence. The PSD of the sequence $y(n)$ is given by

$$S_y(f) = T_s' \rho_0 |B(f)|^2 / |A(f)|^2, \quad |f| \leq 1/2T_s' \quad (12.2.23)$$

where $\rho_0 = E[x^2(n)]$ and

$$A(f) = 1 + \sum_{k=1}^p a(k) \exp(-j2\pi k f T_s') \quad (12.2.24a)$$

and

$$B(f) = \sum_{k=0}^q b(k) \exp(-j2\pi k f T_s') \quad (12.2.24b)$$

It should be pointed out that in general $S_y(f)$ will be an approximation to the actual PSD, but one that can be made as accurate as desired by increasing p and q . The discrete time interval T_s' should be appropriate for $\mathbf{e}(t)$. As was mentioned before, the bandwidth of $\mathbf{e}(t)$ is much smaller than the information rate, hence it should be sampled less often. We will take advantage of this property in the next section.

The model is specified by finding a set of coefficients $\{a(k), b(k)\}$ such that that $S_y(f)$ is a reasonable facsimile of the component of $S_e(f)$ at hand. A set of $p + q + 1$ equations needs to be solved in order to determine these coefficients. The Yule–Walker method (see, e.g., Ref. 22) is an efficient procedure for doing so, and is implemented in commercially available packages such as MATLAB. Evidently, we want to keep p, q as small as possible, consistent with a reasonable fit (which to some extent is a subjective matter). As a specific example, the spectrum of Figure 12.22 was first fit to an ARMA model with $p = q = 20$. We call this an ARMA(p, p) model, with $p = 20$ in this case; the model spectrum (as a function of normalized frequency) is shown as the dashed curve in Figure 12.22. It can be seen that the agreement with the original PSD is good for λ about greater than three, but leaves something to be desired for lower frequencies. Next, an ARMA(p, p) model with $p = 40$ was tried, the resulting model spectrum being shown in Figure 12.22 as the dotted curve, which is seen to be quite a better fit. The model coefficients were scaled so that in both cases the total power in either model is the same as that for the true PSD.

The use of the $p = 20$ or $p = 40$ model presents the typical tradeoff between accuracy and run time. In order to get a better sense of this option (of course, we are not limited to these two numbers), let us first define $S_{20}(\lambda)$ and $S_{40}(\lambda)$ as the model spectra for $p = 20$ and $p = 40$,

respectively. Let us also define $P_{20}(m)$ and $P_{40}(m)$ as the power in successive 1-Hz intervals under the corresponding PSDs:

$$P_k(m) = \int_m^{m+1} S_k(\lambda) d\lambda \quad (12.2.25)$$

where k is either 20 or 40. Correspondingly, we define $P_\epsilon(m)$ as the power in successive 1-Hz intervals under $S_\epsilon(\lambda)$,

$$P_\epsilon(m) = \int_m^{m+1} S_\epsilon(\lambda) d\lambda \quad (12.2.26)$$

A sensible measure of the model fidelity is the normalized error $\Delta_p(m)$ defined as

$$\Delta_p(m) = \frac{|P_k(m) - P_\epsilon(m)|}{P} \quad (12.2.27)$$

where P is the total power under any of the spectra, which, it is recalled, has been forced to be the same. This measure gives a good sense of the possible impact of model error on the end result. Figure 12.24 shows $\Delta_p(m)$ for both models, up to m equal to 10, after which the error is negligible. Evidently, the $p = 40$ model is superior, but based on Figure 12.24, the improved model fidelity is probably not worth the increased run time.

The preceding discussion has dealt with the untracked phase noise for one oscillator. As mentioned above, it will usually be convenient to handle separately the various contributors to $\epsilon(t)$, so a model development and analysis similar to that just described must be performed for each contributor to the residual phase noise.

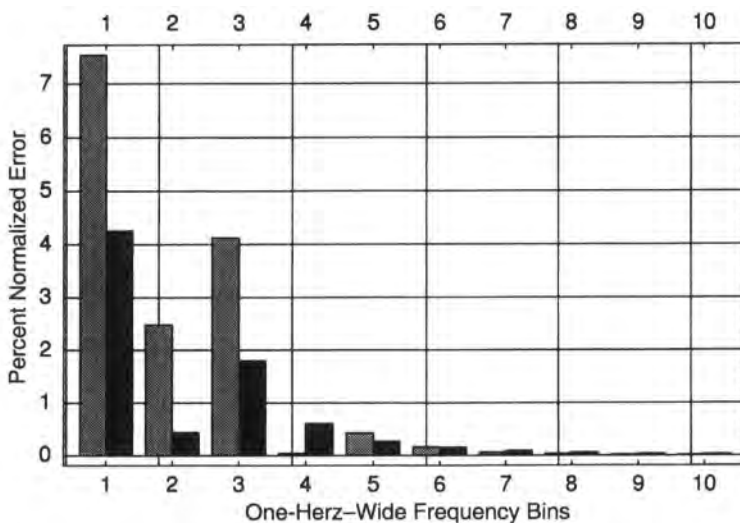


Figure 12.24. Normalized error for the two ARMA models as a function of frequency measured in 1-Hz intervals. Gray, ARMA(20, 20); black, ARMA(40, 40).

12.2.6. A Mixed Quasianalytic Generator for the Error Sequence

We now outline a mixed QA (MQA) approach for generating an error sequence, in which the generation of random processes and analytics are combined. The sampled value of the I&D detector at time nT_b in the I channel is

$$Z_n = \int_{(n-1)T_b}^{nT_b} D_I(t) dt$$

and substituting from (12.2.4), using the fact that I and Q channels are offset by half a bit, results in

$$\begin{aligned} \frac{1}{A_c} Z_n = & I(n) \int_{(n-1)T_b}^{nT_b} \cos[\epsilon(t)] dt + Q(n-1) \int_{(n-1)T_b}^{nT_b - T_b/2} \sin[\epsilon(t)] dt \\ & + Q(n+1) \int_{nT_b - T_b/2}^{nT_b} \sin[\epsilon(t)] dt + \int_{(n-1)T_b}^{nT_b} (1/A_c) n_1(t) dt \end{aligned} \quad (12.2.28)$$

Because two Q-channel bits can straddle an I-channel bit, there are four possible crosstalk combinations, as shown in Figure 12.25. As we said earlier, we assume in this problem that $\epsilon(t)$ varies slowly with respect to T_b^{-1} . Figure 12.26 shows a slice of $\epsilon(t)$ to indicate the time scale. We define an interval of time KT_b over which we can well approximate $\epsilon(t)$ by a straight line. This is not necessarily the largest sampling interval compatible with the bandwidth of $\epsilon(t)$, but we choose it deliberately in this fashion because the integration in (12.2.28) can then be completed in terms of the values of $\epsilon(t)$ at the endpoints of the intervals so defined. Thus, our choice of K allows us to write

$$\epsilon(t) = \epsilon_{r-1} + \frac{\epsilon_r - \epsilon_{r-1}}{KT_b} [t - (r-1)KT_b], \quad (r-1)KT_b \leq t \leq rKT_b \quad (12.2.29)$$

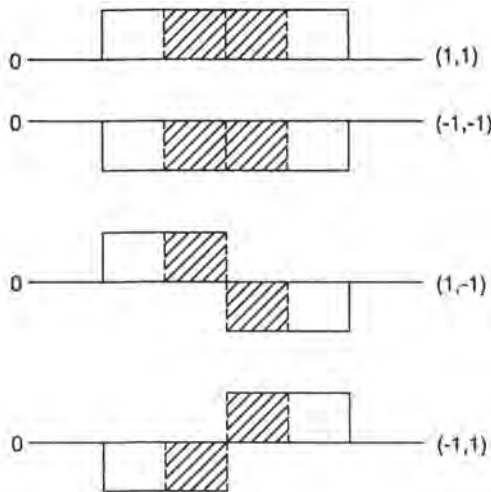


Figure 12.25. Illustration of the four crosstalk possibilities.

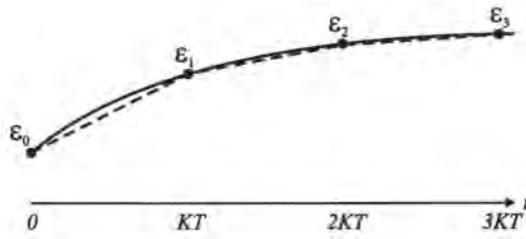


Figure 12.26. Illustration of the slowly varying nature of the residual phase noise.

for $r = 0, \pm 1, \pm 2, \dots$. Thus, we can generate values of $\epsilon(rKT_b)$ and obtain values for $\epsilon(nT_b)$ by *linear interpolation*. Using (12.2.29) in (12.2.28) produces the following result:

$$\begin{aligned} \frac{1}{A_c} Z_n &= \frac{I(n)}{m_r} \{ \sin(\epsilon_{r-1} + k\Delta\epsilon_r) - \sin[\epsilon_{r-1} + (k-1)\Delta\epsilon_r] \} \\ &+ \frac{Q(n-1)}{m_r} \{ \cos[\epsilon_{r-1} + (k-0.5)\Delta\epsilon_r] - \cos[\epsilon_{r-1} + (k-1)\Delta\epsilon_r] \} \\ &+ \frac{Q(n+1)}{m_r} \{ \cos(\epsilon_{r-1} + k\Delta\epsilon_r) - \cos[\epsilon_{r-1} + (k-0.5)\Delta\epsilon_r] \} + N_I(n) \end{aligned} \quad (12.2.30)$$

where

$$m_r = (\epsilon_r - \epsilon_{r-1})/KT_b$$

$$\Delta\epsilon_r = (\epsilon_r - \epsilon_{r-1})/K$$

$$k = n \bmod K$$

$$N_I(n) = \text{a normal r.v. with mean zero and variance } \sigma^2$$

$$\sigma^2 = N_0 T_b / A_c^2$$

$$N_0 = \text{one-sided input noise spectral density}$$

We can use (12.2.30) to generate a decision variable every T_b seconds, compare its value to a decision threshold, make a decision, compare that decision to the transmitted bit, and produce an error sequence. The process is illustrated in Figure 12.27. The entire process is quite efficient since it requires only the calculations in (12.2.30) once per bit and the generation of one normal random variable once per bit. The I- and Q-channel sequences are also randomly generated as bit-spaced ± 1 variables, and as discussed earlier, the phase noise values only need to be generated every K bits, a typical value of K being on the order of 50-100. The above procedure produces an error sequence of length on the order of 10^9 in about 2.5 hr on a Sun Sparc 10 workstation.

12.2.7. Postprocessing

A particular error sequence is generated for one value of signal-to-noise ratio ($1/\sigma^2$) and one value of rms phase noise (σ_ϵ). This sequence forms a file which can then be rearranged any number of ways to correspond to different interleaver arrays such that one dimension of

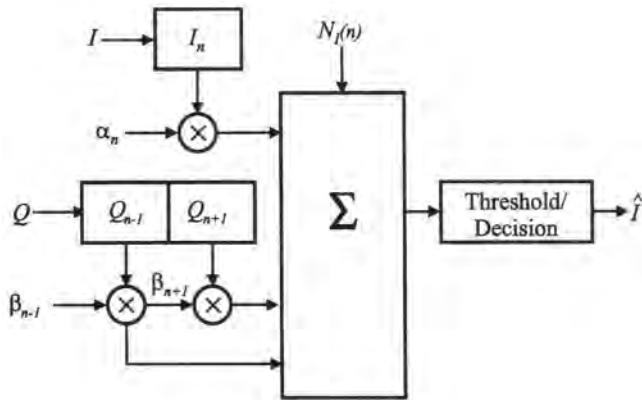


Figure 12.27. Block-diagram representation of the quasianalytical processing for producing an error sequence.

the array is the length of one codeword. The number of ones in a codeword is the number of errors. The error-correcting capability of the code is then applied to the readout codewords. A simplified algorithm is adequate for present purposes: for example, if the number of errors is $\leq t$, turn the ones into zeros, otherwise do nothing. After the entire file has been processed in this fashion, count the remaining number of ones and divide by the total number of bits: this is the estimated BER. A “generic” result is shown in Figure 12.28. It is generic in the sense that for a number of cases studied, the thresholding behavior shown is representative. There are two regions of error rate behavior, having different slopes, as indicated. The transition between the two regions is not always this sharp, but there appears to be always a small range of interleaver depths greater than which will more or less imitate an independent error channel, and less than which can lead to rapid degradation. The particular scaling on the

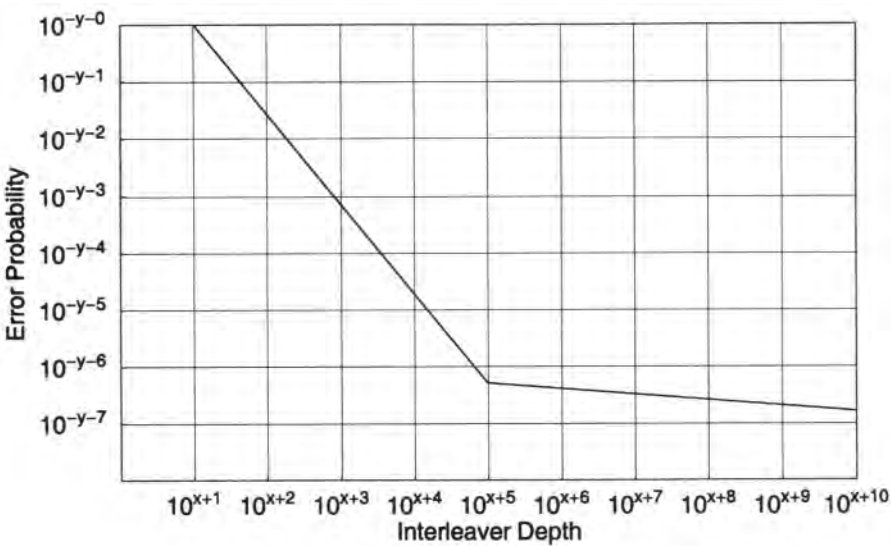


Figure 12.28. Typical (“generic”) behavior of the error probability as a function of interleaver depth. The actual curve and the location of the “knee” depend on the specific code, thermal noise level, and phase noise characteristics.

abscissa and ordinate (i.e., the values of y and x in the exponent of the scale labels) depend in general on the thermal noise level, the rms phase noise, the phase noise spectrum, the code's error-correcting capability, and the codeword length.

12.2.8. Conclusions

In this case study we have illustrated one approach to a computationally demanding problem. The approach combines a number of methodological and computational ideas to effect complexity reduction. The first step was to simplify the problem by idealizing all but the most relevant sources of impairment for the problem at hand. In any particular problem, this kind of simplification relies on judgment which may need to be validated after the fact. Another step in complexity reduction is achieved by omitting explicit simulation of encoders and decoders, since the decoded performance can be approximated well enough from the postprocessing files structured into blocks of codeword length. Perhaps the most significant construct for this study is the development of a *residual* phase noise model, which effectively eliminates the need to simulate frequency sources and consequently devices intended to process them such as frequency converters and phase tracking loops. This is an application of what was called equivalent random process methodology. Although this methodology was applied within an otherwise idealized system, it should also be applicable in less restricted situations. With the assumptions and techniques applied to this point we could now develop a *mixed quasianalytic* detection process in which the analytical portion allows us to deal with only symbol-spaced samples at the detector output and, additionally, even further-spaced samples of the residual phase noise process by taking advantage of multirate sampling.

12.3. Case Study III: Exploring the Effects of Linear and Nonlinear Distortions and Their Interactions on MSK-Modulated Signals: A Visual Approach

12.3.1. Introduction

In Chapter 2, it was mentioned that a unique aspect of simulation is the capability to produce the evolution of a waveform in time, and that the ability to observe this waveform, or certain of its properties, at different points in the system can provide a great deal of insight to the system engineer which otherwise might be difficult to obtain. The objective of this case study is to demonstrate the utility of this dynamic aspect of simulation through the specific case to be described. As will be seen, while the nature of the insight is largely qualitative and visual, it leads to a much sharper understanding of the relationship between performance (i.e., BER) and equipment parameters, which can actually be exploited to improve performance. Consider the nonlinear communication system model in Figure 12.29, which employs MSK modulation. Both the transmitter and channel segment distortions are modeled as the cascade of a linear filter followed by a memoryless nonlinearity exhibiting AM/AM and AM/PM conversion. As was discussed in Chapter 5, this is one of the (simpler) possible models that can be used for modeling nonlinear amplifiers. The filter, however, not only may represent the frequency-selective behavior of the device, but can incorporate any such behavior preceding the amplifier. To complete the description of the system, the receiver performs carrier phase recovery, demodulation, zero-forcing equalization, and bit detection. The details of these operations are not central to the following discussion.

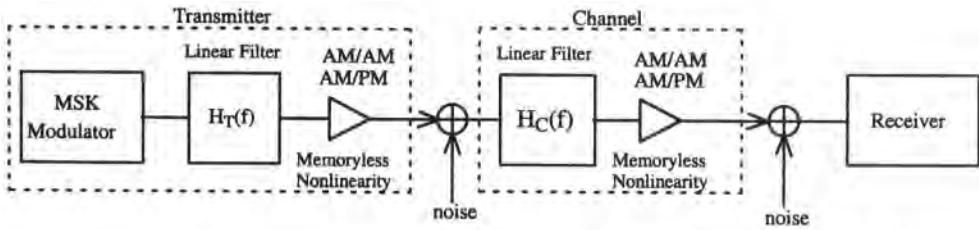


Figure 12.29. Simplified block diagram of system in Case Study III (from Ref. 23, © IEEE, 1996).

In order to study the effects of linear distortions and of their interaction with the nonlinearity, it is necessary to have a parametric description of the linear filters. Such a parameterization is also useful for specification purposes. Toward this end, we choose the modeling methodology outlined in Chapter 2, whereby we describe both the amplitude and phase characteristics over the passband by low-order polynomial fits and their associated residuals, in addition to a suitable definition of the passband and a means for describing the “skirts” of the filter. For simplicity, we will assume the filter passband (amplitude characteristic) is defined by a “brickwall” filter over the data rate bandwidth, and we will take the polynomials to be of second order for the amplitude response and of third order for the phase response. In other words, the filter imperfections are described by linear amplitude, parabolic amplitude, parabolic phase, cubic phase, residual amplitude, and residual phase (see Figure 12.30). Residual amplitude and residual phase are, by definition, components of the amplitude and phase, respectively, which cannot be described by the polynomial of the given order. Residuals generally have a ripple-like appearance, but do not have analytically describable structure. We should also note that the first-order (linear) term in the phase polynomial does not contribute to distortion, hence is ignored from now on. But its existence is implied in order to make the composite phase characteristic physically realizable.

The motivation for this study originated from simulation results of the system of Figure 12.29, which indicated that BER performance is highly sensitive to the relative polarities of the parabolic phase distortion and of the AM/PM characteristics that follow the linear filters. We will say that $AM/PM < 0$ ($AM/PM > 0$) if the phase shift versus input power curve exhibits a negative (positive) slope over the entire operating range. For the tradeoffs performed on this system in which the AM/PM characteristics were negative, the average required E_b/N_0 for a certain target BER was 14.9 dB when positive parabolic phase was used in both segments and 17.9 dB for negative parabolic phase. When positive AM/PM was used, the opposite trend was observed, although the disparity in required E_b/N_0 was not as dramatic. As a further illustration, Figure 12.31 shows the BER as a function of the degree of parabolic phase in $H_T(f)$ with $AM/PM < 0$ in both the transmitter and channel. It can be seen that the minimum BER occurs at $+30^\circ$, which suggests the possibility for a kind of self-equalization by biasing the parabolic phase in $H_T(f)$ in the positive direction.

These results provide the motivation to probe the linear and nonlinear filtering effects on MSK signals and to understand how the interaction between parabolic phase and AM/PM has the observed effect on BER performance. The insight we seek will be provided by simulation-generated scatter plots obtained by sampling the complex envelope at the input to the demodulator. (Equivalently, the sampled waveform can be thought of as the I and Q components of the demodulated waveform, with an ideal demodulator.) Before looking into this interaction, it is useful to clarify the distortion effects of a linear filter by itself. Therefore, in the next section we first look at the effects of the polynomial distortions as characterized by

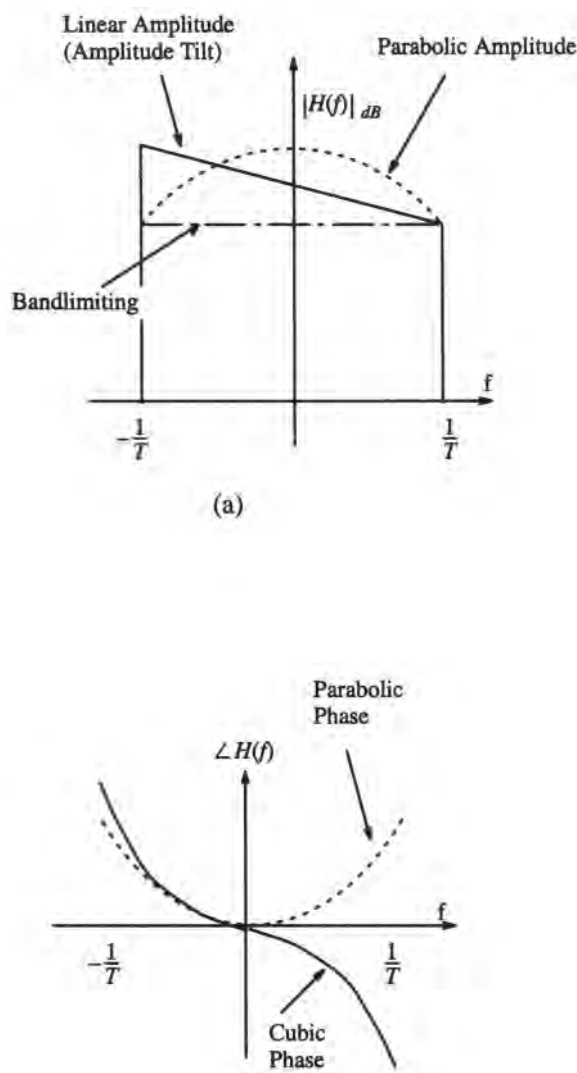


Figure 12.30. Illustration of the decomposition of filter transfer function into polynomial representations. (a) Amplitude characteristic; (b) phase characteristic (from Ref. 23, © IEEE, 1996).

the sampled complex envelope prior to data filtering. There are different possible observation points, for example, at the output of the data filter (matched filter). However, the effects we are seeking to understand are more clearly evident prior to detection, because the narrower bandwidth of the data filter tends to blur these effects. Therefore, studying the effects of distortions at the center sampling instant prior to data filtering will provide us with a good discriminator for understanding the differences among the various filter distortions. The simulation-generated scatter plots show that linear filter distortions cause specific transformations of ideally modulated MSK signals. The transformations can also be explained by viewing MSK-modulated signals as clockwise- and counterclockwise-rotating phasors. In the section following, we then study the nonlinear distortions produced by instantaneous AM/AM and AM/PM conversions. By applying these nonlinear transformations to the

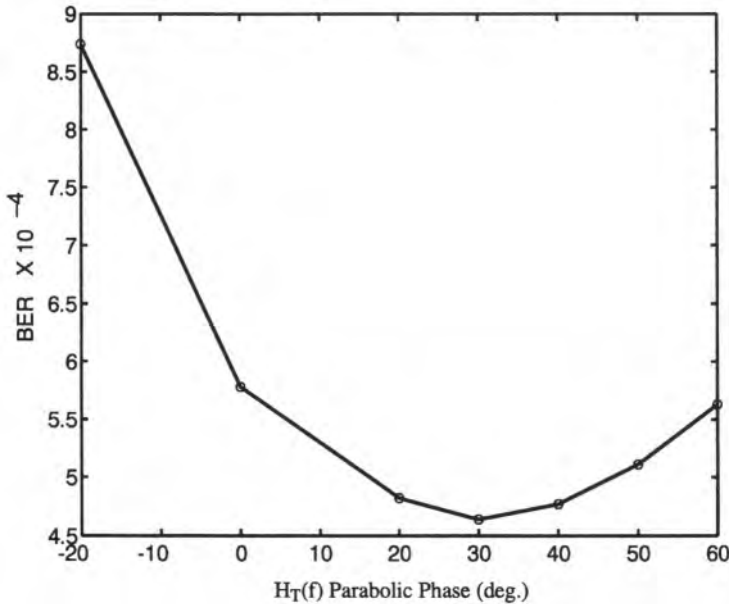


Figure 12.31. An example of BER performance as a function of filter parabolic phase component (from Ref. 23, © IEEE, 1996).

scatter plots associated with linear filter distortions, we then show how certain linear filter distortions can precompensate for subsequent nonlinear distortions. Specifically, we demonstrate that the phase variations at the output of a memoryless nonlinearity with $AM/PM < 0$ are less when preceded by positive parabolic phase distortion as opposed to negative parabolic phase. Similarly, positive parabolic phase distortion prior to a nonlinearity with $AM/PM > 0$ causes more phase variation than does negative parabolic phase.

It should be mentioned that the sign of the AM/PM characteristic is not a choice that can be made in actual hardware. The nature of the AM/PM curve is inherent to the type of device that is being used. Traveling-wave tubes will tend to have negative slope, while in solid-state amplifiers the slope can be positive or negative.

12.3.2. Linear Filter Distortions

12.3.2.1. Preliminaries

The real and imaginary components $I(t)$ and $Q(t)$, respectively, of a lowpass-equivalent MSK signal are shown in Figure 12.32. For an MSK signal, one of four states can be detected at any sampling instant $kt/2$, where T is the symbol period and k is an integer. These four states are $(0, -1)$, $(0, 1)$, $(-1, 0)$, and $(1, 0)$ where the states are denoted by their real and imaginary values at the sampling instant, i.e., (real, imaginary). The scatter plot for an ideal MSK-modulated signal is shown in Figure 12.33. The scatter plot shows the location of the sampled lowpass-equivalent signal in the complex plane. The horizontal axis corresponds to the signal amplitude of the real part (I channel) and the vertical axis corresponds to the signal amplitude of the imaginary part (Q channel). The dotted circle indicates the unit circle for reference, as most of the plots are normalized for unity average power.

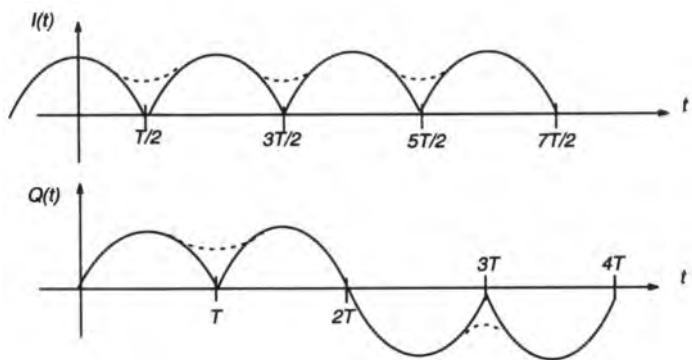


Figure 12.32. Ideal and filtered MSK waveforms (from Ref. 23, © IEEE, 1996).

As is well known, the phase of the (undistorted) MSK signal varies linearly and increases or decreases by 90 deg in a symbol interval. The visualization of MSK signals as clockwise- or counterclockwise-rotating phasors can be used to provide a qualitative understanding of linear filtering effects. Since we will be sampling the complex envelope at nominal decision times (indicated by the four positions around the circle in Figure 12.33), it will be helpful to focus on the dynamics of the waveform near a sampling instant in the complex envelope domain. Without loss of generality, suppose we assume that the signal is passing through the point (1, 0) at the time $t = 0$. Therefore, the signal transitions passing through (1, 0) can exhibit only four possible trajectories, which are shown in Figure 12.34. The real and imaginary parts of $s(t)$ for each of the trajectories of the rotating phasor are shown in Figure 12.35. Note that $s_1(t)$ and $s_2(t)$ correspond to phasors which change direction as they pass through (1, 0) while $s_3(t)$ and $s_4(t)$ represent clockwise- and counterclockwise-rotating phasors, respectively. As noted, the interpretation of MSK signals as clockwise- and counterclockwise-rotating phasors can be used to provide a qualitative understanding of linear

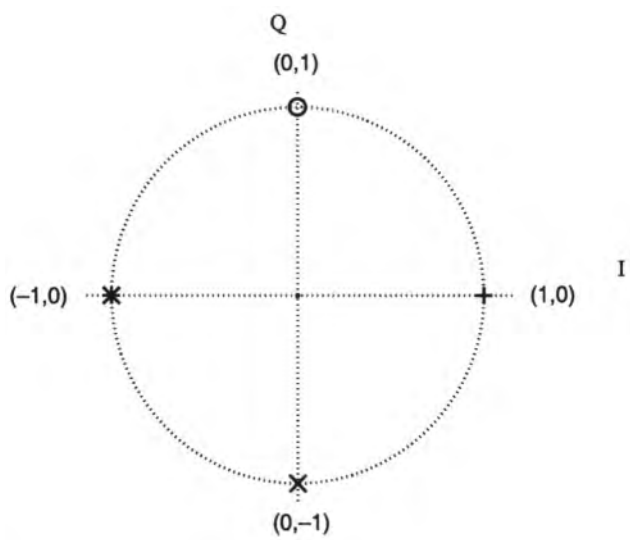


Figure 12.33. Signal space representation of MSK, showing ideal values of the sampled signal.

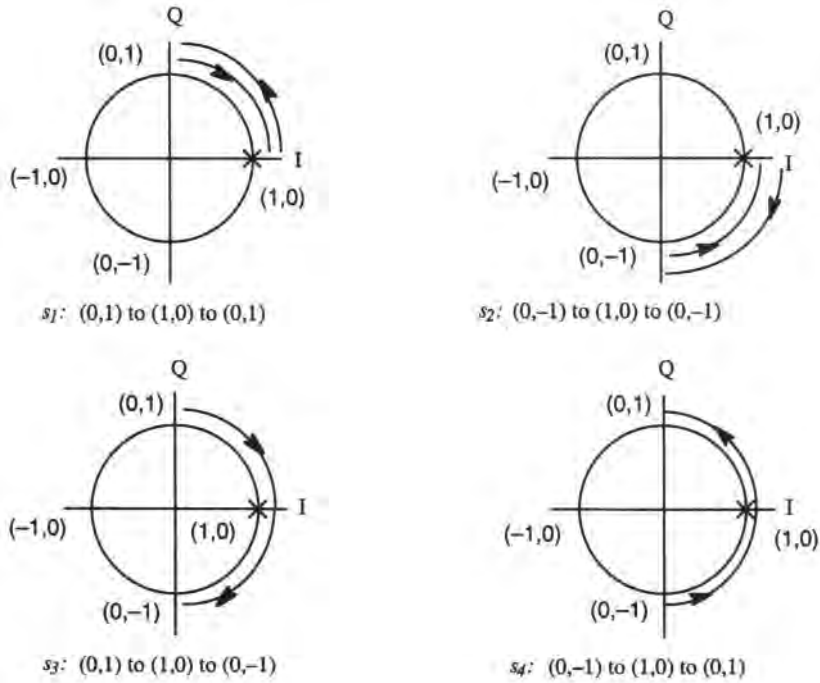


Figure 12.34. MSK signal transitions as the signal changes states (from Ref. 23, © IEEE, 1996).

filter effects. That is, in keeping with FSK view of MSK, linearly increasing or decreasing phase means that the instantaneous frequency is either above or below the carrier. Thus, the gain of a clockwise- rotating phasor is affected by the amplitude response at negative frequencies. Similarly, the gain of a counterclockwise-rotating phasor is a result of the amplitude response at positive frequencies. Alternatively, the advance or delay of a rotating phasor is a result of the group delay at the frequency of interest.

12.3.2.2. Bandlimiting

Bandlimiting is inherent in all of the linear filter distortions considered here. A scatter plot of an MSK signal filtered by an ideal bandlimited channel is shown in Figure 12.36. The clusters in the scatter plots are numbered according to their associated signal trajectory as defined in Figure 12.34. As mentioned earlier, the filter used is an ideal “brick wall” with a cutoff frequency at one half the data rate and a zero-phase response. Observe that the single points that occur in each quadrant for the ideal MSK scatter plot (Figure 12.33) are now each separated into a cluster of three. This phenomenon results from the fact that bandlimiting removes sharp transitions in the signal’s phasor trajectory as shown in Figure 12.32. These sharp transitions are caused by the change of direction associated with signals $s_1(t)$ and $s_2(t)$ from Figure 12.35. Similar observations have been made in Ref. 24.

12.3.2.3. Linear Amplitude

If we consider linear amplitude distortion (the term linear here means the amplitude is a straight line in the decibel domain), which is also referred to as amplitude tilt, we recognize

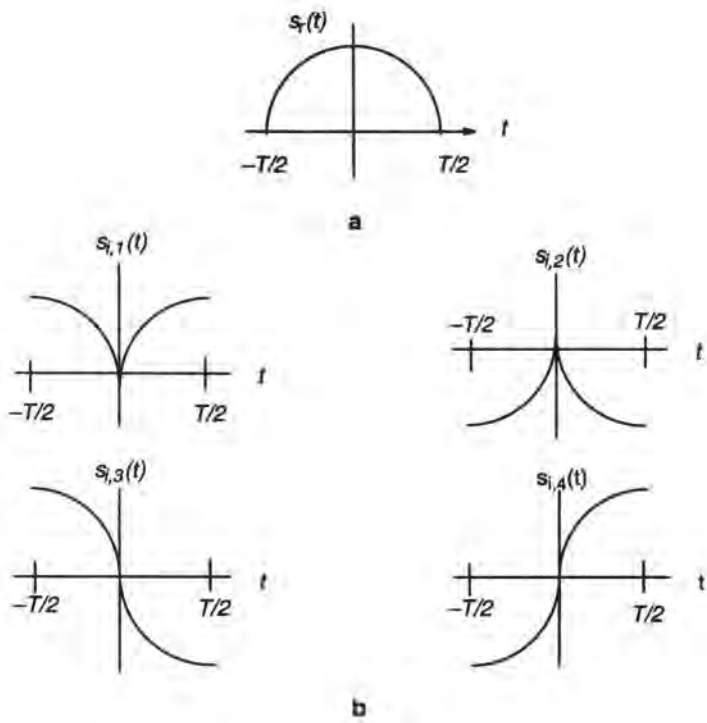


Figure 12.35. (a) The real part of an MSK signal; (b) the imaginary part for different transitions (from Ref. 23, © IEEE, 1996). $s_n(t) = s_r(t) + js_{i,n}(t)$ for $n = 1, 2, 3$, and 4.

that positive frequencies are attenuated more or less than negative frequencies, depending on the polarity of the tilt. Since a rotating phasor in the clockwise direction corresponds to negative frequencies, we can expect attenuation of $s_3(t)$ when the tilt is positive. Likewise, we can expect amplification of $s_4(t)$. The opposite effect occurs with negative tilt. The scatter plots for linear amplitude distortion are shown in Figure 12.37. Note that the sampling instants associated with $s_3(t)$ and $s_4(t)$ are dispersed inside or outside the unity envelope circle according to the polarity of the amplitude tilt. The sampling instants associated with $s_1(t)$ and

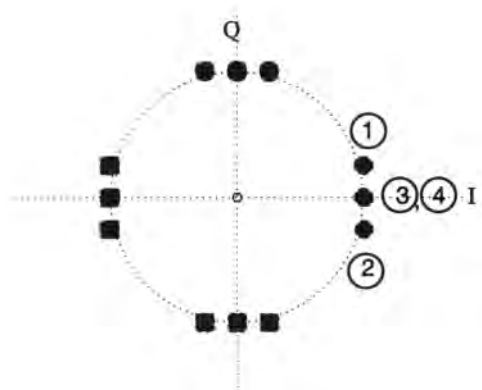


Figure 12.36. Scatter plot for MSK signal distorted by an ideal brickwall filter (from Ref. 23, © IEEE, 1996).

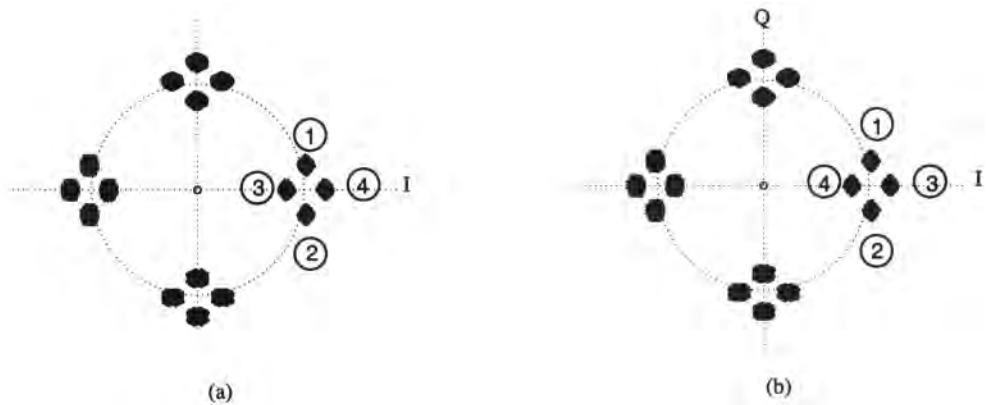


Figure 12.37. Scatter plot for MSK signal distorted by “linear” amplitude distortion, (a) Positive tilt (slope); (b) negative tilt (slope) (from Ref. 23, © IEEE, 1996).

$s_2(t)$ experience no net gain or attenuation due to the change of direction, but are affected by the bandlimiting as described in the previous section.

12.3.2.4. Parabolic Amplitude

The effect of parabolic amplitude distortion (again, amplitude here is measured in decibels) is shown in Figure 12.38. For positive parabolic amplitude, both clockwise- and counterclockwise-rotating phasors experience gain so that each of the four constellation points move outside the unit circle. For negative parabolic amplitude distortion, all of the trajectories experience attenuation. In addition to all of the points moving inside the unit circle, the effect is similar to what one would expect from severe bandlimiting.

12.3.2.5. Parabolic Phase

The effect of parabolic phase distortion on an MSK signal is shown in Figure 12.39. For both polarities of parabolic phase distortions, the sampling instants associated with $s_3(t)$ and

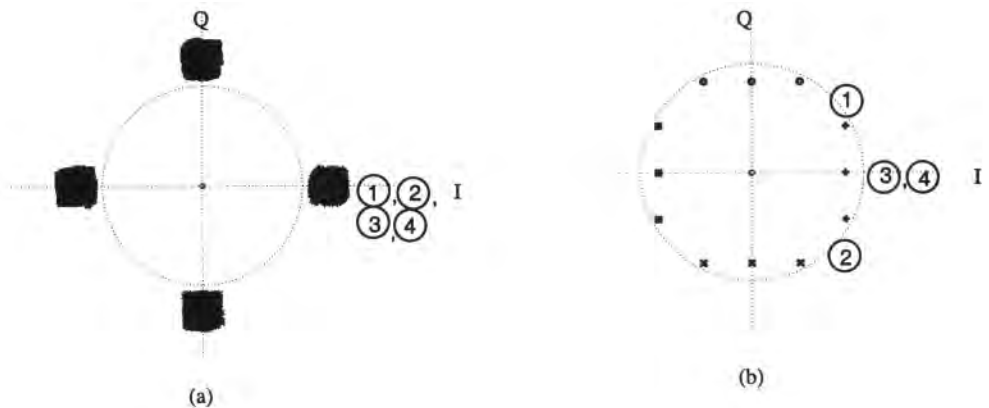


Figure 12.38. Scatter plot for MSK signal distorted by parabolic amplitude distortion, (a) Positive amplitude (concave up); (b) negative amplitude (concave down) (from Ref. 23, © IEEE, 1996).

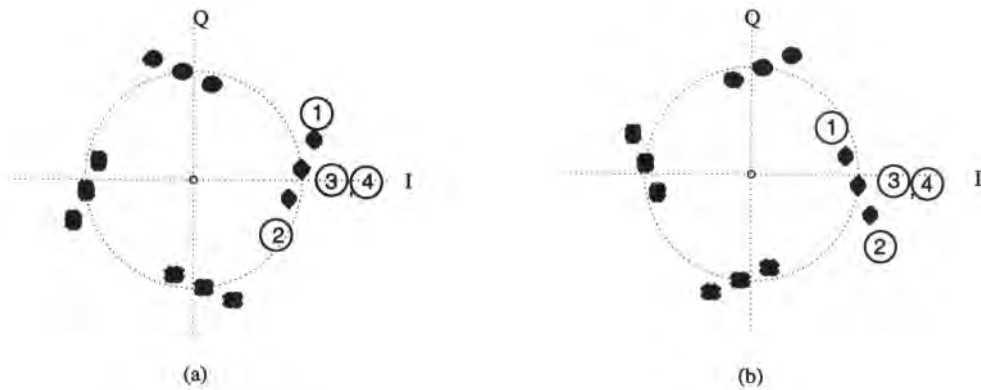


Figure 12.39. Scatter plot for MSK signal distorted by parabolic phase distortion, (a) Positive parabolic phase; (b) negative parabolic phase (from Ref. 23, © IEEE, 1996).

$s_4(t)$ do not experience noticeable distortion. However, those associated with $s_1(t)$ and $s_2(t)$, which change direction, cause a rotation of the clusters associated with each of the constellation points. This phenomenon can be attributed to the fact that for positive parabolic phase, negative frequencies are delayed while positive frequencies are advanced. The effect is opposite for the case of negative parabolic phase. Also note that envelope fluctuations more significant than those caused by bandlimiting are created even though the gain of the filter is constant over the data rate bandwidth. A sketch of the mathematical proof for the effect of parabolic phase distortion can be found in Ref. 23.

12.3.2.6. Cubic Phase

The effect of cubic phase distortion is shown in Figure 12.40. Unlike parabolic phase, there is no difference in the scatter plots between positive and negative polarities. This can be explained by observing that all signal trajectories experience an advance for positive cubic phase distortion and a delay for negative cubic phase. The bandlimiting effect dominates, as shown in Figure 12.40.

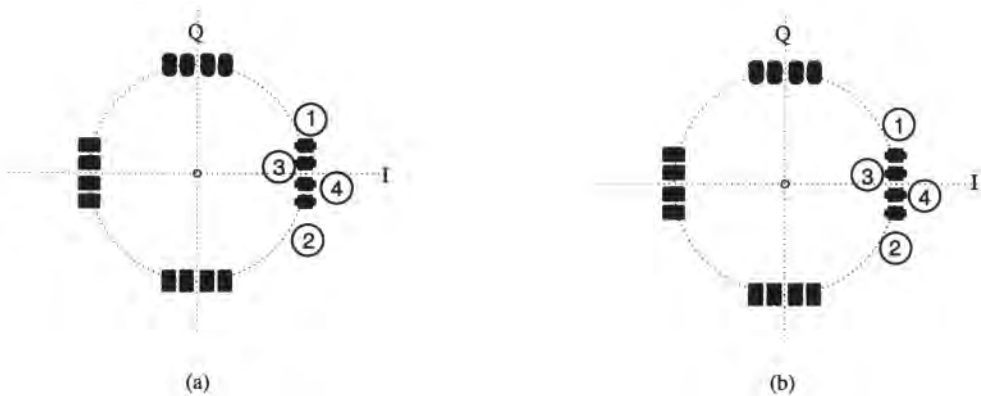


Figure 12.40. Scatter plot for MSK signal distorted by cubic phase distortion, (a) Positive cube; (b) negative cube (from Ref. 23, © IEEE, 1996).

12.3.2.7. *Residual Amplitude and Phase*

To demonstrate the effect of residual amplitude and phase, a filter response from a measured channel was fitted to a second-order and a third-order polynomial for the amplitude and phase, respectively. The polynomial components were then removed from the filter so that only the residual remained. These residuals are shown in Figure 12.41. Their effects are shown in Figure 12.42. Notice that the residuals of the amplitude and phase cause the appearance of random dispersion of the samples. This can be explained by the theory of paired echoes⁽²⁵⁾ in which ripples in the frequency domain result from echoes, or intersymbol interference, in the time domain, which is known to cause dispersion of the constellation points.

12.3.2.8. *Combined Effects of Linear Filter Distortions*

The combined effect of parabolic phase and residual amplitude and phase is shown in Figure 12.43. Note the combination of dispersion resulting from the residual distortion and the rotation of the clusters caused by the parabolic phase distortion.

12.3.3. **Memoryless Nonlinear AM/AM and AM/PM Distortions**

In this section, we study the distortions produced by memoryless AM/AM and AM/PM nonlinear conversions and demonstrate the effect of these transformations on the scatter plot. The nonlinear device is assumed to be in saturation for this analysis and therefore has the effect of “compressing” the signal and forcing constant envelope. A typical power-out (P_{out}) versus power-in (P_{in}) curve for this type of device is shown in Figure 12.44a. On the scatter

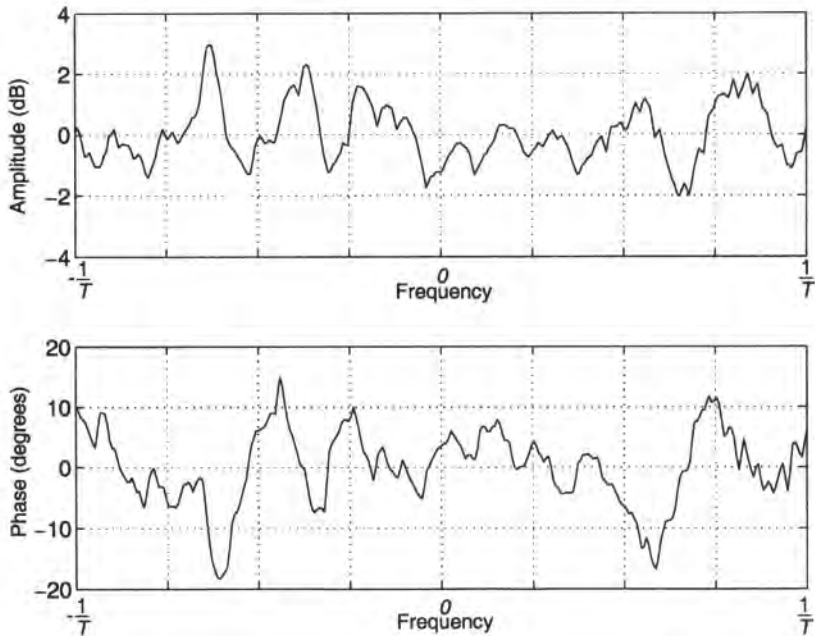


Figure 12.41. Amplitude and phase residual components Case Study III.

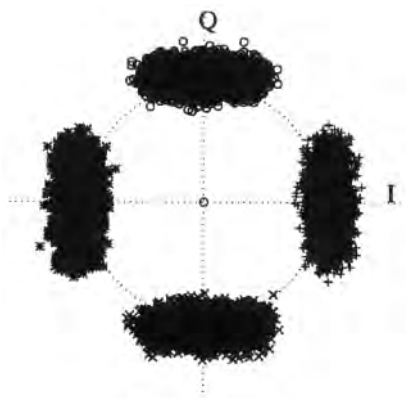


Figure 12.42. Scatter plot for MSK signal distorted by the residual components (from Ref. 23, © IEEE, 1996).

plot, the AM/AM effect will be evident from the compression of points onto the unit circle. In plotting, we have normalized the signal amplitude by $\sqrt{P_o}$ so that the average instantaneous power is unity.

The AM/PM conversion dictates the change in phase associated with input amplitude fluctuations. A typical phase-out versus power-in curve in which the AM/PM is negative (slope of the curve is negative) is shown in Figure 12.44b. Since the nonlinearity is modeled as a memoryless device, inputs whose instantaneous power is less than (greater than) unity envelope will be shifted in phase by some value greater than (less than) ϕ_0 . Here ϕ_0 is the nominal phase offset caused by the nonlinearity, and this will be tracked out by the demodulator. The net result of these AM/AM- and AM/PM-induced phase variations as seen on the scatter plots is illustrated by Figure 12.44c. Notice that samples outside the unit circle are shifted clockwise (negative phase shift) and samples inside the unit circle are shifted counterclockwise (positive phase shift).

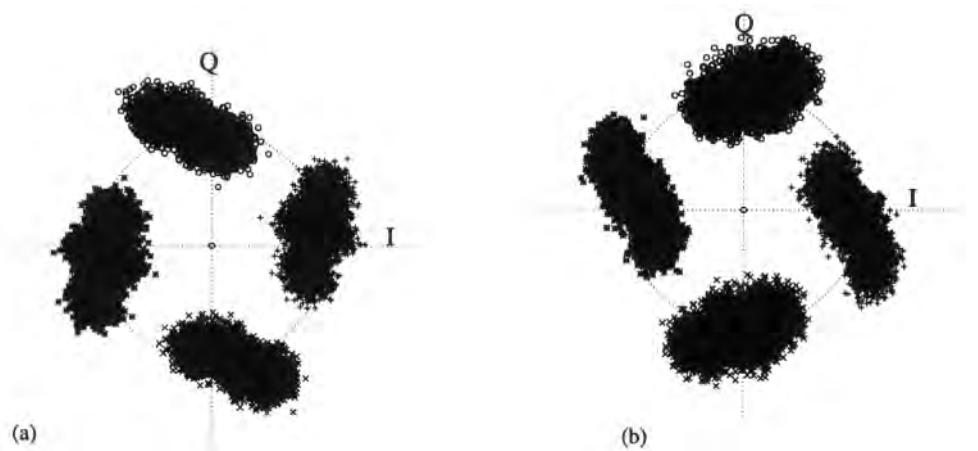


Figure 12.43. Scatter plots for MSK signal distorted by the combination of parabolic phase and the amplitude and phase residuals. (a) Positive parabolic phase; (b) negative parabolic phase (from Ref. 23, © IEEE, 1996).

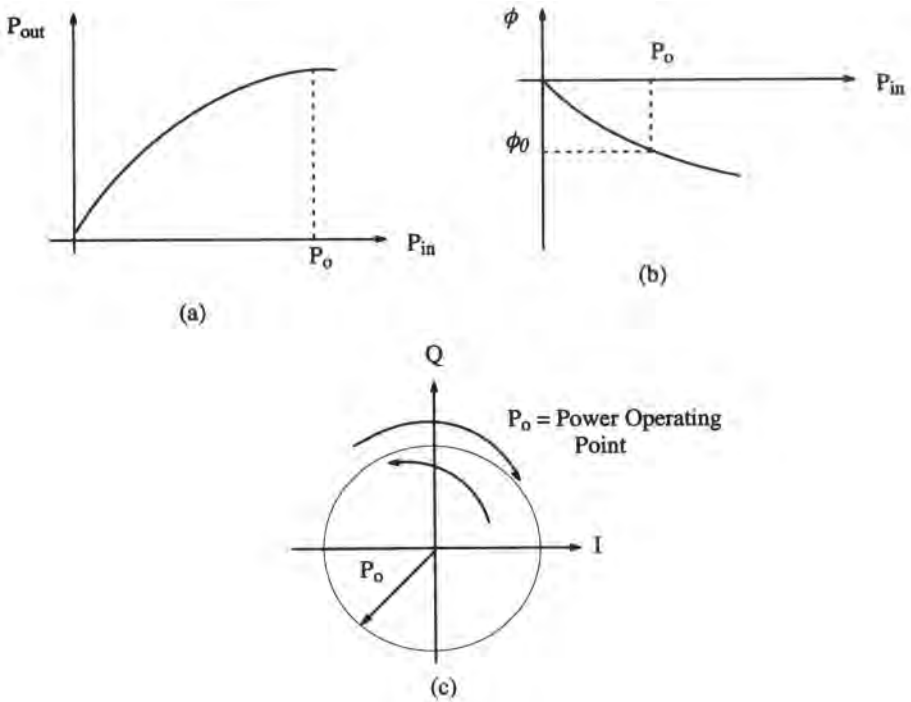


Figure 12.44. Description of the nonlinearity. (a) Typical compression characteristic; (b) typical phase characteristic; (c) qualitative description of the nonlinearity's effect on the signal (from Ref. 23, © IEEE, 1996).

12.3.4. Nonlinear Filter Distortions

In Section 12.3.2, the effects of linear filter distortions on MSK-modulated signals at the sampling instant were readily seen in the scatter plots. In the previous section, we showed how memoryless AM/AM and AM/PM conversions produce both compression of the envelope and either positive or negative phase shifts with respect to the nominal phase. The effect of a nonlinear filter, which can be modeled as the cascade of a linear filter and a memoryless AM/AM and AM/PM, can be studied by combining the results seen in the previous sections. In this section, we will focus on the combined effect of parabolic phase distortion and AM/AM and AM/PM to explain the significant impact on BER performance discussed in Section 12.3.1. Specifically, we will demonstrate that AM/PM-induced phase variations can either be counteracted or compounded depending upon the polarity of the parabolic phase distortion prior to the memoryless nonlinearity. The result of the counteracting or compounding is a decrease or increase, respectively, of the net phase variations.

In order to see the combined effects of linear and nonlinear distortions, an MSK-modulated signal was passed through a linear filter exhibiting negative parabolic phase and amplitude and phase residuals. The linearly filtered output was passed through a memoryless nonlinearity with $AM/PM < 0$ and repeated for $AM/PM > 0$. The linearly filtered complex signal was sampled at the center of the I-channel bits and plotted in Figure 12.45, as asterisks for $I = -1$ and as plus signs for $I = +1$. The output of the memoryless nonlinearity was sampled at the same time instants. The endpoints of the lines in Figure 12.45 indicate where the signal sample is mapped to after the nonlinearity and enables us to see the “movement” of

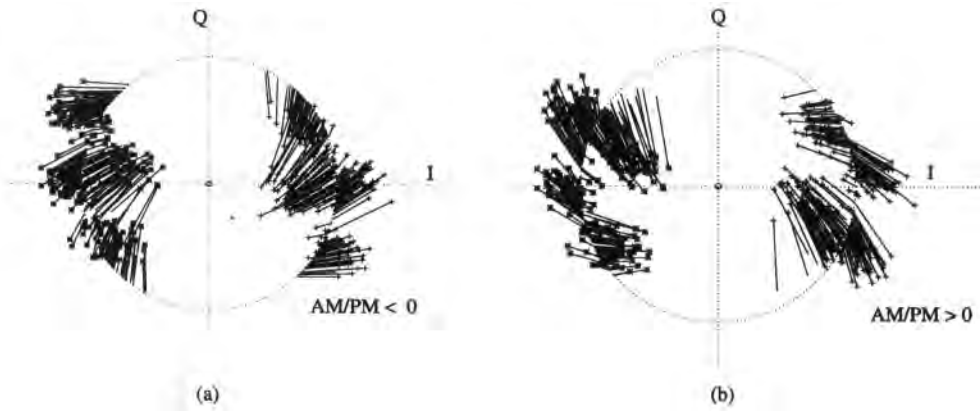


Figure 12.45. Movement of sampled signal points due to the presence of the nonlinearity (from Ref. 23, © IEEE, 1996).

samples on the scatter plot caused by the AM/AM and AM/PM. First, it can be seen by viewing the endpoints of the lines that the AM/AM causes a compression of the envelope for both AM/PM polarities. Second, note that negative (positive) AM/PM causes a clockwise (counterclockwise) rotation of the samples which lie outside the unit circle.

The combined effects of the linear filtering and the memoryless nonlinearity are more clearly illustrated by considering the effects with pure parabolic phase distortion, Figure 12.46 shows the effect produced by a memoryless nonlinearity with $AM/PM < 0$ on the linearly filtered signals of Figure 12.39. For positive parabolic phase, the clockwise rotation of points outside the unit circle and counterclockwise rotation of points inside the unit circle with $AM/PM < 0$ tends to restore the points back to their original position. However, for negative parabolic phase the opposite effect occurs. Therefore, $AM/PM < 0$ results in less phase variation when preceded by positive parabolic phase than for negative parabolic phase. Note that the constellations are rotated by an arbitrary amount which accounts for the nominal

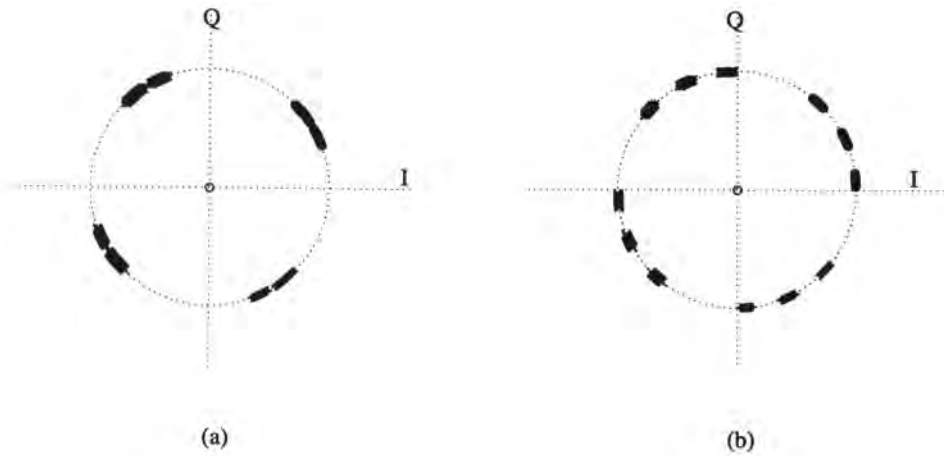


Figure 12.46. Scatter plots for MSK signal distorted by a combination of the memoryless nonlinearity and parabolic phase. (a) Positive parabolic phase; (b) negative parabolic phase (from Ref. 23, © IEEE, 1996).

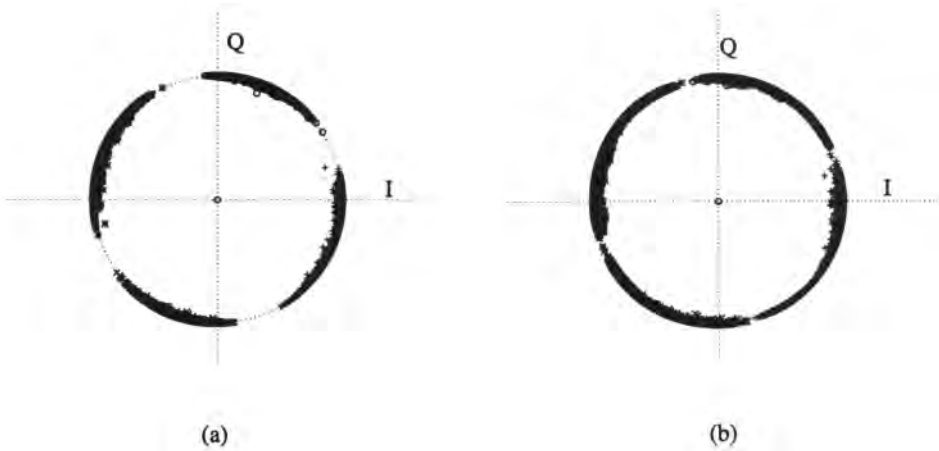


Figure 12.47. Scatter plots for MSK signal distorted by a combination of the memoryless nonlinearity, amplitude and phase residuals, and parabolic phase. (a) Positive parabolic phase; (b) negative parabolic phase (from Ref. 23, © IEEE, 1996).

phase of ϕ_0 that would be tracked out at the demodulator, but is not removed for these plots. Figure 12.47 shows the effect produced by $AM/PM < 0$ for a combination of gain/phase residuals and positive versus negative parabolic phase. Again, the dispersion or net phase variations are less in the case of positive parabolic phase due to the “restoring” action of the $AM/PM < 0$. Since dispersion at the sampling instant brings the I and Q values closer to the decision boundaries, the greater dispersion caused by a combination of $AM/PM < 0$ ($AM/PM > 0$) and negative (positive) parabolic phase will result in a larger BER.

12.3.5. Conclusions

We have shown how linear filter distortions affect MSK-modulated signals. We have also shown how AM/AM and AM/PM memoryless nonlinearities transform the signal samples of the scatter plots. We have studied the effects of a particular class of nonlinear filters on MSK-modulated signals which can be modeled as the cascade of a linear filter and a memoryless nonlinearity. Finally, we demonstrated that positive parabolic phase tends to compensate for the AM/PM -induced phase variations of the memoryless nonlinearity for $AM/PM < 0$, and that negative parabolic phase tends to compound the phase variations for $AM/PM < 0$. This increase or decrease in the net phase variations directly affects BER performance.

12.4. Case Study IV: Performance Evaluation of a CDMA Cellular Radio System

12.4.1. Introduction

The problem addressed in this case study is the performance evaluation of a “generic” CDMA cellular radio system. Below, we will make clear in what sense *performance evaluation* is to be taken. The system is “generic” in the sense that it imitates many of the salient features of current IS-95-like CDMA systems,^(26–28) but is not intended to duplicate

any particular existing system. In Chapter 9 we discussed the methodology for simulating a voice communication system operating over a fading channel. That methodology is based on the general principle outlined in Chapter 2 that an effective approach for a complicated enough problem is to partition it into a set of more manageable problems, and for each of those subproblems to further practice complexity reduction (simplify) to the extent possible. In the case at hand, two levels of partition are natural, corresponding to the *waveform* and *symbol* levels, respectively. The inputs and outputs of these two levels of simulation are displayed in Figures 12.48 and 12.49, respectively. A third level, which we do not deal with at all here, is the subjective evaluation corresponding to a particular discrete-level performance.

The waveform-level simulation is used to evaluate the behavior of the portion of the system consisting of modulators, channels, filters, etc. It is generally used to study the effect on the waveform of various parameters of these subsystems, which ultimately will be reflected in the detection performance of the system. In the case of a coded system, the detection performance in question is the “raw” sequence of errors which correspond to hard decisions at the detector. Among other things, this sequence yields, of course, the average error rate (BER) at the input to the decoder. Waveform-level simulation is computationally expensive because, among other reasons, it proceeds at some sampling rate f_s samples/s, typically corresponding to 8–16 samples/bit. However, once hard decisions (or soft decisions, for that matter) have been made, it normally suffices to process one sample per symbol (the logical or real value representing that symbol) since most digital equipment, such as error-correction decoders, interleavers, voice coders, etc., is designed to operate on that sequence of symbol values. It is therefore sensible to decouple the waveform-level from the symbol-level

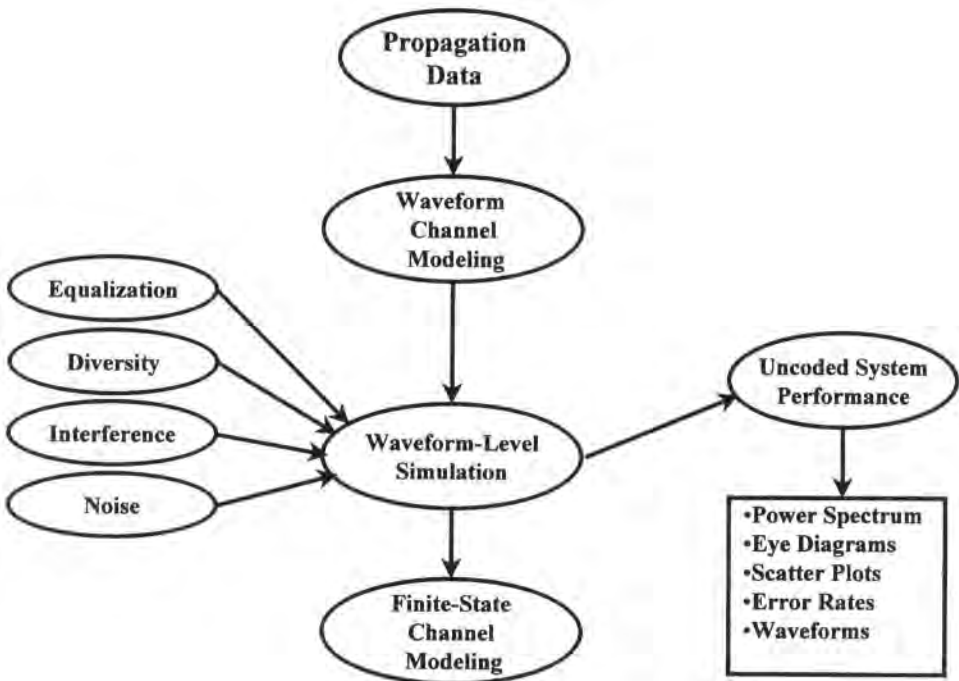


Figure 12.48. Waveform-level simulation.

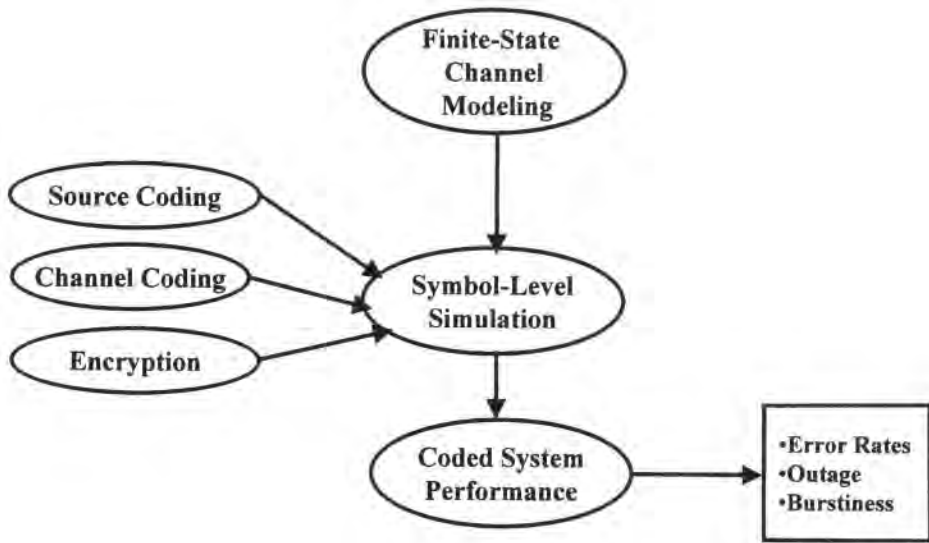


Figure 12.49. Symbol-level simulation.

simulations since repeated discrete-level simulations would not then be burdened each time with the “front-loading” of the waveform-level simulation.

The decoupling in question is achieved by synthesizing a finite-state channel model (as defined in Chapter 9), which in principle is a complete description of the system for the processing in question. The finite-state channel model is derived from the error sequence mentioned above, which is produced by the waveform-level simulation.^(29,30) In this study, the specific objective was to obtain a finite-state channel model that would account well enough for the observed error sequence. The error sequence, of course, has to be long enough to provide a statistically sufficient observation. As was discussed in Chapter 9, the family of hidden Markov models (HMM) provides a relatively general construct for representing the behavior of random discrete sequences, and in this case study we will see that particular members of this family are good candidates for modeling the error sequences produced by the system under study.

Because the derivation of an appropriate HMM is our main goal here (a subsidiary goal is the computation of frame error rate), and because the character of the error sequence is primarily dictated by the fading nature of the channel, it is reasonable to simplify as much as possible the models for the parts of the system which will have a substantially lesser effect on the objectives of our study. The very same point was made to justify the simplifications that we needed to make in Case Study II. Consequently, we omit here explicit simulation of the synchronization system, which is quite complex in a typical system. Acquisition and tracking are, of course, central issues in an actual system, and the reader is referred to Ref. 31, for example, for discussions of these issues. Similarly, individual components, such as the modulator, were assumed to perform their functions essentially ideally.

This case study employs a number of techniques and ideas discussed in other chapters, as outlined here.

- As already discussed, this case study is a case in point for illustrating the methodology of *partitioning and simplifying complex problems*.

- Because multiple processes with a wide range of bandwidths are involved, *multirate techniques* are used to advantage to improve the computational efficiency.
- The generation of a *Rayleigh fading channel process* along the lines discussed in Chapter 9 will be explicitly detailed.
- *The synthesis of an HMM* will be demonstrated, further elaborating the corresponding discussion of Chapter 9.

12.4.2. Brief Description of a CDMA Cellular System

A simplified block diagram of an air “link” in a CDMA system is shown in Figure 12.50, a link consisting of the transmission path either from a user to the base station, or from the base station to a user. In the first instance, the link is referred to as the *reverse link*, and in the second case the link is called the *forward link*. In either case, the voice signal to be transmitted is first voice-encoded to a rate of 9.6 kb/s. This bit stream is then error-correction-encoded (by the channel encoder), interleaved, and sent over the radio channel. Of course, a modulator and amplifier are implicit. This channel is the “wireless” part of the link, where fading occurs and external interference makes itself felt. At the receiver, deinterleaving takes place first, followed by demodulation, error-correction decoding, and voice decoding to the original message.⁽²⁶⁾ It may be noted that in the reverse link, the recovery of the original voice signal may not be performed because the base station has a number of retransmission possibilities. If there were only mobile users, recovery to the voice-encoded 9.6-kb/s level is all that would be necessary in principle. The main differences between the reverse and forward links in Figure 12.50 lie in the method of modulation and demodulation, and the specifics of the channel encoder, which are as follows:

1. The channel encoder of the forward link is a constraint-length-9, rate-1/2 convolutional encoder, which produces a 19.2-kb/s signal.
2. The channel encoder of the forward link is a constraint-length-9, rate-1/3 convolutional encoder, which produces a 28.8-kb/s signal.

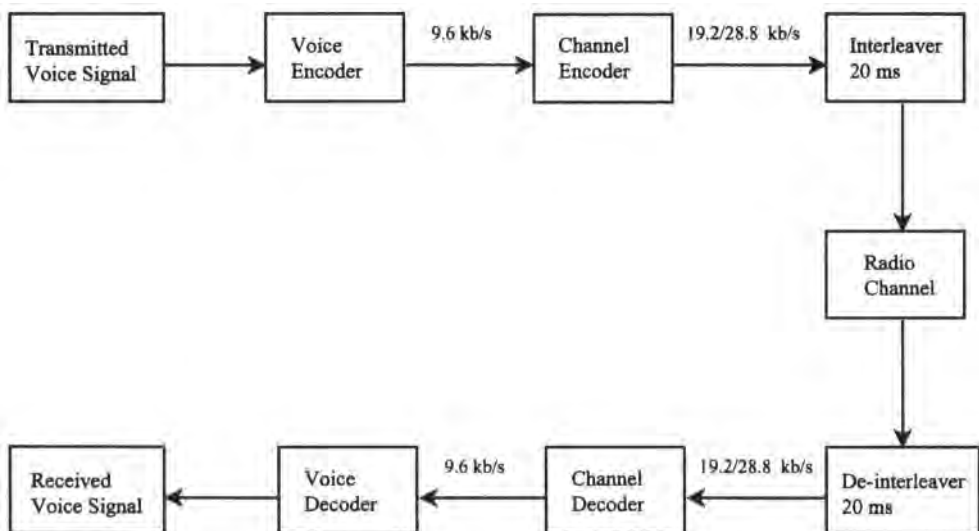


Figure 12.50. Block diagram of a radio link in the CDMA cellular system.

The interleaver frame is $T_f = 20$ ms long, which corresponds to a frame length of 576 and 384 encoded symbols, respectively, for the reverse and forward links. The particulars of the modulation schemes and the remainder of the system will be exposed in Section 12.4.3 for the reverse link and Section 12.4.4 for the forward link.

12.4.3. Reverse Radio Link Simulation

The reverse radio link, as mentioned, is the communication channel from remote (mobile) users to their base station. Because the base station receives signals from many users, it is not efficient to employ a coherent modulation/demodulation scheme, which would burden the receiving equipment with multiple synchronization subsystems (one for each user). Hence, a noncoherent scheme was devised, as detailed below. Of course, timing synchronization will still be required for making decisions, but as mentioned in the introduction, we do not explicitly simulate this function.

The system is also assumed to practice so-called “Rake” reception,^(32,33) which is consistent with the channel model to be described. Rake reception takes advantage of a channel characterized by resolvable rays, by individually collecting and then combining the energy from each of the rays. The Rake receiver is itself a tracking structure which can be considered another type of synchronization subsystem. Again, this function did not need to be explicitly simulated for present purposes.

12.4.3.1. The Simulation Model of the Reverse Radio Link

We now describe the various components of the simulation model: the transmitter, the channel, and the receiver. A more detailed block diagram of the reverse link is shown in Figure 12.51.

12.4.3.1.1. The Transmitter Since the objective of our study is to develop a finite-state channel model, it is not necessary to replicate the voice coding function (which is fairly complex), nor for that matter the channel encoding. Thus, our input signal can be directly an emulated channel-encoded sequence of 28.8 kb/s,[†] which we generate via a random binary sequence generator. This sequence is transformed into a waveform having 8 samples/bit. The data are then “modulated,” i.e., multiplied, by one of a set of orthogonal 64-bit sequences, sometimes referred to as Walsh functions,⁽²⁸⁾ in the following way.[‡] The 64-bit Walsh functions are 64 in number, hence each sequence represents a unique mapping to some 6-bit word. The “Walsh modulator” in the block diagram represents this mapping. Thus, successive 6-bit words $w^{(k)}(t)$, where k identifies one of the 64 possible 6-tuples, is mapped to a corresponding 64-bit Walsh function denoted $W_k(t)$. Evidently, the Walsh function must operate in real time at a rate which is $(64/6)$ times greater than the input sequence, namely 307.2 kb/s. The bit sequence input to the Walsh modulator is upsampled by a factor of 8, making 1843.2 ksamples/s, which is then compatible with sampling the output of the Walsh modulator at 6 samples/bit ($307.2 \times 6 = 1843.2$).

In the example system, the output of the Walsh modulator is split into two rails or channels, each of which is multiplied by a different spreading code (PN sequence), say $PN_1(t)$ and $PN_2(t)$, respectively. The PN sequences are de Bruijn sequences (see Chapter 7) of period

[†]In our description we will use real-time bit rates, but of course in the simulation context these numbers only indicate relative rates since the simulation does not process at real-time speed.

[‡]In general, there exist orthogonal sets with 2^k members and 2^k bits per sequence for all integer k .

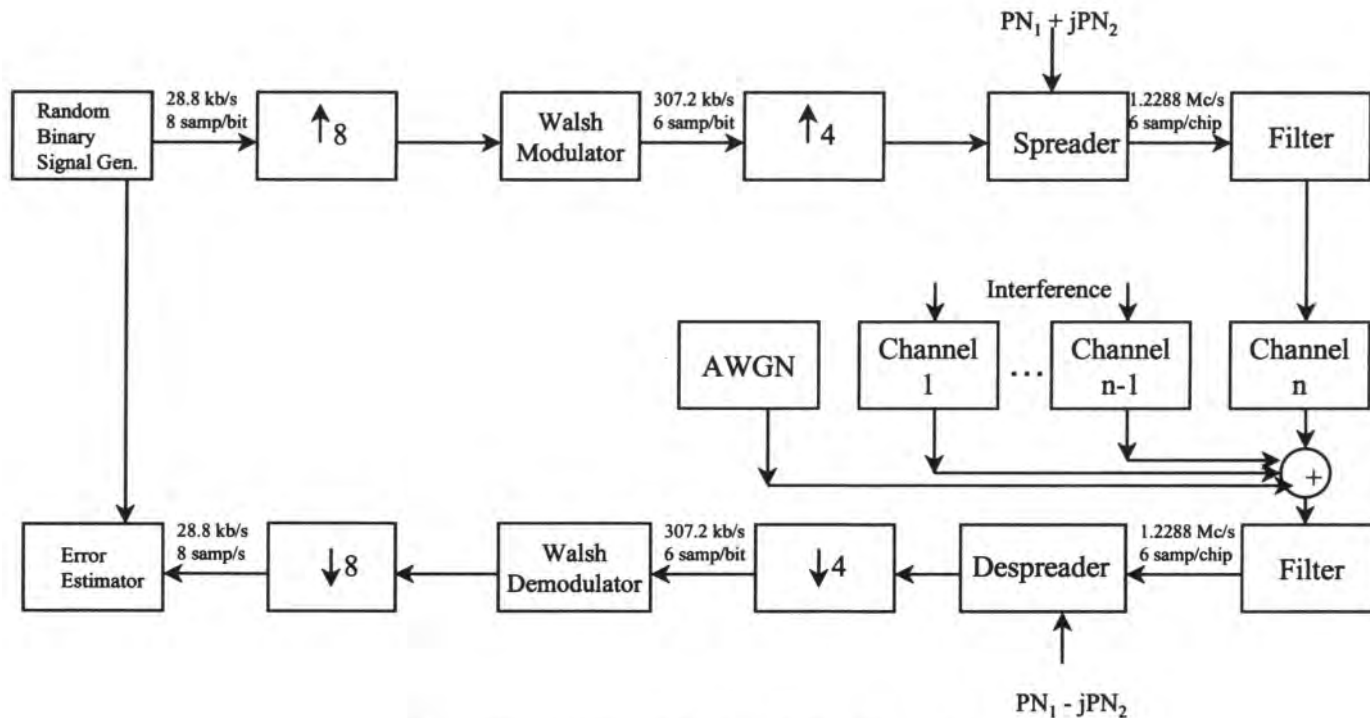


Figure 12.51. Block diagram of the reverse radio link simulation.

2^{15} operating at the rate of 1.2288 megachips/s (Mc/s); a de Bruijn sequence is simply a maximum-length linear shift register sequence with a single zero inserted. The processing gain of the system is thus $1.2288 \times 10^6 / 28.8 \times 10^3 = 42.67$, or 16 dB. The output of the Walsh modulator is upsampled by a factor of 4, which then permits us to sample the output of the spreader at 6 samples/chip. The I and Q rails are then fed to an OQPSK modulator, the offset being 1/2 chip between the two rails. The modulated carrier is then filtered by the transmitter filter, $h_T(t)$, which is a square-root raised-cosine filter with 0.35 rolloff. Since OQPSK modulation is a linear process, we can write for the complex envelope of the transmitted signal

$$S_T(t) = W_k(t)[PN_1(t) + jPN_2(t)] * h_T(t) \quad (12.4.1)$$

for some k in every 0.20833-ms interval (corresponding to one Walsh sequence). It should be emphasized that the *same* data are transmitted on both quadrature channels, which reduces the susceptibility to interference.⁽²⁸⁾

12.4.3.1.2. The Channel The wireless propagation channel model selected for this case study belongs to the “discrete multipath” category, as defined in Chapter 9, and is one of the reference channel models selected by the standards bodies to be used for PCS in the 2-GHz region. Specifically, the model is the vehicular outdoor channel model consisting of three Rayleigh paths with relative delays of 0, 1.5, and 14.0 μ s, respectively (see the Appendix in Chapter 9). The Doppler frequency for this model is 200 Hz, and the Doppler spectrum is the Jakes spectrum. The complex lowpass-equivalent impulse response for this channel is

$$\tilde{c}(\tau, t) = \sum_{i=1}^3 \rho_i(t) e^{j\theta_i(t)} \delta(t - \tau_i) \quad (12.4.2)$$

where τ_i , ρ_i , and θ_i are the delay, amplitude, and phase, respectively, of the i th path. The relative strengths of the three rays (i.e., the ρ_i) are 0, -3, and -6 dB, in the same order as listed for the delays. The channel rate of variation, as stated, is on the order of 200 Hz, which is much slower than the symbol stream by orders of magnitude. Therefore, generating the channel samples [i.e., samples of $\rho_i(t)$ and $\theta_i(t)$] at the higher rate would not only be wasteful, but would probably lead to roundoff problems. This is an appropriate place to use multirate techniques, which we applied in the simulation. The particulars of the technique for generating and interpolating samples of the channel response $\rho_i(t)e^{j\theta_i(t)}$ for any one of the rays are given in the Appendix to this chapter.

The propagation channel supports many users, and this represents (potential) interference to any particular user. For the example, a variable number (1–7) of in-cell and out-of-cell interferers could be generated. These interferers were modeled independently as real users with their own distinct propagation channels and spreading codes. Of course, in some situations an interferer may travel over much of the same path as the desired signal, but in most cases it is reasonable to assume the channel is different (uncorrelated) for each interferer. The received power levels of the interferers are assumed to be the same as that of the desired signal for the sake of simplicity. While the modeling just described is the most realistic, some computation can be saved by assuming that the combination of the aggregate interference and the receiver noise is well approximated by a Gaussian random process. This would not be a good approximation for a single interferer, since it has a tune-varying magnitude, but the sum of many interferers, even if individually fluctuating, would be reasonably represented as Gaussian with a fixed standard deviation. We will return to this point below.

12.4.3.1.3. *The Receiver* The received signal is the sum of the desired signal (as affected by the channel), the interference, and the receiver front-end noise:

$$S_R(t) = S_T(t) * \tilde{c}(\tau, t) + I(t) + n(t) \quad (12.4.3)$$

where $I(t)$ is the aggregate interference, a sum of terms similar to the desired signal, and $n(t)$ is the noise. The received signal is filtered by the matching square-root raised-cosine filter $h_R(t)$. This filtered signal is then despread by the appropriate delayed complex conjugate sequence on each Rake “finger” (as they are called), as shown in Figure 12.52. Since we are not reproducing the Rake tracking, we know *a priori* the relative delays τ_i . Hence, the despread and delayed signal out of each finger is given by

$$S_D(t, i) = S_R(t) * h_R(t)[PN_1(t - \tau_i) - jPN_2(t - \tau_i)], \quad i = 1, 2, 3 \quad (12.4.4)$$

Combining (12.4.3) and (12.4.4) produces

$$S_D(t, i) = 2\rho_i(t)e^{j\theta_i(t)}W_k(t - \tau_i) * h_R(t) * h_T(t) + N(t) \quad (12.4.5)$$

since $PN \cdot PN^* = 2$ and where $N(t)$ denotes the combination of despread noise and interference. PN code synchronization is implicit in (12.4.5), but we do not explicitly simulate that mechanism, for the reasons given before. The despread signals in each finger are time-aligned by delaying each amount of time $\tau_{\max} - \tau_i$, where $\tau_{\max} = \max(\tau_1, \tau_2, \tau_3)$, and then down-sampled by a factor of 4. The time-aligned signals, $S'_D(t, i)$ are identical to (12.4.5) with $W_K(t - \tau_i)$ replaced by $W_k(t - \tau_{\max})$ for all i .

The detection of each multipath component is done by correlating the signal with each of the 64 orthogonal Walsh functions. The correlator outputs (independent of $O_i(t)$ because of “noncoherent” detection) determine a 64-element vector for each i , as follows:

$$\begin{aligned} C_{i,j} &= \left| \int S'_D(t, i) W_j(t - \tau_{\max}) dt \right|^2 \\ &= 4|\rho_i(t)|^2 \left| \int [W_k(t - \tau_{\max}) * h_R(t) * h_T(t) + N(t)] W_j(t - \tau_{\max}) dt \right|^2 \\ &\text{for } j = 1, 2, \dots, 64 \end{aligned} \quad (12.4.6)$$

because the channel amplitude $\rho_i(t)$ can be considered practically constant over an integration interval (which corresponds to about 0.2 ms, as indicated before, while the channel time constant is on the order of $200^{-1} = 5$ ms, or about 25 times longer). The correlator outputs for the three multipath components are added, and the decision is made on the basis of the largest sum, i.e., one selects the transmitted Walsh sequence as that sequence $\hat{W}_j$ that satisfies

$$\hat{W}_j = \max_j \left(\sum_{i=1}^3 C_{i,j} \right) \quad (12.4.7)$$

The 6-bit word corresponding to $\hat{W}_j$ is then produced as the detected data.

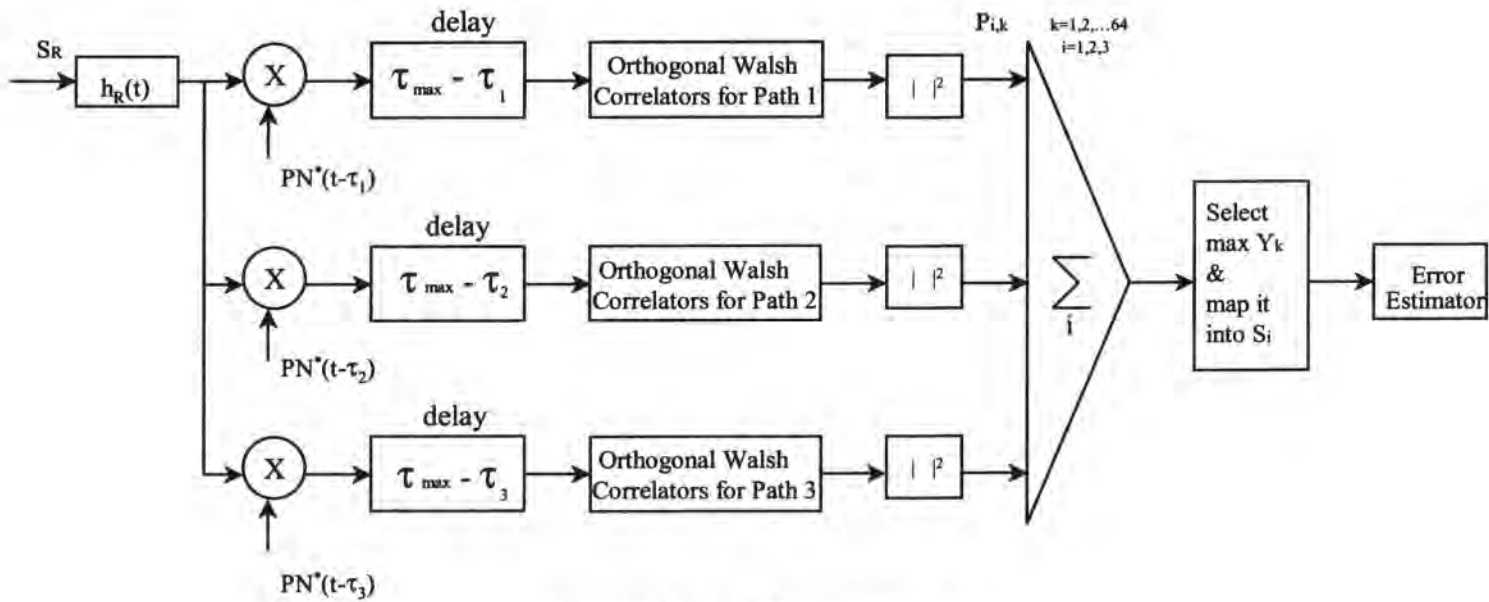


Figure 12.52. Receiver of the reverse radio link.

In evaluating (12.4.6b), as we mentioned earlier, $N(t)$ arises from the product of simulated interferers and noise with the local PN sequences. It could be that $N(t)$ is reasonably well approximated by a Gaussian random process. In that case, some processing time could be saved by recognizing that the term

$$\int N(t) dt$$

in (12.4.6b) could simply be directly generated only once per Walsh sequence as a properly scaled normal random number.

In any case, once the 6-bit word is produced, direct comparison with the transmitted word reveals whatever errors have occurred. Errors are flagged as 1's and no-errors as 0's. In this fashion, we produce an error sequence $\mathbf{e}$. This error sequence is used in two ways. As discussed later, it is the basis for synthesizing a discrete channel model, and it is also used to produce an estimate of the *frame error rate* (FER), a typical performance measure for this type of channel. A frame is considered in error if more than a certain number of errors occurs. A frame here is 576 bits long, so the sequence $\mathbf{e}$ is subdivided into 576-bit segments and the frame declared in error or not, as appropriate.

12.4.3.2. Simulation Run-Length Requirement

The frame error rate, as mentioned, is the measure of performance for the portion of the system ending at the demodulator output. If the number of errors is $\geq k$, say, a frame is in error, otherwise it is not. The probability of k or more errors in a frame, p_k , is estimated by $\hat{p}_k = n_k/N$, where n_k is the number of errored frames and N is the number of frames simulated. An errored frame is an "error event" as defined in Chapter 11. As was discussed there, the reliability of $\hat{p}_k$ is determined both by n_k and N , as well as the degree of correlation between error events. For independent error events, any of the formulations in either Section 11.2.3.1 (based on the binomial distribution), Section 11.2.3.2 (based on the Poisson approximation), or Section 11.2.3.3 (based on the normal approximation) of Chapter 11 can be used in the proper circumstances. Chapter 11 also discusses the fact that more observations are needed for a given confidence level if the error events are correlated. Because the channel is slowly time-varying, the question of correlation between frame error events arises. The rate of change of the channel is embodied in the space-time correlation function, which for the Jakes spectrum is given by $R(\tau) = 2\pi J_0(2\pi f_d \tau)$, where J_0 is the zeroth-order Bessel function (see Figure 9.8b in Section 9.1). A good rule of thumb for the decorrelation time is $T_{uc} = 5/(\pi f_d)$. Since $f_d = 200$, we calculate $T_{uc} = 8$ ms. For the transmission rate of 28.8 kb/s, this corresponds to 230 bits, which is about 40% of the frame duration of 576 bits. Therefore, we can consider the frame errors to be roughly uncorrelated, although there will obviously be edge effects in adjoining frames. In any case, for sizing the run length, only a rough idea is needed. In Chapter 11 we saw that to obtain a reliable estimate, a good rule of thumb for the number of event observations is on the order of 10 times (or more) the reciprocal of the event error rate. Thus, for an FER of 10^{-3} , we need to observe at least 10,000 frames. This gives a confidence interval roughly $(2 \times 10^{-3}, 0.5 \times 10^{-3})$ for about 95% confidence level.

12.4.3.3. Simulation Runs

Simulation runs were performed at a fixed SNR of 6 dB at the input to the receive filter, where the SNR is defined at the unfaded signal level. One to seven interferers were simulated.

The target FER was 10^{-3} , hence 10,000 frames were simulated for each case. This FER was roughly reached with one interferer present. When the number of interferers is larger, performance is degraded, hence the 10,000 frames are more than sufficient. Observe that 10,000 frames corresponds to 200 s of real time. With a Doppler bandwidth of 200 Hz, this represents 40,000 fading cycles, and would therefore appear to be a reasonably statistically significant sample for purposes of estimating the parameters of the discrete channel model. For the latter purpose, however, unlike the FER estimation case, there does not appear to be a simple formulation for determining how many observations we need in order to ascertain the reliability of the derived model.

Error sequences were recorded for one, three, and five interferers, with SNR = 6 dB, and used to infer a discrete HMM model, as described later.

12.4.4. The Forward Radio Link

The forward radio link is the communication channel from the base station to each individual mobile user in a cell. As depicted in Figure 12.50, the reconstituted voice signal is voice-coded to a 9.6-kb/s digital stream, although in some switching arrangements this bit stream may be directly available. This bit stream is then convolutionally encoded with a rate-1/2 code to 19.2 kb/s. For each user, this encoded stream is spread by two “codes,” a Walsh function with period 64 and two PN (de Bruijn) sequences (for orthogonal carriers) with period 2^{15} . Both the Walsh and PN sequences operate at 1.2288 Mc/s. It is important to note that these two sequences play a different role than they do in the reverse link. In the forward link, the base station uses the *same* PN sequences for *all* users in its cell. Each cell uses its own unique PN sequences. These PN sequences, therefore, are not used to isolate same-cell users from one another, but to discriminate against out-of-cell users. On the other hand, for each user within its cell, the base station uses a different (repeating) Walsh function sequence, which essentially nulls out within-cell interference because of the orthogonality of the Walsh sequences. This will be discussed further below.

12.4.4.1. The Simulation Model of the Forward Radio Link

We now describe the various components of the simulation model, namely the transmitter and the receiver. A more detailed block diagram of the reverse link is shown in Figure 12.53. The channel characteristics are taken as identical to those already presented for the reverse channel.

12.4.4.1.1. The Transmitter For present purposes, we can consider the “source” to be the 19.2-kb/s encoded bit stream, which we will label $S_r(t)$. It is initially sampled at 8 samples/bit, resulting in a sample rate $f_{s,i} = 153.6 \text{ ksamples/s}$. This signal is then upsampled and interpolated by a factor of 64 in order to be compatible with the spreading signals to be applied at 1.2288 Mc/s. Observe that because $1.2288 \times 10^6 / 19.2 \times 10^3 = 64$, there is exactly one Walsh sequence period per convolutionally encoded bit. The 64-upsampling also implies that the number of samples per chip is 8, so that the corresponding sampling rate is $f_{s,sys} = 9.8304 \text{ megasamples/s}$. As with the reverse link, the signal is split for transmission over quadrature carriers, and each of the two rails is spread with a different PN sequence. These two rails are fed to a QPSK modulator, filtered by a square-root cosine filter, and transmitted over the channel. Thus, the lowpass equivalent of the transmitted signal for user m

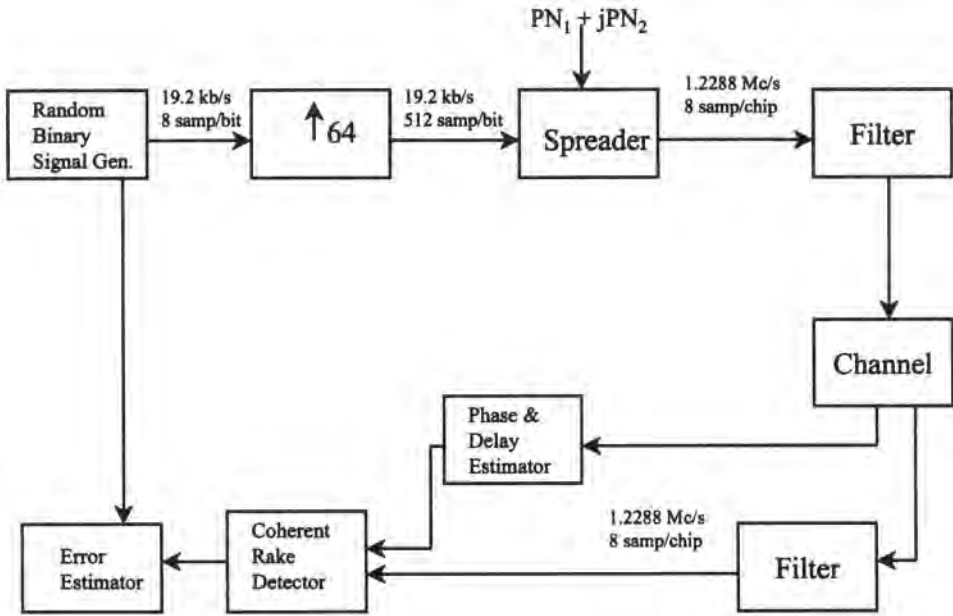


Figure 12.53. Block diagram of the modeled forward radio link.

can be written as

$$S_{T,m}(t) = S_{I,m}(t)W_m(t)[PN_1(t) + jPN_2(t)] * h_T(t) \quad (12.4.8)$$

with appropriate repetition of the Walsh and PN sequences.

12.4.4.1.2. The Receiver Figure 12.54 shows the operations in the receiver. The received signal $S_{R,m}(t)$ for the m th user is the transmitted signal passed through the channel and filtered by the matching receiver filter, with the usual front-end noise added, as well as other user interference:

$$S_{R,m}(t) = [S_{T,m}(t) * \tilde{c}(\tau, t) + I(t) + n(t)] * h_R(t) \quad (12.4.9)$$

The receiver performs *coherent* demodulation in this case, so that carrier phase information is required. In actuality, carrier phase as well as symbol synchronization are estimated from a transmitted pilot common to all users in the cell. In this simulation, the synchronization process is not performed, or rather it is performed ideally from the information that is already known. As before, Rake finger tracking is simply given by the delays τ_i in the channel model, and phase tracking is known from the simulated channel itself, i.e., the sequence of samples of $\theta_i(t)$ for $i = 1, 2, 3$. Thus, synchronization for each Rake finger is performed simply by feeding $-\theta_i(t)$ to the receiver. This is in fact an emodiment of the *hard-wired* synchronization methodology presented in Chapter 8. After bringing to baseband, the signal is then despread by the appropriate delayed sequences. The lowpass equivalent of the demodulated lowpass-equivalent signal for the i th ray of the m th user is then

$$S_D(t, i) = S_{R,m}(t)e^{-j\theta_i(t)}W_m(t - \tau_i)[PN_1(t - \tau_i) + jPN_2(t - \tau_i)], \quad i = 1, 2, 3 \quad (12.4.10)$$

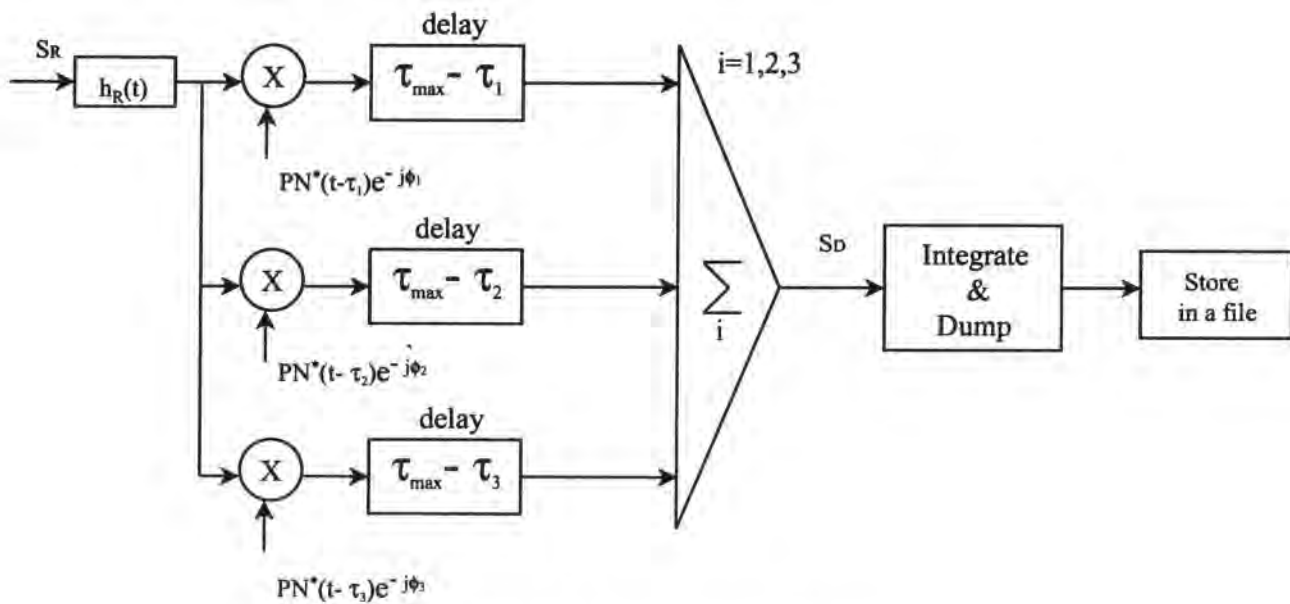


Figure 12.54. Block diagram of a coherent rake detector.

As with the reverse link, each of the signals in (12.4.10) is delayed by $\tau_{\max} - \tau_i$ to mutually align them. These delayed signals are then added to form

$$S_D(t) = \sum_{i=1}^3 S_D(t - (\tau_{\max} - \tau_i), t) \quad (12.4.11)$$

which then becomes the input to the detector, which is an integrate-and-dump (I&D) filter. At the input to the I&D filter the signal is filtered and down sampled by a factor of 64 and the I&D filter performs the final detection on a bit-by-bit basis on the 19.2-kb/s stream.

Earlier we made a point about elimination of in-cell interference because of the orthogonality of the Walsh functions. To elaborate somewhat on this point, observe that for some other user, say user n , the corresponding output of the integrator reduces to

$$\int_0^{T_w} S_{I,n}(t) W_n(t) W_m(t) dt = 0$$

because $S_{I,n}(t) = \pm 1$ over the bit period of the encoded stream, which is identical to the Walsh sequence period T_w . To the extent that there is some distortion, the previous integral will in practice not be quite zero, but still very small.

12.4.4.2. QA Performance Evaluation of the Forward Link in a Bursty Environment

The error rate performance of the forward link is of course affected by noise and interference. Since the receiver is linear, the Gaussian noise at the receiver input is still Gaussian at the I&D filter output. The interference from out-of-cell signals and the residual interference from in-cell signals can be regarded as the sum of many small signals, which can be approximated by a Gaussian process. The accuracy of this approximation is also improved by the fact that this interference is broadband, and the I&D filter (whose bandwidth is on the order of 64 times smaller) will tend to Gaussiannize this interference signal. Hence we will be justified by approximating the sum of the receiver noise and the interference as some equivalent Gaussian noise. The associated output of the I&D filter is thus a Gaussian-distributed voltage, which does not have to be explicitly generated. Instead, we can simply simulate without noise and interference, and inject a properly scaled Gaussian random number at the I&D filter output to simulate their presence. This is simply a form of the quasianalytical error evaluation methodology presented in Chapter 11.

To be specific, the error generation mechanism is shown in Figure 12.55. The signal presented to the decision device for the k th symbol, at sampling epoch τ , is given by

$$\hat{V}_R(k, \tau) = V_R(k, \tau) + V_n(k, \tau) \quad (12.4.12)$$

where $V_R(k, \tau)$ is the noiseless analog voltage at the I&D filter output for the k th symbol, at sampling epoch τ ; and $V_n, I(k\tau)$ is the output of the Gaussian random number generator for the k th symbol, at sampling epoch τ .

The voltage $\hat{V}_R(k, \tau)$ is compared to the transmitted sequence, $V_T(k)$, and an error sequence $\mathbf{e}$ is generated, where $e_k = 1$ if an error is detected, and $e_k = 0$ if no error occurs. The error sequence $\mathbf{e}$ can be partitioned, as before, into frame-length segments to compute a frame error rate, and it also serves as the basis for extracting a discrete channel model.

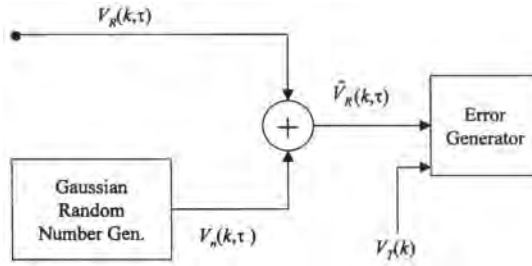


Figure 12.55. Block diagram of the direct error sequence generator.

12.4.4.3. Simulation Runs

The same kinds of considerations arise as were discussed for the reverse link in terms of the required number of frames for a given confidence in the FER. We simulated the system for values of E_b/N_0 of 4, 5, 6, 7, and 8 dB. The estimated FER is about 10^{-3} for $E_b/N_0 = 8$ dB, and the run length of 10,000 frames again provides adequate reliability in the results, all the more so for lower E_b/N_0 since the error rate is then greater.

Because the average bit error rate is relatively small, it is convenient to develop a shorthand notation for the error sequence, rather than record the actual sequence of 1's and 0's, most of which will be 0's. One concise representation is an interval representation (see Section 9.4), where the numbers in the sequence represent an error-free run between errors. For example, the error sequence 0010000000110001 is represented by (2, 7, 0, 3). An alternative representation was also presented in Chapter 11, whereby the error sequence segment just given would be summarized by $0^210^71^20^31$. In any case, these error sequences are used for finite-state channel modeling using an HMM, as described next.

12.4.5. Finite-State Channel Characterization

To obtain a finite-state channel model for both the reverse and forward cellular radio systems, we apply methods described in Section 9.4.

According to our model, the discrete channel is symmetric (zero and one bits are affected the same way), and therefore it can be described by a sequence of bit errors. The simulated sequence of bit errors is bursty and we model it with an hidden Markov model (HMM). For the binary channel an HMM is defined by the 4-tuple $(\mathbf{S}, \mathbf{\Pi}, \mathbf{A}, \mathbf{B})$, where $\mathbf{S} = \{s_1, s_2, \dots, s_n\}$ is the set of Markov chain states, $\mathbf{\Pi}$ is a vector of state initial probabilities, $\mathbf{A} = [a_{ij}]_{n,n}$ is a matrix of state transition probabilities ($a_{ij} = \Pr\{s_j | s_i\}$), and $\mathbf{B} = [b_{ik}]_{n,m}$ is a matrix of input-to-output symbol transition (i.e., error) probabilities [see (9.96)].

Recall from the definition of $\mathbf{B}$ in Chapter 9 that $b_{ik} = b_i(\mathbf{e}_k)$ is the probability of observing error symbol $\mathbf{e}_k$ given that the system is in state i . In the present case only two error symbols are recognized — $k = 1: \mathbf{e}_1 = 0 \rightarrow$ no error, and $k = 2: \mathbf{e}_2 = 1 \rightarrow$ error — so that the observed sequence $\mathbf{e}$ consists of only 1's and 0's, as already noted. (More general outcome spaces can be defined; see also Chapter 11.)

As an example, if $n = 2$, the HMM is called the Elliott-Gilbert^(34,35) model, which can be interpreted as a channel with two states: G (for “good”) and B (for “bad”). The model state transition probability matrix has the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (12.4.13)$$

In the good state (state 1), errors occur with small probability b_{12} , while in the bad state (state 2), the bit error probability b_{22} is larger. The appropriate probabilities for no errors occurring are $b_{11} = 1 - b_{12}$ for state 1, and $b_{21} = 1 - b_{22}$ for state 2. Thus we have

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \quad (12.4.14)$$

Because $b_{11} = 1 - b_{12}$, the matrix $\mathbf{B}$ is completely characterized by the conditional probability vector $\mathbf{b}_1 = (b_{12}, b_{22})$. The model state transition diagram is shown in Figure 12.56.

Consider another example; for $n = 3$, the HMM represents a three-state model. The model state transition probability matrix has the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (12.4.15)$$

In state 1, errors occur with probability b_{12} , in state 2 the bit error probability is b_{22} , and in state 3 the bit error probability is b_{32} . Thus we have

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \quad (12.4.16)$$

The model state transition diagram is shown in Figure 12.57.

12.4.5.1. HMM Parameter Estimation

The Markov model for a discrete channel is described by the state transition matrix $\mathbf{A}$ and by the error probability matrix $\mathbf{B}$. An iterative procedure for estimating these parameters

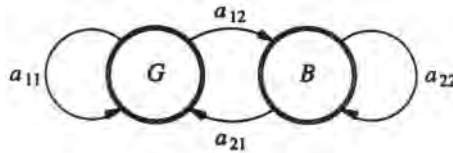


Figure 12.56. Elliot-Gilbert model state diagram.

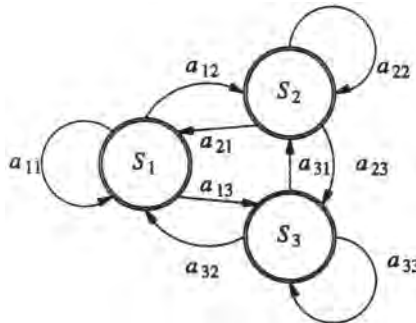


Figure 12.57. Model for a Markov chain three-state diagram.

from a given set of simulated or measured error sequences was developed by Baum and Welch⁽³⁶⁾ and is a special case of the EM algorithm.^(37,38) This iterative procedure described in Section 9.6 is intended to converge to the maximum-likelihood estimate of the model $(\Pi, \mathbf{A}, \mathbf{B})$ given the number of states and the observation sequence.

12.4.5.2. Discrete Channel Modeling

We describe below the results of fitting HMMs using the Baum–Welch algorithm to the error sequences produced by the waveform-level simulations for both the reverse and forward links of the CDMA cellular radio system.

We experimented with two discrete channel models, the two-state Gilbert model (shown in Figure 12.56) and a three-state model (shown in Figure 12.57).

12.4.5.2.1. The Reverse Link The reverse radio link system was simulated (Section 12.4.3) using the waveform-level simulator with variable amounts of interference applied to the input of the receiver. In addition, AWGN noise with SNR = 6 dB was also applied at the input of the receiver. We analyzed the impact of a variable amount of cochannel interference on the model parameters.

After applying the Baum–Welch algorithm to a two-state model, we obtained the following model parameters for the discrete channel with one, three, and five interferers:

$$\begin{aligned}
 &\text{One interferer} \\
 \Pi &= \begin{bmatrix} 0.996393 \\ 0.003607 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.999537 & 0.127852 \\ 0.000463 & 0.872148 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.574161 & 0.425839 \end{bmatrix} \\
 &\text{Three interferers} \\
 \Pi &= \begin{bmatrix} 0.984419 \\ 0.015581 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.998537 & 0.092446 \\ 0.001463 & 0.907554 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.539846 & 0.460154 \end{bmatrix} \\
 &\text{Five interferers} \\
 \Pi &= \begin{bmatrix} 0.967388 \\ 0.032612 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.995998 & 0.118733 \\ 0.004002 & 0.881267 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.503516 & 0.496484 \end{bmatrix}
 \end{aligned}$$

For the three-state model we obtained the following results

$$\begin{aligned}
 &\text{One interferer} \\
 \Pi &= \begin{bmatrix} 0.996582 \\ 0.001168 \\ 0.002250 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.999579 & 0.000000 & 0.186320 \\ 0.000421 & 0.641147 & 0.000000 \\ 0.000000 & 0.358853 & 0.813680 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.440941 & 0.559059 \\ 0.607305 & 0.392695 \end{bmatrix} \\
 &\text{Three interferers} \\
 \Pi &= \begin{bmatrix} 0.984527 \\ 0.005865 \\ 0.009608 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.998645 & 0.000000 & 0.138841 \\ 0.001354 & 0.772499 & 0.000078 \\ 0.000000 & 0.227501 & 0.861081 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.491358 & 0.508642 \\ 0.564860 & 0.435140 \end{bmatrix} \\
 &\text{Five interferers} \\
 \Pi &= \begin{bmatrix} 0.953933 \\ 0.025066 \\ 0.021000 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.996738 & 0.122975 & 0.001383 \\ 0.001102 & 0.800820 & 0.187699 \\ 0.002160 & 0.076206 & 0.810917 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.384690 & 0.615310 \\ 0.963458 & 0.036542 \end{bmatrix}
 \end{aligned}$$

12.4.5.2.2. *The Forward Link* The result of fitting HMMs to the results of the waveform-level simulation of the forward link yields the following model parameters:

$$\begin{aligned}
 & \text{4 dB} \\
 \mathbf{\Pi} &= \begin{bmatrix} 0.985002 \\ 0.014998 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.996330 & 0.240474 \\ 0.003670 & 0.759526 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.782483 & 0.217517 \end{bmatrix} \\
 & \text{6 dB} \\
 \mathbf{\Pi} &= \begin{bmatrix} 0.991005 \\ 0.008995 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.997357 & 0.290565 \\ 0.002643 & 0.709435 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.779764 & 0.220236 \end{bmatrix} \\
 & \text{8 dB} \\
 \mathbf{\Pi} &= \begin{bmatrix} 0.997205 \\ 0.002795 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.997205 & 0.287374 \\ 0.002795 & 0.712626 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1.000000 & 0.000000 \\ 0.831187 & 0.168813 \end{bmatrix}
 \end{aligned}$$

For the three-state model we obtained the following results:

$$\begin{aligned}
 & \text{4 dB} \\
 \mathbf{\Pi} &= \begin{bmatrix} 0.987199 \\ 0.003649 \\ 0.009152 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.997031 & 0.334190 & 0.186705 \\ 0.001368 & 0.585774 & 0.018046 \\ 0.001601 & 0.080035 & 0.795249 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0.999794 & 0.000206 \\ 0.647589 & 0.352411 \\ 0.810841 & 0.189159 \end{bmatrix} \\
 & \text{6 dB} \\
 \mathbf{\Pi} &= \begin{bmatrix} 0.993116 \\ 0.002284 \\ 0.004600 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.998045 & 0.426803 & 0.212247 \\ 0.001047 & 0.521940 & 0.010233 \\ 0.000908 & 0.051257 & 0.777520 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0.999799 & 0.000201 \\ 0.626481 & 0.373519 \\ 0.792298 & 0.207702 \end{bmatrix} \\
 & \text{8 dB} \\
 \mathbf{\Pi} &= \begin{bmatrix} 0.994031 \\ 0.002085 \\ 0.003884 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 0.998206 & 0.458940 & 0.209489 \\ 0.001032 & 0.487307 & 0.012437 \\ 0.000761 & 0.053753 & 0.778074 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0.999849 & 0.000151 \\ 0.634515 & 0.365485 \\ 0.806539 & 0.193461 \end{bmatrix}
 \end{aligned}$$

12.4.5.3. *The Number of States*

The number of states of the Markov chain is an important parameter of the HMM. There are several papers that consider the estimation of the number of states.^(39,40) The problem of selecting the number of states is a special case of the problem of the parametric complexity of a statistical model. This problem is usually solved using confidence limits.

On the other hand, the number of states can be judiciously selected depending on the model application. A repeated application of the fitting process by incrementing the number of states by one will generally converge fairly rapidly to an apparent “correct” number of states. In the following sections we demonstrate that a two-state Gilbert model is quite adequate for calculating probability distributions of intervals between consecutive errors and probability distributions of the number of errors in a given interval.

12.4.5.4. Probability Distribution of Error-Free Intervals

It can be shown⁽³⁸⁾ that for the HMM the error-free intervals l have a matrix-geometric cumulative probability distribution given by

$$\frac{\Pi P(1)[I - P'(0)]1}{\Pi P(1)1} \quad (12.4.17)$$

where we define

$$P(e) = AC(e), \quad e = 0, 1$$

and $C(e)$ is an $n \times n$ diagonal matrix whose nonzero elements are the columns of the matrix B . For example, if $e = 0$, the diagonal entries in $C(0)$ are $(b_{11}, b_{21}, \dots, b_{n1})$ and for $e = 1$, the diagonal entries in $C(1)$ are $(b_{12}, b_{22}, \dots, b_{n2})$.

The results of computing this probability for the two-state and three-state models are compared with the results of the waveform-level simulation of the reverse link in Figures 12.58–12.60. We also plot in these figures for comparison the results obtained for the model with independent errors (channel without memory). As we can see, the model with two states is quite adequate.

12.4.5.5. Probability Distribution of the Number of Errors in a Block

The probability distribution $P(m, n)$ of the number of errors m in a block of length n is a matrix-binomial distribution⁽³⁸⁾ which can be obtained recursively by the following forward algorithm:

$$\begin{aligned} p_1(0) &= \Pi P(0), & p_1(1) &= \Pi P(1) \\ p_n(m) &= 0 & \text{for } m < 0 \text{ or } m > n \\ p_n(m) &= p_{n-1}(m)P(0) + p_{n-1}(m-1)P(1) \\ P(m, n) &= p_n(m)1 & \text{and } 1 &= [1, 1, \dots, 1]^T \end{aligned} \quad (12.4.18)$$

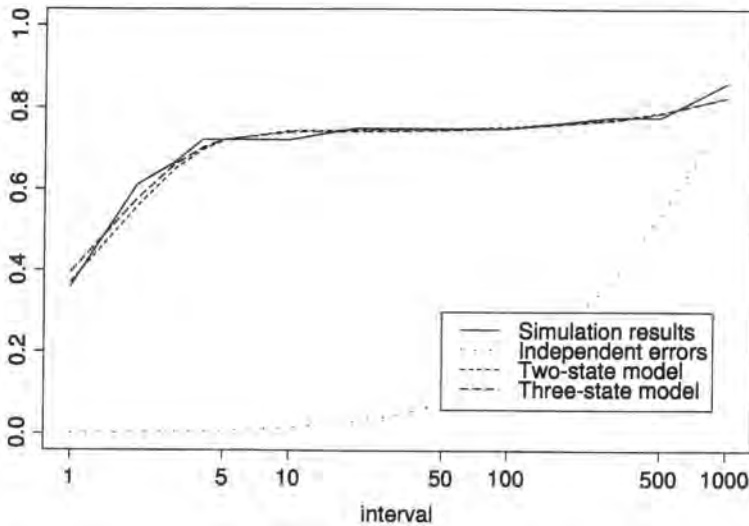


Figure 12.58. Cumulative probabilities of the error-free intervals for one interferer in the reverse link.

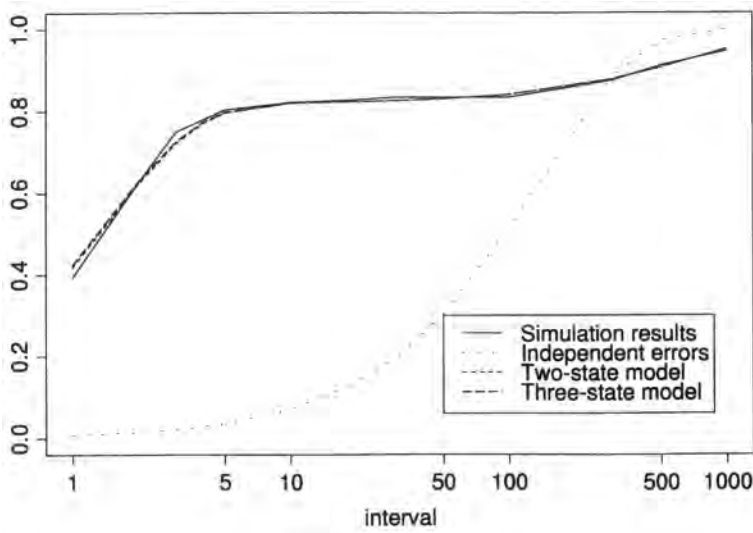


Figure 12.59. Cumulative probabilities of the error-free intervals for three interferers in the reverse link.

The form (12.4.18) reduced for the Gilbert model was given in (11.2.108)–(11.2.110). The cumulative distribution corresponding to this matrix-binomial distribution for the two-state and three-state models are compared with the results of the waveform-level simulation in Figures 12.61–12.63 for the reverse link and in Figures 12.64–12.66 for the forward link. We also plot in these figures for comparison the results obtained for the model with independent errors (channel without memory). As we can see, the model with two states is quite adequate.

12.4.6. Conclusion

Waveform-level simulation of a discrete channel is computationally expensive and is impractical to use for evaluating the performance of digital devices (such as error correction/detection coders, scramblers, interleavers, etc). However, waveform-level simulation can

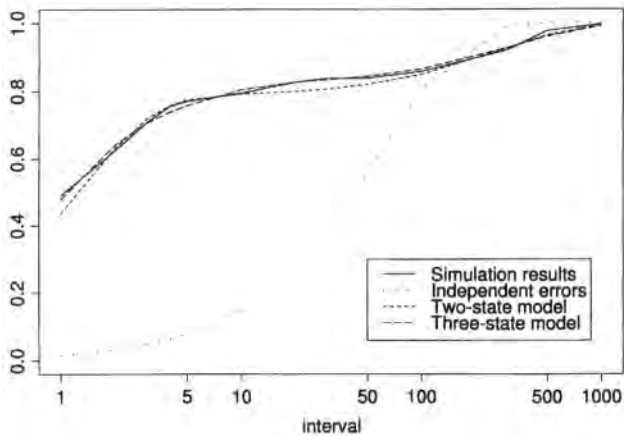


Figure 12.60. Cumulative probabilities of the error-free intervals for five interferers in the reverse link.

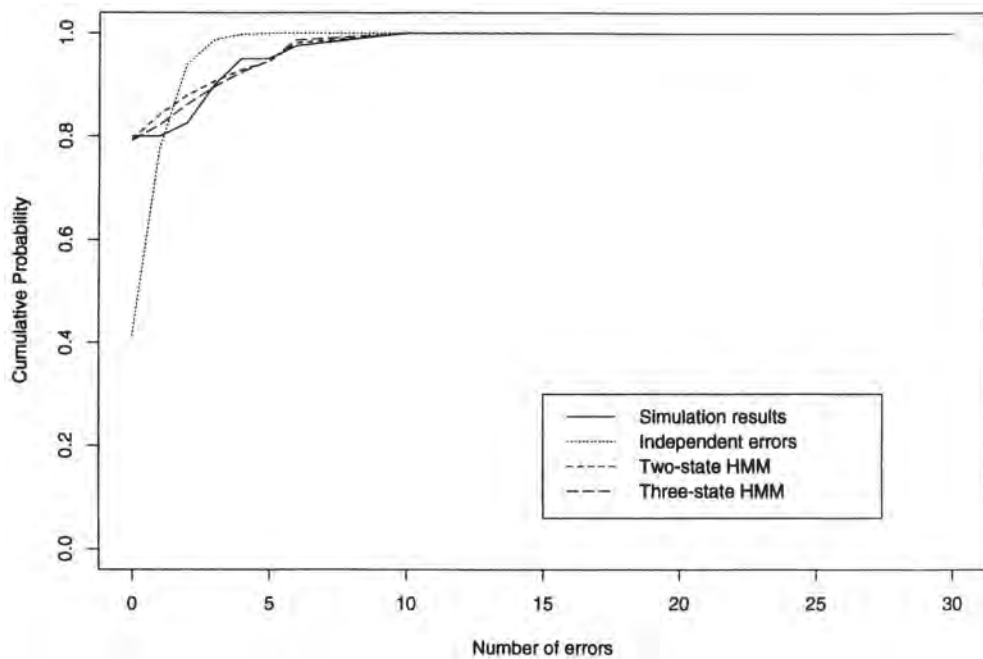


Figure 12.61. Cumulative probabilities of a number of errors in a block of 576 bits, one interferer in the reverse link.

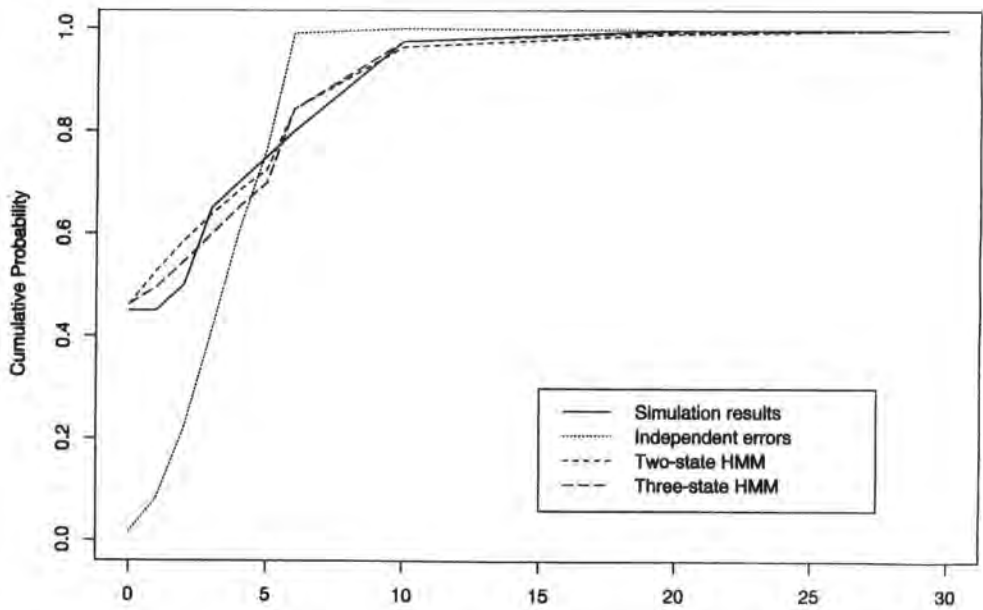


Figure 12.62. Cumulative probabilities of a number of errors in a block of 576 bits, three interferers in the reverse link.

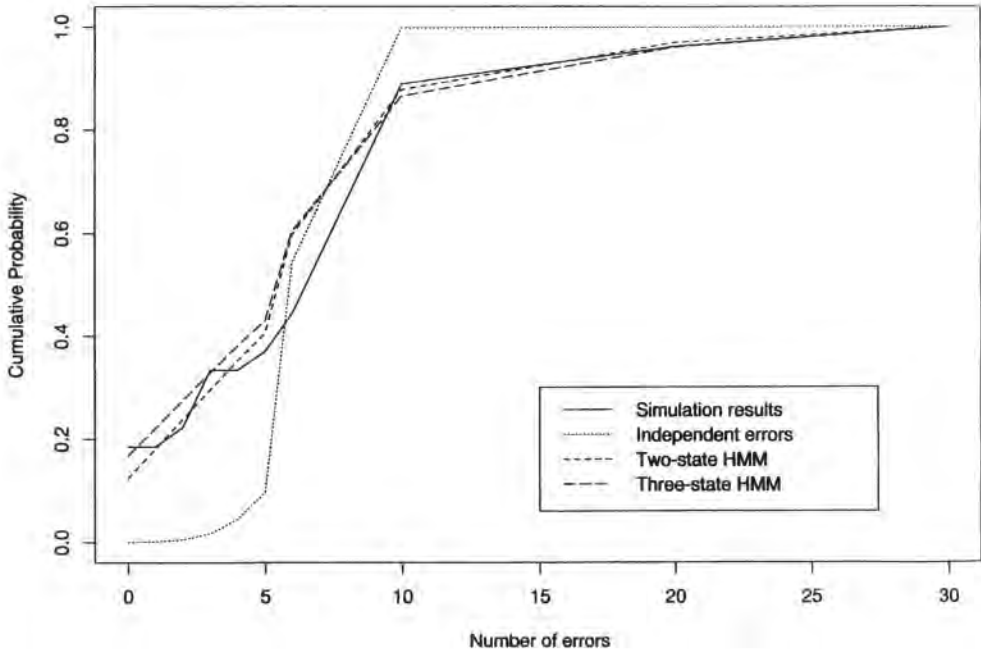


Figure 12.63. Cumulative probabilities of a number of errors in a block of 576 bits, five interferers in the reverse link.

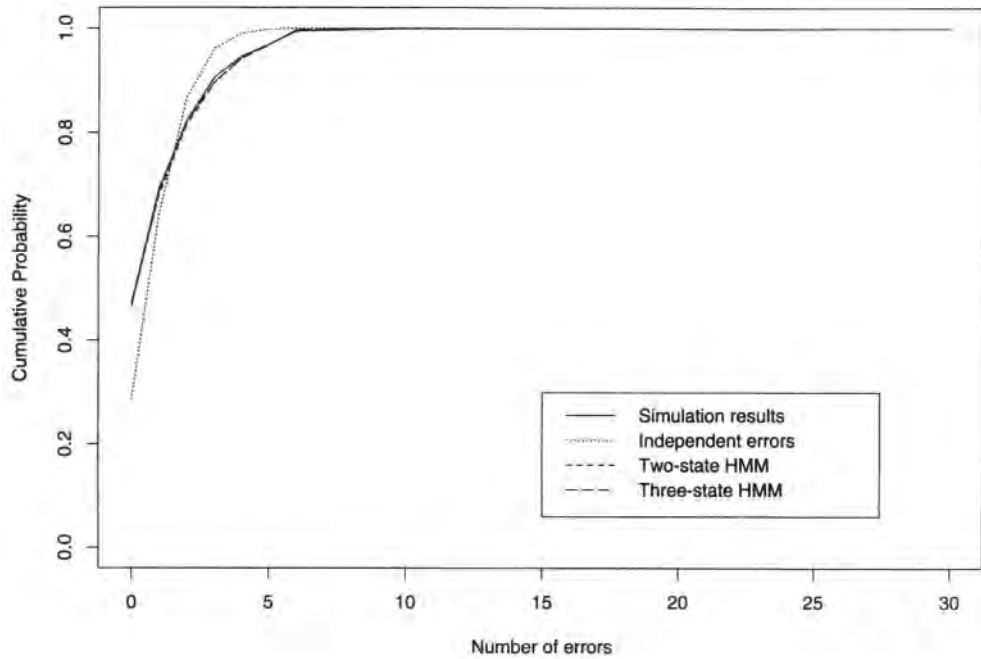


Figure 12.64. Cumulative probabilities of a number of errors in a block of 576 bits, with $E_b/N_0 = 4$ dB in the forward link.

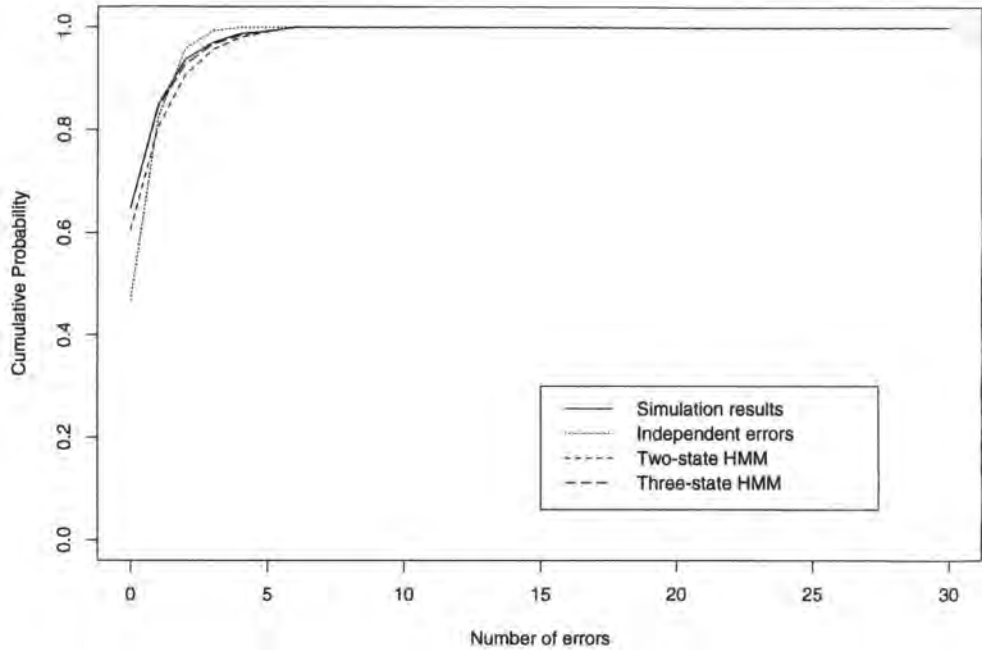


Figure 12.65. Cumulative probabilities of a number of errors in a block of 576 bits, with $E_b/N_0 = 6$ dB in the forward link.

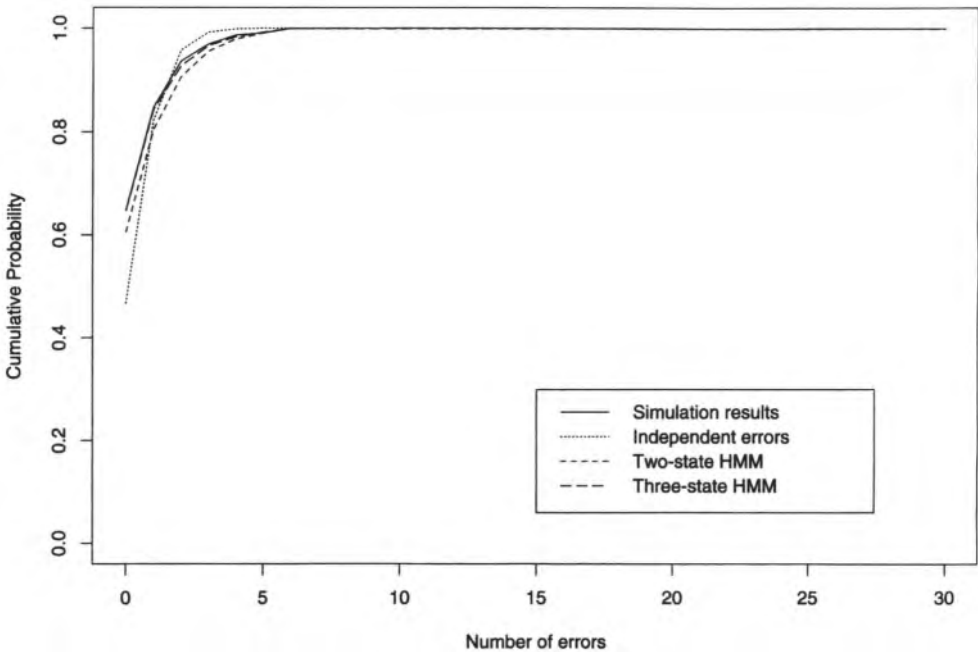


Figure 12.66. Cumulative probabilities of a number of errors in a block of 576 bits, with $E_b/N_0 = 8$ dB in the forward link.

be used to generate enough data for fitting a finite-state channel model. Since HMMs are general enough to approximate a large variety of random processes, they are good candidates for modeling the finite-state channel. In this case study, we have demonstrated that for the communication system described, the finite-state channel can be modeled successfully with the Gilbert model, which is a special case of the HMM.

12.5. Appendix: Simulation of the Tap-Gain Functions for a Rayleigh Fading Channel

12A.1. Introduction

A block diagram of a simulation model of a tap-gain function generated for a Rayleigh fading mobile channel is shown in Figure 12.A.1. A complex Gaussian signal with a flat spectrum $P_f(f)$ is generated by the signal generator. This signal is shaped by a filter with a frequency response $H_j(f)$ to have the Jakes power spectrum $P_j(f)$ and a normalized total power.

The bandwidth of the tap-gain function, which is determined by the maximum Doppler frequency f_D , is very narrow as compared with the bandwidth of the transmitted signal. Therefore, to increase accuracy and decrease the computational load, the tap-gain function is sampled at a much lower rate than the sampling rate of the system signal. The signal sampling rate of the tap-gain function is then expanded by a factor L and interpolated to be compatible with the system signal sampling rate.

To reduce computational effort, the weight of the tap-gain σ_t can be applied before the sampling rate expansion.

12A.2. Estimation of the Sampling Rates and Expansion Ratios

The CDMA signal chip rate is $f_c = 1.2288 \text{ Mc/s}$, assuming a sampling rate of 8 samples/chip; the system sampling rate is $f_{s,\text{sys}} = 9.8304 \times 10^6 \text{ samples/s}$.

The maximum Doppler frequency is computed from the velocity of the channel

$$f_d = f_0 \frac{v}{c} \quad (12.A.1)$$

where v is the velocity of the vehicle (m/s), c is the speed of light ($3.08 \times 10^8 \text{ m/s}$), f_0 is the carrier frequency, and f_D is the maximum Doppler frequency.

For a vehicle moving at $v = 30 \text{ m/s} = 108 \text{ km/hr}$ and a carrier frequency $f_0 = 2.0 \text{ GHz}$ the maximum Doppler frequency is $f_d = 200 \text{ Hz}$. A good rule of thumb is to use between 6 and 12 samples/period of the highest frequency of an analog signal. Thus, we approximate

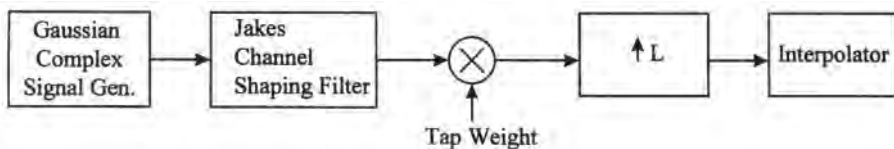


Figure 12.A.1. Block diagram of a single-ray Rayleigh channel model.

the channel sampling rate generated by the Gaussian signal generator, $f_{s,t} = 10f_d = 2000$ samples/s. The expansion factor is then an integer close to the ratio of the sampling rates,

$$L = \left\lfloor \frac{f_{s,sys}}{f_{s,t}} \right\rfloor = 4915 \quad (12.A.2)$$

where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x .

The tap-gain signal sampling frequency is then recalculated to be

$$f_{s,t} = \frac{f_{s,sys}}{L} = 2000.0814 \quad (12.A.3a)$$

and the sampling period of the channel signal is

$$T_{s,t} = \frac{1}{f_{s,t}} = 5.0 \times 10^{-4} \quad (12.A.3b)$$

12A.3. The Channel Shaping Filter

The impulse response of the channel shaping filter with the Jakes frequency response was shown in Section 9.1.3.5.3 and is repeated below for convenience:

$$h_j(t) = C f_d x^{-1/4} J_{1/4}(x) \quad (12.A.4)$$

where $J_{1/4}(\cdot)$ is the fractional Bessel function and $x = 2\pi f_d |t|$. The normalized impulse response $h_j(t)/C f_d$ is shown in Figure 12.A.2. Equation (12.A.4) can be well approximated by an FIR filter.

12A.4. FIR Implementation

The FIR shaping filter $h_j(n)$ is implemented by truncating and discretizing the impulse response $h_j(t)$. Assume we want to truncate the tails of $|h_j(t)|$ so that they are 20 dB below the maximum value, which implies (from Figure 12.A.2) that the truncation points are $x_{t1} = -20$ and $x_{t2} = 20$. The time span of the truncated filter is then $T = (x_{t2} - x_{t1})/(2\pi f_d) = 3.18 \times 10^{-2}$. Thus the FIR filter has $M = \lfloor T/T_{s,t} \rfloor = 64$ taps. To make the filter causal, we introduce a delay $T/2$ that corresponds to a linear phase in the frequency response $\exp(-j\pi f T)$,

$$h_j(n) = h \left[n \frac{T}{M} - \frac{T}{2} \right] \quad (12.A.5)$$

The output of the shaping filter is

$$y_j(n) = \sum_{m=0}^{M-1} h_j(m) x_G(n-m) \quad (12.A.6)$$

The computational effort is 64 multiplications per tap-gain function signal sample n .

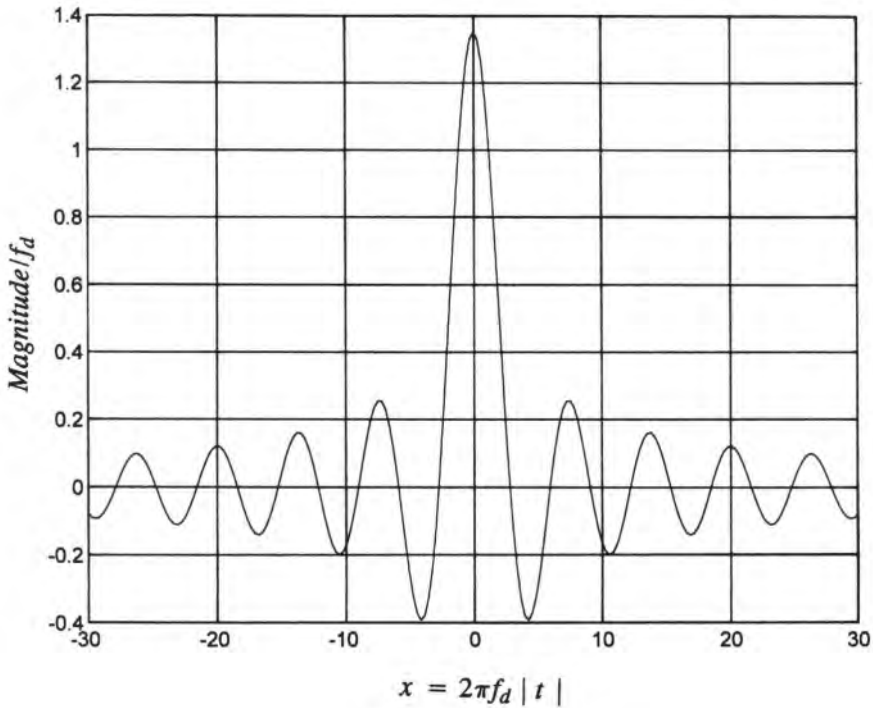


Figure 12.A.2. Impulse response of the Jakes fitter.

To reduce the Gibbs phenomenon distortion due to truncation, the shaping filter should be windowed using, e.g., a Hamming window. The impulse response of the FIR shaping filter is then

$$h_{fw}(n) = h_j(n)w_H(n) \quad (12.A.7)$$

where the Hamming window is

$$w_H(n) = 0.54 + 0.46 \frac{\cos\{2\pi[n - (M-1)/2]\}}{M}, \quad 0 \leq n \leq M-1 \quad (12.A.8)$$

12A.5. IIR Implementation

An alternative implementation is to use an approximating IIR filter. One possibility might be an eighth-order elliptic filter that approximates the Jakes tap-gain function filter response. If the filter is implemented by four biquadratic sections, the number of multiplications per tap-gain function signal sample is 16.

12A.6. Comparison of IIR and FIR Filter Implementation

The FIR filter implementation requires more multiplications (64) than the IIR (16). However, since the sampling rate of the tap-gain function signal is very low compared with the system sampling rate, this is insignificant.

The magnitude of the IIR filter can be well matched to the Jakes response, but the IIR filter cannot be designed to have a linear phase. An FIR filter can be designed with a perfect linear phase. The significance of that fact is not known.

12.A7. Sampling Rate Expansion and Interpolation

Since the sampling rate expansion is very large ($L = 4915$), we use a bandlimited (sinc) interpolation with a polyphase implementation as described in Section 3.5. The sinc function is truncated at $x = \pm 8$, and windowed using a Hamming window. All the coefficients of the filters, N for each of the L filters, were precomputed before the simulation.

It is also possible to use more efficient interpolation techniques. One possibility is a hybrid interpolation that combines both a sinc and linear interpolation. The sinc interpolation would be used to compute only a few samples, and for the remaining samples we can use a linear interpolation. The linear interpolation is adequate if the distance between the low-rate samples is not too large. This combination combines the accuracy of the sinc interpolator with efficiency of the linear interpolator.

References

1. A. J. Viterbi, *Principles of Coherent Communication*, McGraw-Hill, New York (1966).
2. D. P. Stapor and P. Hayes, On the number of symbols processed in quasianalytic communications link simulation, Presented at 2nd IEEE International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks, University of Massachusetts, Amherst, Massachusetts, October 12–14, 1988.
3. A. H. Stroud and D. Secrest, *Gaussian Quadrature Formulas*, Prentice-Hall, Englewood Cliffs, New Jersey (1966).
4. G. Evans, *Practical Numerical Integration*, Wiley, New York (1993).
5. W. Gautschi, *Numerical Analysis: An Introduction*, Birkhäuser, Boston (1997).
6. P. Balaban, H. P. Corrales, and V. K. Prabhu, Statistical performance estimation of digital radio over fading channels, in *Conference Record of the IEEE International Conference on Communications ICC '91*, Denver, Colorado (1991).
7. W. T. Barnett, Multipath propagation at 4, 6, and 11 GHz, *Bell Syst. Tech. J.* **51**(2), 321–361 (1972).
8. W. C. Lindsey, Phase-shift keyed signal detection with noise reference signals, *IEEE Trans. Aerospace Electron. Syst.* **AES-2**(4), 393–401 (1966).
9. D. Divsalar and M. K. Simon, The use of interleaving for reducing noisy reference loss in trellis-coded modulation systems, *IEEE Trans. Commun.* **38**(8), 1190–1198 (1990).
10. E. Zehavi and G. Kaplan, Phase noise effects on M -ary PSK trellis codes, *IEEE Trans. Commun.* **39**(3), 373–379 (1991).
11. R. T. Ray, M. C. Jeruchim, and D. Hatzipapafotiou, The analysis of phase noise degradation in a coded communication link, in *Proc. MILCOM*, Monterey, California (1990).
12. T. J. Kolze, S. T. McBride, and H. L. Berger, Phase noise effects on coding gain in forward error control coded communications, in *AIAA 16th International Communications Satellite Systems Conference*, Washington, D.C. (1996), pp. 277–286.
13. J. R. Alexovich and R. M. Gagliardi, The effect of phase noise on noncoherent digital communications, *IEEE Trans. Commun.* **38**(9), 1539–1548 (1990).
14. M. C. Jeruchim, R. J. Wolfe, C. Garthwaite, and S. McGlynn, A computer-aided analysis of the effect of phase noise in interleaved block-coded communication links, *IEEE J. Select. Areas Commun.* **11**(3), 465–473 (1993).
15. J. Rutman, Characterization of phase and frequency instabilities in precision frequency sources: Fifteen years of progress, *Proc. IEEE* **66**(9), 1048–1074 (1978).
16. J. Rutman and F. L. Walls, Characterization of frequency stability in precision frequency sources, *Proc. IEEE* **79**(6), 952–960 (1991).
17. W. P. Robins, *Phase Noise in Signal Sources (Theory and Applications)*, Peregrinus, London (1982).
18. H. Meyr and G. Ascheid, *Synchronization in Digital Communications*, Vol. 1, Wiley, New York (1990).

19. R. M. Gagliardi, *Introduction to Communications Engineering*, 2nd ed., Wiley, New York (1988).
20. N. J. Kasdin, Discrete simulation of colored noise and stochastic processes and $1/f^\alpha$ power law noise generation, *Proc. IEEE* **83**(5), 802–827 (1995).
21. R. S. Blum, A simple model for fractional non-Gaussian processes, in *Proc. IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, Philadelphia (1994).
22. B. Friedlander and M. Porat, The modified Yule–Walker method of ARMA spectral estimation, *IEEE Trans. Aerospace & Electron. Syst.* **AES-20**(2), 158–173 (1984).
23. G. Maskarinec and D. Castor, Linear and nonlinear distortion effects on MSK modulated signals, in *Conference Proceedings, IEEE MILCOM 96*, McLean, Virginia (1996), Vol. 1, pp. 308–315.
24. D. Morais and K. Feher, The effects of filtering and limiting on the performance of QPSK, offset QPSK, and MSK systems, *IEEE Trans. Commun.* **COM-28**(12), 1999–2009 (1980).
25. H. Wheeler, The interpretation of amplitude and phase distortion in terms of paired echoes, *Proc. IRE* **40**(6), 359–385 (1939).
26. ANSI J-STD-008, Personal Station–Base Station Compatibility Requirements for 1.8 to 2.0 GHz Code Division Multiple Access (CDMA) Personal Communications Systems (March 1995).
27. K. S. Gilhousen *et al.*, On the capacity of cellular CDMA systems, *IEEE Trans. Vehic. Technol.* **40**(2) 303–312 (1991).
28. A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, Addison-Wesley, Reading, Massachusetts (1995).
29. S. Sivaprakasam and K. S. Shanmugan, An equivalent Markov model for burst errors in digital channels, *IEEE Trans. Commun.* **43**, 1347–1355 (1995).
30. P. Balaban, D. Li, and W. Turin, Performance evaluation of the reverse link of a CDMA cellular system using simulation, Presented at GLOBECOM'98, Sydney, Australia.
31. A. Polydoros and S. Glisic, Code synchronization: A review of principles and techniques, in *Code Division Multiple Access Communications*, S. Glisic and P. Leppanen (eds.), Kluwer Academic, Boston (1995).
32. R. Price and P. E. Green, A communication technique for multipath channels, *Proc. IRE* **46**, 555–570 (1958).
33. J. G. Proakis, *“Digital Communications”*, 2nd ed., McGraw-Hill, New York (1989).
34. E. O. Elliott, A model for the switched telephone network for data communications, *Bell. Syst. Tech. J.* **44**, 89–119 (1965).
35. E. N. Gilbert, Capacity of a burst-noise channel, *Bell Syst. Tech. J.* **39**, 1253–1266 (1960).
36. L. E. Baum, T. Petrie, G. Soules, and N. Weiss, A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *Ann. Math. Stat.* **41**, 164–171 (1970).
37. A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. R. Stat. Soc.* **76**, 341–353 (1977).
38. W. Turin, *Digital Transmission Systems: Performance Analysis and Modeling*, McGraw-Hill, New York (1998).
39. J. Ziv and N. Merhav, Estimating the number of states of a finite source, *IEEE Trans. Inf. Theory* **IT-38**(1), 61–65 (1992).
40. J. Rissanen, Complexity of strings in the class of Markov sources, *IEEE Trans. Inf. Theory* **IT-32**(4), 526–532 (1986).

This page intentionally left blank

Problems and Projects

The following problems and projects are intended to extend understanding in various directions, for example, by proving results in the text, or by applying techniques that are discussed to specific situations. In some of the projects you are asked to “write” a program or a simulation. This can be interpreted as pseudocode, an algorithm, a flowchart, executable code in some language, or a “script” in some applications package, as may be appropriate to the case at hand. Some of the projects are suitable as term projects and could be team efforts. While individual problems are associated with particular chapters, many involve material that spans the subject matter of several chapters.

Chapter 3

P3.1. Lowpass Equivalent. Write a program to implement a Hilbert transform (a) in the time domain, (b) in the frequency domain.

P3.2. Lowpass Equivalent for SSB. The Hilbert transform is often used in describing single-sideband modulation (SSB). Show a lowpass equivalent implementation of such a scheme using the Hilbert transform of problem P3.1.

P3.3. Complex Envelope. Let $s(t)$ be a bandpass signal and $h(t)$ a bandpass filter, both more or less arbitrary. Starting from the convolution integral for the output, $e(t) = s(t) * h(t)$, show that the formulation for the output in terms of lowpass equivalents derived from the Hilbert transform provides the strictly correct solution (i.e., does not depend on bandlimitedness).

P3.4. Complex Envelope. As in Problem P3.3, let $s(t)$ be a bandpass signal and $h(t)$ a bandpass filter. The bandpass output is often computed using $\tilde{s}(t)$ as the input signal, defined through $\tilde{s}(t) = \text{Re}\{\tilde{s}(t)e^{j2\pi f_c t}\}$. Starting from the convolution integral for the output, $e(t) = s(t) * h(t)$, show that the error in doing so vanishes if $s(t)$ is strictly bandlimited, and obtain thereby integral expressions for the error in this procedure if this is not the case.

P3.5. Sampling. Given a rectangular pulse $x(t) = p_{T/2}(t)$ sampled at a rate $f_s > 1/T$, (a) compute the ratio of the aliased power to the signal power within the band $|f| \leq 1/T$; (b) repeat part (a) for a triangular pulse of the same duration.

P3.6. Prefiltering. It is common to prefilter a signal such as the pulse in problem P3.5 to reduce aliasing. How would you implement such a filter? Discuss the sampling rate

and the bandwidth for the prefilter, and the impact of these parameters on the aliasing. Discuss the tradeoff between aliasing and distortion introduced by prefiltering.

P3.7. Sampling Rate. Let $\tilde{y}(t) = e^{j\beta x(t)}$, where $x(t)$ is bandlimited to $|f| \leq B/2$. Determine the proper sampling rate (“small” aliasing error) in terms of B as a function of β . Consider (a) a sinusoid of frequency $B/2$, (b) two sinusoids, the highest frequency being $B/2$, (c) a bandlimited Gaussian random process.

P3.8. Aliasing. Suppose $x(t)$ is bandlimited with one-sided bandwidth B_1 , and a filter has bandlimited transfer function B_2 . The output of the filter will contain no aliasing error if $f_s \geq 2B$, where $B = \max(B_1, B_2)$. Show that a less stringent condition to produce no aliasing is $f_s \geq (B_1 + B_2)$.

P3.9. Aliasing Error. Suppose $x(t)$ is bandlimited to $|f| \leq B$, so that that it can be represented by the usual expansion in terms of sinc functions. Let $x_N(t)$ be a truncated version

$$x_N(t) = \sum_{-N}^N x\left(\frac{n}{2B}\right) \text{sinc}(2Bt - n)$$

Show that the *magnitude* of the error is given by

$$|x(t) - x_N(t)| \leq \sqrt{2Be_N}$$

where

$$e_N = \frac{1}{2B} \sum_{|n| > N} |x^2(n/2B)|$$

P3.10. Combined Error Due to Truncation, Aliasing, and Windowing. Let $x_c(t)$ be a real, continuous signal, modeled as a stationary ergodic random process, and let $h_c(t)$ be the impulse response of a baseband filter. We want to approximate the filter output

$$y_c(t) = \int_0^\infty h_c(\tau) x_c(t - \tau) d\tau$$

with the *windowed* discrete convolution

$$y_d(n) = \sum_{k=0}^{M-1} W(k) h_c(kT_s) x_c(kT_s - nT_s) \approx y_c(nT_s)$$

where the window $W(k)$ is such that $W(k) = 0$ for $k < 0$ and for $k \geq M$. Now define the mean-square error (MSE) η as $E[(y_c(nT_s) - y_d(n - p))^2]$, where p is a possible “delay” that can be introduced to minimize η . (a) Show that the MSE is given by

$$\eta = \int_{-(1/2T_s)}^{(1/2T_s)} \sum_{l=-\infty}^{\infty} S_{x_c}\left(f - \frac{l}{T_s}\right) \cdot \left| H_c\left(f - \frac{l}{T_s}\right) - H(e^{j2\pi f T_s}) \right|^2 df$$

where $S_{x_c}(f)$ is the power spectral density (PSD) of $x_c(t)$, $H_c(f) = G_c(f)e^{j\omega p T_s}$, $G_c(f)$ is the Fourier transform of $h_c(t)$, and $H(e^{j2\pi f T_s})$ is the discrete Fourier transform of $W(k)h_c(kT_s)$. (b)

Obtain reductions of this expression if $x_c(t)$ is bandlimited to $|f| \leq 1/2T_s$, and if its PSD is white.

P3.11. Differentiation of a Bandpass Signal. If $x(t)$ is a bandpass signal, and its complex envelope is defined through $x(t) = \text{Re}\{\tilde{x}(t)e^{j\omega_c t}\}$, show that

$$\frac{d}{dt}x(t) = \text{Re}\left\{\left[(j\omega_c)\tilde{x}(t) + \frac{d}{dt}\tilde{x}(t)\right]e^{j\omega_c t}\right\}$$

P3.12. Interpolation. Derive Equations (3.5.18) and the specific form for N_1 and N_2 indicated in the text immediately following.

P3.13. Interpolation. Write a program to implement a third-order spline interpolation (or use use a “canned” one), (a) Generate a sequence of randomly spaced samples of a well-behaved function (e.g., by accessing a uniform random number generator; see Chapter 7) and apply spline interpolation to produce a “continuous” curve; (b) apply linear interpolation to the original samples (c) apply parabolic interpolation to the original samples. Superimpose and compare the interpolated curves.

P3.14. Interpolation. Generate samples of a raised-cosine pulse with one-sided bandwidth $W = 0.75R_b$, where R_b is the bit rate (see Chapter 8). Use a sample spacing $T_s = (1.5R_b)^{-1}$ and space the samples on either side of $t = 0$, spanning ± 5 bits; i.e., do not sample at the peak of the pulse. Regenerate the raised-cosine pulse using (a) bandlimited interpolation (truncated to ± 5 bits), (b) windowed bandlimited interpolation, (c) linear interpolation. Suppose you sampled the pulse at $t = 0$ to make a decision. Calculate the relative error and the impact on the error rate associated with the different interpolation methods.

P3.15. Multirate Simulation System. Consider the implementation of sampling rate conversion (“multirate”) in a simulation. (a) Describe different kinds of systems where sampling rate conversion may improve the efficiency of simulation. Draw simulation block diagrams showing sampling rate conversion operations as identifiable blocks. (b) For the cases above, calculate the run-time improvement due to multirate simulation. Perform this calculation both with and without accounting for the time spent in the sampling rate conversion blocks. (c) Suggest one (or more) efficient approach for the design of interpolating filters, [see also M. Pent, L. LoPresti, M. Mondin, and L. Zaccagnini, Multirate sampling techniques for simulation of communication systems, IASTED International Symposium on Modeling, Identification, and Control, Grindewald, Switzerland (February 1987), and V. Castellani, M. Mondin, M. Pent, and P. Secchi, Simulation of spread spectrum communication links, Presented at 2nd IEEE International Workshop on Computer-Aided Modeling, Analysis, and Design of Communication Links and Networks, Amherst, Massachusetts (October 12–14, 1998)].

P3.16. Discrete Fourier Transform and Fourier Series. Show that the Fourier series of a periodic function with period P , bandlimited to $|f| \leq M \Delta f$, where $\Delta f = P^{-1}$, is identical to the N -point discrete Fourier transform of the function if $N > 2M$. (Such a function is called a trigonometric polynomial.)

P3.17. Circular Convolution. If Y_n is the DFT of y_n , X_n is the DFT of x_n , and H_n is the DFT of h_n , show that if $Y_n = X_n H_n$, then y_n is the circular convolution of x_n and h_n .

P3.18. Simulation of a Network Analyzer. Systems are typically characterized by their frequency domain response. In practice, such responses are measured using so-called “swept-tone” measurements, performed with a network analyzer. Develop a “discrete-time” network analyzer for your simulation. Discuss the difference between *swept-tone* and *stepped-*

tone; also discuss how the analyzer results may be influenced by the relationship between the frequencies used and the sampling interval.

P3.19. Aliasing and Leakage Effects in the DFT. Consider a sampled version of a waveform, $x(t) = 10 \sin(2\pi f_0 t)$, $t = kT_s$, $0 \leq k \leq N - 1$. Let $f_0 = 1$ and $T = NT_s$. Compare the DFT of $x(kT_s)$ with the continuous Fourier transform of $x(t)$ for the following conditions: (a) Sampling rate $f_s = 1.6$ samples/s, $T = 20$ s, (b) $f_s = 1.5$ samples/s, $T = 64/3$ s, (c) $f_s = 3.2$ samples/s, $T = 10$ s, (d) $f_s = 3$ samples/s, $T = 32/3$ s, and (e) $f_s = 3$ samples/s, $T = 64/3$ s. Explain the influence of f_s , N , and T on the DFT. How would you mitigate the effects of aliasing and leakage of a large periodic component in the signal. [See Chapter 3, Refs. 3 and 30, see also A. A. Girgis and F. M. Ham, A quantitative study of pitfalls in the FFT, *IEEE Trans. Aerospace Electron. Syst.* **AES-16**, 434-439 (1980).]

Chapter 4

P4.1. Analog Butterworth Filter. Given a lowpass Butterworth filter with a passband attenuation α_p at the frequency f_p and a stopband attenuation α_r at the frequency f_r , determine the filter order n and the 3-dB bandwidth f_b .

P4.2. Relationship between Lowpass Prototype and Lowpass Equivalent. For the lowpass Butterworth filter in P4.1, design a corresponding bandpass filter. Find the lowpass equivalent of this bandpass filter. Compare the lowpass Butterworth filter transfer function to that of the lowpass equivalent as a function of the ratio of bandwidth to center frequency. Show that, as this ratio $\rightarrow 0$, the two transfer functions coalesce.

P4.3. Analog Elliptic Filter. Given an analog fourth-order elliptic filter: (a) Find the coefficient for the biquadratic sections for the normalized lowpass filter with $\alpha_p = 1$ dB and $\alpha_s \geq 40$ dB and the ratio of the stopband to passband frequencies $f_r/f_p = 1.5$. (b) Show the biquadratic realization of the filter. (c) Perform a lowpass to bandpass transformation as in (4.1.15) with $f_b = \omega_b/2\pi = 3$ Hz and (1) $f_g = \omega_g/2\pi = 4$ Hz, (2) $f_g = 400$ Hz. (d) Plot the frequency responses of the lowpass and bandpass filters. (e) Discuss the implications of the band transformations on the lowpass equivalent filter.

P4.4. Bandpass Filter in the Discrete Domain. Given a bandpass filter specified by poles and zeros, you have to formulate a lowpass-equivalent discrete filter using the impulse-invariant transformation. Under what conditions do the negative-frequency poles *not have* to be eliminated?

P4.5. Application of the Bilinear Transform. Provide a discrete model for the filter specified in problem P4.1 using the bilinear transform with the sampling rates (a) $f_s = 3f_r$, (b) $f_s = 15f_r$. Discuss the implications. Plot the frequency response and compare with those of Problem P4.1.

P4.6. Frequency-Domain FIR Filter. How would you determine if an FIR filter has sufficient samples in the frequency domain? (Hint: Obtain the impulse response and integrate its square.)

P4.7. Frequency-Domain Interpolation. Show that augmenting the impulse response of an FIR filter with zeros produces interpolating samples in the frequency domain.

P4.8. Gibbs Phenomenon Distortion in an Ideal Filter. Consider a sampled version of an ideal lowpass FIR filter in the frequency domain (see Figure 4.1a). The IFFT produces a truncated impulse response. (a) Explain why the Gibbs phenomenon distortion is invisible in

the frequency domain. (b) Augment the impulse response with zeros and take the FFT. Explain the results.

P4.9. *Overlap-and-Add Method.* Write a program to implement the overlap and add method of filtering.

P4.10. *Windowed Response.* Redo Problem P4.8 and apply (a) the Hamming window (b) the Bartlett window (see also Chapter 10). Compare the results.

P4.11. *Filter Synthesis from Measured Data.* Given a set of measured amplitude and phase points, how would you go about synthesizing an IIR filter from these points?

P4.12. *Impulse Response Comparison.* For the filter in Example 4.1.7, obtain the discrete-time filter using the bilinear z -transform. Compare the frequency response, as a function of f_s , to that of the solution given in the example.

P4.13. *Gibbs Phenomenon Distortion in a Raised-Cosine Filter.* Simulate the raised cosine filter (see Section 8.9) with a frequency rolloff $\beta = 0.25$ using a frequency-domain FIR filter implementation. What is the Gibbs phenomenon distortion with (a) a rectangular window, (b) a Hamming window?

P4.14. *Filtering and Multirate.* Suppose that you want to filter a signal of bandwidth W with a filter of bandwidth B . (The definition of bandwidth is not critical for this problem.) Even if $W \gg B$, we know that the bandwidth of the filter output will be at most on the order of B . Do we still have to generate samples of the input signal (and therefore also samples of the filter impulse response) at the rate $2W$? What can be done to create efficient sampling at the filter output on the order of $2B$ samples/s?

P4.15. *Effects of Aliasing and Truncation.* Transmit a single rectangular pulse of duration T into a 0.5-dB ripple three-pole Chebyshev filter with $f_p T = 0.7, 0.8, \dots, 1.2$. Approximate the filter impulse response by an FIR filter and superimpose the various output waveforms computed as a function of the number of samples per unit of T and the length of the impulse response.

P4.16. *Effects of Aliasing and Truncation.* Repeat Problem P4.15 using the bilinear transformation for the same values of sampling rate, and compare the two sets of results.

P4.17. *Effects of Aliasing and Truncation.* For the filter of Problem P4.15 now transmit a short pseudonoise sequence (say 15 bits), using eight samples per bit. (a) Simulate using an FIR model with $f_p T = 1.0$ and a truncation equal to three time constants. (b) Repeat (a) using a bilinear transformation for the filter, and compare the two resulting sequences.

P4.18. *Efficiency of the FFT.* It was shown that when using the FFT, the number of multiplications for K segments of filtered data is $K(N \log_2 N + N) + \frac{1}{2} N \log_2 N$, where the number of points used in the FFT is $N = L + M - 1$, L is the actual number of sample points, and $M - 1$ is the number of padded zeros. Obtain the “effective” number of multiplications μ_e per processed bit, assuming m samples per bit, and taking into account the number of useful bits in the OA or OS method. For a given m , plot μ_e as a function of N and observe that there is an optimum value of N , which depends on M .

P4.19. *Impulse-Invariant Transformation.* Develop an analytical expression for the impulse response $h(t)$ of a lowpass five-pole Butterworth filter with 3-dB bandwidth B equal to 1. (a) Obtain the sampled values $h(kT_s)$ for $T_s = 0.4B^{-1}$. (b) Using the pole values, generate the impulse response using a discrete impulse as the input to the impulse-invariant model given by (4.1.66), and verify that it is the same as in (a).

P4.20. *Filtering with Finite Precision.* Generate a sequence of Gaussian random numbers with variance equal to 1, at sampling rate f_s , and prepare to filter it with a four-pole, 0.25-dB-ripple, Chebyshev filter, with cutoff frequency of $0.1f_s$, by using the bilinear trans-

formation. Quantize the generated samples and the filter coefficients at various levels of quantization (beginning with the “unquantized” values normally produced), and compare the corresponding filter outputs.

P4.21. Bandwidth Expansion of a Linear Time-Varying System. Show that the relationship in Equation (4.2.8) implies the one in Equation (4.2.9).

P4.22. Discrete LTV Model. Formulate a discrete (sampling) LTV model for a signal with a bandwidth $|f| \leq B$ and an impulse response of Example 4.2.2.

P4.23. Separable LTV Model. Formulate a separable LTV model for the impulse response of Example 4.2.2.

P4.24. Inverse of LTV System. The discrete impulse response of an LTV system is given in matrix form $h(m, n) = A(m, n)$. Prove that the inverse of the impulse response $h^{-1}(m, n)$ is the inverse of the matrix $A(m, n)$. Discuss the practicality of this procedure in simulation.

Chapter 5

P5.1. Nonlinear Spectral Spreading. Assume the output of a nonlinear system is given by (5.2.3) for $x(t)$ a finite-energy signal. Show that the output spectrum is given by (5.2.4).

P5.2. Nonlinear Spectral Spreading. Assume you have a pair of AM/AM and AM/PM characteristics (you can create these curves or use published data; see, e.g., those in Ref. 10 of Chapter 5). Generate a Gaussian random process and filter it with a filter whose bandwidth is appreciably less than (say, one fifth of) the simulation bandwidth (f_s). (a) Obtain the PSD of the filter output (see Chapter 10 for PSD estimation); (b) pass the filter output through the AM/AM characteristic *only* and obtain the PSD of the output; (c) repeat part (b) using the AM/PM characteristic *only*; (c) repeat part (b) with *both* the AM/AM and AM/PM characteristics.

P5.3. Nonlinear Spectral Spreading. Repeat Problem P5.2, but instead of a Gaussian process, let the input be an unfiltered random binary waveform with at least 16 samples/bit.

P5.4. The Chebyshev Transform. (a) Use the substitution $A \cos \alpha$ in (5.2.9a) to show that

$$a_1 = \frac{2}{\pi} \int_{-A}^A \frac{(u/A)F(u) du}{\sqrt{A^2 - u^2}}$$

is the first term of a Chebyshev expansion of the function F defined over the interval $[-A, A]$. (b) Conclude that $a_1 = 0$ if F is an even function: explain in intuitive terms why that is so.

P5.5. Memoryless Nonlinearity. Show that for *any* “ordinary” function F , the coefficient g_2 defined by (5.2.9b) is *always* zero.

P5.6. Describing Function of a Hysteresis Nonlinearity. Obtain Equations (5.2.14) starting from the definitions (5.2.9). *Hint.* Break up the integral into four segments corresponding to the four sides of the hysteresis characteristic.

P5.7. Hard-Limiter. Show that for a hard-limiter, the general form (5.2.15) reduces to (5.2.17a) for the baseband case, and implies (5.2.17b) for the bandpass case.

P5.8. Effects of Finite Sampling. A sinusoidal signal $x(t) = A \sin 2\pi t$ is applied to a hard-limiter: the output, $y(t) = -1$ for $x(t) \leq 0$ and $y(t) = 1$ for $x(t) > 0$, is sampled at the rate $f_s = 8$ samples/s. Examine $y(t)$ in the time and frequency domains. How would you alleviate the anomalous behavior? Consider the effects of sampling rate and sampling phase.

P5.9. Limiter Suppression. Limiters have an effect known as “suppression” where large signals decrease smaller signals. Consider two sinusoids $A_1 \cos(2\pi f_1 t)$ and $A_2 \cos(2\pi f_2 t)$ with $A_2 < A_1$. Let B_1 and B_2 be the corresponding outputs at the original frequencies. Plot the ratio B_1/B_2 versus A_1/A_2 . Make sure you sample sufficiently rapidly.

P5.10. Soft-Limiter. Show that for a limiting amplifier (or soft-limiter), the general form reduces to (5.2.18a) for the baseband case, and implies (5.2.18b) and (5.2.19) for the bandpass case. Show that (5.2.19) reduces to (5.2.17b) as $l \rightarrow 0$.

P5.11. Asymmetric Limiter. Consider the asymmetric limiter

$$\begin{aligned} y &= (L_1/l_1 x), & 0 \leq x \leq l_1 \\ &= L_1, & x > l_1 \end{aligned}$$

and

$$\begin{aligned} y &= (L_2/l_2 x), & -l_2 \leq x \leq 0 \\ &= -L_2, & x < -l_2 \end{aligned}$$

with $l_2 < 0$. (a) Obtain the coefficient a_1 . (b) Show that b_1 is still zero, i.e., the asymmetry is not a factor.

P5.12. Power Series Model. Fill in the steps to show that (5.2.24) is true.

P5.13. Effectively Memoryless Nonlinearity. Consider the following model for a nonlinearity (see Ref. 9, Chapter 5): The input signal $x(t) = r(t) \cos[\omega_c t + \theta(t)]$ is applied to a memoryless nonlinearity F , then split into $N+1$ delay lines each with delay τ_n , $n = 0, 1, \dots, N$, recombined, and passed through a zonal filter. Thus, the output is described by

$$y(t) = \sum_{n=0}^N F_1(r(t - \tau_n)) \cos[\omega_c(t - \tau_n) + \theta(t - \tau_n)]$$

where F_1 means first-zone output. Assume the specific (envelope-dependent) form for the delay $\tau_n = n\tau_0(r)$. Show that as $\tau_0 \rightarrow 0$ and $N \rightarrow \infty$ such that $N\tau_0(r) \rightarrow \tau(r)$, the output reduces to the form

$$y(t) = g_1(r) \cos[\omega_c t + \theta(t) + \Phi(r)]$$

which shows at least one circumstance under which a model with memory reduces to one which is “effectively” memoryless. As stated in the text, “instantaneous” may be a better term here.

P5.14. Nonlinear Memoryless Amplifier Modeling. Show that the standard measured AM/AM and AM/PM characteristics, i.e., power out versus power in, and phase out versus power in, can be meaningfully interpreted in discrete time as instantaneous envelope

out versus instantaneous envelope in, and instantaneous phase out versus instantaneous envelope in. Discuss the conditions on T_s for this interpretation to be meaningful.

P5.15. Two-Tone Signal. Consider the two-tone signal

$$x(t) = A \cos(2\pi f_c t + \theta_A) + a \cos(2\pi f_{\pm} t + \theta_a)$$

where $A, a > 0$, θ_A and θ_a are arbitrary phases, and

$$f_{\pm} \equiv f_c \pm \Delta f$$

represents the frequency of the small-tone that is above (below) the large-tone frequency f_c by the positive offset Δf in the high-side (low-side) injection case. Using the results of Section 3.4, show that $x(t)$ can be expressed in the equivalent form (5.2.27), where $A(t)$ is as in (5.2.35a), $\theta(t)$ is as in (5.2.35b), and $\alpha \equiv a/A$, which need not be small for this result to hold.

P5.16. Two-Tone Dynamic Model. Consider the two-tone model output $y(t)$ in (5.2.28) with $[gA(t)]$ and $\Phi[A(t)]$ replaced with the more general relations in (5.2.37) for the high-side injection case (that is, taking the plus sign in these functions). (a) Expand the exact expressions for $A(t)$ and $\theta(t)$ of the two-tone input given in (5.2.35) about $\alpha = 0$ and show that they can be expressed as

$$A(t) = A[1 + \alpha \cos(2\pi \Delta f t + \theta_a - \theta_A)] + \mathcal{O}(\alpha^2)$$

and

$$\theta(t) = \theta_A + \alpha \sin(2\pi \Delta f t + \theta_a - \theta_A) + \mathcal{O}(\alpha^2)$$

(b) Expand $y(t)$ about $\alpha = 0$, keeping terms only up to order α , as follows:

$$y(t) = y(t)|_{\alpha=0} + \left. \frac{\partial y(t)}{\partial \alpha} \right|_{\alpha=0} \alpha + \mathcal{O}(\alpha^2)$$

Substituting the expansions for $A(t)$ and $\theta(t)$ from (a) into this expansion, show that the two-tone dynamic model output can be expressed as in (5.2.38), with the sideband phase components $\Phi_{\pm}$ given explicitly by

$$\Phi_+ = \Phi_+(A, \Delta f) = \theta_a + \theta_+, \quad \Phi_- = \Phi_-(A, \Delta f) = 2\theta_A - \theta_a + \theta_-$$

where

$$\theta_{\pm} = \theta_{\pm}(A, \Delta f) = \Phi(A, \Delta f) \pm \tan^{-1} \left[\frac{A\Phi'(A, f)}{1 \pm \frac{Ag'(A, \Delta f)}{g(A, \Delta)}} \right]$$

P5.17. Intermodulation Products. Convince yourself that the bandpass nonlinearity model produces the correct intermodulation products in the first zone, through the following simple example. Let

$$y = a_1x + a_3x^3 \quad (\text{A})$$

represent a simple power series nonlinearity (you may take $a_1 = 1$ for convenience) and let

$$x = A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t) \quad (\text{B})$$

(a) Obtain the output in the first-zone by applying (B) to (A) directly. (b) From (a) define the implied envelope transfer characteristic for the first zone $g_1(A)$. (c) Express (B) as $A(t) \cos[\omega_0 t + \theta(t)]$ and obtain the first-zone output by using the form $g_1(A(t)) \cos[\omega_0 t + \theta(t)]$ directly. Show that the result is identical to that in (a). (d) Use simulation (for specificity, you may use $a_3 = 0.1$) to generate sampled values of the lowpass equivalent of (B) into the envelope transfer characteristic, and compare to the results of (a). (e) Use simulation to generate samples of the actual signal (B) into the actual nonlinearity (A), and compare to the previous results.

P5.18. Intermodulation Products. Assume you have a pair of AM/AM and AM/PM characteristics. (a) Generate several sinusoids of equal amplitude (begin with three and increase as desired) and input them into the AM/AM characteristic *only*. Study the locations and amplitudes of the output tones when the input tones have equal spacing in frequency and when they do not. (b) Repeat part (a) with the AM/PM characteristic *only*. (c) repeat part (a) with *both* the AM/AM and AM/PM characteristics.

P5.19. Intermodulation Products: Effects on BER. Assume the nonlinearity in the previous problem is followed by a receiver with additive Gaussian noise and a detection filter (choose the filter type and use the BT product of 0.7), and consider three modulated quaternary phase-shift-keying (QPSK) input carriers with equal data rates $R = 1/T$. The upper carrier is spaced in frequency at $2R$ above the center carrier, and the lower carrier at $2.5R$ below the center carrier. (a) Obtain the BER performance for the center carrier; (b) compare with the performance in the absence of adjacent channels.

P5.20. Nonlinear Predistortion. Assume you have a pair of AM/AM and AM/PM characteristics. Construct a memoryless nonlinear “predistorter,” a nonlinear device preceding the nonlinearity such that the cascade acts as a linear amplifier. [See, e.g., A. A. M. Saleh and J. Salz, Adaptive linearization of power amplifiers in digital radio systems, *Bell Syst. Tech. J.* **62**(4), 1019–1033 (1983).]

P5.21. Nonlinearity with Memory: The PSB Model. Show that the model of Figure 5.13 is the correct combination of the AM/AM and AM/PM models given in Figures 5.11 and 5.12.

P5.22. Nonlinearity with Memory: The Saleh Model. Show that the model given by (5.3.3) results in the block diagram descriptions given in the legend of Figure 5.14.

P5.23. Nonlinearity with Memory: More on the Saleh Model. Show that the model specified by (5.3.3) possesses the property that the phase transfer characteristic $\psi(f) \rightarrow 0$ as $A \rightarrow 0$, which necessitates reinserting the small-signal phase $\Phi_0(f)$. Is the location of this block uniquely determined by the model?

P5.24. Nonlinearity with Memory: Model Structure. Show that an NLWM model whose structure is a filter–memoryless nonlinearity–filter cascade will necessarily produce swept-tone AM/AM characteristics that are combinations of horizontal and vertical translates

of one another on dB scales; and similarly for the AM/PM characteristics with phase the dependent variable and the power axis in dB units.

P5.25. Nonlinearity with Memory: More on Model Structure. Show that the PSB and the Saleh models are generally different. Under what conditions are the models identical (for swept-tone inputs)?

P5.26. Abuelma'atti Model. Show that the Abuelma'atti model does not constrain the inphase nonlinearities $S_p(A, f)$ to be horizontal and/or vertical translates of one another (as a function of frequency); and similarly for the quadrature nonlinearities $S_q(A, f)$.

P5.27. Three-box Model with Least-Squares Fit. In the model of Section 5.3.2.2, the phase of the two filters is ambiguous. Describe some ways in which you might split the derived phase characteristics between the two filters. How might you validate your choice?

P5.28. Power-Series Nonlinearity-Filter Volterra Kernels. Consider the nonlinear branch of the polyspectral model shown in Figure 5.23a, where the nonlinearity is given by the finite power series

$$g(x) = \sum_{n=0}^N a_n x^n(t)$$

(a) First show that the output $y_v(t)$ of the branch is given by

$$y_v(t) = \sum_{n=0}^N a_n \int_{-\infty}^{\infty} h(\tau_1) x^n(t - \tau_1) d\tau_1$$

where $h(\tau_1)$ denotes the impulse response of the filter $H_2(f)$. As a consequence, we can individually derive the Volterra kernels for each term of the power series.

(b) Consider the $n = 0$ term of power series. Show that the zeroth-order Volterra kernel h_0 and its frequencytransform $H_0(f)$ are as in (5.3.22). Similarly, derive separately the results for $n = 1$. For $n \geq 2$, substitute the relation

$$x^n(t - \tau_1) = \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{n-1 \text{ integrals}} \prod_{k=1}^n x(t - \tau_k) \prod_{l=2}^n \delta(\tau_1 - \tau_l) d\tau_l$$

into the n th component of the expression for $y_v(t)$ in (a), place the result in the form of (5.3.8), and then identify the n th-order kernel to be as claimed in (5.3.22a). Calculate the multi-dimensional Fourier transform of the kernels via

$$H_n(f_1, \dots, f_n) = \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{n-1 \text{ integrals}} h_n(\tau_1, \dots, \tau_n) \left[\exp \left(-j2\pi \sum_{k=1}^n f_k \tau_k \right) \right] d\tau_1 \cdots d\tau_n$$

to arrive at the results in (5.3.22b).

P5.29. LPE Power Series Nonlinearity-Filter Polyspectral Model. Derive the LPE of the power series nonlinearity-filter polyspectral model treated in the previous problem as follows, assuming that both of the filters in the model are bandpass in nature, centered around the RF frequency f_c . (a) Use Figure 5.23a to conclude that the linear branch of the model has the standard LPE where $H_1(f)$ is replaced with $H_{L1}(f) = H_1(f + f_c)U(f + f_c)$. For the

nonlinear branch, observe that $y_v(t)$ can be expressed in the form (5.3.7) and (5.3.8), where $h_n(\tau_1, \dots, \tau_n)$ is as in (5.3.22a). For $n = 0$, show that $y_{v0}(t) = 0$, and hence does not contribute to the LPE branch output $\tilde{y}_v(t)$. For $n \geq 1$, substitute the relation

$$h(\tau_1) = \frac{1}{2} [\tilde{h}(\tau_1)e^{j2\pi f_c \tau_1} + \tilde{h}^*(\tau_1)e^{-j2\pi f_c \tau_1}]$$

and a similar one for $x(t)$ to conclude that

$$\begin{aligned} y_{vn}(t) &= \frac{a_n}{2^{n+1}} \int_{-\infty}^{\infty} [\tilde{h}(\tau_1)e^{j2\pi f_c \tau_1} + \tilde{h}^*(\tau_1)e^{-j2\pi f_c \tau_1}] \\ &\times [\tilde{x}(t - \tau_1)e^{j2\pi f_c(t-\tau_1)} + \tilde{x}^*(t - \tau_1)e^{-j2\pi f_c(t-\tau_1)}]^n d\tau_1 \end{aligned}$$

(b) Because only the first-zone terms for $y_v(t)$ apply for the LPE model, argue that only those product terms that contain the exponential $\exp(\pm j2\pi f_c t)$ in the expression for $y_{vn}(t)$ in (a) will survive. Also, use the binomial theorem to show that such terms can only arise when n is an odd number (denoted by $2m + 1$, $m = 0, \dots, M - 1$ or M , the latter depending on whether $N = 2M$ or $2M + 1$, respectively), and furthermore only for the specific terms given by

$$\begin{aligned} &\binom{2m+1}{m+1} [\tilde{x}(t - \tau_1)]^m e^{j2\pi m f_c(t-\tau_1)} [\tilde{x}^*(t - \tau_1)]^{m+1} e^{-j2\pi(m+1)f_c(t-\tau_1)} \\ &+ \binom{2m+1}{m} [\tilde{x}(t - \tau_1)]^{m+1} e^{j2\pi(m+1)f_c(t-\tau_1)} [\tilde{x}^*(t - \tau_1)]^m e^{-j2\pi m f_c(t-\tau_1)} \end{aligned}$$

where

$$\binom{2m+1}{m+1} = \binom{2m+1}{m} = \frac{(2m+1)!}{m!(m+1)!} = \frac{(2m+1)(2m)\cdots(m+2)}{m!} \equiv: c_m$$

(c) Write

$$y_{v,2m+1}(t) = \frac{1}{2} [\tilde{y}_{v,2m+1}(t)e^{j2\pi f_c t} + \tilde{y}_{v,2m+1}^*(t)e^{-j2\pi f_c t}]$$

for the nonzero components of $y_v(t)$. Use the fact that f_c is large compared to the bandwidth of the input signal to argue that terms containing the exponential $\exp(\pm j2\pi f_c k\tau_1)$, $k \neq 0$, provide a vanishingly small contribution to the first zone of $y_{v,2m+1}(t)$ because of their rapid oscillation in τ_1 compared to terms that do not have this factor. Proceed to show that the $(2m + 1)$ th-order LPE branch output is thus given by

$$\begin{aligned} \tilde{y}_{v,2m+1}(t) &= \frac{c_m a_{2m+1}}{2^{2m+1}} \int_{-\infty}^{\infty} \tilde{h}(\tau_1) [\tilde{x}(t - \tau_1)]^{m+1} [\tilde{x}^*(t - \tau_1)]^m d\tau_1 \\ &= \frac{c_m a_{2m+1}}{2^{2m}} \frac{1}{2} \tilde{h}(t) * \{|\tilde{x}(t)|^m \tilde{x}(t)\} \end{aligned}$$

where $*$ denotes the convolution operation. Conclude that the LPE of the nonlinear branch is as in Figure 5.23b, with $H_{L2}(f)$ as the standard LPE of $H_2(f)$, and the equivalent memoryless nonlinearity $g_L(\cdot)$ is as claimed after (5.3.22).

P5.30. LPE of Third-Order Filter-Nonlinearity Polyspectral Model. Derive the LPE of the third-order filter-nonlinearity polyspectral model shown in Figure 5.24a as follows, assuming that all of the filters in the model are bandpass in nature, centered around the RF frequency f_c . (a) Use Figure 5.24a to conclude that the linear branch of the model has the standard LPE where $H_1(f)$ is replaced with $H_{L1}(f) = H_1(f + f_c)U(f + f_c)$. (b) For the filter-squarer nonlinear branch, use the $n = 2$ case of (5.3.8) and (5.3.25a) and the procedure outlined in Problem P5.22 to conclude that the first-zone portion of $y_2(t)$ is zero, and hence does not contribute to the LPE output $\tilde{y}(t)$ [that is, $\tilde{y}_2(t) \equiv 0$]. (c) Proceed similarly for the filter-cuber nonlinear branch and show that its LPE contribution $\tilde{y}_3(t)$ to $\tilde{y}(t)$ is given by

$$\begin{aligned}\tilde{y}_3(t) &= \frac{3}{32} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{h}(\tau_1) \tilde{h}(\tau_2) \tilde{h}^*(\tau_3) \tilde{x}(t - \tau_1) \tilde{x}(t - \tau_2) \tilde{x}^*(t - \tau_3) d\tau_1 d\tau_2 d\tau_3 \\ &= \frac{3}{4} \left[\int_{-\infty}^{\infty} \frac{1}{2} \tilde{h}(\tau) \tilde{x}(t - \tau) d\tau \right]^2 \left[\int_{-\infty}^{\infty} \frac{1}{2} \tilde{h}(\tau) \tilde{x}(t - \tau) d\tau \right]^* \\ &= \frac{3}{4} [\tilde{u}_3(t)]^2 \tilde{u}_3^*(t)\end{aligned}$$

Using the $m = 0$ case of (5.2.12) and the definition of the third-order LPE Volterra kernel given after it, show that

$$\tilde{h}_3(\tau_1, \tau_2, \tau_3) = \frac{1}{4} \tilde{h}(\tau_1) \tilde{h}(\tau_2) \tilde{h}^*(\tau_3)$$

Letting $H_{L3}(f, g, h)$ be the three-dimensional Fourier transform of the third-order LPE Volterra kernel, show that

$$H_{L3}(f, g, h) = \frac{3}{4} H_{L3}(f) H_{L3}(g) H_{L3}^*(-h)$$

where $H_{L3}(f)$ is the standard LPE of $H_3(f)$. Using the last relation for $\tilde{y}_3(t)$ derived above, conclude that the LPE of the filter-cuber branch is as in Figure 5.24b, with $H_{L3}(f)$ as just defined, and the equivalent memoryless third-order nonlinearity $g_{L3}(\cdot)$ given by

$$g_{L3}(\tilde{x}) = \frac{3}{4} \tilde{x}^2 \tilde{x}^*$$

P5.31. Nonlinear Differential Equations. Find the local error E and the order for the backward Euler and trapezoidal integration formulas.

P5.32. Optical Amplifier Model. Given an optical amplifier of the type discussed in Section 5.4.5.1 with parameters $\Gamma = 0.25$, $a = 3 \times 10^{-20} \text{ m}^2$, $N_0 = 10^{24} \text{ m}^3$, $I_0 = 40 \text{ mA}$, $\tau_c = 200 \text{ ps}$, and $P_{\text{sat}} = 1 \text{ mW}$, simulate the amplifier using (a) the explicit second-order Adams-Bashforth integration method, (b) the implicit trapezoidal integration method with Newton-Raphson root-finding, (c) the implicit trapezoidal integration method with a unit delay in each feedback loop. Compare the results. Discuss the choice of the sampling interval T_s .

Chapter 6

P6.1. Moments of Sums of Random Variables. The estimator of the mean of a random variable has the form

$$Y = \frac{1}{N} \sum_{k=1}^N X_k$$

where X_k are N independent samples from the underlying pdf $f_X(x)$. Assuming the underlying pdf is $N(0,1)$, find the mean and variance of the estimator Y .

P6.2. Moments of Sums of Random Variables. The estimator for the variance of a zero-mean Gaussian has the form

$$Y = \frac{1}{N} \sum_{k=1}^N X_k^2$$

Find the mean and variance of the estimator in terms of the unknown variance of the underlying pdf of X and the sample size N .

P6.3. Moments of Sums of Random Variables. Suppose the sampled values of the output of the receive filter in a communication system has the form

$$Z = \sum_{m=1}^3 h_m A_m$$

where A_m is an independent sequence of octal symbols with values $-7, -5, -3, -1, 1, 3, 5$, and 7 , and h_m represents sampled values of the impulse response with values $h_1 = 0.3$, $h_2 = -0.1$, and $h_3 = 0.05$. Compute the first eight moments of the ISI.

P6.4. Transformation of Random Variables. Transformations of random variables are used for generating random numbers. Find the pdf of the following transformations: (a) $Y = -\log(X)$, where X is uniform in $[0, 1]$. (b) $Y = X^2$, where X is Gaussian $(0, 1)$. (c) $Y = \sqrt{X_1^2 + X_2^2}$, where X_1 and X_2 are independent Gaussian $(0, 1)$ variables.

P6.5. Vector-Valued Transformations. X_1 and X_2 are two independent, zero-mean, unit-variance Gaussian variables. Y_1 and Y_2 are defined by the transformations

$$Y_1 = \sqrt{X_1^2 + X_2^2} \quad \text{and} \quad Y_2 = \tan^{-1}(X_1/X_2)$$

Find the joint pdf of Y_1 and Y_2 , and the marginal pdf's of Y_1 and Y_2 .

P6.6. Bounds and Approximations. Plot the Chebyshev and the Chernoff bounds as well as the exact values for $P[X > a]$ versus a , $a > 0$, for the following pdf's of X : (a) Uniform $[0, 1]$. (b) Exponential. (c) $N(0, 1)$.

P6.7. Bounds and Approximations. Derive the union bound for the MPSK system in which the decision metric has the form $Z = X + N$, where X is a complex signal having values that are uniformly spaced on a circle of radius A , $X = Ae^{-j\theta}(\theta = 2\pi k/M; k = 0, 1, 2, \dots, M-1)$, with equal probability, and $N = N_p + jN_q$ is a complex noise sample whose components are uncorrelated zero-mean Gaussian variables with a variance σ^2 . Express the probability of error as a function of A^2/σ^2 and M .

P6.8. Bounds and Approximations. Repeat the previous problem for a 16-QAM constellation in which the complex signal component X has 16 equally likely values which are uniformly spaced at the corners of a rectangular grid $X = mA + jnA, m, n = -3, -1, 1, 3$.

P6.9. Sampling Rate. Suppose we want to sample a zero-mean Gaussian process with the autocorrelation function $R_{xx}(\tau) = \exp(-\alpha|\tau|)$. Compute the signal to aliasing power ratio as a function of sampling rate normalized by α (e.g., for $\alpha T_s = 5, 10, 20, 50, 100$).

P6.10. Noise Bandwidth of Filters. Compute the noise bandwidth of a Butterworth filter of order 2, 4, 6, 8, 10. Assume that the filters have a 3-dB bandwidth of 1 Hz.

P6.11. Noise Bandwidth of Filters. Consider a simple analog filter with an impulse response of $h(t) = e^{-t}$ for $t > 0$. (a) Compute the noise bandwidth of the analog filter. (b) Suppose the analog filter is simulated as an IIR filter using a sampling rate of n samples/s. What is the noise bandwidth of the filter for $n = 10, 20, 50$, and 100 ? (c) Suppose the filter is simulated as an FIR filter using a truncated impulse response of N samples, $N = 2n, 8n$, and $16n$, where n is the sampling rate. Find the noise bandwidth as a function of n and N . Explain the results.

P6.12. Lowpass-Equivalent Representation. Consider a bandpass system in which the received radiofrequency signal has the form $Y(t) = X(t) + N(t)$, where $X(t) = A \cos(2\pi f_0 t + \theta)$ is the signal component, θ is uniform in $[0, 2\pi]$, and $N(t)$ is bandlimited Gaussian noise with a PSD of $N_0/2$ in the interval $f_0 - B < |f| < f_0 + B$ and 0 elsewhere ($f_0 \gg B$). (a) Find the lowpass-equivalent representation in the time domain and frequency domain for the signal and noise components. (b) Find the signal to noise power ratio, defined as $E|X(t)|^2 / E|n(t)|^2$, for the bandpass and lowpass representations. Are they the same for the two representations?

P6.13. Lowpass-Equivalent Representations. Consider a complex bandlimited Gaussian process $n(t)$ with a power spectral density

$$S_{nn}(f) = \begin{cases} 1.0 & \text{for } f_0 - B \leq |f| \leq f_0 \\ 0.5 & \text{for } f_0 \leq |f| \leq f_0 + B \\ 0 & \text{elsewhere } (f_0 \gg B) \end{cases}$$

Find the lowpass-equivalent representation of $n(t)$ of the form $\tilde{n}(t) = n_c(t) + jn_s(t)$. That is, find the power spectral densities of the quadrature components and also the cross-correlation between them.

P6.14. Quantization Effects. Finite-precision arithmetic may affect simulation and implementation accuracies. Suppose we are simulating an FIR filter of the form $Y = X_1 + X_2 + \cdots + X_n$. (a) If the X_i are independent random variables with a uniform pdf in the interval $[0, 1]$ and $n \gg 1$, find the output pdf. (b) If the X_i and the Y_i are quantized to eight levels over their respective ranges, find the mean and variance of Y and the mean square error defined as

$$\int_{y^{(\min)}}^{y^{(\max)}} (y - y_q)^2 f_Y(y) dy$$

where y_q is the quantized version of the output.

P6.15. Response of Linear Systems to Random Inputs. The input to a second-order Butterworth filter with a 3-dB bandwidth of 4 kHz is a lowpass Gaussian process with a two-sided psd of $N_0/2$ over $-4000 < f < 4000$ and 0 elsewhere. Find the mean and variance of the output of the filter.

P6.16. Response of Nonlinear Systems to Random Inputs. The input $X(t)$ to a memoryless nonlinearity is a real-valued, lowpass, Gaussian process with a two sided PSD

$N_0/2$ for $|f| < B$ and 0 elsewhere. Find the mean, variance, and the pdf of the output of the nonlinearity for the following cases:

$$\begin{aligned}
 \text{(a)} \quad Y(t) &= \begin{cases} 1 & \text{if } X(t) > 0 \\ -1 & \text{if } X(t) < 0 \end{cases} \\
 \text{(b)} \quad Y(t) &= \begin{cases} X(t) & \text{if } |X(t)| < A \\ -1 & \text{if } X(t) < -A \end{cases} \\
 \text{(c)} \quad Y(t) &= X^2(t)
 \end{aligned}$$

Chapter 7

P7.1. Uniform Random Number Generator. Implement the following algorithms for generating uniform random numbers: (a) Linear congruential. (b) Wichman–Hill algorithm. (c) Marsaglia–Zaman algorithm.

P7.2. Uniform Random Number Generators. Compare the histograms of the output of the above three algorithms using 1 million samples and 100 bins of width 0.01. Do the comparison visually by plotting the histograms and by applying the chi-square goodness-of-fit test with $\alpha = 0.05$.

P7.3. Uniform Random Number Generators. Prove that the period of a random number generator defined by a recursion, such as the linear congruential, Wichman–Hill, and Marsaglia–Zaman generators, must have a finite period.

P7.4. Uniform Random Number Generators. (a) Show that the period of a linear congruential generator of modulus M can have a period of at most M ; (b) show that a multiplicative linear congruential generator with prime modulus M has period equal to $M - 1$.

P7.5. Exponential. Write a program to generate exponentially distributed random numbers; use the transform method.

P7.6. Rayleigh. Repeat problem 7.5 for Rayleigh distribution.

P7.7. Discrete RV. Write programs to generate random numbers from the following discrete distributions: (a) gamma $\Gamma(\alpha, \beta)$; (b) Poisson $P(\lambda)$; (c) geometric $G(\theta)$; (d) binomial $B(n, p)$.

P7.8. Gaussian. Write a program to generate samples from a Gaussian distribution using the Box–Muller method.

P7.9. Arbitrary pdf. Assume that X is a continuous random variable with a pdf $f_X(x)$ defined on a finite interval $[a, b]$. Write a program to generate samples of X using the method described in Figure 7.4. Assume uniform quantizing.

P7.10. Gaussian. Write a program to generate Gaussian random variables using the inverse transform method and the following approximation for the distribution function:

$$1 - F(x) = Q(x) = \int_x^{\infty} \frac{1}{(2\pi)^{1/2}} e^{-t^2/2} dt, \quad x > 0$$

$$\approx \left[\frac{1}{(1 - a)x + a(x^2 + b)^{1/2}} \right] \frac{e^{-x^2/2}}{(2\pi)^{1/2}}$$

where $a = 1/\pi$ and $b = 2\pi$.

P7.11. Computational Efficiency. Compare the computational efficiency of the Gaussian random number generators described in (7.2.6) and Problems 7.8 and 7.10, i.e., generate, say, 10,000 samples using each method and compare the run times.

P7.12. Discrete RV. Assume that X is a discrete random variable and $P(x = x_i) = p_i, i = 1, 2, \dots, N$. Given $(x_i, p_i), i = 1, 2, \dots, N$, write a program to generate samples of X .

P7.13. Acceptance/Rejection Method. Assume that analytical expressions are given for a pdf $f_X(x)$ and a bounding function $k f_W(x)$ and the inverse cumulative $F_W(x)$. Write a program to generate samples of X using the acceptance/rejection method.

P7.14. Random Binary Sequence. Write a program to generate a random binary sequence using the uniform RNG with $P(X = 0) = p$ and $P(X = 1) = 1 - p$.

P7.15. Binary PN Sequence. Write a program to generate binary PN sequences for register lengths ranging from 6 to 16.

P7.16. M-ary PN Sequence. Write a program to generate octal and quaternary PN sequences for register lengths ranging from 2 to 5.

P7.17. Colored Gaussian Process. Write a program to generate sampled values of zero-mean white Gaussian noise with a given power spectral density (use a frequency-domain filter).

P7.18. Correlated Gaussian Sequences. Derive the algorithm for generating a Gaussian vector $\mathbf{Y} = [Y_1, Y_2, Y_3]^T$ with a mean vector of zero and a covariance matrix given by [each component process $Y_i(t)$ is white]

$$\Sigma_Y = \begin{bmatrix} 1.5 & 0.5 & 1.25 \\ 0.5 & 1.0 & 0.5 \\ 1.25 & 0.5 & 1.5 \end{bmatrix}$$

P7.19. Correlated Gaussian Sequences. Consider a bandlimited complex Gaussian process with a power spectral density

$$S_{nn}(f) = \begin{cases} 1.0 & \text{for } f_0 - B \leq |f| \leq f_0 \\ 0.5 & \text{for } f_0 \leq |f| \leq f_0 + B \\ 0 & \text{elsewhere } (f_0 \gg B) \end{cases}$$

For the lowpass-equivalent representation of $n(t)$ of the form $\tilde{n}(t) = n_c(t) + jn_s(t)$, find the power spectral densities of the quadrature components and also the cross-correlation between them. Develop a procedure for generating sampled values of the complex samples of the lowpass-equivalent process.

P7.20. Gaussian Process with arbitrary PSD. Generate a temporally correlated sequence of samples from a zero-mean Gaussian process with a PSD

$$S_{YY}(f) = \frac{1}{\sqrt{1 - (f/f_D)^2}} \quad \text{for } |f| < f_D \quad \text{and zero elsewhere}$$

(this is the so-called Jakes Doppler spectrum used in modeling the Doppler spectrum in mobile communication channels). Assume that f_D can range from 10 to 100 Hz. Choose an appropriate sampling rate and implement the spectral shaping filter as an FIR filter.

P7.21. Gaussian Process with a Specified Autocorrelation Function. Suppose we want to generate sampled values of a Gaussian process with an autocorrelation function $R_{XX}(\tau) = \exp(-\alpha|\tau|)$. Choose an appropriate sampling rate as a function of α and develop a 10th-order AR model for generating sampled values of the process. (Find the coefficients of the AR model using sampled values of the autocorrelation function in the Yule-Walker equations.)

P7.22. Chi-Square and KS Tests. Write a program to implement the chi-square and KS goodness-of-fit tests. Assume that the histogram intervals, counts, and the underlying true bin probabilities $p_i, i = 1, 2, \dots, m$, will be specified by the user in a table form. Your program should read a file containing N sampled values of the output of the RNG and apply the chi-square and KS tests at a user-specified value for the significance level.

P7.23. Correlation. Write a program to compute the cross-correlation between two vectors $\mathbf{X}$ and $\mathbf{Y}$ of dimension N . The output should be an array of the normalized correlation coefficients $\rho_{XY}(k)$ defined as

$$\rho_{XY}(k) = \frac{1}{N-k} \sum_{m=1}^{N-k} X(m)Y(m+k), \quad k = 0, 1, 2, \dots, N-1$$

(See Chapter 10 for a DFT-based procedure.)

P7.24. Correlation. Let $X(k)$ and $Y(k)$ be two random sequences with $Y(k) = aX(k-m)$. Derive a procedure for estimating a and m and write a program to implement it (a is a positive constant, m is a positive integer).

P7.25. Correlation. Generate a sequence of 10,000 independent Gaussian numbers. Compute and plot the normalized autocorrelation function of the sequence for lags ranging from 0 to 9500 for three different seeds. Does the RNG exhibit any significant correlation? Do the results differ much as a function of the seed?

P7.26. Correlation Test. Generate $Y(k) = X(k) + X(k+100)$, $k = 1, 2, \dots, 10,000$. Apply the correlation test to $Y(k)$.

P7.27. Miscellaneous. Suppose the intersymbol interference in a communication system can be expressed as

$$Z_k = \sum_{m=1}^3 h_m A_{k-m}$$

where A_j is an independent sequence of octal symbols with values $-7, -5, -3, -1, 1, 3, 5$, and 7 , and h_m represents sampled values of the impulse response with values $h_1 = 0.3, h_2 = -0.1$ and $h_3 = 0.05$. Generate a PN sequence of appropriate length that will produce all possible ISI values. Obtain a histogram of the ISI values. Could the ISI distribution be approximated by a Gaussian? (use the chi-square goodness-of-fit test).

Chapter 8

P8.1. Multitone Source Model. Multiple sinusoids (tones) are sometimes used as a source model. One source that can be emulated by multiple tones is a “white noise load,” which is a band of uniform-PSD Gaussian noise with “notches.” Study via simulation the approach of $\sum_{i=1}^N A_i \cos(\omega_i t + \theta_i)$ to a normal distribution, as a function of N . The phases θ_i are independent and uniformly distributed on $[0, 2\pi]$. Discuss the influence of the A_i on the Gaussianness and on the spectral density of the sum.

P8.2. Convolutional Encoder. Write a program to implement a general convolutional encoder.

P8.3. Viterbi Decoding. Write a program to implement a “practical” version of the Viterbi algorithm; i.e., use soft decisions, truncated path memory, etc. Test your program on a system model.

P8.4. Frequency Modulation. Write a program to simulate a frequency modulator. Interpreting the FM modulator as a nonlinear device (why?), discuss the required sampling rate.

P8.5. M-ary QAM Signal Generation. Assume you have available in your library only binary sequence generators. You want to simulate an M -QAM signal, where M is a perfect square. Show that you can build up an M -QAM modulator hierarchically. In particular, (a) show that you can build a QPSK modulator from two binary sequence generators, and (b) show that you can build a 16-QAM modulator from two QPSK modulators.

P8.6. Modulator Specification for M-QAM. Consider a 64-QAM signal whose constellation has the appearance of Figure 11.36. The ideal location of the symbols is given by the coordinates $(a_m A, b_n A)$, where $a_m, b_n \in \{\pm 1, \pm 3, \pm 5, \pm 7\}$. These ideal locations cannot be exactly realized in hardware. Develop a specification for an actual modulator in terms of a maximum tolerance around each point in the constellation. Assuming an ideal channel and demodulator, establish the degradation in BER performance as a function of the tolerance for two cases: (a) the “worst-case” degradation, and (b) the degradation when the actual symbol location is uniformly distributed over the tolerance range.

P8.7. Quadrature Modulation/Demodulation. Consider the following system. A QAM modulator generates a signal $X(t) = X_1(t) \cos(2\pi f_c t) + X_2(t) \sin(2\pi f_c t)$, which is sent through a linear bandpass channel with transfer function $H_{bp}(f)$. The demodulator multiplies the received signal by $\cos(2\pi f_c t + \theta)$ to produce the in-phase baseband signal $Y_1(t)$, and by $\sin(2\pi f_c t + \theta)$ to produce the quadrature baseband signal $Y_2(t)$. The angle θ is a demodulator reference error (static phase error). (a) Show that the output signals are given by

$$Y_1(t) = X_1(t) * h_{11}(t) + X_2(t) * h_{12}(t)$$

$$Y_2(t) = X_2(t) * h_{22}(t) + X_1(t) * h_{21}(t)$$

where $h_{11} = h_{22}$ and $h_{21} = -h_{12}$ and

$$h_{11}(t) = h_r(t) \cos \theta + h_i(t) \sin \theta$$

$$h_{12}(t) = h_i(t) \cos \theta - h_r(t) \sin \theta$$

where h_r and h_i are, respectively, the real and imaginary parts of the lowpass-equivalent impulse response. (b) Express h_r and h_i in terms of the real and imaginary parts of the

lowpass-equivalent transfer function. [Hint: Use odd and even decomposition (see Section 3.4).]

P8.8. The Complex Envelope of CPM Signals. Write a program to generate the complex envelope of CPM signals. Discuss the required sampling rate.

P8.9. MSK Modulation. A pictorial representation of the possible values of the complex envelope is often called a *signal space diagram*, examples of which are shown in Figure 8.14. (a) Show that the signal space diagram for MSK is a circle. Label the transition from quadrant to quadrant with the corresponding symbol transitions. (b) Write a program to implement MSK in quadrature form. (c) Obtain a signal space diagram from your simulation.

P8.10. Trellis-Coded Modulation. (a) Write a program to implement a (Ungerboeck) trellis encoder using an 8-ary PSK signal constellation and a Viterbi decoder. For a code with parity-check matrix $\begin{bmatrix} x^3 + 1 & x & x^2 \end{bmatrix}$ [see, e.g., R. E. Blahut, *Digital Transmission of Information*, Addison-Wesley, Reading, Massachusetts (1990), pp. 272–273, for the interpretation], an AWGN channel, and matched filter detection in the receiver, plot simulated BER vs. E_b/N_0 for soft decision quantized to 4 and 8 bits; (b) obtain the BER as in part (a), as a function of phase error; (c) place a transmission filter in the path of the signal (to create ISI) and repeat parts (a) and (b); (d) draw the trellis for enough branches to obtain the free distance d_f and the number of paths N_f possessing that distance; (e) compare the various results to the first-order approximation for the BER, $p \approx (1/m)N_f B_f Q(d_f/2\sigma)$, where m is the number of bits/symbol and B_f the average number of bit errors on the divergent paths of distance d_f .

P8.11. Frequency Demodulation. Write a program to implement the following types of frequency demodulator: (a) A discriminator; this device ideally extracts the derivative of the phase of the input; (b) a delay-line discriminator. For the above demodulators, obtain via simulation the output SNR versus the input SNR curve for sinusoidal tone signal. (Hint: There are different ways to implement differentiation in discrete time; see, e.g. Ref. 41 in Chapter 8.)

P8.12. Discrete-Time Differentiation. Prove that a bandlimited differentiator has the discrete-time impulse response given by (8.8.16).

P8.13. Discrete-Time Differentiation. Obtain Equation (8.8.20) and the coefficient values b_k when $m = 3$. Form the table of divided differences for the coefficients a_k .

P8.14. Discrete-Time Differentiation. Consider a random process $X(t)$ with power spectral density $S_X(f) = k_1 \exp(-k_2|f|)$. Pass $X(t)$ through an ideal differentiator: $Y(t) = dX(t)/dt$. Assume $X(t)$ and $Y(t)$ are sampled at a rate f_s samples/s to form discrete-time sequences. Plot the signal-to-aliasing noise ratio for both $X(t)$ and $Y(t)$ as a function of f_s .

P8.15 Equalizer Convergence. (a) Show that the MSE for a tapped-delay-line equalizer is a convex function of the tap gains. (b) Discuss the implications for equalizer convergence. (c) Write a program to implement an MSE equalizer. (d) Experiment with the initial values of the tap coefficients and the coefficient step sizes.

P8.16 Equalizer Implementation. Some systems employ *baseband* equalizers and some use IF (intermediate-frequency) equalizers. Assume a QAM system. (a) Draw the actual block diagram for each type. (b) Draw the simulation (lowpass-equivalent) block diagram for each type.

P8.17 Equalizer Tap Spacings. (a) Discuss the properties of the MSE equalizer as a function of the tap spacing T' . (b) Discuss the implications of T' for the simulation of the equalizer. (Hint: Consider the symbol synchronization aspect of the relation between T' and T_s .)

P8.18 Equalization of Static Phase Error. (a) Show that a TDL equalizer will always compensate for static phase error. (b) In an otherwise undistorted system, how many taps are required?

P8.19 Baseband Equalization by Channel Covariance Matrix Inversion. Consider a baseband four-level PAM signal operating through a linear channel with finite impulse response $h_0 = 1, h_2 = -0.1, h_k = 0, k > 2$. Find the optimal tap gains for an MMSE equalizer by the covariance matrix inversion method. What should be the number of taps of the equalizer?

P8.20 Bandpass Equalization by Channel Covariance Matrix Inversion. Consider a QPSK system. The channel is linear. Assume the channel transfer function is represented by a five-pole, Chebyshev bandpass filter, with $\alpha_p = 0.5$ dB, $f_p = 0.6R_s$, and $f_c/R_s = 10$, where α_p is the passband ripple parameter, f_p is the passband edge frequency, R_s is the symbol rate, and f_c is the carrier frequency. (a) Obtain the (complex) response to a unit pulse at the input to the I channel. (b) Find the optimal (complex) tap gains for a 5-tap, 7-tap, and 9-tap MMSE equalizer by the covariance matrix inversion method. (c) Evaluate the BER with and without equalization using a QA technique (see Sections 11.2.7 and 12.1).

P8.21 Equalized Mean-Square Error. Show that the minimum mean-squared error of the LMS equalizer is given by (8.9.37).

P8.22 PSK Demodulation Phase Ambiguity Resolution. (a) Show that data-derived phase estimation for M -ary PSK using an M th-power loop produces an M -fold ambiguity. (b) How would you implement your simulation so as to resolve this ambiguity?

P8.23 Binary CPFSK Demodulation. Consider a binary CPFSK signal with deviation ratio $h = n/m$, where n, m are integers. (a) Show that, in principle, a phase and timing recovery structure can be implemented by first raising the signal to the m th power, followed by two phase-locked loops centered at $mf_c \pm n/2T$, where f_c is the carrier frequency and T the symbol duration. The PLLs are followed by a multiplier, the output of which is fed to two bandpass filters, one at $2mf_c$ and the other at n/T . Show the block diagram, including the processing following the bandpass filters that must be done to recover the carrier and the clock. (b) How might you implement such a structure in simulation without explicitly simulating phase-locked loops (treat them as narrowband filters)?

P8.24 Phase Noise Equivalent Process. The residual phase noise in a demodulated signal is due to the sum of *tracked* thermal noise and *untracked* oscillator noise. Write a program to generate this residual noise (assuming its statistics are Gaussian) by using a “white” noise source and appropriate filtering; assume your specifications are a spectral shape and an rms value for each component.

P8.25 Phase Noise and Block Coding. Show that correlated noise, such as phase noise, has a worse effect on block-coded performance than “wideband” noise. Consider the case of an idealized OQPSK system only AWGN corrupts the system, except for phase noise at the demodulator; prior to the decision device. Formulate the (average) probability of bit error for a block-coded system with hard decisions. Postulate a reasonable decoding rule. For simplicity you may assume the correlated noise is slow enough to remain essentially constant over a code block.

P8.26. Carrier Phase Synchronization. Develop a lowpass-equivalent simulation to obtain the behavior of a “times-four” loop. Consider the use of multirate techniques, as appropriate. The system configuration is as follows. The input $r(t)$ is the sum of a QPSK signal $s(t)$ and additive white Gaussian noise $n(t)$. This sum is input to a “front-end” filter with transfer function $H_1(f)$, then into a “quadrupling device,” which delivers $r^4(t)$. This signal is input to a narrowband filter $H_4(f)$, the output of which is the input to a second-order

PLL with $\zeta = 0.707$. The carrier frequency is initially the same as the rest frequency of the VCO, and the latter is assumed ideal. (a) Plot the phase error as a function of time for several values of input E_b/N_0 and for several combinations of bandwidths for $H_1(f)$, $H_4(f)$, and ω_n ; choose reasonable values for these parameters; (b) repeat part (a) with different values of rms phase noise on the carrier; choose a “representative” spectrum for the phase noise, and model it as a Gaussian random process. (c) Repeat part (b) with different values of the input carrier from the VCO rest frequency.

P8.27. Digital Phase-Locked Loops. Investigate the implementation of digital phase-locked loops (DPLL). (a) Write the nonlinear difference equation governing a DPLL. (b) Discuss the simulation of a DPLL. (c) Discuss the relationship between the simulation of an APLL (i.e., a discrete-time approximation) and a DPLL.

P8.28. Calibration. Find the energy in the impulse response for a raised-cosine filter and for a square-root raised-cosine filter as a function of the excess bandwidth factor β .

P8.29. Calibration. Describe in detail a calibration procedure for establishing the decision regions at the receiver for QAM signaling and for PSK signaling.

P8.30. Calibration. Establish the correct demodulated I- and Q-channel noise spectral densities for a single-sided noise PSD at RF of N_0 . Assume a demodulator constant equal to $2k_d$.

P8.31. Calibration. Set up a calibration procedure for determining the noise power at a detector filter (“matched” filter) output given the noise PSD at the receiver input. Consider determining the noise bandwidth by inserting a discrete impulse at the receiver input.

P8.32. Calibration for QA Simulation. Discuss in detail how you would calibrate your simulation for simulating an M -QAM system using the QA technique. In particular, describe how you would set the standard deviation of the equivalent Gaussian noise source which is added analytically to the signal at the input to the “virtual” decision device. Consider evaluating the receiver noise bandwidth by simulation means and by analytical means. Consider evaluating “ E_b ” in two ways: one that uses the transmitted signal (e.g., a PN sequence), and one that is independent of the actual signal sent. One might be called “received” E_b and the other “available” E_b ; discuss the meaning and usefulness of one or the other.

Chapter 9

P9.1. Uncorrelated Scattering Channel. It is often stated that for most physical channels, the channel correlation function can be written as $R_{\tilde{c}}(\tau_1, \tau_2, \Delta t) = R_{\tilde{c}}(\tau_1, \Delta t)\delta(\tau_1 - \tau_2)$. This reflects the uncorrelated scattering assumption. Develop a plausibility argument for supporting this assertion.

P9.2. Calibrating Simulation of Fading Channels. You want to simulate a frequency-selective Rayleigh fading channel. Describe step by step how to calibrate your simulation both with respect to the fading channel and the receiver noise. State what information you will need in order to do your calibration. Assume you have both shadow fading and multipath.

P9.3. More on Calibration. For the separable model of the scattering function $S(\tau, \nu) = p(\tau)S(\nu)$ [see Equations (9.1.40)], assuming a Jakes spectrum, $S(\nu) = A/[\pi f_D \sqrt{1 - (\nu/f_D)^2}]$, determine the constant A necessary to normalize the power in $S(\nu)$ to unity.

P9.4. Fading Channel Impulse Response. Verify that the impulse response corresponding to the Jakes channel model is given by Equation (9.1.65).

P9.5. Correlated Tap-Gain Model. Show that the matrix $\mathbf{R}_0$ whose entries are defined by (9.1.42b) is positive-definite.

P9.6. Filtered Channel Responses. Show that Equations (9.1.53)–(9.1.57) are true.

P9.7. Diffuse Correlated Tap-Gain Model. For the model given by (9.1.42), with $p(\tau) = e^{-\tau^2/T^2}$ and $T = 1/B$, (a) compute the matrix $\mathbf{R}_0$ and (b) obtain the corresponding lower triangular matrix (Cholesky decomposition) $\mathbf{L}$ defined by (9.1.47b).

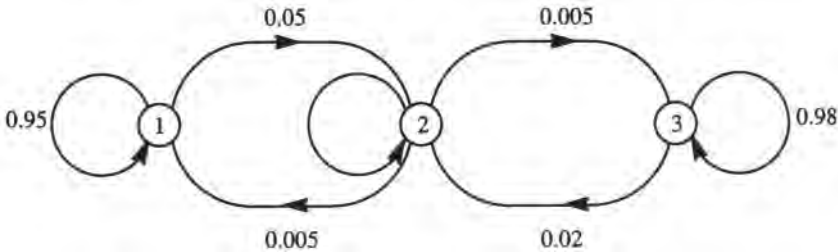
P9.8. Discrete-Channel Tap-Gain Model. Given a discrete-channel model with a delay power profile for hilly terrain recommended as a reference model for GSM application [Appendix to Chapter 9, Table 9.A.2, column (1)], establish an N -tap delay line model along the lines of (9.1.61), with $T = 1/B = 5 \mu\text{s}$. What is a reasonable value for N ?

P9.9. Rummmler Model. Prove that the amplitude and phase characteristics for the Rummmler channel model are given by (9.1.75a) and (9.1.75b), respectively.

P9.10. Rummmler Model. Demonstrate that for the Rummmler channel model, (9.1.74), the transfer function is minimum-phase when $b < 1$ and nonminimum-phase for $b \geq 1$. Discuss the implications of the latter type for equalization.

P9.11. Discrete Channel: Baum–Welch (BW) Algorithm. Write a program to estimate the parameters of a Markov model from an error sequence using the BW algorithm. Assume that the number of states and an initial set of parameter values are given.

P9.12. Discrete Channel: Baum–Welch Algorithm. Generate an error sequence of 100,000 bits using the Markov model given below:



$P[\text{bit error}|\text{state } 1] = 0.0$, $P[\text{bit error}|\text{state } 2] = 0.001$, and $P[\text{bit error}|\text{state } 3] = 1.0$. Choose initial state probabilities to satisfy the stationarity condition. (a) Estimate the parameters of the model from the error sequence using the BW algorithm; assume that the number of states is given ($= 3$); assume a suitable initial guess of parameter values that are “close” to the true values of the parameters of the Markov model. (b) Repeat with an initial guess that is not very close to the true values. (c) Compare the true values of the parameters and the estimated values.

P9.13. The decision metric in a binary communication system has the form

$$Z_k = A_k S_k + n_k$$

where $S_k = \pm 1$ is the symbol (bit) value, n_k is the noise, which is assumed to be an independent sequence of $N(0, 0.1)$, and A_k is the time-varying attenuation due to fading. The sampled values of the attenuation A_k is modeled as a random sequence,

$$A_k = 1.0 + R_k; \quad R_k = N(0, 0.25); \quad E\{R_k R_{k+m}\} = \exp(\alpha|m|); \quad \alpha = 0.01$$

(a) Outline a procedure for *deriving* a three-state Markov model for this case. (b) Simulate 100,000 bits through this model and generate the error sequence and estimate the parameters of a three-state Markov model for this system.

Chapter 10

P10.1. Average Level Estimation. Assume a waveform has the properties of a Gaussian process with average value μ and variance σ^2 . Derive an expression for a two-sided confidence interval for μ as a function of the number of sampled waveform values.

P10.2. Average Power Estimation. Show that the pdf of $(\hat{P}_{av}|s(t))$, the average power estimator conditioned on the signal, is noncentral gamma.

P10.3. Estimation of Distribution Function. The Kolmogorov–Smirnov (KS) test is a goodness-of-fit test for distributions (see, e.g., Ref. 5 of Chapter 10). It is used to test assumptions regarding the true distribution approximated by the empirical distribution. (a) Develop a program to apply the KS test to a sequence of “random” numbers. Use your program to test your favorite random number generator (see also Chapter 7).

P10.4. Histogram Construction. Write a program that will generate a histogram from simulation-generated data. (Note: most commercial math analysis/simulation packages have a built-in histogram function.)

P10.5. Histogram. Generate 100 samples of a uniformly distributed random variable, and compute a histogram of the 100 samples using a bin width of 0.05. (a) Plot the histogram; does it “look” uniform? (b) Repeat for different values of the seed for the uniform RNG. (c) Repeat for a sample size of 1000.

P10.6. Estimation of Two-Dimensional pdf. Find the form of an estimator of a two-dimensional probability density function and establish its properties.

P10.7. Power Spectral Density Estimation. (a) Show that the variance of the periodogram does not decrease with N , i.e., that Equations (10.5.24) and (10.5.25) are true for a Gaussian process. (b) Verify (a) through simulation.

P10.8. Power Spectral Density Estimation. (a) Write a program to implement a smoothed PSD estimator; incorporate at least one window. (b) Apply the estimator to a system composed of a cascade of a source, a modulator, a filter, and a memoryless nonlinearity. (c) Obtain and compare the PSD at the output of the modulator, the filter, and the nonlinearity.

P10.9. Power Spectral Density Estimation. Another method of reducing the variance of a PSD estimator, aside from the Bartlett or Welch periodograms, is by smoothing in the frequency domain. The smoothed (discrete) frequency-domain estimator $\tilde{G}_{XX}(k)$ is defined as a running average, $\tilde{G}_{XX}(k) = [1/(2M+1)] \sum_{l=-M}^M \tilde{P}_{XX}(k+l)$, where $\tilde{P}_{XX}$ is the unsmoothed periodogram. (a) Show that $\tilde{G}_{XX}$ has decreased variance, increased bias, and decreased resolution, for a fixed number of sample points. (b) Verify (a) through simulation.

P10.10. Power Spectral Density Estimation. Generate a set of samples $\{X(k)\}$, $k = 0, 1, \dots, N-1$, of a stationary random process at a sampling rate f_s samples/s. Estimate the PSD of the process as follows. Let $F(k)$, $k = [-N/2], \dots, 0, \dots, [N/2-1]$, be the FFT of $\{X(k)\}$. Let $G(k) = a|F(k)|^2$, where a is chosen to ensure

$$\frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 T_s = \sum_{k=-N/2}^{N/2-1} a |F(k)|^2 \Delta f$$

where $T_s = f_s^{-1}$ and $\Delta f = (NT_s)^{-1}$. Obtain the smoothed PSD $\tilde{G}_{xx}(k)$ using the method of problem P10.9 for $M = 2$ and display the spectral plot $10 \log \tilde{G}_{xx}(k)$ versus k .

P10.11. Power Spectral Density Estimation. Generate a random binary sequence of 512 bits with $\Pr(\text{zero}) = \Pr(\text{one}) = 1/2$. Encode these bits, using a sampling rate of 16 samples/bit, into the following formats: (a) NRZ; (b) RZ with 50% pulse width. Estimate the PSD of these waveforms using the PSD estimator of problem 10.9.

P10.12. Power Spectral Density Estimation. Encode a sequence of 512 bits of a random binary waveform into an “unbalanced” NRZ waveform in which “0” is represented by amplitude -1 for T_0 seconds (or N_0 samples) and “1” is represented by an amplitude of $+1$ for T_1 seconds (or N_1 samples). Estimate the PSD of this waveform and compare it to that of the balanced NRZ waveform. Choose different values of the parameters; start with $T_0 = 7/16$, $T_1 = 9/16$, $T_s = 1/16$.

P10.13. Estimation of Phase and Delay. Show that the estimates of delay and phase given by the cross-correlation method are identical to the maximum-likelihood estimators when the system has unknown gain, delay, and phase, but is otherwise undistorted.

P10.14. Estimation of Phase and Delay. Write a program to obtain the phase and delay of a system by using the cross-correlation technique. Implement the cross-correlation via FFT methods. (See also Problems P7.21 and 7.22.)

P10.15. Eye Pattern/Scatter Plot. Write a program that will generate an eye pattern. (b) Write a program that will generate a scatter plot.

P10.16. Eye Diagram. (a) Generate the eye diagram for a binary sequence when the received pulse corresponding to $+1$ is given by

$$g(t) = \begin{cases} 0, & t < 0 \\ 1 - e^{-t/T}, & 0 \leq t \leq T \\ (1 - e^{-t/T})e^{(t-T)/T}, & t \geq T \end{cases}$$

taking into account the ISI generated over five pulse intervals; (b) repeat part (a) for a four-level signal $\pm 1, \pm 3$.

Chapter 11

P11.1 SNR Estimation. (a) Show that the SNR definition that minimizes (11.1.6) implies (11.1.7) and (11.1.8). Also derive (11.1.9) and thence (11.1.14). Implement the SNR estimator in your simulation.

P11.2. SNR Estimation. Prove that the optimum delay (the delay that maximizes the cross-correlation between input and output of a black box) is given by the DFT procedure indicated in (11.1.20). Implement the optimum delay estimator in your simulation.

P11.3. SNR Estimation. This problem extends the coverage in Section 11.1 to complex signals. Let $s(t)$ be a periodic complex waveform and $\mathbf{s} = (s(kT_s))$, $k = 1, \dots, N$, its sampled sequence. Let $x(t)$ be the corresponding (complex) output and $\mathbf{x} = (x(kT_s))$, $k = 1, \dots, N$, its sampled sequence. For d an integer between 0 and $N - 1$, define the right-shifted sequence $\mathbf{s}_d = (s(N - d - 1), \dots, s(1), s(2), \dots, s(N - d))$, T_s being understood in the arguments. For row vectors $\mathbf{a}$, $\mathbf{b}$ the inner product is defined as $\mathbf{a} \cdot \mathbf{b}^H$ and the norm squared of $\mathbf{a}$, $\|\mathbf{a}\|^2 = \mathbf{a} \cdot \mathbf{a}^H$, where the superscript H indicates complex conjugate transpose. Let $\mathbf{y} = A\mathbf{s}_d$, where A is a complex scalar, and define $u(A, d) = \|\mathbf{x} - A\mathbf{s}_d\|^2 = \|\mathbf{x} - \mathbf{y}\|^2$. If A_* , d_* minimize $u(A, d)$, then the SNR is given by $|A_*|^2 \|\mathbf{s}\|^2 / u(A_*, d_*)$. (a) Show that $u(A, d)$

is minimized by the choice of A and d that maximize $\text{Re}(\mathbf{x} \cdot \mathbf{y}^H)$. (b) Show that this is equivalent to choosing the A and d that maximize $|\mathbf{x} \cdot \mathbf{y}^H|$. (c) Show that $\mathbf{A}_* = \mathbf{x} \cdot \mathbf{s}_d^H / \|\mathbf{s}\|^2$.

P11.4. The Monte Carlo Method for Evaluating Integrals. We have seen [e.g., in (11.2.1)] that estimating the BER is equivalent to evaluating an integral. Typically we do not know the integrand, and in the MC method we evaluate the integral by observing its randomly generated values. The MC method can also be used to evaluate integrals with known integrands that might otherwise be difficult to solve. To illustrate the method, evaluate by MC means the integral

$$I = \int_0^5 x^3 e^{-x} dx$$

Use several sets of random numbers with different sizes (e.g., 100, 1000) and compare to the true value. *Hint:* Write I as an expectation.

P11.5. Constructing a Monte Carlo BER Curve. (a) Devise a smooth interpolation routine to “best-fit” a BER curve from a set of Monte Carlo estimates at different values of E_b/N_0 ; take into account the relative reliability of these different estimates. (b) Apply the results of part (a) to an actual simulation.

P11.6. Confidence Interval for BER. (a) Formulate a one-sided confidence interval for BER based on the binomial distribution and on the normal approximation. (b) Obtain the one-sided confidence intervals for 90%, 95%, and 99% confidence levels. (c) Discuss the implications of two-sided versus one-sided confidence intervals for the BER.

P11.7. BER Estimation for a Symbol with a Random Number of Occurrences. Consider the MC BER estimator (11.2.3) for a particular symbol when the total number of such transmitted symbols is random. (a) Show that this estimator is unbiased. (b) Show that the estimator variance is given by (11.2.24).

P11.8. Dependent Errors. Starting with (11.2.27), show that (11.2.28) is true.

P11.9. Sequential Estimation for BER. Derive Equation (11.2.32a).

P11.10. Tail Extrapolation. (a) Derive the form of the tail extrapolation estimator, (11.2.46) and (11.2.47). (b) Quantify the error term $\epsilon(t)$ as a function of the exponent parameter $\mathbf{v}$. (c) Show that the best-fit tail extrapolation estimator with three equally spaced pseudothresholds (in decibel domain) is given by (11.2.50). (d) Write a program to implement the tail extrapolation estimator.

P11.11. Bias of the IS Estimator. (a) Show that the IS estimator (11.2.58) for the output version is unbiased. (b) Show that the IS estimator for the input version (11.2.65) is unbiased if no impulse response truncation takes place. (c) Show that if impulse response truncation occurs, there must be some estimator bias.

P11.12. Variance of the IS Estimator. (a). Obtain the variance of the BER estimator (11.2.59) for the input version of IS. (b) Obtain the variance of the BER estimator (11.2.70) for the output version of IS.

P11.13. Conditional Importance Sampling. The form of IS conditioned on simulating only a relatively few (worst) ISI patterns requires the identification of those patterns. For a linear system, suppose you had the response $p(t)$ corresponding to a 1; set up a recursive computation to rank the ISI patterns.

P11.14. Variance of the CIS and IIS Estimators. Derive Equations (11.2.73a) and (11.2.73b) in Example 11.2.5.

P11.15. Optimum Value of Bias for CIS. Suppose we want to estimate $P(X > T)$, $X \sim \mathcal{N}(0, 1)$, using conventional importance sampling. Derive an expression for variance of the estimator as a function of T and sample size N . Find the optimum value of the bias, i.e., the factor by which the variance should be increased.

P11.16. CIS with Memory. Find the optimum biasing for X_1, X_2 in order to evaluate $P(X_1 + X_2 > T)$, with $X_i \sim \mathcal{N}(0, \frac{1}{2})$. Compare the variance reduction possible to that in Problem P11.15.

P11.17. Comparing Different Estimation Methods. Consider a binary communication system in which a random binary source (± 1 V) sampled at 8 samples/bit is added to a noise source $\mathcal{N}(0, \frac{1}{2})$, and the sum input to a detection filter with impulse response

$$h(kT_s) = \begin{cases} 1, & 0 \leq t \leq T_b \\ -0.1, & T_b \leq t \leq 2T_b \end{cases}$$

where T_b is the bit duration. The output is sampled at multiples of T_b in order to make a decision. Estimate the probability of error using the following techniques. (a) An analytical (exact) expression; (b) Monte Carlo, with 10^5 bits; (c) quasianalytical, with 100 random bits; (d) conventional importance sampling with 10 random bits; (e) improved importance sampling with 10 random bits. Compare the estimated values to one another. Repeat the simulations 10 times and estimate the bias and variance of the estimators.

P11.18. QA Method for PAM. Derive equations for use in QA simulation that give the specific error rates for PAM (Section 11.2.7.3). The specific error rate is the probability of deciding on a particular symbol other than the transmitted one

P11.19. QA Method for QAM. Derive an equation for QA application for the error rate of symbols on the periphery of a QAM rectangular constellation.

P11.20. QA Method for PSK. Show that Equation (11.2.101) is true. Bound the error analytically, and estimate the computational requirement for the bound relative to the exact result (11.2.99).

P11.21. QA Method with ISI. Consider a binary system with received waveform containing ISI and additive Gaussian noise

$$s(t) = \sum_{k=-\infty}^{\infty} a_k g(t - kT) + n(t)$$

with $a_k = \pm 1$. (a) Show that the average BER is given by

$$p = E \left\{ \frac{1}{2} \operatorname{erfc} \left[\frac{g(t_0) + \sum_{k \neq 0} a_k g(t_0 - kT)}{\sqrt{2}\sigma} \right] \right\}$$

where σ^2 is the variance of the noise. (b) For SNR varying from 5 to 20 dB, estimate p by simulation when the number of ISI terms is equal to eight (four on either side of t_0) for the pulse

$$g(t) = \frac{\sin(\pi t/T) \cos(0.5\pi t/T)}{\pi t/T - 1 - t^2/T^2}$$

(c) Repeat part (b) with 12 ISI terms.

P11.22. QA Method with Interference. The average error rate of a binary PSK signal $A \cos(2\pi f_c t + \phi_k)$ corrupted by additive Gaussian noise and cochannel interference

$I \cos(2\pi f_c t + \phi_i + \theta)$, where ϕ_k and ϕ_i are the respective modulations, and θ is taken to be uniformly distributed on $(0, 2\pi)$, is given by

$$p = E \left[\frac{1}{2} \operatorname{erfc} \left\{ \frac{A + I \cos \theta}{\sqrt{2\sigma}} \right\} \right]$$

where σ^2 is the variance of the noise. Evaluate this expression by simulation for various values of SNR, and plot P_N , the running simulation average, versus N , the current number of generated values of θ . Observe the rate at which p_N tends to the “true” value [for the latter, see, e.g., V. K. Prabhu, Error rate considerations for coherent phase-shift-keyed systems with co-channel interference, *Bell Syst. Tech. J.* **48**(3), 1304–1310 (1969)].

P11.23. Error Probability Estimation by Statistical Averaging Method. Write a program to determine the BER via the method of averaging a QA estimate (obtained without phase and timing jitter) over a distribution of phase and timing errors (the technique discussed in Section 11.2.7.8).

P11.24. QA Method for BER Evaluation of a System. Consider a system which consists of the following cascade of elements: A transmit filter (six-pole Butterworth with $BT = 1.0$); a nonlinear amplifier with characteristics given as follows (BO = backoff):

Input BO	Output BO	Output phase
−18	−18	0°
−15	−14	1°
−12	−11	2°
−9	−7	3°
−6	−4	5°
−3	−2	8°
0	0	10°
3	−2	13°

an additive noise source with PSD $N_0/2$; and a receive filter (four-pole Butterworth), with variable BT . (a) For an MPSK signal (try a couple values of M), plot BER as a function of receiver BT in the range 0.5, ..., 1.2, for different values of E_b/N_0 . Find the optimum receiver bandwidth, and plot BER versus E_b/N_0 for the optimized filter. How different is the optimum bandwidth for different E_b/N_0 ? (b) Now input a 16-QAM signal, but fix the receive BT at 0.7. Run simulations for 0, −3, −6, −9 dB input backoff. For a given E_b/N_0 at saturation, plot the BER as a function of backoff and determine the optimum backoff for several values of E_b/N_0 .

P11.25. QA Method with Coding. Obtain an expression for the error magnification in the BER estimate using the transfer function bound, if the channel transition probabilities are obtained from MC simulation.

This page intentionally left blank

A Collection of Useful Results for the Error Probability of Digital Systems

It is a standard part of methodology to check the results of a simulation by comparing them to certain known benchmarks. This is useful both for debugging and validation purposes. In the first instance, of course, we set up the simulation so that it *should* yield the known results. These known results typically apply to theoretical models that are idealizations of real systems. Therefore, strictly speaking, they cannot “validate” a simulation of a real system, but they do provide a “sanity check” in the sense that they serve as lower bounds on the (error probability) performance of real digital systems. Furthermore, if the simulated results differ from the theoretical results by an amount that seems excessive, that may be symptomatic of inadequate models. Therefore, as an aid to this checking process, we present in this Appendix a short collection of known formulas, approximations, bounds, and curves for the error probability for a variety of digital communication systems.

The general setting for these known results is as follows. During a signaling interval, the transmitter selects one of M waveforms $\{s_m(t)\}$, $m = 1, 2, \dots, M$, which can be represented in the form

$$s_m(t) = \text{Re}\{u_m(t)e^{j2\pi f_c t}\}, \quad m = 1, 2, \dots, M, \quad 0 \leq t \leq T \quad (\text{A.1})$$

where $u_m(t)$ is the complex equivalent lowpass waveform (for practical purposes, the same as the complex envelope). Each waveform (symbol) is characterized by an energy per symbol

$$E_m = \int_0^T s_m^2(t) dt = \frac{1}{2} \int_0^T |u_m(t)|^2 dt, \quad m = 1, 2, \dots, M \quad (\text{A.2})$$

and related to other waveforms by a cross-correlation coefficient

$$\rho_{jm} = \frac{1}{2(E_j E_m)} \int_0^T u_j(t) u_m^*(t) dt \quad (\text{A.3})$$

The average energy per symbol, assuming equiprobable symbols, is given by

$$E_s \triangleq E = \frac{1}{M} \sum_{m=1}^M E_m \quad (\text{A.4})$$

and the average power by

$$P_{av} = E/T \quad (\text{A.5})$$

We assume typically that $M = 2^k$, that is, each transmission corresponds to k bits of information. Then, the *energy per bit* E_b is

$$E_b = \frac{1}{k} E \quad (\text{A.6})$$

The noise is assumed white (or uniform PSD) Gaussian with one-sided noise power spectral density N_0 in W/Hz. The common signal-to-noise measure that is used for comparing one scheme against another is the ratio of energy per bit to noise power spectral density, namely,

$$\rho_b \triangleq E_b/N_0 \quad (\text{A.7})$$

We can also interpret (A.7) as

$$\rho_b = \frac{E_b}{N_0} \cdot \frac{kT}{kT} \quad (\text{A.8a})$$

$$= P_{av}/N_0 R_b \quad (\text{A.8b})$$

namely, as the average signal-to-noise ratio in a bandwidth equal to the (total) bit rate, $R_b = k/T$.

In the preceding, it has been implicitly assumed that the channel is distortionless, with unity gain. If the channel has gain α , that is, the received signal is $\alpha s_m(t)$, then it is sensible to define ρ_b as the *received* value of E_b/N_0 , i.e.,

$$\rho_b = \alpha^2 E_b/N_0 \quad (\text{A.9})$$

and if α is not constant, as in a fading channel, then we are interested in the average value of (A.9),

$$\bar{\rho}_b = E(\alpha^2) E_b/N_0 = E(\alpha^2) \rho_b \quad (\text{A.10})$$

where ρ_b here is the value corresponding to $\alpha = 1$.

A comment on notation. Results for a number of cases are given in terms of the function erfc , which is defined as

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-u^2} du \quad (\text{A.11})$$

The same results are also often expressed in terms of a function Q , defined as

$$Q(y) = \frac{1}{(2\pi)^{1/2}} \int_y^\infty e^{-u^2/2} du \quad (\text{A.12})$$

These two formulations are related by

$$Q(y) = \frac{1}{2} \operatorname{erfc}(y/\sqrt{2}) \quad (\text{A.13a})$$

or

$$\frac{1}{2} \operatorname{erfc}(x) = Q(\sqrt{2}x) \quad (\text{A.13b})$$

In Table A.1, we use the notation P_b to indicate the probability of *bit* error, and P_M to indicate the probability of error of an M -ary symbol.

Table A.1. Formulas, Bound, and Approximations for Error Probabilities

M-ary coherent PSK ($M = 2^k$)		Ref. 1
1.	$M = 2$	$P_b = \frac{1}{2} \operatorname{erfc}(\sqrt{\rho_b})$ Figure A.1(F)
2.	$M = 4$	$P_b = \frac{1}{2} \operatorname{erfc}(\sqrt{\rho_b}), \rho_b = \rho/2$
3.	$M > 2$	$P_M = 1 - \int_{-\pi/M}^{\pi/M} p(\theta) d\theta$ Figure A.1 (A, B, C)
where		
$p(\theta) = \frac{1}{2\pi} e^{-\rho} \left[1 + (4\pi\rho)^{1/2} \cos \theta e^{\rho \cos^2 \theta} \times \frac{1}{2\pi} \int_{-\infty}^{(2\pi)^{1/2} \cos \theta} e^{-u^2/2} du \right]$		
3.1.	$M > 2$	$P_M \approx \operatorname{erfc}[(k\rho_b)^{1/2} \sin \frac{\pi}{M}]$
3.2.	$M > 2$	$P_b \approx \frac{1}{k} P_M = \frac{1}{k} \operatorname{erfc}[(k\rho_b)^{1/2} \sin \frac{\pi}{M}]$
Partially coherent MPSK (noisy phase reference)		Refs. 2, 3
4.	$M \geq 2$ General formulation:	
$P_m = 1 - \int_{-\pi}^{\pi} \int_{-(\pi/M)}^{(\pi/M)} f(\theta \phi) P(\phi) d\theta d\phi$ $= \int_{-\pi}^{\pi} P_M(\phi) p(\phi) d\phi$		
where ϕ is phase error of local oscillator reference		
5.	$M = 2$ Auxiliary carrier reference; first order PLL phase-error statistics	
$P_b = \int_{-\pi}^{\pi} \frac{1}{2} \operatorname{erfc}(\sqrt{\rho_b} \cos \phi) \frac{\exp(\alpha \cos \phi)}{2\pi I_0(\alpha)} d\phi$		Figure A.2 (A–E)
6.	$M = 2$ Squaring loop, first-order PLL:	
$P_b = \int_{-\pi}^{\pi} \frac{1}{2} \operatorname{erfc}\left(\sqrt{\rho_b} \cos \frac{\phi}{2}\right) \frac{\exp(\alpha \cos \phi)}{2\pi I_0(\alpha)} d\phi$		Figure A.2 (F–J)

(continued)

Table A.1. (continued)

Partially coherent MPSK (noisy phase reference), cont'd.		Ref. 1
where		
$\alpha = \rho_b \frac{\delta}{4}; \delta = \frac{R_b}{B_L} \left(1 + \frac{W/R_b}{\rho_b} \right)^{-1}$		
W is the input bandwidth; R_b is the bit rate; B_L is the one-sided loop bandwidth		
7.	$M = 4$ Quadrupling loop; first-order PLL:	Ref. 4
$P_b = \int_{-\pi}^{\pi} \{ \operatorname{erfc}[\sqrt{\rho} \cos((\pi + \phi)/4)] + \operatorname{erfc}[\sqrt{\rho} \sin((\pi + \phi)/4)] \} \times \frac{\exp(\alpha \cos \phi)}{2\pi I_0(\alpha)} d\phi$		
where $\alpha = \rho_b \beta$ and β is a complicated function given in Ref. 4		
8.	Any M M th power loop or M -phase Costas:	
$P_M = \int_{-\pi}^{\pi} P_M(\phi/M) p(\phi) d\phi$		
where $P_M(\phi/M)$ is error probability conditioned on phase error ϕ/M		
Binary PSK with noisy symbol synchronization		Ref. 3
9.	$P_b = \int_{-0.5}^{0.5} 0.25 \{ \operatorname{erfc}(\sqrt{\rho_b}) + \operatorname{erfc}[\sqrt{\rho_b}(1 - 2 \lambda)] \} \times \frac{\exp[\cos 2\pi\lambda/(2\pi\sigma_\lambda^2)]}{I_0[(1/2\pi\sigma_\lambda^2)]} d\lambda$	
where σ_λ^2 is the variance of the normalized timing error		
M -ary coherent PSK with interference		Ref. 5
10.	$M > 2$	$\frac{(M-1)}{M} \hat{P}_M \leq P_M \leq \hat{P}_M$
where		
$\hat{P}_M = \int_{-\Omega}^{\Omega} \operatorname{erfc}[(k\rho'_b)^{1/2} [\sin(\pi/M) + R]] p(R) dR$		
$p(R)$ is the pdf of R , where $R = \sum_{j=1}^K r_j \cos \lambda_j$; $\Omega = \sum_{j=1}^K r_j$; r_j is the interference-to-signal ratio for j th interferer; λ_j is the phase of j th interferer relative to phase of unmodulated desired carrier; $\rho'_b = (E_b/N_0)/WT$; W is input bandwidth; T is symbol duration; K is the number of interferers		
10.1.	$M = 2$	Exact expression for P_2 is 0.5 times above integral
11.	$M > 2$	$P_M \leq \operatorname{erfc}[(k\rho'_{b,\text{eq}})^{1/2} \sin(\pi/M)]$ $\rho'_{b,\text{eq}} = (E_b/N_{0,\text{eq}})/WT; N_{0,\text{eq}} = N_0(1 + r);$

Table A.1. (continued)

<i>M</i> -ary coherent PSK with interference, cont'd.		
<i>r</i> is the total interference-to-signal ratio (Note: Above bound is equivalent to replacing interference by equal power thermal noise)		
Differentially encoded/differentially decoded binary signaling		
12.	$P_b = 2P_2(1 - P_2)$, where P_2 is the error probability of any coherent binary scheme	
<i>M</i> -ary PAM signaling		Ref. 1
13. Any <i>M</i>	$P_M = \frac{M-1}{M} \operatorname{erfc}\left(\frac{3}{M^2-1} k\rho_b\right)^{1/2}$	Figure A.1 (F-I)
<i>M</i> -ary QAM-rectangular constellation; <i>k</i> even		Ref. 1
14. $M \geq 4$	$P_M = 2\left(1 - \frac{1}{\sqrt{M}}\right) \operatorname{erfc}\left(\frac{3}{2(M-1)} k\rho_b\right)^{1/2} \\ \times \left[1 - \frac{1}{2}\left(1 - \frac{1}{\sqrt{M}}\right) \operatorname{erfc}\left(\frac{3}{2(M-1)} k\rho_b\right)^{1/2}\right]$	Figure A.1 (F, J, K)
15.	$P_M \leq 2 \operatorname{erfc}\left(\frac{3}{2(M-1)} k\rho_b\right)^{1/2}$	
<i>M</i> -ary coherent orthogonal signaling ($M = 2^k$)		Ref. 1
16. Any <i>M</i>	$P_M = \frac{1}{(2\pi)^{1/2}} \int_{-\infty}^{\infty} \{1 - [1 - 0.5 \operatorname{erfc}(u/\sqrt{2})]^{M-1}\} \\ \times e^{[-u - (2\rho)^{1/2}]^2} du$	Figure A.3 (A-F)
16.1.	$P_b = \frac{2^{k-1}}{2^k - 1} P_M$	
17.	$P_M \leq \frac{M-1}{2} \operatorname{erfc}[(\rho/2)^{1/2}], \text{ where } \rho = k\rho_b$	
Differentially encoded/differentially detected PSK (DPSK)		Ref. 1
18. $M = 2$	$P_b = \frac{1}{2} \exp(-\rho_b)$	Figure A.1 (D)
19. $M = 4$	$P_B = \int_b^{\infty} u \exp[-(a^2 + u^2)/2] I_0(ab) du \\ - \frac{1}{2}(ab) \exp[-(a^2 + b^2)/2] \\ a \approx (1.08)(\rho_b/2)^{1/2}; b \approx (2.61)(\rho_b/2)^{1/2}$	Figure A.1 (E)
<i>M</i> -ary noncoherent orthogonal signaling		Ref. 1
20. $M = 2$ (FSK with frequency separation a multiple of $1/T$)	$P_b = \frac{1}{2} \exp(-\rho_b/2)$	Figure A.3 (K)
21. Any <i>M</i>	$P_M = \sum_n^{M-1} (-1)^{n+1} \binom{M-1}{n} \frac{1}{n+1} e^{-nk\rho_b/(n+1)}$	Figure A.3 (G-J)

(continued)

Table A.1. (continued)

<i>M</i> -ary noncoherent orthogonal signaling		
21.1.	$P_b = \frac{2^{k-1}}{2^k - 1} P_M$	
<i>M</i> -ary continuous-phase frequency-shift-keying (CPFSK)		Ref. 6
22. $M = 2$	$h = 0.5$ (MSK), $P_b = \frac{1}{2} \text{erfc}(\sqrt{\rho_b})$ (Note: This is equivalent to coherent binary PSK)	
23.	For other selected values of M , h (deviation ratio), and n (receiver "memory") see figures	Figure A.4
Binary signaling in a frequency-nonselective, slowly varying fading channel		Ref. 1
24.	Rayleigh fading channel; L th-order diversity ($L \geq 1$): $P_b \approx \left(\frac{1}{N\bar{\rho}_b} \right)^L \binom{2L-1}{L}$ for $\bar{\rho}_b \gg 1$; $\bar{\rho}_b$ is the average E_b/N_0 per diversity channel	
24.1. $L = 1$	$P_b = 1/N\bar{\rho}_b$	
24.2. $N = 4$	Coherent PSK	
24.3. $N = 2$	Coherent, orthogonal FSK, DPSK	
24.4. $N = 1$	Noncoherent, orthogonal FSK	
25.	Ricean fading channel with coherent PSK and noisy phase reference:	Ref. 7
	$P_b = \frac{1}{2\pi} \int_0^\infty \int_{-\pi}^\pi \frac{1}{2} \text{erfc}(\sqrt{\rho_b} \alpha \cos \phi) \times \frac{\exp(A\alpha S \cos \phi)}{I_0(\alpha S)} \frac{\alpha}{\sigma^2} \exp\left(-\frac{\alpha^2 + \gamma^2}{2\sigma^2}\right) \times I_0\left(\frac{\alpha\gamma}{\sigma^2}\right) d\phi d\alpha$ $E(\alpha^2) = \gamma^2 + 2\sigma^2$ ($\gamma = 0$ corresponds to Rayleigh fading), S is the loop SNR for a first-order loop	Figure A.5
Binary block codes		Ref. 8
26.	Soft-decision decoding using coherent PSK: $P_w \leq \sum_{m=2}^M \frac{1}{2} \text{erfc}[(\rho_b R_c w_m)^{1/2}]$ P_w is "word" error probability; R_c = code rate = k/n ; $M = 2^k$; w_m is the weight of m th code word (other than all-zero word)	
27.	$P_2 < \frac{1}{2} \exp(-\rho_b R_c d_{\min} + k \ln 2)$ $d_{\min}$ is the minimum distance	
28.	Hard-decision decoding; channel error probability = p : $\sum_{m=t+2}^n P(m, n) \leq P_w \leq \sum_{m=t+1}^n P(m, n)$ $P(m, n) = \binom{n}{m} p^m (1-p)^{n-m}$ $t = \text{Int}[(d_{\min} - 1)/2]$	

Table A.1. (continued)

Binary block codes, cont'd.		
29.	$P_w \leq (2^k - 1)[4p(1 - p)]^{d_{min}/2}$	
30.	$P_b \approx \frac{d_{min}}{n} \sum_{m=i+1}^{d_{min}} P(m, n) + \frac{1}{n} \sum_{m=d_{min}+1}^n mP(m, n)$	Figure A.6 (A–C)
Binary convolutional codes		Ref. 8
31.	Soft-decision decoding using coherent PSK: $P_b < \frac{1}{2} \sum_{d=d_{free}}^{\infty} \beta_d \operatorname{erfc}[(\rho_b R_c d)^{1/2}]$ β_d are coefficients of D in the transfer function derivative, $= \left. \frac{\partial T(D, N)}{\partial N} \right _{N=1}$	Figure A.1 (L, M)
32.	Hard-decision decoding; channel error probability = p : $P_b < \left. \frac{\partial T(D, N)}{\partial N} \right _{N=1, D=[4p(1-p)]^{1/2}}$	Figure A.6 (D–F)

[Note: Refer to encoder block diagram, Figure 8.12; if $k > 1$, expressions 31 and 32 are divided by k]

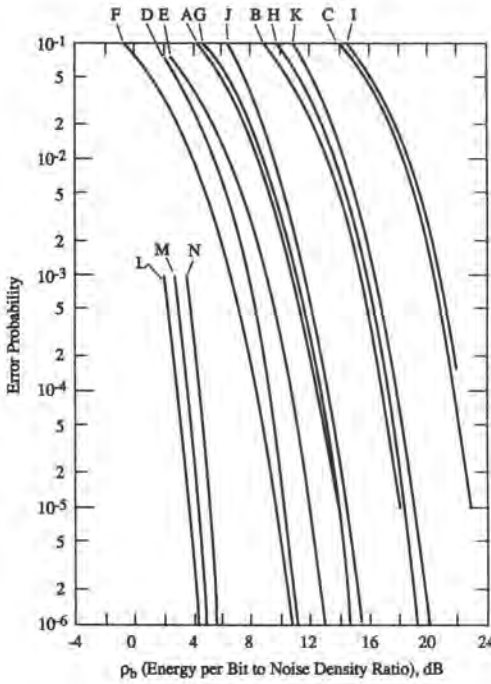


Figure A.1. Bit and symbol error probabilities for various coherent modulation schemes. A, 8-PSK: P_8 ; B, 16-PSK: P_{16} ; C, 32-PSK: P_{32} ; D, 2-DPSK: P_b ; E, 4-DPSK: P_b ; F, 2-PSK, 4-PSK, 4-QAM, 2-PAM: P_b ; G, 4-PAM: P_4 ; H, 8-PAM: P_8 ; I, 16-PAM: P_{16} ; J, 16-QAM: P_{16} ; K, 64-QAM: P_{64} . L–N, Rate-1/2 convolutional codes; code parameters (L, d_f) ; Viterbi decoding; PSK modulation; L, $L = 9, d_f = 12: P_b$; M, $L = 7, d_f = 10: P_b$; N, $L = 5, d_f = 7: P_b$.

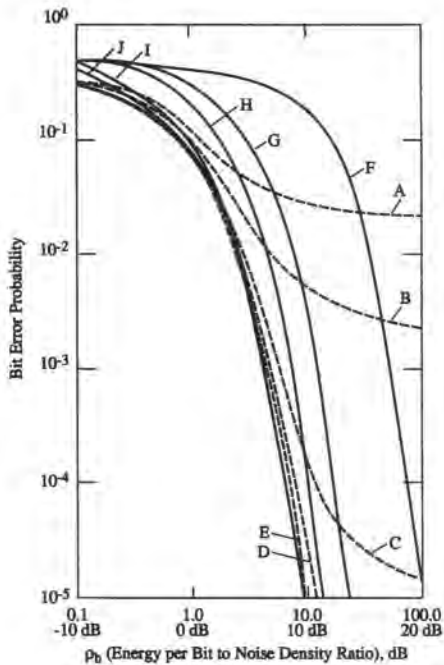


Figure A.2. Effect of phase noise on binary PSK error probability. A–E, Auxiliary carrier: A, Loop SNR = 3dB; B, loop SNR = 7dB; C, loop SNR = 10dB; D, loop SNR = 15dB; E, loop SNR = 20dB. F–J, Squaring loop; F, $\delta = 0.1$; G, $\delta = 0.5$; H, $\delta = 1.0$; I, $\delta = 5.0$; J, $\delta = 10.0$.

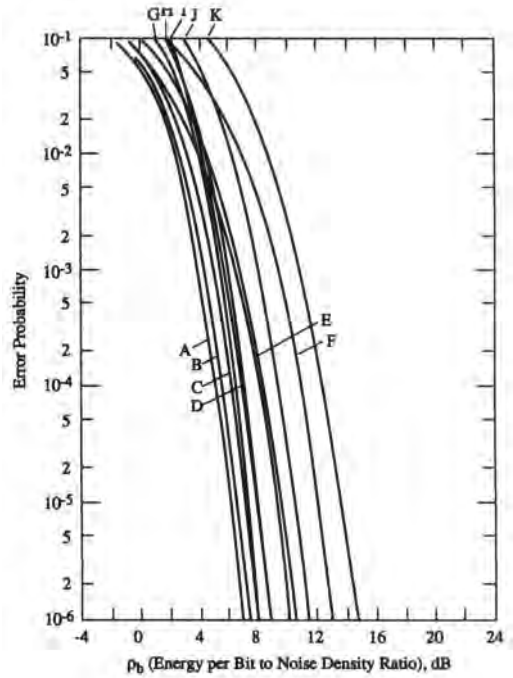


Figure A.3. Bit error probability for coherent and noncoherent M -ary orthogonal signaling. A-F, Coherent detection: A, $M=64$: P_b ; B, $M=32$: P_b ; C, $M=16$: P_b ; D, $M=8$: P_b ; E, $M=4$: P_b ; F, $M=2$: P_b . G-K, Noncoherent detection: G, $M=32$: P_b ; H, $M=16$: P_b ; I, $M=8$: P_b ; J, $M=4$: P_b ; K, $M=2$: P_b .

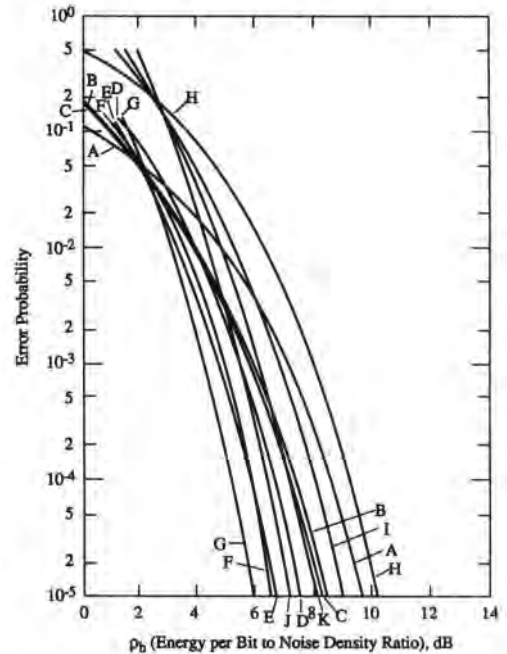


Figure A.4. Bit and symbol error probabilities for continuous phase frequency-shift-keying (CPFSK) for coherent and noncoherent detection. A-G, Coherent detection: A, 2-CPFSK, $h = 0.715$, $n=2$: P_b ; B, 2-CPFSK, $h = 0.715$, $n=3$: P_b ; C, 2-CPFSK, $h = 0.715$, $n=4$: P_b ; D, 4-CPFSK, $h=1.75$, $n=2$: P_b ; E, 4-CPFSK, $h=0.8$, $n=3$: P_b ; F, 8-CPFSK, $h=0.879$, $n=2$: P_b ; G, 8-CPFSK, $h = 0.879$, $n=3$: P_b . H-K, Noncoherent detection: H, 2-CPFSK, $h = 0.715$, $n=3$: P_b ; I, 2-CPFSK, $h=0.715$, $n=5$: P_b ; J, 4-CPFSK, $h = 0.8$, $n=3$: P_b ; K, 4-CPFSK, $h = 0.8$, $n=5$: P_b .

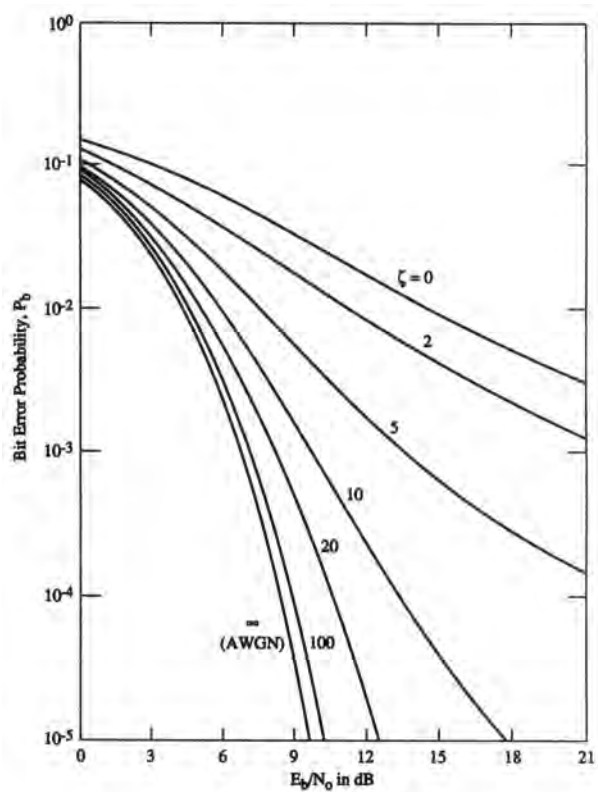


Figure A.5. Bit error probability for slow-fading Ricean channel with noisy phase reference. Reference loop SNR=15dB: ζ is ratio of specular to diffuse energy.

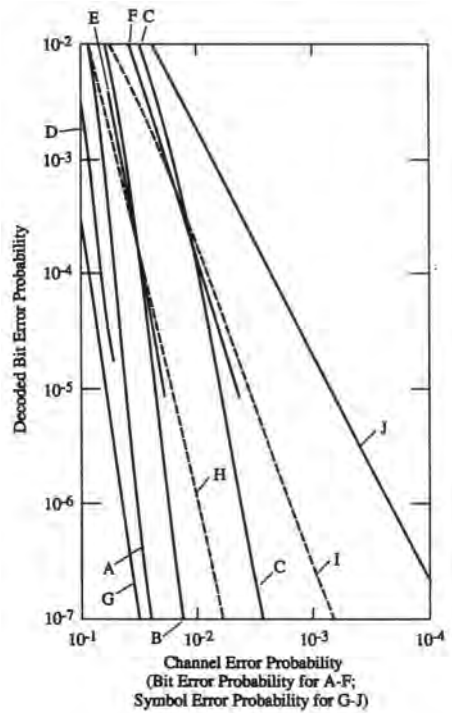


Figure A.6. Decoded bit error probability as a function of channel error probability. A–C, (n, k, t) BCH code: A, $n = 127, k = 36, t = 15$; B, $n = 127, k = 64, t = 10$; C, $n = 127, k = 92, t = 5$. D–F, Convolutional codes, hard decision: D, $L = 7, R = 1/3$; E, $L = 7, R = 1/2$; F, $L = 9, R = 3/4$. G–J, Modulation; 32-orthogonal, Reed–Solomon code: $n = 31, t$ -error correcting. G, $t = 8$. H, $t = 4$. I, $t = 2$. J, $t = 1$.

References

1. J. G. Proakis, *Digital Communications*, 2nd ed., McGraw-Hill, New York (1989).
2. S. A. Rhodes, Effect of noisy phase reference on coherent detection of offset-QPSK signals, *IEEE Trans. Commun.* **COM-22**(8), 1046–1055 (1974).
3. W. C. Lindsey and M. K. Simon, *Telecommunication Systems Engineering*, Prentice-Hall, Englewood Cliffs, New Jersey (1973).
4. J. J. Spilker, Jr., *Digital Communications by Satellite*, Prentice-Hall, Englewood Cliffs, New Jersey (1977).
5. D. M. Jansky and M. C. Jeruchim, *Communication Satellites in the Geostationary Orbit*, 2nd ed., Artech House, Norwood, Massachusetts (1987).
6. T. A. Schonhoff, Symbol error probabilities for M -ary CPFSK: Coherent and noncoherent detection, *IEEE Trans. Commun.* **COM-24**(6), 644–652 (1976).
7. J. W. Modestino and S. Y. Mui, Convolutional code performance in the Rician fading channel, *IEEE Trans. Commun.* **COM-24**(6), 592–606 (1976).
8. J. P. Odenwalder, *Error Control Coding Handbook*, Linkabit Corp., San Diego, California (1976).

This page intentionally left blank

Appendix B

Gaussian Tail Probabilities $Q(x)$ and an Approximation $\hat{Q}(x)$

$$Q(x) = \frac{1}{(2\pi)^{1/2}} \int_x^{\infty} e^{-u^2/2} du$$

$$\hat{Q}(x) = \left[\frac{1}{(1-a)x + a(x^2+b)^{1/2}} \right] \frac{1}{(2\pi)^{1/2}} e^{-x^2/2}$$

x	$Q(x)$	$\hat{Q}(x)$	x	$Q(x)$	$\hat{Q}(x)$
0	5.00E-01	5.00E-01	2.7	3.47E-03	3.46E-03
0.1	4.60E-01	4.58E-01	2.8	2.56E-03	2.55E-03
0.2	4.21E-01	4.17E-01	2.9	1.87E-03	1.86E-03
0.3	3.82E-01	3.78E-01	3.0	1.35E-03	1.35E-03
0.4	3.45E-01	3.41E-01	3.1	9.68E-04	9.66E-04
0.5	3.09E-01	3.05E-01	3.2	6.87E-04	6.86E-04
0.6	2.74E-01	2.71E-01	3.3	4.83E-04	4.83E-04
0.7	2.42E-01	2.39E-01	3.4	3.37E-04	3.36E-04
0.8	2.12E-01	2.09E-01	3.5	2.33E-04	2.32E-04
0.9	1.84E-01	1.82E-01	3.6	1.59E-04	1.59E-04
1.0	1.59E-01	1.57E-01	3.7	1.08E-04	1.08E-04
1.1	1.36E-01	1.34E-01	3.8	7.23E-05	7.23E-05
1.2	1.15E-01	1.14E-01	3.9	4.81E-05	4.81E-05
1.3	9.68E-02	9.60E-02	4.0	3.17E-05	3.16E-05
1.4	8.08E-02	8.01E-02	4.5	3.40E-06	3.40E-06
1.5	6.68E-02	6.63E-02	5.0	2.87E-07	2.87E-07
1.6	5.48E-02	5.44E-02	5.5	1.90E-08	1.90E-08
1.7	4.46E-02	4.43E-02	6.0	9.87E-10	9.86E-10
1.8	3.59E-02	3.57E-02	6.5	4.02E-11	4.02E-11
1.9	2.87E-02	2.86E-02	7.0	1.28E-12	1.28E-12
2.0	2.28E-02	2.26E-02	7.5	3.19E-14	3.19E-14
2.1	1.79E-02	1.78E-02	8.0	6.22E-16	6.22E-16
2.2	1.39E-02	1.39E-02	8.5	9.48E-18	9.48E-18
2.3	1.07E-02	1.07E-02	9.0	1.13E-19	1.13E-19
2.4	8.20E-03	8.17E-03	9.5	1.05E-21	1.05E-21
2.5	6.21E-03	6.19E-03	10.0	7.62E-24	7.62E-24
2.6	4.66E-03	4.65E-03			

Note: The parameters a and b can be optimized for various objectives. The values $(1/\pi, 2\pi)$ optimize the form $\hat{Q}(x)$ in the sense of providing the best lower bound on $Q(x)$ for $x \geq 0$. Here, best means the smallest absolute relative error $\varepsilon(x) = |[\hat{Q}(x) - Q(x)]/Q(x)|$. For an extensive discussion, see P. O. Börjesson and C.-E. W. Sundberg, Simple approximations of the error function $Q(x)$ for communications applications, *IEEE Trans. Commun.* **COM-27**(3), 639–643 (1979).

This page intentionally left blank

Appendix C

Coefficients of the Hermite Polynomials

$$H_n(x) = \sum_{i=0}^n c_i x^i$$

	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
H_0	1										
H_1	0	2									
H_2	-2	0	4								
H_3	0	-12	0	8							
H_4	12	0	-48	0	16						
H_5	0	120	0	-160	0	32					
H_6	-120	0	720	0	-480	0	64				
H_7	0	-1680	0	3360	0	-1344	0	128			
H_8	1680	0	-13,440	0	13,440	0	-3584	0	256		
H_9	0	30,240	0	-80,640	0	48,384	0	-9216	0	512	
H_{10}	-30,240	0	302,400	0	-403,200	0	161,280	0	-23,040	0	1024

Extracted from M. Abramowitz and I. Stegun *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, U.S. Department of Commerce, National Bureau of Standards, Applied Mathematics Series, No. 55 (June 1964).

This page intentionally left blank

Some Abscissas and Weights for Gaussian Quadrature Integration

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^n w'_i f(x'_i)$$

$\pm x'_i$	w'_i
$n=2$	
0.70710 67811 86548	8.86226 92545 28E - 01
$n=3$	
0.00000 00000 00000	1.18163 59006 04E - 00
1.22474 48713 91589	2.95408 97515 09E - 01
$n=4$	
0.52464 76232 75290	8.04914 09000 55E - 01
1.65068 01238 85785	8.13128 35447 25E - 02
$n=5$	
0.00000 00000 00000	9.45308 72048 29E - 01
0.95857 24646 13819	3.93619 32315 22E - 01
2.02018 28704 56086	1.99532 42059 05E - 02
$n=6$	
0.43607 74119 27617	7.24629 59522 44E - 01
1.33584 90740 13697	1.57067 32032 29E - 01
2.35060 49736 74492	4.53000 99055 09E - 03
$n=7$	
0.00000 00000 00000	8.10264 61755 68E - 01
0.81628 78828 58965	4.25607 25261 01E - 01
1.67355 16287 67471	5.45155 82819 13E - 02
2.65196 13568 35233	9.71781 24509 95E - 04
$n=8$	
0.38118 69902 07322	6.61147 01255 82E - 01
1.15719 37124 46780	2.07802 32581 49E - 01
1.98165 67566 95843	1.70779 83007 41E - 02
2.93063 74202 57244	1.99604 07221 14E - 04
$n=9$	
0.00000 00000 00000	7.20235 21560 61E - 01
0.72355 10187 52838	4.32651 55900 26E - 01
1.46855 32892 16668	8.84745 27394 38E - 02
2.26658 05845 31843	4.94362 42755 37E - 03
3.19099 32017 81528	3.96069 77263 26E - 05

Extracted from M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, U.S. Department of Commerce, National Bureau of Standards, Applied Mathematics Series No. 55 (June, 1964).

This page intentionally left blank

Chi-Square Probabilities

The table below gives percentage points from the χ_m^2 distribution. That is, values of $\chi_{m;\alpha}^2$, where m represents degrees of freedom and

$$\int_0^{\chi_{m;\alpha}^2} \frac{1}{2\Gamma(m/2)} \left(\frac{y}{2}\right)^{(m/2)-1} e^{-y/2} dy = 1 - \alpha$$

For $m < 100$, linear interpolation is adequate. For $m > 100$, the chi-square distribution can be approximated by a Gaussian distribution with mean m and variance $2m$.

m/α	0.990	0.975	0.950	0.050	0.025	0.010
1	1.57088E - 04	9.82069E - 04	3.93214E - 04	3.84146E - 00	5.02389E - 00	6.63490E - 00
2	2.01007E - 02	5.06356E - 02	1.02587E - 01	5.99147E - 00	7.37776E - 00	9.21034E - 00
3	1.14832E - 01	2.15795E - 01	3.51846E - 01	7.81473E - 00	9.34840E - 00	1.13449E + 01
4	2.97110E - 01	4.84419E - 01	7.10721E - 01	9.48773E - 00	1.11433E + 01	1.32767E + 01
5	5.54300E - 01	8.31211E - 01	1.14548E - 00	1.10705E + 01	1.28325E + 01	1.50863E + 01
6	8.72085E - 01	1.23734E - 00	1.63539E - 00	1.25916E + 01	1.44494E + 01	1.68119E + 01
7	1.23904E - 00	1.68987E - 00	2.16735E - 00	1.40671E + 01	1.60128E + 01	1.84753E + 01
8	1.64648E - 00	2.17973E - 00	2.73264E - 00	1.55073E + 01	1.75346E + 01	2.00902E + 01
9	2.08791E - 00	2.70039E - 00	3.32511E - 00	1.69190E + 01	1.90228E + 01	2.16660E + 01
10	2.55821E - 00	3.24697E - 00	3.94030E - 00	1.83070E + 01	2.04831E + 01	2.32093E + 01
11	3.05347E - 00	3.81575E - 00	4.57481E - 00	1.96751E + 01	2.19200E + 01	2.47250E + 01
12	3.57056E - 00	4.40379E - 00	5.22603E - 00	2.10261E + 01	2.33367E + 01	2.62170E + 01
13	4.10691E - 00	5.00874E - 00	5.89186E - 00	2.23621E + 01	2.47356E + 01	2.76883E + 01
14	4.66043E - 00	5.62872E - 00	6.57063E - 00	2.36848E + 01	2.61190E + 01	2.91413E + 01
15	5.22935E - 00	6.26214E - 00	7.26094E - 00	2.49958E + 01	2.74884E + 01	3.05779E + 01
16	5.81221E - 00	6.90766E - 00	7.96164E - 00	2.62962E + 01	2.88454E + 01	3.19999E + 01
17	6.40776E - 00	7.56418E - 00	8.67176E - 00	2.75871E + 01	3.01910E + 01	3.34087E + 01
18	7.01491E - 00	8.23075E - 00	9.39046E - 00	2.88693E + 01	3.15264E + 01	3.48053E + 01
19	7.63273E - 00	8.90655E - 00	1.01170E + 01	3.01436E + 01	3.28523E + 01	3.61908E + 01
20	8.26040E - 00	9.59083E - 00	1.08508E + 01	3.14104E + 01	3.41696E + 01	3.75662E + 01
25	1.15240E + 01	1.31197E + 01	1.46114E + 01	3.76525E + 01	4.06465E + 01	4.43141E + 01
50	2.97067E + 01	3.23574E + 01	3.47642E + 01	6.75048E + 01	7.14202E + 01	7.61539E + 01
100	7.00648E + 01	7.42219E + 01	7.79295E + 01	1.24342E + 02	1.29561E + 02	1.35807E + 02

This page intentionally left blank

Index

- Acceptance-rejection method, 381–383
- Adjacent channel interference (ACI), 406
- Algebra of LTV Systems, 135
- Aliasing, 87; *see also* Sampling
- Autocorrelation function
 - properties, 329, 335
 - of stationary processes, 335
- Autoregressive moving average models
 - ARMA model, definition, 342
 - applied, in Case Study II, 804
 - AR model, 343
 - MA model, 343
 - autocorrelation, 343
 - parameter estimation, 349
 - Yule–Walker equations, 349
 - power spectral density, 343, 804
 - vector ARMA model, 399
- Baseband modulation, 417–425
 - differential encoding, 420
 - line coding, 420–425
 - partial response signaling, 425
- BER
 - estimation of, 678–757
 - conceptual framework, 683–686
 - importance sampling, 710–734
 - interval measures, 697–703
 - mixed quasianalytical, 754–757
 - moment method, 787–790
 - Monte Carlo, 686–703
 - quasianalytical, 737–754
 - run-time implications, 679–683
 - tail extrapolation, 703–710
 - evaluation of
 - block-coded systems, 748–753
 - convolutionally coded systems, 749–750, 753
 - formulas for, 881–885
- Bias, *see also* Expected value of estimators
 - asymptotic, in tail extrapolation, 707
 - statistical, 628
- Biased density, 711–712, 715
- Bilinear transformation, 173–178
 - frequency prewarping, 176–177, 179
 - frequency warping, 176
- Biquadratic sections
 - continuous, 112–113
 - canonic form, 113
 - discrete, 166
 - cascade interconnection, 168
 - parallel interconnection, 169
- Bounds, *see also* Inequalities
 - Chebyshev, 317
 - Chernoff, 318
 - union, 318
- Calibration of simulations, 534–539
- Carrier recovery
 - for BPSK, 504–506
 - Costas loop, 507, 527–528
 - effect on BER, 753–754
 - estimation of phase, 655–661
 - block processing, 660–661
 - cross-correlation, 655–657
 - lowpass equivalent, 507, 512
 - Mth power loop, 511–512
 - PLL, 495–534
 - phase error distribution, 662–664
 - for QPSK, 510–512
 - simulation, 495, *et seq.*
 - squaring loop, 506
- CDMA, *see also* Multiplexing/multiple access
 - applied, in Case Study IV, 822–848
- Central limit theorem, 320
- Channel capacity
 - AWGN, 425
 - discrete memoryless, 426
- Channel (fading) simulation methodology: *see* Simulation methodology
- Channels, channel models: *see* Finite-state channel models; Free-space channels; Guided media channels; Multipath fading channels

- Cochannel interference (CCI), in Case Study IV, 829–831
- Coded modulation, 449–451
 - block-coded modulation, 449–450
 - trellis-coded modulation, 450–451
- Coded systems, *see also* Error control coding
 - hard decision,
 - with dependent errors, 751–753
 - with independent errors, 748–751
 - quasianalytic simulation of, 748–753
 - soft decision, 753
- Communication system, generic block, diagram, 408
- Confidence interval
 - for average level, 635–636
 - for BER, based on
 - binomial distribution, 688–691
 - normal distribution, 691–694
 - Poisson distribution, 498–500
 - defined, 629–631
- Convolution property, 68–69
- Convolution property for periodic signals, 73
- Convolution theorem
 - applied to filtering, 135
 - convolution sum, 63
- Correlated Gaussian sequences
 - generation of spatially correlated Gaussian sequences
 - Cholesky method, 388–399
 - covariance factorization method, 398
 - generation of temporally correlated Gaussian sequences
 - ARMA models, 394
 - FIR method, 397
 - spectral factorization method, 396
 - spatial correlation, 394
 - temporal correlation, 394
- Correlation coefficient, 295
- Correlogram, 646–647
- Covariance
 - matrix, 296
 - scalar, 295
- Cross-correlation
 - for estimating delay, 655–656
 - for estimating SNR, 671–673
- Cross-correlation function
 - properties, 336
 - of stationary processes, 335–336
- Cross PSD
 - definition, 338
 - properties, 338–339
- Delay
 - effect on phase-locked loop, 524, 531–534
 - estimation of, 655–657, 658–659
 - in SNR estimation, 675–678
- Demodulation/demodulators
 - analog (*cont.*)
 - envelope demodulator, 461
 - noncoherent, 460–466
 - PLL, 465–466
 - square-law demodulator, 461
 - digital
 - coherent, 457–460
 - noncoherent, 460–466
 - quadrature form, 457–458
- Dependent errors, 696–697, 751–753
- Difference equation, 165; *see also* ARMA models
- Differential equations
 - linear time-invariant (LTI), 110
 - transfer function, 111
 - linear time-varying (LTV), 192–193
 - impulse response, 192
 - nonlinear: *see* Nonlinear differential equations
- Differentiation, discrete-time, 462–465
- Discrete Fourier transform (DFT), 119
- Discrete (Markov) channel models: *see* Finite-state channel models
- Discrete (sampling) model for LTV systems, 195–196
- Discrete-time systems: *see* Systems
- Distribution: *see* Random variables
- Empirical distribution, 640–641
- Equalizers, 474–481
 - applied, in Case Study I, 773–776
 - fractionally spaced, 475
 - LMS algorithm, 478
 - simulation, 480–481
 - synchronous, 475
 - TDL structure, 474
 - weight calculation, 476–480
 - by covariance inversion, 479–480
 - gradient algorithm, 476–477
- Error control coding
 - bandwidth expansion, 750
 - block coding/decoding, 428–431
 - computational load, 430
 - block error distribution
 - based on gap distribution, 700
 - for a Gilbert channel, 751–753
 - code performance
 - binary block code, 749
 - Reed–Solomon, 749
 - code rate, 428, 431
 - coding gain, 751
 - convolutional coding/decoding, 431–433
 - computational load, 432
 - interleaving
 - in Case Study II, 794
 - word error probability, 700
 - quasianalytical method, applied to, 748–753
 - Reed–Solomon code, 428, 749
 - simulation of, 427, 748–753
 - Viterbi algorithm, 431–432
- Error indicator function, 687

Error probability: *see* BER; Estimation of BER
 Estimation (of)

- average level, 631–635
- average power, 636–640
- bit-error-rate (BER), 678–757
- carrier phase: *see* Carrier recovery
- error-free intervals, 697–703
- power spectral density, 645–654
- probability density function, 640–641
- probability distribution function, 641–645
- signal-to-noise ratio, 670–678
- symbol timing: *see* Timing recovery

Estimator

- bias of, 628
- consistent, 628
- variance of, 629

Expected value of estimator (of)

- average level, 632
- average power, 637
- BER (importance sampling), 718
- BER (Monte Carlo), 694
- power spectral density, 651–652
- probability density, 653–654

Expected values

- approximate computation, 321–323
- conditional, 295
- covariance
 - matrix, 296
 - scalar, 295
- mean, 293, 296
- standard deviation, 293
- variance, 293

Eye diagram, 664–666

- applied, in Case Study I, 780

Fading: *see* Multipath fading channels

Fast Fourier transform (FFT), 120–121

F distribution, 304

- in SNR estimation, 694

Feedback loops, 47; *see also* Phase-locked loops

- simulation of, 47

Filters, *see also* FIR filters; IIR filters

- adaptive, 474–481
- biquadratic decomposition, 112–113, 166
- classical filters, 136–141
 - Bessel, 140
 - Butterworth, 137
 - Chebyshev, 138–139
 - elliptic, 139
 - frequency transformations, 141
 - lowpass equivalent, 142–144
- Doppler: *see* Multipath fading channels

FIR filters: *see* Finite impulse response (FIR) filters

- functional filters, 481–482
 - amplitude ripple, 481
 - amplitude tilt, 482
 - cubic phase, 482

Filters (*cont.*)

functional filters (*cont.*)

- parabolic gain, 482
- parabolic phase, 482

ideal filters, 136

- bandpass, 136
- bandstop, 136
- high-pass, 136
- low-pass, 136

IIR filters: *see* Infinite impulse response filters

matched, 472–473

minimum MSE, 470

Wiener filter, 470

noise bandwidth

- computation, 662
- definition, 262

pole-zero, 143

lowpass equivalent, 143

preemphasis/deemphasis, 348

pulse shaping, 468–469; *see also* FIR filters

quadrature, 77

raised cosine, 468

in Case Study I, 767–768

recursive: *see* Infinite impulse response

simulation (of)

FIR filters, 149–165

IIR filters, 165–181

summary, 182–184

spectral shaping, 467

square root, 469, 472

in Case Study I, 767–768

tabular filters, 483–484

Finite impulse response (FIR) filters, 147, 149–165

with DFT, 154–158

block processing, 159

example, 161–162

for nonperiodic signals, 101–107

overlap-and-add, 155–156

overlap-and-save, 156–158

time domain, 149–151

windowing, 150–151

Finite-state channel models, 596–609

applied, in CDMA example (Case Study IV), 838

Baum–Welch algorithm, 606

backward recursion, 608

forward recursion, 606

definition, 596–597

examples

binary symmetric, 598

HMM, 602

M-ary, 598

soft-decision, 598

Fritchman model, 604

Gilbert model, 604

HMM: (hidden Markov model), 600

modified, 608

interval (error-free) simulation: *see* Estimation of error-free intervals

interval simulation, 388

- Finite-state channel models (*cont.*)
 - model types
 - with memory, 599
 - memoryless, hard-decision, 598
 - memoryless, soft decision, 598
 - parameter estimation, 606;
 - Baum–Welch algorithm, 606
- Formatting/source coding, 414–417
 - A/D conversion, 414
 - quantization, 415–416
- Fourier analysis
 - continuous, 66–74
 - discrete, 118–126
- Fourier series, 65
- Fourier transform, 66–67
 - for periodic signals, 72
 - properties, 67–70
 - table of, 129
- Fourier transform for periodic signals, 35, 96
- Free-space (almost) channels, 586–591
 - absorption, 587
 - ionospheric effects, 589
 - rain attenuation, 587–588
 - tropospheric effects, 587
- Frequency domain representation
 - linear time-invariant (LTI), 63–74, 119–126
 - linear time-varying (LTV), 188–189
- Frequency response
 - discrete, 124
 - Fourier transform, 70
 - Laplace transform, 114
 - z transform, 124
- Galois field, 390
- Gaussian distribution
 - approximation, 301
 - table (of Q function), 891
- Gaussian quadrature
 - applied, in Case Study I, 787–790
- Gaussian random process
 - bandpass, 354–357
 - envelope, 356
 - lowpass equivalent, 354–355
 - phase, 356
 - quadrature model, 354
 - definition, 351–352
 - ergodicity, 333
 - stationarity, 352
 - white PSD, 352–353
 - autocorrelation, 352
 - bandlimited, 353
- Generalized exponential distribution, 304
 - applied to BER estimation, 703, 741, 743–744
- Generation of random sequences
 - correlated sequences, 392
 - PN sequences, 386–390
 - sampled white noise, 383, 385
 - uncorrelated sequences, 384
- Gibbs (the) phenomenon, 71, 150
- Gilbert model, 246, 392
- Glivenko’s theorem, 641
- Guided media channels, 591–595
 - optical fibers, 593
 - waveguides, 591
- Hilbert transform, 75–77
 - properties, 77–78
- Histogram, 641–642
- Importance sampling, 710–734
 - biasing, 719–721, 724–732
 - conventional, 720
 - implementation considerations, 713–717, 732–734
 - improved, 721
 - variance, 718
- Importance splitting, 734–737
- Impulse-invariant transformation, 170–173, 178
 - aliasing, 170
 - for rational functions, 171
- Impulse response
 - continuous, 62
 - discrete, 62
- Incomplete gamma function, 741
- Inequalities, 317–319
 - Chebyshev, 317
 - Chernoff, 318
- Infinite impulse response (IIR) filters, 165–181, 184
 - classical, continuous, 136–141
 - lowpass equivalent, 142–144
 - impulse-invariant
 - mapping from continuous to discrete: *see* Bilinear;
 - pole-zero (example of), 172
 - realization using biquadratic expansion, 79
 - continuous, 112–113, 80
 - discrete, 166–169
- Information sources
 - analog signals, 411–413
 - filtered random processes, 413
 - multiple tones, 412
 - single tone, 412
 - binary PN sequences, 386
 - digital signals, 413–414
 - M-ary PN sequences, 390
- Interconnection of systems
 - cascade connection of biquadratic sections
 - continuous, 112
 - discrete, 168
 - in frequency domain, 70
 - of LTI systems, 108–110
 - of LTV systems, 190–192
 - parallel connection of biquadratic sections
 - continuous, 112
 - discrete, 166–169
- Interference
 - in Case Study IV, 829–831
 - in CDMA, 490
 - in FDMA, 490

- Interleaving
 - applied, in Case Study II, 794
- Interpolation, 89–105
 - bandlimited, 86, 96–98
 - cubic spline, 100–105
 - linear, 49, 98–100
 - in multirate application, 88–89, 93–96
- Laplace transform, 106–107
 - properties, 107–108
 - table of, 130
- Large-deviations theory, 724–732
- Linear time-invariant (LTI) systems, *see also* Filters;
 - FIR filters; IIR filters
 - continuous, 62, 70, 110–115
 - discrete, 62, 115–118
- Linear time-invariant (LTI) systems: discrete, 22, 72–124
 - convolution sum, 22
 - impulse response, 22
- Linear time-varying (LTV) systems, 184–196
 - impulse response, 186–187
 - properties of, 190–192
 - superposition integral, 187
- Line coding, 420–425
 - biphase, 423
 - correlative, 421
 - delay, 423
 - differential, 420
 - Miller, 421
 - NRZ, 422–423
 - partial response, 425
 - RZ, 423
 - spectral shaping, 424
- Lowpass equivalents, 74–83
 - applied to filtering, 81–82
 - bandpass filter, 79–81
 - bandpass nonlinearity, 219–220
 - classical filters, 142–144
 - complex envelope, 75
 - modeling, 82–83
 - modulated signal, 78–79
 - preenvelope (analytic signal), 77
 - of random signals, 362–366
 - in system analysis, 79–82
- Mapping from continuous to discrete: *see* Bilinear transformation; Impulse-invariant transformation
- Methodology, *see also* Simulation methodology
 - applied, chapter 12
- Mixed quasianalytical method, 754–757
- Mobile radio channels: *see* Multipath fading channels
- Modeling
 - concepts of, 17–26
 - errors in, 28–36
- Modulation/modulators, 433–457
 - analog
 - AM, 433–434 327
 - FM, 434
 - PM, 434
 - quadrature, 434–435
 - digital
 - ASK, 438
 - coded modulation, 449–451
 - CPFSK, 445–446
 - CPM, 443–449
 - general quadrature, 435–438
 - modeling considerations, 451–455
 - MSK, 446–447
 - OFDM, 439–442
 - OQPSK, 438
 - Pi/4 QPSK, 438–439
 - PSK, 438
 - QAM, 438
 - applied, in Case Study I, 766
 - signal constellation, 437
- Moment method, 322
 - applied to BER evaluation, in Case Study I, 787–790
- Moments
 - of finite sums, 322–323
 - recursive computation, 323
 - in Case Study I, 787–790
- Monte Carlo simulation, definition, 15, 316, 371–373
 - applied to BER estimation, 686–703
 - variations, 26–27, 373
 - modified MC, 373
 - pure MC, 371
- Multipath fading channels, 546–586
 - delay
 - rms, 557
 - spread, 556
 - diffuse multipath model, 561–572
 - Cholesky factorization, 567
 - example, with correlated tap gains 570
 - filtered delay-power profile, 568
 - separable scattering function, 565
 - tapped delay-line models, 563–565
 - discrete multipath model, 572–575
 - example, with filtered model, 572
 - filtered discrete model, 572
- Doppler
 - fast fading, 560, 561
 - Jakes autocorrelation function, 558
 - Jakes spectrum, 558
 - slow fading, 560, 561
 - Spread, 560
- fading
 - coherence BW, 557
 - coherence time, 560
 - frequency non-selective (flat), 554, 558
 - frequency selective, 554, 558
 - fading, large scale (shadow)
 - lognormal shadowing, 549

- Multipath fading channels (*cont.*)
 - fading, small scale (multipath)
 - delay power profile, 554
 - Doppler spectrum, 554
 - random process model, 553–554
 - Rayleigh, 552
 - Rician, 552
 - indoor wireless channel models, 576–583
 - factory and open plan building, 577
 - office building, 578
 - ray-tracing, 547, 582
 - line-of-sight (LOS) radio channel model
 - LOS radio Case Study, 769–793
 - minimum-phase, 585
 - non-minimum phase, 586
 - Rummler model, 583–584
 - lowpass equivalent, 550–551
 - path loss measurements, 549
 - path loss (shadowing) models, 548
 - COST-231 model, 548
 - Hata model, 548
 - ray-tracing, 547, 582
 - propagation effects, 546
 - reference models for GSM applications, 614
 - Doppler spectrum types, 616
 - hilly terrain, 615
 - rural areas, 614
 - urban areas, 615
 - reference models for PCS applications, 617
 - reference models for UMTS-IMT-2000 applications, 618
 - decorrelation length, 619
 - impulse response (delay spread), 619
 - indoor, 618
 - outdoor to indoor and pedestrian, 618
 - vehicular, 618
 - scattering function, 554
 - simulation of, 561–576
 - LOS radio, 783, 785
 - spaced-frequency correlation function, 557
 - spaced time correlation function, 558
 - statistical characterization, 551–554
 - tap-gain models
 - correlated, 565–567
 - generation of tap-gain processes, 575–576
 - multirate, 845–846
 - uncorrelated, 564
 - tapped delay-line models
 - for diffuse channels, 563
 - for discrete channels, non-uniformly spaced, 572
 - for discrete channels, uniformly spaced, 573
 - wideband channel measurements, 561
 - WSSUS model, 553
- Multiplexing/multiple access, 484–491
 - CDMA, 489–491
 - FDMA, 486–487
 - PDMA, 484–485
 - SDMA, 484–485
 - TDMA, 487
- Multirate processing, 83, 87–89, 91–96
- Multivariate Gaussian, 305–308
 - conditional, 306
 - joint, 305
 - linear transformation, 306
 - marginal, 306
 - moments, 308
 - properties, 306
- Noise
 - band-limited, 352
 - Gaussian, 350–356
 - impulse noise, 348–350
 - quadrature representation, 354–356
 - shot noise, 346–348
 - thermal noise, 352
 - white, 352
- Nonlinear differential equations, 257–275
 - applied to optical amplifier, 271–275
 - integration formulas, 261–262
 - explicit, 263
 - implicit, 263–266
 - stability, 267, 269–270
 - truncation error, 268–269
- Nonlinear systems, 203–256
 - analytic bandpass memoryless models
 - Chebyshev transform, 210
 - describing function, 210
 - limiters, 213
 - power series, 214
 - analytic models with memory, 237–252
 - polyspectral models, 245–252
 - Volterra series, 237–245
 - baseband models
 - general, 206–207
 - hysteresis, 211
 - limiter, 212–213
 - block models, memoryless, 215–223
 - AM/AM, AM/PM, 215, 218–219, 220–221
 - lowpass equivalent, 219–220
 - quadrature model, 217
 - serial model, 216
 - block models with memory, 227–237
 - Abuelma'atti model, 232–234
 - instantaneous frequency model, 255
 - nonlinear parametric model, 253
 - power-dependent transfer function model, 252
 - Poza-Sarkozy-Berger model, 227–229
 - Saleh's model, 154
 - three-box models, 236–237
 - two-box models, 234–236
 - intermodulation products, 221–223
 - measurement techniques, 275–284
 - time-domain, 280–284
 - two-tone, 278–280
 - VNA, 275–277
 - nonlinearities with memory, classification, 224–227
 - sampling rate, 207–209

- Optical amplifier model, 271–275
- Optical fiber model
 - multimode, 370–373
 - single mode, 370–373
- Optical source models, 492–494
 - rate equations, 492
 - simulation model, 493–494
 - block diagram, 493
 - explicit form, 494
- Order of computation, 524
- Order of execution, 524
- Order statistics, 315
- Outage probability
 - definition, 790–791
 - estimation methodology
 - example in Case Study I, 781–793
- Overlap-and-add: *see* Finite impulse response (FIR)
- Overlap-and-save: *see* Finite impulse response (FIR)
- Parseval's theorem
 - applied to SNR estimation, 687
 - for continuous signals, 32
 - for discrete signals, 95
 - for periodic signals, 26
- Periodogram, 647–648
- Phase-locked loop
 - assembled model, 522–529
 - effect of delay on, 531–534
 - distribution of phase error, 662–663
 - as an FM demodulator, 465–466, 529–531
 - modeling considerations, 514–515
 - stand-alone model, 515–522
- Phase noise
 - ARMA model for residual, 804
 - averaging effects of, on BER, 753–754
 - distribution, in a PLL, 662–663
 - equivalent process methodology, 502–504
 - general model for, 800
 - residual, 799–803
 - effect on block-coded systems, Case Study II, 808
 - tracked, 800
 - untracked, 800
 - variance, of phase error, 663
- Poisson Process, 346
- Power spectral density (PSD), 336–340
 - bandpass, 337
 - bandwidth, 337
 - definition, 336
 - estimation of, 645–654
 - lowpass, 337
 - power calculation, 337
 - properties, 336
 - of random sequences, 339
- Probability density function (pdf)
 - conditional, 295
 - definition, 293
 - estimation of, 645–654
 - joint, 294
 - marginal, 294
- Probability distribution function, 293
 - estimation of, 640–641
- Probability mass function, 293
- Pseudo-BER, 705
- Pseudonoise (PN) sequences, 386–392
 - binary, 386–389
 - generation, 386–388, 389
 - octal, 391
 - properties, 387–389
 - quaternary, 392
- Pseudothreshold, 705
- Quadrature rule
 - formula, 323–324
 - weights for Gaussian, 895
- Quantization, 415–416
 - of a Gaussian pdf, 369
 - A-law, 416
 - MSE, 367, 369
 - mu-law, 415
 - nonuniform, 368–369
 - about simulating, 416–417
 - uniform, 367–368
- Quasianalytical (QA) method, 737–754
 - applied to binary systems, 740–743
 - applied to hard-decision-decoding, 748–753
 - applied to PAM systems, 743–744
 - applied to PSK systems, 745–748
 - applied to QAM systems, 744–745
 - applied to soft-decision decoding, 753–754
 - fast QA, applied to Case Study I, 786–787
 - incorporating jitter, 753–754
- Quasistatic approach to simulation, 570, 653
- Radio channels: *see* Multipath fading channels
- Radio-relay system, Case Study I, 763–793
- Random number generators (RNG)
 - arbitrary pdf, 377
 - acceptance/rejection method, 381–382
 - transformation method, 377
 - exponential, 379
 - gamma, 379
 - Gaussian, 383–384
 - Box-Muller method, 383
 - correlated sequences, 392
 - summation method, 383
 - white noise, 383
 - geometric, 379
 - Poisson, 379
 - seed, 374, 387
 - summary of algorithms, 384
 - uniform RNG, algorithms for 374–37
 - linear congruential, 374
 - Marsaglia–Zaman, 376–377
 - Wichman–Hill, 376
- Random processes
 - classification, 326–327
 - continuous amplitude, 326–327
 - continuous time, 326–327

- Random processes (*cont.*)
 - classification (*cont.*)
 - discrete amplitude, 326–327
 - discrete time, 326–327
 - complex-valued, 326–327
 - definitions, 326–327
 - ensemble averages, 329–331
 - autocorrelation, 329
 - autocovariance, 329
 - correlation coefficient, 330
 - ergodicity, 331
 - mean, 329
 - member function, 326
 - notations, 326
 - simulation implications, 333
 - stationary, 331
 - strict sense, 331
 - wide sense, 331
 - time averages, 332
 - transformation, 357
- Random sequences
 - ARMA model, 342
 - applied to residual phase noise, 804
 - autoregressive (AR), 342
 - binary PN, 386
 - independent, 340
 - Markov 340
 - M-ary PN, 389
 - moving average (MA), 342
 - PSD of, 339, 343
 - random binary, 345
 - random M-ary, 344
 - white, 340
- Random variables
 - complex-valued, 297
 - continuous, 293
 - chi-square, 303
 - exponential, 301
 - F, 304
 - gamma, 302
 - Gaussian, 301, 304
 - generalized exponential, 304
 - Rayleigh, 302
 - Rice, 356
 - student's t, 303
 - uniform, 300
 - definition, 292
 - discrete, 293
 - binomial, 298
 - Erlang, 302
 - geometric, 299
 - multinomial, 304
 - negative binomial, 299
 - Poisson, 299
 - uniform, 298
 - real-valued, 292
 - scalar (univariate), 294
 - vector (multivariate), 294
- RF sources, 491–492
 - phase noise, 491
- Roundoff noise
 - FFT, 182
 - FIR filters, 182
 - IIR filters, 181
- Sampling
 - of deterministic signals
 - impulse sampling, 83–85
 - sampling rate for nonlinear systems, 207–208
 - sampling theorem, 86
 - of random processes, 363–367
 - aliasing, 364
 - aliasing error, 365
 - bandpass, 365
 - lowpass, 363–364
 - minimum rate, 363–364
 - reconstruction, 363
 - sampling principle, 363–364
- Sampling rate, for simulation, 365; *see also* Multirate
- Scatter diagram, 666–667
 - applied, in Case Study III, 815–817, 819, 821–822
- Semianalytic: *see* Quasianalytical
- Separable model for LTV 132
- Shot noise, 251–252
- Signals, 14–56–59
 - continuous, 56–57
 - discrete, 57–59
 - periodic, 72
- Signal-to-noise ratio estimation, 670–678
- Simulation methodology
 - approach, to performance evaluation, 1–2, 16–17, 26–28
 - calibration, 534–539
 - defined, 13–14
 - error sources, 28–36
 - for fading channels, 611
 - speech coder, 613
 - symbol level, 612
 - waveform level, 611
 - of feedback loops. 46–47, 495–534
 - hardware environment, 45
 - hardware in the loop, 25–26
 - modeling, 17–26
 - devices, 21–22
 - hierarchy, 18–19
 - hypothetical systems, 23–25
 - partitioning, 19
 - random processes, 22–23
 - systems, 21
 - software environment, 42–45
 - and systems engineering, 49–52
 - validation of, 36–41
- Source coding: *see* Formatting/source coding
- Source signals, 411–414
- Specifications, 601, 637, 697
- Spectral factorization, 60, 396, 344
- Spectrum for deterministic signals
 - Fourier, 67
 - for periodic signals, 65

- Spline interpolation, 100–105
- Spread spectrum techniques
 - CDMA, 489–491
 - direct sequence (DS), 489
 - generic block diagram, 489
 - multirate, in Case Study I, 845–848
- Standard deviation 293
- Synchronization, *see also* Phase-locked loop; Carrier recovery; Timing recovery
 - block delay estimator, 658–659
 - block phase estimator, 660–661
 - carrier recovery methods
 - BPSK, 504–506
 - Costas loop, 527–528
 - MPSK, 510–512
 - PLL, 514–524
 - QPSK, 510–512
 - estimating carrier phase, 655–657, 660–661
 - estimating system delays, 655–657, 658–659
 - methods of simulating, 498–504, 514–534
 - equivalent random process model, 502–504
 - hardwired, simulation method, 500–502
 - timing recovery
 - BPSK, 506–510
 - QPSK, 513
- Systems
 - linear time-invariant (LTI), 55–184
 - selection of simulation method, 182–184
 - linear time-varying (LTV), 184–196
 - nonlinear: *see* Nonlinear systems
- Systems engineering and simulation, 636–640
- Tables
 - biquadratic coefficients for classical filters, 198–201
 - chi-square probabilities, 897
 - correlation theorem, 129
 - Fourier transform, 129
 - frequency convolution, 128
 - Gaussian quadrature rules, 895
 - Gaussian tail probabilities, 891
 - Hermite polynomial coefficients, 893
 - Laplace transform, 130
 - Parseval's theorem, 128
 - z transform, 130
- Tapped-delay line, *see also* Equalization
 - FIR filter model, 150
 - LTV system model, 196
- Testing of RNGs, 400–405
 - chi-square, 404
 - Durbin–Watson, 403–404
 - goodness of fit, 405
 - KS, 404
 - tests for
 - periodicity, 402–403
 - stationarity, 400–401
 - uncorrelatedness, 402–403
- Test signals, 412–413
- Time-reliability product, 631; *see also* variance
- Timing recovery
 - bias, 754
 - for BPSK, 506–510
 - delay and multiply method, 508–510
 - early-late gate method, 510
 - effect on BER, 753–754
 - estimation of time delay, 655–657, 658–659
 - jitter, 754
 - for QPSK, 512
- Transfer function
 - of a code, 753
 - continuous, 108
 - for linear differential equations, 111
 - discrete, 117–118
 - for difference equation, 166
- Transformation from continuous to discrete: *see* Bilinear, impulse invariant
- Transformation of random processes, 357–361
 - autocorrelation and PSD, 357
 - autocorrelation of the output, 357
 - filtering, 358
 - noise bandwidth, 358–359
 - Gaussian, 357
 - integration, 360
 - mean of the output, 357
 - output of
 - linear Systems, 357–360
 - nonlinear systems, 361
 - time-varying systems, 362
 - PSD, 357
 - PSD of the output, 357
 - stationarity, 357
- Transformation of random variables 309–316
 - continuous, 310
 - discrete, 310
 - linear, 313
 - nonlinear, 316
 - scalar, 310
 - sum, 314
 - vector, 313
- Validation, 36–41
- Variance, of the estimator (of)
 - average level, 632–635
 - average power, 638–639
 - BER (importance sampling), 718
 - BER (Monte Carlo), 694–697
 - BER (tail extrapolation), 707–708
 - power spectral density, 652–653
 - probability density, 644–645
- Windows
 - for filters, 72, 150–151, 161–162
 - for power spectral density estimator, 648–652
- Wireless channels: *see* Multipath fading channels; Free-space channels
- z transform, 115–116
 - properties, 117–118
 - table of, 130

This page intentionally left blank