Advances in Industrial Control

Steven X. Ding

Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems

Springer

Advances in Industrial Control

Series editors

Michael J. Grimble, Glasgow, UK
Michael A. Johnson, Kidlington, UK

For further volumes:
http://www.springer.com/series/1412

Steven X. Ding

Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems

Springer

Steven X. Ding
Institute for Automatic Control and
Complex Systems (AKS)
University of Duisburg-Essen
Duisburg
Germany

ISSN 1430-9491 ISSN 2193-1577 (electronic)
ISBN 978-1-4471-6409-8 ISBN 978-1-4471-6410-4 (eBook)
DOI 10.1007/978-1-4471-6410-4
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2014935149

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To My Parents and Eve Limin

Series Editors' Foreword

The series *Advances in Industrial Control* aims to report and encourage technology transfer in control engineering. The rapid development of control technology has an impact on all areas of control discipline. New theory, new controllers, actuators, sensors, new industrial processes, computer methods, new applications, new philosophies…, new challenges. Much of this development work resides in industrial reports, feasibility study papers, and the reports of advanced collaborative projects. The series offers an opportunity for researchers to present an extended exposition of such new work in all aspects of industrial control for wider and rapid dissemination.

The classifier "data-driven" for some control design methods, and fault detection and isolation (FDI) techniques appears to date from the 1990s. However the idea of going directly from plant data to a control design has a well established lineage arguably commencing with the PID controller design techniques of Ziegler and Nichols. This type of controller design principle found further development through for example, the Åström relay experiment, the phase-locked loop scheme and the iterative feedback tuning method. Despite the fact that these methods use online plant responses whether they would be classed as "data-driven" is an interesting question. So, what are the characteristics that identify a technique as "data-driven"? Some ideas are found in the early chapters of this *Advances in Industrial Control monograph, Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems* written by Professor Steven X. Ding of the University of Duisburg–Essen, Germany. After an introductory chapter, and then a chapter describing the case study examples to be used to demonstrate and illustrate the techniques in the monograph, Professor Ding devotes a chapter to a basic statistical approach to fault detection introducing hypothesis-testing ideas, and test statistics such as the T^2 statistic. Later chapters describe techniques based on methods like principal components analysis, partial least squares analysis, and canonical variate analysis. Thus, one feature of data-driven methods here is that they are used with large, often noisy, process data sets and typical tools are those based on statistica concepts and algorithms.

A look at the industrial motivation for these methods also adds interesting background for "data-driven" classifier. In industry, and particularly the process industries, the widespread use of computer systems, along with new sensor

hardware generates significant quantities of real-time process data that is often stored as large historical process data archives. It has been our own past experience that industrial operatives frequently asked what could be done with both the real-time and archived historical data. The analysis of large data sets (data that is often contaminated with process noise) naturally leads to the use of statistical techniques for the extraction of information that is used to measure process performance, identify fault conditions, and produce the evidence needed to direct performance improvements. Some *Advances in Industrial Control* monographs that may be useful reading for these problems are given here.

Statistical process control is one group of methods that can be used to tackle these large process data sets and produce indicators for fault and "out of control" conditions in manufacturing and industrial processes. The *Advances in Industrial Control* series has published a monograph of recent research in this field. The reader should find *Multivariate Statistical Process Control* by Zhiqiang Ge and Zhihuan Song (ISBN 978-1-4471-4512-7, 2012) of interest for new ideas in this field.

A different approach to dealing with process monitoring and assessment uses knowledge-based concepts and the methods of data mining. Xue Z. Wang has an *Advances in Industrial Control* monograph, *Data Mining and Knowledge Discovery for Process Monitoring and Control* (ISBN 978-1-85233-137-5, 1999) that covers such ideas. The monograph *Computational Intelligence in Time Series Forecasting* by Ajoy K. Palit, and Dobrivoje Popovic (ISBN 978-1-85233-948-7, 2005) may be of interest.

As previously mentioned, the problem of how to exploit the large data sets arising from heavily instrumented production lines is one that is particularly relevant to the process and chemical industries. Evan L. Russell, Leo H. Chiang and Richard D. Braatz wrote on data-driven FDI methods in an *Advances in Industrial Control* series monograph *Data-driven Techniques for Fault Detection and Diagnosis in Chemical Processes* (ISBN 978-1-85233-258-7, 2000). This monograph was revised for our sister series *Advanced Textbooks in Control and Signal Processing* as *Fault Detection and Diagnosis in Industrial Systems* with authors Leo H. Chiang, Evan L. Russell, and Richard D. Braatz (ISBN 978-1-85233-327-0, 2001).

Returning now to the work of Professor Steven X. Ding; he has had a long association with research—both theory and applications—in the FDI and control field. His monograph *Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms and Tools* was well received and went into a second edition in our series (ISBN 978-1-4471-4798-5, 2013). We are now pleased to welcome into the *Advances in Industrial Control* series this companion volume: *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems* that focuses on statistical data-based fault diagnosis and the implementation of model-based methods in a statistical data framework.

Glasgow, Scotland, UK

M. J. Grimble
M. A. Johnson

Preface

My activity in the field of data-driven process monitoring and diagnosis began with the visit of Prof. B. Huang from the University of Alberta as an AvH-fellow (fellow of Alexander von Humboldt Foundation) in our institute in 2003. I have learnt some data-driven methods and was so impressed by the first successful application of a simple data-driven detection algorithm on a test bed in our laboratory. Motivated by this success, a research group working on data-driven methods was established in our institute in 2004. This working group has been active in European research projects NeCST (2004–2007) and PAPYRUS (2010–2013), in national research grant ZIM-EVA (2010–2012) and in a number of industrial collaborative projects.

For about three decades, model-based fault diagnosis in dynamic automatic control systems has been a research focus of our institute. This tradition has a strong influence on our research activities in the field of data-driven process monitoring and diagnosis. Much attention has been paid to the development of data-driven fault diagnostic methods for dynamic processes and to the establishment of a link between data-driven and model-based methods. Driven by our collaboration with the automotive and process industry, our recent research is dedicated to the data-driven design of fault-tolerant and lifetime management of automatic control systems.

This book is a summary of our work of the past 10 years. Since the summer semester of 2013, the draft version of this book also serves as lecture notes for the Master's course on *Fault Diagnosis and Fault Tolerant Systems,* offered in the Department of Electrical Engineering and Information Technology (EIT) at the University of Duisburg-Essen. It is worth mentioning that the main results and methods described in this book are presented in the form of algorithms, and detailed case studies are included in most chapters. In fact, this book is so structured that it can also be used as a self-study book for engineers in the application fields of automatic control.

This book would not be possible without the valuable support of many people. I would like to thank Prof. Huang for his help in the starting period of our work, Prof. P. Zhang for her remarkable contributions in these 10 years, Dr. Naik and Dr. Yin for their excellent work as the first member of our working group,

Prof. Jämsä-Jounela from the University of Aalto in Finland and Prof. Sauter from the Université de Lorraine in France for the unforgettable collaboration in the European projects, and Prof. Jeinsch and Dr. Engel for the joint national research grant.

I would especially like to thank my Ph.D. students and co-workers for their contributions to the case studies. They are Ms. L. Li (Chaps. 11,12 and 15), Mr. Z. Chen (Chaps. 6,7 and 11), Mr. H. Hao (Chaps. 2, 3, 5 and 6), Mr. H. Luo (Chaps. 2–4 and 9), Mr. K. Zhang (Chaps. 4, 6 and 7) and Mr. Y. Zhang (Chaps. 10, 14 and 15). In addition, I am greatly indebted to Mr. T. Könings for the extensive editorial corrections.

Finally, I would like to express my gratitude to Oliver Jackson and Charlotte Cross from Springer-Verlag and the Series Editor for their valuable support.

Duisburg, December 2013 Steven X. Ding

Contents

Notation

$\forall$	For all
$\in$	Belong to
$\equiv$	Identically equal
$\approx$	Approximately equal
:=	Defined as
$\Rightarrow$	Implies
$\Leftrightarrow$	Equivalent to
$>>$ ($<<$)	Much greater (less) than
max (min)	Maximum (minimum)
sup (inf)	Supremum (infimum)
$\mathcal{R}$	Field of real numbers
$\mathcal{R}^n$	Space of real n-dimensional vectors
$\mathcal{R}^{n\times m}$	Space of n by m matrices
X^T	Transpose of X
$X^\perp$	Orthogonal complement of X
X^{-1}	Inverse of X
X^-	Pseudo-inverse of X (including left or right inverse)
$X(i:j,p:q)$	Submatrix consisting of the ith to the jth rows and the pth to the qth columns of X
$col(X)$	vectorize X, $col(X) = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \in \mathcal{R}^{nm}$, for $X = [x_1 \quad \cdots \quad x_m] \in \mathcal{R}^{n\times m}, x_i \in \mathcal{R}^n,\ i = 1,\cdots,m$
$rank(X)$	Rank of X
$trace(X)$	Trace of X
$\det(X)$	Determinant of X
$\lambda(X)$	Eigenvalue of X
$\bar{\sigma}(X)$ ($\sigma_{\max}(X)$)	Largest (maximum) singular value of X
$\underline{\sigma}(X)$ ($\sigma_{\min}(X)$)	Least (minimum) singular value of X
$\sigma_i(X)$	The ith singular value of X
$diag(X_1,\cdots,X_n)$	Block diagonal matrix formed with $X_1,\cdots,X_n$

$\|\cdot\|_E$	Euclidean norm of a vector
$\|\cdot\|_F$	Frobenius norm of a matrix
prob$(a < b)$	Probability that $a < b$
$\mathcal{N}(a, \Sigma)$	Gaussian distribution with mean vector a and covariance matrix Σ
$x \sim \mathcal{N}(a, \Sigma)$	x is distributed as $\mathcal{N}(a, \Sigma)$
$\chi^2(m)$	Chi-squared distribution with m degrees of freedom
$\mathcal{E}(x)$	Expectation of x
$var(x)$	Variance of x

Part I
Introduction, Basic Concepts and Preliminaries

Chapter 1
Introduction

It is the state of the art that automatic control systems are widely integrated into complex industrial processes. In order to meet ever increasing demands for high production and product quality as well as for economic and ecological operations, today's industrial processes have become more complex and their degree of automation is significantly growing. This development calls for more system reliability, dependability and safety. Associated with this, process monitoring, diagnosis and fault-tolerant control receive considerably enhanced attention, both in the engineering and research domains.

1.1 Basic Concepts

Fault diagnosis is a classical engineering area. The major and well-established technologies can be classified into

- *Hardware redundancy based fault diagnosis*: The core of this technology consists of the reconstruction of the process components using the identical (redundant) hardware components. A fault in the process component is then detected if the output of the process component is different from one of its redundant components. The main advantage of this scheme is its high reliability and the direct fault isolation. The use of redundant hardware results in, on the other hand, high costs, and thus the application of this scheme is usually restricted to a number of essential and safety-relevant components.
- *Signal processing based fault diagnosis*: On the assumption that certain process signals carry information about the faults to be detected and this information is presented in the form of "symptoms", a fault diagnosis can be achieved by suitable signal processing. Typical symptoms are time domain functions like magnitudes, arithmetic or quadratic mean values, limit values, trends, statistical moments of the amplitude distribution or envelope, or frequency domain functions like spectral power densities, frequency spectral lines, cepstrum analysis, etc. The signal

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_1, © Springer-Verlag London 2014

processing based methods are mainly used for those processes in the steady state. Their efficiency for the detection of faults in dynamic systems, which have a wide operating range, is considerably limited.

- *Statistical data based fault diagnosis:* The major feature of this technology is the availability of large amount of process data, both historical data collected and recorded during process operations, and online measurement data. The application of this class of diagnosis methods principally consists of two phases: (i) training, in which the historical data sets are presented as a priori knowledge of the process under monitoring and transformed to a diagnostic system or algorithm (ii) online running, in which the online measurement data are processed in the diagnostic system or using the diagnostic algorithm for a reliable fault detection and identification. Similar to the signal processing based methods, this technique is mainly applied to static processes.
- *Analytical model based fault diagnosis*: The core of the analytical model based technology is the availability of an analytical process model that presents a mathematical description (model) of the process dynamics and major features. A model-based fault diagnostic system consists of two parts: (i) generation of a so-called *residual signal,* which is the difference between the measured process variables and their estimates based on the process model (ii) *residual evaluation* and *decision making*. The model-based methods are, due to the embedded analytical process model, powerful in dealing with fault diagnosis in dynamic processes.
- *Knowledge-based fault diagnosis*: Knowledge-based fault diagnosis is based on a qualitative model which represents a priori knowledge of the process under monitoring. Fault diagnosis is then realized by running well-developed search algorithms. The core of a knowledge-based fault diagnosis system is an expert system which consists of (i) a knowledge base (ii) a data base (iii) an inference engine and (iv) an explanation component. Knowledge-based fault diagnosis technique is receiving increasing attention in dealing with fault diagnosis in complex technical processes.

In the context of reliable and safe automatic control systems, *fault-tolerant control* (FTC) is closely associated with fault diagnosis. Generally speaking, FTC is the immediate step after a successful fault diagnosis to reconfigure the controller and control system so that a fail-safe operation of the automatic control system is guaranteed. For a feedback control system, the system stability is an essential requirement. In the context of FTC addressed in this book, stability is the fail-safe operation of automatic control systems.

The focus of this book is on process data based fault diagnosis and the implementation of (analytical) model-based methods in the statistical data based framework. We call it *data-driven design of fault diagnosis systems*. In addition, in the context of lifetime management of automatic control systems, data-driven design and implementation of fault-tolerant control will be addressed.

In this book, we deal with fault diagnosis for the following three essential tasks:

- *Fault detection*: detection of the occurrence of faults in the functional units of the process, which lead to undesired or unacceptable behavior of the whole system

- *Fault isolation*: localization (classification) of different faults
- *Fault identification*: determination of the type, magnitude and cause of the fault.

Often, we use the short form FDI to represent these three tasks or fault diagnosis in general.

1.2 Motivation

1.2.1 Data-Driven and Model-Based FDI

Data-driven techniques are widely applied in the process industry for process monitoring and diagnosis purposes. Among numerous data-driven schemes, the multivariate analysis (MVA) technique with principal component analysis and partial least squares as its representative methods is, due to its simplicity and efficiency in processing huge amount of process data, recognized as a powerful tool for addressing statistical process monitoring and diagnosis problems.

The major objective of integrating an automatic control system into a complex technical process is to cope with process dynamics, nonlinearity in the process and disturbances existing around the process. In order to deal with the existing dynamics in the process under consideration, dynamic MVA algorithms, recursive, fast moving window and multiple-mode variants of the standard MVA algorithms have been developed in recent years. The MVA technique is generally applied to static or dynamic processes in the steady state, and delivers optimal performance for high level process monitoring and diagnosis in large-scale systems. In comparison, the model-based FDI technique, exploiting the application of advanced system and control theory, provides a more efficient and powerful tool to investigate FDI issues in highly dynamic systems and control loops, which are generally located at the process level. The possible high FDI performance is often achieved at the cost of a complex modeling process and, based on it, a sophisticated FDI system design procedure. In practice, the decision for the application of a data-driven or a model-based FDI scheme is often made by a trade-off between engineering costs and FDI performance.

In this book, FDI methods will be introduced, which link the data-driven and model-based FDI. Concretely, data-driven schemes will be developed for the preliminary design and construction of an FDI system, which is then realized in the form of a model-based FDI system and further optimized during its operation.

1.2.2 Fault-Tolerant Control and Lifetime Management

In the framework of lifetime management of automatic control systems, FTC is only a special action that ensures a fail-safe operation under real-time conditions if some

components in the process or/and in the automatic control system fail or become faulty. In practice, further scenarios are of interests:

- component or device replacement in a maintenance or repair action
- changes in the operation conditions and in the environment around the process
- aging behavior in the components.

It is well-known that robust, adaptive and fault-tolerant control techniques are powerful tools to deal with the possible performance degradation in these scenarios. In comparison, only few research efforts have been made to integrate controller configuration and optimization into the maintenance and lifetime management plan.

In the last part of this book, FTC issues will be addressed in the context of lifetime management of automatic control systems, in which FTC architectures, the function and priority of the embedded controllers play a central role.

1.2.3 Information Infrastructure

In the past two decades, rapid development in information infrastructure of modern automatic control systems can be observed. The most significant developments include:

- data acquisition becomes multifarious: wide application of advanced sensor techniques, integration of intelligent actuators and process components equipped with smart sensors or sensing components into automatic control systems and control loops, enhanced application of sensor networks
- networking is the state of the art: using the well-developed networking techniques, data transmissions and information exchanges within and between automatic control systems become more efficient, fast and reliable
- high computation power is available: more and more computers of the new generation are integrated into automatic control systems
- the capacity for data archival storage increases significantly: as a result, a long-term and reliable archiving of process data becomes possible.

Consequently, the fundamentals for the application of data-driven techniques, that is,

- sufficient and well-archived data collected and recorded during process operations and maintenance actions
- sufficient real-time process data and
- high-speed computation

are well established. Further progress and development in this area can be expected in the future. Hence, the development of the data-driven FDI and FTC methods can also be viewed as a logical and reasonable outcome of the rapid development in computer, microelectronic and communication technologies.

1.3 Outline of the Contents

This book is concerned with three thematic areas: (i) some basic detection and MVA methods for process monitoring and diagnosis in Chaps. 3–7 and 11 in part (ii) data-driven design of I/O data set model and observer-based FDI schemes in Chaps. 8–12 (Chap. 11 in part) (iii) data-driven design of fault-tolerant control systems in the context of lifetime management of automatic control systems in Chaps. 13–15 and 12 in part. In the first thematic area, only static or dynamic processes in the steady state are addressed, while in the latter two thematic areas the focus is on the dynamic automatic control systems and feedback control loops. Below, each of these chapters is described and commented upon.

Chapter 2 is dedicated to the introduction and description of two real laboratory systems and one widely used benchmark process. They will be adopted in the case studies throughout this book and applied to illustrate and demonstrate the achieved theoretical results and the application of diagnosis and control algorithms.

In Chap. 3, a basic statistic fault detection problem and its variants are formulated and their solutions are investigated. Although this fault detection problem is presented in the simplest form, a number of the subsequent studies can be considered as an extension of this problem. Also important is the introduction of some basic concepts, detection schemes and standard statistical methods as tools, which are frequently used in the subsequent chapters and useful for a systematic study of process monitoring and diagnosis.

The analog (basic) fault detection problems for processes with deterministic disturbances are addressed in Chap. 4. Although in the subsequent chapters these fault detection problems will not be studied explicitly, extensions of the results to dynamic processes with deterministic disturbances are straightforward, due to the analogy between the problem formulations in Chaps. 3 and 4. A useful result in this chapter is the different interpretations of the problem solution and the idea of viewing the fault detection formulation as a disturbance estimation problem, which leads to a simplified threshold setting.

In Chap. 5, the first and most popular MVA method, principal component analysis (PCA), and its applications in process monitoring and diagnosis are introduced. In comparison with the fault detection schemes presented in Chap. 3, the applications of PCA technique to process monitoring and fault detection are reviewed. Some critical comments direct the reader's attention to the correct use and understanding of PCA technique for monitoring and detection purposes.

Partial least squares (PLS) regression introduced in Chap. 6 is the second MVA method addressed in this book in the context of process monitoring, fault diagnosis and prediction of key process variables. The focus of our study is on the comparison between PLS and least squares (LS) for regression purposes.

Chapter 7 deals with the application of canonical correlation analysis (CCA), the third MVA method, to process monitoring. Different from its traditional application in system identification, we are concerned with monitoring and detection issues in static or steady processes.

Chapter 8 gives a brief introduction to some basic model and observer-based fault diagnostic schemes, which are essential for our subsequent studies on data-driven design of FDI and FTC systems. The last section in this chapter is dedicated to the different types of input-output (I/O) data set models.

Beginning with Chap. 9, dynamic processes are under consideration. We first deal with FDI problems based on the I/O data set models. This study is similar to the application of the MVA methods to solving FDI problems and serves as a link to the FDI and FTC schemes introduced in the next chapters.

In Chap. 10, data-driven design of observer-based FDI systems is studied. The core of this work is the data-driven realization of the so-called kernel representation of dynamic systems, which results in a parametrization of observer-based residual generators. The recursive implementation of the data-driven kernel representation leads to an observer that can be used for the FDI and state estimation purposes. The approaches and algorithms presented in this chapter are useful not only for FDI but also for FTC issues to be addressed in Chaps. 13–15.

Chapter 11 consists of two parts. In the first part, recursive computation algorithms for singular value decomposition are presented. They will then be applied to approaching adaptive MVA based fault detection. The second part of this chapter is devoted to the adaptive implementation of the kernel representation by using the adaptive observer technique. The adaptive algorithm proposed in this part will also be applied in the FTC framework. By means of these two adaptive schemes, the FDI systems designed by data-driven methods become robust against changes in the process dynamics.

In order to be adaptive to the changes in the environment around the process being monitored, in Chap. 12 iterative optimization of data-driven designed estimators and residual generators is addressed. The basic idea behind this work is to optimize the estimation by means of iterative interactions between the estimator and the environment around the process. To this end, the iterative generalized LS serves as the computation tool and, based on it, iterative recursive LS and Kalman filter algorithms are developed.

In the last part of this book, FTC and lifetime management of automatic control systems are studied in the data-driven design framework. At first, an FTC architecture is presented in Chap. 13, which is constructed based on the observer-based realization of Youla parametrization of stabilizing controllers. Further variants of the Youla parametrization result in the functionalization and prioritization of the control parameters in the context of FTC implementation. This results provides the control theoretical basis for a data-driven implementation of lifetime management of automatic control systems.

The data-driven design of feedback and feed-forward controllers is presented in Chap. 14. The core of this study is the data-driven realization of the so-called image representation of process dynamics, a dual form of the kernel representation. An integration of this controller design scheme and the algorithm given in Chap. 10 for the observer construction leads to a data-driven design of an observer-based automatic control system.

According to the prioritization of the control parameters, the lifetime management strategy is realized in two phases. As described in Chap. 15, the higher prioritized parameters, which are the parameters designed in the data-driven fashion and guaranteeing the system stability, that is, the fail-safe operation, are updated using the adaptive control schemes under the real-time condition. This is an event-driven procedure, in which a successful fault detection or a maintenance action or a change in the operation condition define an event. After a successful updating of the higher prioritized parameters, system performance will be optimized by running iterative optimization algorithms for lower prioritized parameters.

1.4 Notes and References

In the last two decades, a great number of monographs on process monitoring, diagnosis and fault-tolerant control have been published. Among them, for instance,

- [1–10] are dedicated to the model-based FDI techniques
- [11, 12] are focused on the data-driven process monitoring and FDI methods and
- [8, 13] deal with FTC issues.

Concerning the recent surveys on these topics, we recommend the reader the following representative papers:

- [14–17] on the model-based FDI methods
- [18–21] on the data-driven process monitoring and FDI
- [22] on the knowledge-based fault diagnosis.

Process monitoring, fault diagnosis and fault-tolerant control form an interdisciplinary field. For a success in this field, good knowledge of MVA and statistics, linear algebra and matrix theory, linear system theory, adaptive and robust control theory is necessary. Throughout this book, known and well-developed methods and algorithms from these thematic areas will serve as the major tools for our study. Among the great number of available books on these topics, we would like to mention the following representative ones:

- statistical methods and MVA [1, 23]
- matrix theory [24, 25]
- linear system theory [26, 27]
- filter theory [28]
- adaptive control theory [29]
- robust control theory [30].

References

1. Basseville M, Nikiforov I (1993) Detection of abrupt changes—theory and application. Prentice-Hall, New York
2. Gertler JJ (1998) Fault detection and diagnosis in engineering systems. Marcel Dekker, New York
3. Mangoubi R (1998) Robust estimation and failure detection. Springer, London
4. Chen J, Patton RJ (1999) Robust model-based fault diagnosis for dynamic systems. Kluwer Academic Publishers, Boston
5. Patton RJ, Frank PM, Clark RN (eds) (2000) Issues of fault diagnosis for dynamic systems. Springer, London
6. Gustafsson F (2000) Adaptive filtering and change detection. Wiley, Chichester
7. Simani S, Fantuzzi S, Patton RJ (2003) Model-based fault diagnosis in dynamic systems using identification techniques. Springer, London
8. Blanke M, Kinnaert M, Lunze J, Staroswiecki M (2006) Diagnosis and fault-tolerant control, 2nd edn. Springer, Berlin
9. Isermann R (2006) Fault diagnosis systems. Springer, Berlin
10. Ding SX (2013) Model-based fault diagnosis techniques—design schemes, algorithms and tools, 2nd edn. Springer, London
11. Russell EL, Chiang L, Braatz RD (2000) Data-driven techniques for fault detection and diagnosis in chemical processes. Springer, London
12. Chiang LH, Russell EL, Braatz RD (2001) Fault detection and diagnosis in industrial systems. Springer, London
13. Mahmoud M, Jiang J, Zhang Y (2003) Active fault tolerant control systems. Springer, London
14. Venkatasubramanian V, Rengaswamy R, Yin K, Kavuri S (2003) A review of process fault detection and diagnosis part I: quantitative model-based methods. Comput Chem Eng 27:293–311
15. Zhang P, Ding SX (2008) On fault detection in linear discrete-time, periodic, and sampled-data systems (survey). J Control Sci Eng 2008:1–18
16. Mangoubi R, Desai M, Edelmayer A, Sammak P (2009) Robust detection and estimation in dynamic systems and statistical signal processing: intersection, parallel paths and applications. Eur J Control 15:348–369
17. Hwang I, Kim S, Kim Y, Seah C (2010) A survey of fault detection, isolation, and reconfiguration methods. IEEE Trans Control Syst Tech 18:636–653
18. Venkatasubramanian V, Rengaswamy R, Kavuri S, Yin K (2003) A review of process fault detection and diagnosis part III: process history based methods. Comput Chem Eng 27:327–346
19. Qin SJ (2003) Statistical process monitoring: basics and beyond. J Chemom 17:480–502
20. Qin SJ (2009) Data-driven fault detection and diagnosis for complex industrial processes. In: Proceedings of IFAC SAFEPROCESS Symposium, pp 1115–1125
21. Ding SX (2014) Data-driven design of monitoring and diagnosis systems for dynamic processes: a review of subspace technique based schemes and some recent results. J Process Control. 24:431–449
22. Venkatasubramanian V, Rengaswamy R, Kavuri S (2003) A review of process fault detection and diagnosis part II: qualitative models and search strategies. Comput Chem Eng 27:313–326
23. Härdle WK, Simar L (2012) Applied multivariate statistical analysis, 3rd edn. Springer, Berlin
24. Gantmacher FR (1959) The theory of matrices. Chelsea Publishing Company, New York
25. Strang G (2009) Introduction to linear algebra, 4th edn. Wellesley-Cambridge Press, Wellesley
26. Chen CT (1984) Linear system theory and design. Holt Rinehart, Winston
27. Kailath T (1980) Linear systems. Prentice-Hall, Englewood Cliffs
28. Kailath T, Sayed A, Hassibi B (1999) Linear estimation. Prentice Hall, New York
29. Tao G (2003) Adaptive control design and analysis. Wiley-Interscience, New York
30. Zhou K (1998) Essential of robust control. Prentice-Hall, Englewood Cliffs

Chapter 2
Case Study and Application Examples

In this chapter, three application examples are introduced, which will be applied to illustrate the major process monitoring, diagnosis and control schemes presented and studied in this book. The first two are real laboratory control systems, which are widely used both in the model-based and data-driven fault diagnosis research. The third one is a benchmark process that is mainly applied for the test and demonstration of data-driven control and monitoring schemes. The objective of introducing these application examples in the first part of this book, immediately after *Introduction,* is to provide the reader with the useful application background and understandings of some basic technical concepts in the process monitoring and fault diagnosis field.

2.1 Three-Tank System

A three-tank systems, as sketched in Fig. 2.1, has typical characteristics of tanks, pipelines and pumps used in chemical industry and thus often serves as a benchmark process in laboratories for process control. The model and the parameters of the three-tank system introduced here are from the laboratory setup DTS200.

2.1.1 Process Dynamics and Its Description

Applying the incoming and outgoing mass flows under consideration of Torricelli's law, the dynamics of DTS200 is modeled by

$$\mathcal{A}\dot{h_1} = Q_1 - Q_{13}, \mathcal{A}\dot{h_2} = Q_2 + Q_{32} - Q_{20}, \mathcal{A}\dot{h_3} = Q_{13} - Q_{32}$$
$$Q_{13} = a_1 s_{13} \text{sgn}(h_1 - h_3)\sqrt{2g|h_1 - h_3|}$$
$$Q_{32} = a_3 s_{23} \text{sgn}(h_3 - h_2)\sqrt{2g|h_3 - h_2|}, Q_{20} = a_2 s_0 \sqrt{2gh_2}$$

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_2, © Springer-Verlag London 2014

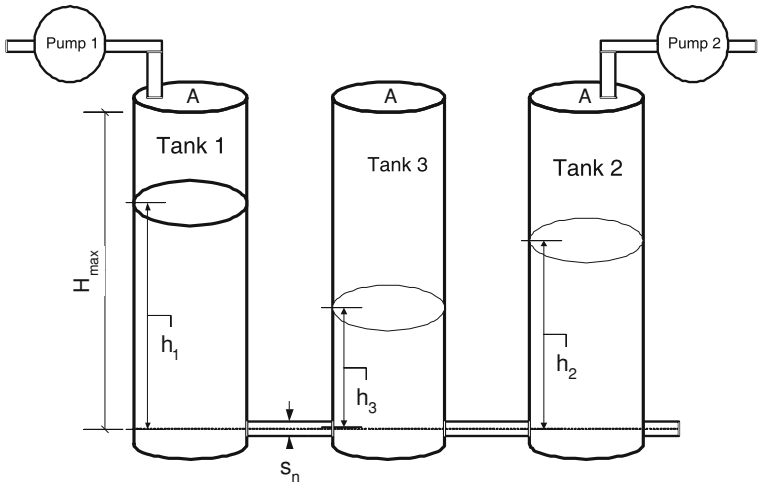


Fig. 2.1 DTS200 setup

where

- Q_1, Q_2 are incoming mass flow (cm^3/s)
- Q_{ij} is the mass flow (cm^3/s) from the ith tank to the jth tank
- $h_i(t), i = 1, 2, 3$, are the water level (cm) in each tank and measurement variables
- $s_{13} = s_{23} = s_0 = s_n$.

The parameters are given in Table 2.1.

In most of our studies, we deal with linear systems. The linear form of the above model can be achieved by a linearization at an operating point as follows:

$$\dot{x} = Ax + Bu,\ y = Cx \tag{2.1}$$

$$x = \begin{bmatrix} h_1 - h_{1,o} \\ h_2 - h_{2,o} \\ h_3 - h_{3,o} \end{bmatrix},\ u = \begin{bmatrix} Q_1 - Q_{1,o} \\ Q_2 - Q_{2,o} \end{bmatrix},\ Q_o = \begin{bmatrix} Q_{1,o} \\ Q_{2,o} \end{bmatrix}$$

$$A = \frac{\partial f}{\partial h}\Big|_{h=h_o},\ B = \begin{bmatrix} \frac{1}{\mathcal{A}} & 0 \\ 0 & \frac{1}{\mathcal{A}} \\ 0 & 0 \end{bmatrix},\ C = \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{bmatrix}$$

where $h_{i,o}, i = 1, 2, 3$, $Q_{1,o}$, $Q_{2,o}$ denote the operating point under consideration and

$$f(h) = \begin{bmatrix} \frac{-a_1 s_{13} \mathrm{sgn}(h_1 - h_3)\sqrt{2g|h_1 - h_3|}}{\mathcal{A}} \\ \frac{a_3 s_{23} \mathrm{sgn}(h_3 - h_2)\sqrt{2g|h_3 - h_2|} - a_2 s_0 \sqrt{2gh_2}}{\mathcal{A}} \\ \frac{a_1 s_{13} \mathrm{sgn}(h_1 - h_3)\sqrt{2g|h_1 - h_3|} - a_3 s_{23} \mathrm{sgn}(h_3 - h_2)\sqrt{2g|h_3 - h_2|}}{\mathcal{A}} \end{bmatrix},\ h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}.$$

Table 2.1 Parameters of DTS200

Parameters	Symbol	Value	Unit
Cross section area of tanks	$\mathcal{A}$	154	cm^2
Cross section area of pipes	s_n	0.5	cm^2
Max. height of tanks	H_{max}	62	cm
Max. flow rate of pump 1	$Q_{1_{max}}$	100	cm^3/s
Max. flow rate of pump 2	$Q_{2_{max}}$	100	cm^3/s
Coeff. of flow for pipe 1	a_1	0.46	
Coeff. of flow for pipe 2	a_2	0.60	
Coeff. of flow for pipe 3	a_3	0.45	

In the steady state at the operating point, it holds

$$A \begin{bmatrix} h_{1,o} \\ h_{2,o} \\ h_{3,o} \end{bmatrix} + B \begin{bmatrix} Q_{1,o} \\ Q_{2,o} \end{bmatrix} = 0 \Longleftrightarrow \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} h_o \\ Q_o \end{bmatrix} = 0. \tag{2.2}$$

It is remarkable that in practice fault diagnosis is often realized as the process runs in the steady state. For this reason, multivariate analysis technique is widely applied to detect faults in the process under consideration.

In our study on the data-driven design of fault diagnosis and fault-tolerant control systems, data will be typically collected at or around an operating point, both in the steady and dynamic operating modes. Serving for the study on adaptive technique, we also consider the case with changes in operating points. It is evident that a linearization at different operating points will result in linear models with different system parameters. System adaptive techniques are powerful tools to deal with such situations.

2.1.2 Description of Typical Faults

Three types of faults are often considered in a benchmark study:

- component faults: leaks in the three tanks, which can be described as additional mass flows out of the tanks,

$$\theta_{A_1}\sqrt{2gh_1}, \theta_{A_2}\sqrt{2gh_2}, \theta_{A_3}\sqrt{2gh_3}$$

  where θ_{A_1}, θ_{A_2} and θ_{A_3} are unknown and depend on the size of the leaks
- component faults: pluggings between two tanks and in the letout pipe by tank 2, which cause changes in Q_{13}, Q_{32} and Q_{20} and thus can be modeled by

$$\theta_{A_4} a_1 s_{13} \mathrm{sgn}(h_1 - h_3)\sqrt{2g|h_1 - h_3|}, \theta_{A_6} a_3 s_{23} \mathrm{sgn}(h_3 - h_2)\sqrt{2g|h_3 - h_2|},$$
$$\theta_{A_5} a_2 s_0 \sqrt{2gh_2}$$

where $\theta_{A_4}, \theta_{A_5}, \theta_{A_6} \in [-1, 0]$ and are unknown

- sensor faults: three additive faults in the three sensors, denoted by f_1, f_2 and f_3
- actuator faults: faults in pumps, denoted by f_4 and f_5.

The influences of these faults can be integrated into the models introduced above. The linear form is given as follows

$$\dot{x} = (A + \Delta A_F)\, x + Bu + E_f f,\ y = Cx + F_f f \tag{2.3}$$
$$\Delta A_F = \sum_{i=1}^{6} A_i \theta_{A_i},\ f = \begin{bmatrix} f_1 \\ \vdots \\ f_5 \end{bmatrix},\ E_f = \begin{bmatrix} 0 & B \end{bmatrix} \in \mathcal{R}^{3\times 5}$$
$$F_f = \begin{bmatrix} I_{3\times 3} & 0 \end{bmatrix} \in \mathcal{R}^{3\times 5}.$$

Those faults modeled by an additive term in the state space representation, e.g. $E_f f$, are called additive faults, while the ones like θ_{A_i} are called multiplicative faults, which may cause changes in the system eigen-dynamics.

2.1.3 Closed-Loop Dynamics

In DTS200, a nonlinear controller is implemented which leads to a full decoupling of the three tank system into

- two linear sub-systems of the first order and
- a nonlinear sub-system of the first order.

This controller can be schematically described as follows:

$$u_1 = Q_1 = Q_{13} + A\,(a_{11} h_1 + v_1\,(w_1 - h_1)) \tag{2.4}$$
$$u_2 = Q_2 = Q_{20} - Q_{32} + A\,(a_{22} h_2 + v_2\,(w_2 - h_2)) \tag{2.5}$$

where $a_{11}, a_{22} < 0$, v_1, v_2 represent two prefilters and w_1, w_2 are reference signals. The nominal (fault-free) closed-loop dynamics is described by

$$\begin{bmatrix} \dot{h}_1 \\ \dot{h}_2 \\ \dot{h}_3 \end{bmatrix} = \begin{bmatrix} (a_{11} - v_1)\, h_1 \\ (a_{22} - v_2)\, h_2 \\ \frac{a_1 s_{13} \mathrm{sgn}(h_1 - h_3)\sqrt{2g|h_1 - h_3|} - a_3 s_{23} \mathrm{sgn}(h_3 - h_2)\sqrt{2g|h_3 - h_2|}}{A} \end{bmatrix} + \begin{bmatrix} v_1 & 0 \\ 0 & v_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}. \tag{2.6}$$

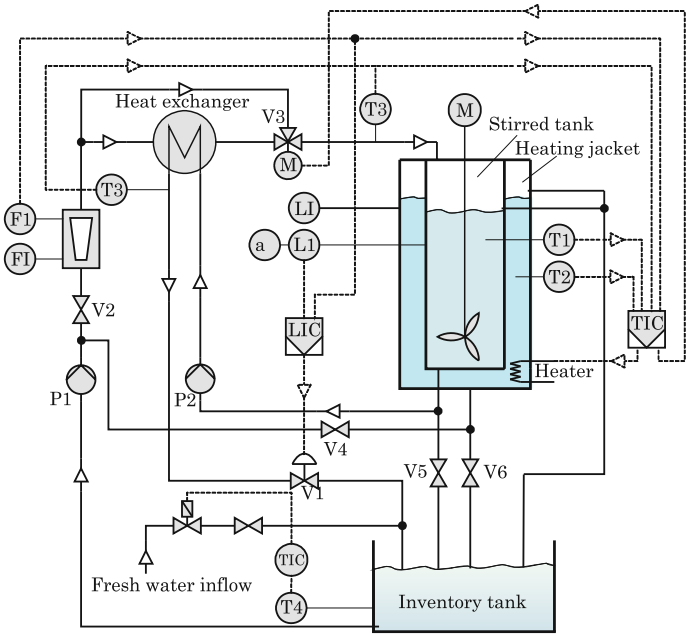


Fig. 2.2 Laboratory setup CSTH

In the steady state, we have

$$\begin{bmatrix} (a_{11}-v_1)\,h_1 \\ (a_{22}-v_2)\,h_2 \\ \frac{a_1 s_{13}\mathrm{sgn}(h_1-h_3)\sqrt{2g|h_1-h_3|}-a_3 s_{23}\mathrm{sgn}(h_3-h_2)\sqrt{2g|h_3-h_2|}}{A} \end{bmatrix} + \begin{bmatrix} v_1 & 0 \\ 0 & v_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = 0. \tag{2.7}$$

It should be mentioned that detecting a fault in the closed-loop configuration is a challenging topic, in particular in the data-driven framework.

2.2 Continuous Stirred Tank Heater

In this section, we briefly introduce a linear model of the laboratory setup continuous stirred tank heater (CSTH), which is a typical control system often met in process industry.

2.2.1 Plant Dynamics and Its Description

Figure 2.2 gives a schematic description of the laboratory setup CSTH, where water is used as the product and reactant. Without considering the dynamic behaviors of

Table 2.2 Technical data of CSTH

Symbol	Description	Unit
V_T	Water volume in the tank	L
H_T	Enthalpy in the tank	J
T_{hj}	Temperature in the heating jacket	°C
$\dot{V}_{in}, \dot{V}_{out}$	Water flows in and out of the tank	l/s
$\dot{H}_{hjT}$	Enthalpy flow from the jacket to the tank	J/s
$\dot{H}_{in}, \dot{H}_{out}$	Enthalpy flows from in- and out-flowing water	J/s
m_{hj}	Water mass in the heating jacket	kg
P_h	Electrical heater power	W = J/s
h_T	Water level in the tank	m
T_T	Water temperature in the tank	°C
m_T	Water mass in the tank	kg
$\dot{m}_{in}, \dot{m}_{out}$	Mass flows in and out of the tank	kg/s
T_{in}, T_{out}	Temperature of the in- and out-flowing water	°C
A_{eff}	The base area of the tank	m^2
c_p	Heat capacity of water	J/kg °C

the heat exchanger, the system dynamics can be represented by the tank volume V_T, the enthalpy in the tank H_T and the water temperature in the heating jacket T_{hj} and modeled by

$$\begin{bmatrix} \dot{V}_T \\ \dot{H}_T \\ \dot{T}_{hj} \end{bmatrix} = \begin{bmatrix} \dot{V}_{in} - \dot{V}_{out} \\ \dot{H}_{hjT} + \dot{H}_{in} - \dot{H}_{out} \\ \frac{1}{m_{hj} \cdot c_p} \left(P_h - \dot{H}_{hjT} \right) \end{bmatrix}. \tag{2.8}$$

The physical meanings of the process variables and parameters used above and in the sequel are listed in Table 2.2. Let

$$\dot{V}_{in} - \dot{V}_{out} := u_1, \, P_h := u_2$$

be the input variables. Considering that

$$\dot{H}_{hjT} = f\left(T_{hj} - T_T\right), \dot{H}_{in} = \dot{m}_{in} c_p T_{in}, \dot{H}_{out} = \dot{m}_{out} c_p T_{out} = H_T \frac{\dot{V}_{out}}{V_T}$$

we have

$$\begin{bmatrix} \dot{V}_T \\ \dot{H}_T \\ \dot{T}_{hj} \end{bmatrix} = \begin{bmatrix} 0 \\ f\left(T_{hj} - \frac{H_T}{m_T \cdot c_p}\right) + \dot{m}_{in} c_p T_{in} - H_T \frac{\dot{V}_{out}}{V_T} \\ \frac{-f\left(T_{hj} - \frac{H_T}{m_T \cdot c_p}\right)}{m_{hj} \cdot c_p} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_{hj} \cdot c_p} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.9}$$

with f denoting some nonlinear function. In CSTH, the water level h_T, the temperature of the water in the tank T_T as well as T_{hj} are measurement variables which satisfy

$$\begin{bmatrix} h_T \\ T_T \\ T_{hj} \end{bmatrix} = \begin{bmatrix} \frac{V_T}{A_{eff}} \\ \frac{H_T}{m_T \cdot c_p} \\ T_{hj} \end{bmatrix}.$$

2.2.2 Faults Under Consideration

Different kinds of faults can be considered in the benchmark study, for instance

- leakage in the tank A leakage will cause a change in the first equation in (2.9) as follows

$$\dot{V}_T = \dot{V}_{in} - \dot{V}_{out} - \theta_{leak}\sqrt{2gh_T} = u_1 - \theta_A\sqrt{\frac{2gV_T}{A_{eff}}} \tag{2.10}$$

  where θ_A is a coefficient proportional to the size of the leakage. It is evident that θ_A is a multiplicative component fault.
- an additive actuator fault in u_1
- additive faults in the temperate enmeshments.

2.3 An Industrial Benchmark: Tennessee Eastman Process

Tennessee Eastman process (TEP) is a realistic simulation of a chemical process that was created by Eastman chemical company in open loop operation. Since its publication by Downs and Fogel in 1993, TEP is widely accepted as a benchmark for control and monitoring studies and used by the control and fault diagnosis communities as a source of data for comparison studies of various control, monitoring and fault diagnosis schemes and methods. In this book, TEP mainly serves for illustrating and demonstrating the data-driven process monitoring and fault diagnosis schemes.

2.3.1 Process Description and Simulation

Figure 2.3 shows the flow diagram of the process with five major units, namely, reactor, condenser, compressor, separator and stripper. The process has two products from four reactants. Additionally, an inert and a by-product are also produced making

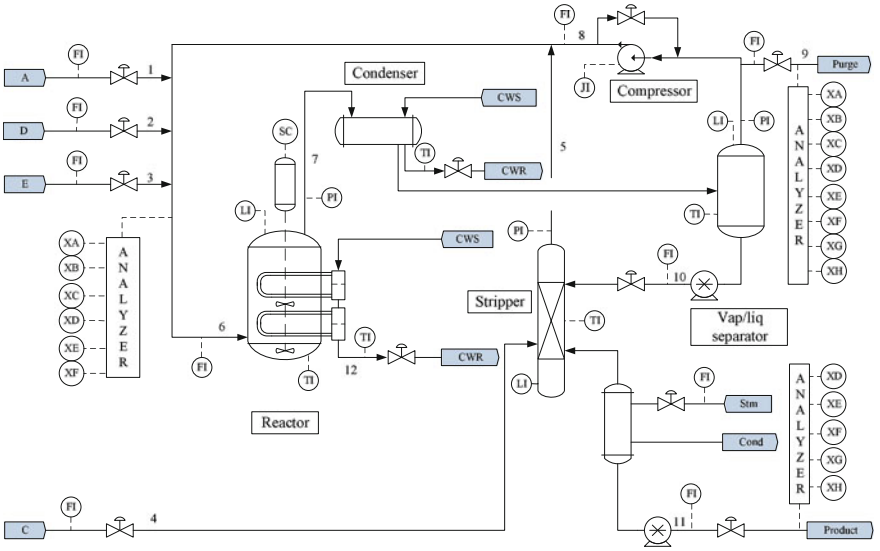


Fig. 2.3 The Tennessee Eastman process

Table 2.3 Process manipulated variables

Variable name	Number	Base value (%)	Units
D feed flow	XMV(1)	63.053	kgh^{-1}
E feed flow	XMV(2)	53.980	kgh^{-1}
A feed flow	XMV(3)	24.644	kscmh
A and C feed flow	XMV(4)	61.302	kscmh
Compressor recycle valve	XMV(5)	22.210	%
Purge valve	XMV(6)	40.064	%
Separator pot liquid flow	XMV(7)	38.100	m^3h^{-1}
Stripper liquid product flow	XMV(8)	46.534	m^3h^{-1}
Stripper steam valve	XMV(9)	47.446	%
Reactor cooling water flow	XMV(10)	41.106	m^3h^{-1}
Condenser cooling water flow	XMV(11)	18.114	m^3h^{-1}

a total of 8 components denoted as A, B, C, D, E, F, G and H. TEP allows total 52 measurements, out of which 11 are manipulated variables and 41 are process variables, as listed respectively in Tables 2.3 and 2.4.

The first TEP simulation was programmed in the FORTRAN code. In the past two decades, different program versions have been developed. For our study in this book, the Simulink code provided by Ricker is used, which is available at the website

http://depts.washington.edu/control/LARRY/TE/download.html

and can be downloaded. This Simulink simulator allows an easy setting and generation of the operation modes, measurement noises, sampling time and magnitudes of

Table 2.4 Process variables

Block name	Variable name	Number
Input feed	A feed (stream 1)	XMEAS(1)
	D feed (stream 2)	XMEAS(2)
	E feed (stream 3)	XMEAS(3)
	A and C feed	XMEAS(4)
Reactor	Reactor feed rate	XMEAS(6)
	Reactor pressure	XMEAS(7)
	Reactor level	XMEAS(8)
	Reactor temperature	XMEAS(9)
Separator	Separator temperature	XMEAS(11)
	Separator level	XMEAS(12)
	Separator pressure	XMEAS(13)
	Separator underflow	XMEAS(14)
Stripper	Stripper level	XMEAS(15)
	Stripper pressure	XMEAS(16)
	Stripper underflow	XMEAS(17)
	Stripper temperature	XMEAS(18)
	Stripper steam flow	XMEAS(19)
Miscellaneous	Recycle flow	XMEAS(5)
	Purge rate	XMEAS(10)
	Compressor work	XMEAS(20)
	Reactor water temperature	XMEAS(21)
	Separator water temperature	XMEAS(22)
Reactor feed analysis	Component A	XMEAS(23)
	Component B	XMEAS(24)
	Component C	XMEAS(25)
	Component D	XMEAS(26)
	Component E	XMEAS(27)
	Component F	XMEAS(28)
Purge gas analysis	Component A	XMEAS(29)
	Component B	XMEAS(30)
	Component C	XMEAS(31)
	Component D	XMEAS(32)
	Component E	XMEAS(33)
	Component F	XMEAS(34)
	Component G	XMEAS(35)
	Component H	XMEAS(36)
Product analysis	Component D	XMEAS(37)
	Component E	XMEAS(38)
	Component F	XMEAS(39)
	Component G	XMEAS(40)
	Component H	XMEAS(41)

Table 2.5 Descriptions of process faults in TE process

Fault number	Process variable	Type
IDV(1)	A/C feed ratio, B composition constant	Step
IDV(2)	B composition, A/C ration constant	Step
IDV(3)	D feed temperature	Step
IDV(4)	Reactor cooling water inlet temperature	Step
IDV(5)	Condenser cooling water inlet temperature	Step
IDV(6)	A feed loss	Step
IDV(7)	C header pressure loss-reduced availability	Step
IDV(8)	A, B, C feed composition	Random variation
IDV(9)	D feed temperature	Random variation
IDV(10)	C feed temperature	Random variation
IDV(11)	Reactor cooling water inlet temperature	Random variation
IDV(12)	Condenser cooling water inlet temperature	Random variation
IDV(13)	Reaction kinetics	Slow Drift
IDV(14)	Reactor cooling water valve	Sticking
IDV(15)	Condenser cooling water valve	Sticking
IDV(16)	Unknown	Unknown
IDV(17)	Unknown	Unknown
IDV(18)	Unknown	Unknown
IDV(19)	Unknown	Unknown
IDV(20)	Unknown	Unknown
IDV(21)	The valve fixed at steady state position	Constant position

the faults. It is thus very helpful for the data-driven study. It is worth to remark that the data sets once generated by Chiang et al. are widely accepted for process monitoring and fault diagnosis research, in which 22 training sets, including 21 faulty and normal operating conditions, were collected to record the process measurements for 24 operation hours. In order to ensure a comparable investigation, in our study the simulator is parameterized in the base operating mode which is comparable with the case considered by Chiang et al.

Note that there are numerous control schemes and structures applied to the TEP simulation. The simulator provided by Ricker simulates the closed loop behavior of the TEP with a decentralized control strategy, which is different from the one adopted by Chiang et al. for the generation of their simulation data.

2.3.2 Simulated Faults in TEP

In the process monitoring and fault diagnosis study, the 21 faults listed in Table 2.5 are considered. These faults mainly affect process variables, reaction kinetics, feed concentration and actuators such as pump valves. Correspondingly, 22 (on-line) test data sets including 48 h plant operation time have been generated in the work

by Chiang et al., where the faults were introduced after 8 simulation hours. By considering the time constants of the process in closed loop, the sampling time was selected as 3 min.

2.4 Notes and references

In this chapter, we have introduced three case and application examples. The first two systems, three-tank system DTS200 and continuous stirred tank heater, a product of the company G.U.N.T. Geraetebau GmbH, are laboratory test beds. For the technical details, the reader is referred to the practical instructions, [1] for DTS200 and [2] for CSTH. It is worth mentioning that a similar CSTH setup has been introduced in [3] for the purpose of benchmark study.

The TEP simulation has first been published by Downs and Fogel in 1993 [4]. In their book on fault detection and diagnosis in industrial systems [5], Chiang et al. have systematically studied process monitoring and fault diagnosis problems on the TEP simulator. For this purpose, 22 training data sets have been generated, which include 21 faulty and the normal operating conditions. These data sets have been available from the Internet and widely adopted by the control and fault diagnosis communities, in particular as a benchmark process for the comparison studied, for instance the work reported in [6]. Ricker has developed a Simulink TEP simulator and presented a control scheme applied to the TEP in [7]. The simulator can be downloaded at his website.

As mentioned previously, a further motivation for introducing these three application cases is that they will serve as application examples for illustrating and demonstrating the results of our study in the forthcoming chapters.

References

1. AMIRA (1996) Three-tank-system DTS200, Practical Instructions. AMIRA, Duisburg
2. GUNT (2011) Experiment Instructions RT682. G.U.N.T, Gerätebau GmbH, Germany
3. Thornhill N, Patwardhan S, Shah S (2008) A continous stirred tank heater simulation model with applications. J Process Control 18:347–360
4. Downs J, Fogel E (1993) A plant-wide industrial process control problem. Comput Chem Eng 17:245–255
5. Chiang LH, Russell EL, Braatz RD (2001) Fault detection and diagnosis in industrial systems. Springer, London
6. Yin S, Ding SX, Haghani A, Hao H, Zhang P (2012) A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. J Process Control 22:1567–1581
7. Ricker N (1996) Decentralized control of the Tennessee Eastman challenge process. J Process Control 6:205–221

Chapter 3
Basic Statistical Fault Detection Problems

In this chapter, we first discuss an essential fault detection problem in the statistical framework. The objective of this study is to learn some basic issues, ideas and solution tools in dealing with fault detection issues. They are useful for our subsequent study. At the end of this chapter, two variations of the essential fault detection problem are then introduced.

3.1 Some Elementary Concepts

In this section, the problem under consideration is described and, associated with it, some essential concepts are introduced.

3.1.1 A Simple Detection Problem and Its Intuitive Solution

We consider the following fault detection problem: Given a measurement model

$$y = f + \varepsilon \in \mathcal{R}$$

with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, where the variance σ^2 is known and $f = 0$ represents the fault-free case, our task consists in detecting a constant fault $f \neq 0$ by means of a number of available (measurement) samples of $y, y_1, \ldots, y_n$.

The following solution is intuitive and consists of two steps:

- computation of the mean value of the n measurements

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{3.1}$$

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_3, © Springer-Verlag London 2014

- decision making according to the detection logic

$$|\bar{y}| > 0 \Longrightarrow \text{faulty, otherwise fault-free.} \tag{3.2}$$

The idea behind this solution is evident: since the fault is represented by a change in the mean of the measurement y, $\bar{y}$ is an estimate of $\mathcal{E}(y)$ and thus also an estimate for f. As a result, a decision for detecting a fault is made based on a simple comparison between the estimated mean value and the (known) mean value in the fault-free case.

On the other hand, we notice that $\bar{y}$ is a random variable satisfying $\bar{y} \sim \mathcal{N}(f, \sigma^2/n)$. It yields

$$prob\{\, |\bar{y}| > 0 \mid f = 0\} > 0.$$

That means it is possible that the decision logic (3.2) triggers an alarm (for indicating a fault), although no fault has occurred. Such an alarm is called false alarm signal. The probability for a false alarm signal is called false alarm rate (FAR).

Note that in the fault-free case $\bar{y} \sim \mathcal{N}(0, \sigma^2/n)$ and thus

$$prob\{\, |\bar{y}| > 0 \mid f = 0\} = 1 - prob\{\, |\bar{y}| = 0 \mid f = 0\} = 1$$

which means an FAR equal to 1. In other words, the detection logic (3.2) is inapplicable in practice. For this reason, zero should be replaced by a positive constant which is called threshold and denoted by $J_{th} > 0$. It is evident that

$$prob\{\, |\bar{y}| > J_{th} \mid f = 0\} = FAR < 1 \tag{3.3}$$

and the larger the threshold J_{th} is, the lower the FAR becomes. If the FAR is considered as the detection performance, our original fault detection problem can be formulated as: given an acceptable FAR α, find the threshold J_{th} so that

$$prob\{\, |\bar{y}| > J_{th} \mid f = 0\} = \alpha. \tag{3.4}$$

Alternatively, we can, for instance, also set the threshold J_{th} to ensure that a fault with a size larger than some value can be detected with a high probability. In that case, the computation of the $FAR = \text{prob}\{|\bar{y}| > J_{th} \mid f = 0\}$ is the major task.

3.1.2 Elementary Concepts in Fault Detection

As a summary and generalization of the above discussion, we now re-study the addressed fault detection problem in the statistical framework of hypothesis testing, and will, in this context, introduce some elementary concepts.

By hypothesis testing, it is assumed that the distribution of a random variable y is a known function of some parameter, for instance $\mathcal{E}(y)$. The so-called null

hypothesis H_o is then tested against an alternative hypothesis H_1. In this context, the fault detection problem described in Sect. 3.1.1 can be formulated as:

$$H_0, \text{ null hypothesis: } f = 0, \text{ fault-free} \tag{3.5}$$

$$H_1, \text{ alternative hypothesis: } f \neq 0, \text{ faulty.} \tag{3.6}$$

The purpose of applying hypothesis testing is to determine whether the sampling (the online process data) supports the rejection of the null hypothesis. In other words, it is to detect if $f \neq 0$. In order to make a reliable decision for or against the rejection of the null hypothesis, that means a decision for fault-free or faulty, a function of the samples will be defined, which is also a random variable. As demonstrated in our study in Sect. 3.1.1, $|\bar{y}|$ is such a test statistic. In our subsequent study, we will denote a test statistic by J, which is defined/selected to be larger or equal to zero and of the properties that

- J is close to zero in the fault-free case (H_0) and
- J is significantly larger than zero in case of a fault (H_1) .

For our purpose, the so-called threshold, denoted by J_{th}, is introduced, which defines two ranges:

$$\mathcal{R}_{f=0} = [0, J_{th}] \text{ and } \mathcal{R}_{f\neq 0} = (J_{th}, \infty).$$

Associated with it, $J \in \mathcal{R}_{f\neq 0}$ indicates the rejection of H_0 (faulty). As a result, the decision logic (fault detection logic) is defined as

$$\begin{cases} J \leq J_{th}, & H_0 \text{ (fault-free)} \\ J > J_{th}, & H_1 \text{ (faulty)} \end{cases}. \tag{3.7}$$

Recall that in general

$$\text{prob}\{J > J_{th} \mid f = 0\} = \alpha > 0$$

that means the probability that a (wrong) decision is made to reject H_0 (to detect a fault) in case $f = 0$ is larger than zero. In hypothesis testing, the probability α is called significance level. In our study on fault detection, this probability is defined as FAR.

Definition 3.1 The probability

$$prob\{J > J_{th} \mid f = 0\} \tag{3.8}$$

is called false alarm rate.

3.1.3 Problem Formulations

We are now in a position to give a more general formulation of the previous fault detection problem in terms of the concepts introduced in the last subsection.

Fault detection problem I (FD-P1): Given

$$y = f + \varepsilon \in \mathcal{R}^m, \quad m \geq 1 \tag{3.9}$$

with $\varepsilon \sim \mathcal{N}(0, \Sigma)$, *where the covariance matrix* Σ *is known and* $f = 0$ *represents the fault-free case*, *an acceptable FAR (significance level)* α, *and suppose that samples (online measurement data) of* $y, y_1, \ldots, y_n$, *are available*. *Find (i) a test statistic J (ii) a corresponding threshold* J_{th} *such that the detection logic* (3.7) *can be used for the detection purpose and FAR* $\leq \alpha$.

We would like to call the reader's attention that the above formulation is a model-based fault detection problem, since we assume that a statistical model for y is available and the model parameters, $\mathcal{E}(y) = 0, Var(y) = \Sigma$, are known a prior. In real applications, $\mathcal{E}(y), Var(y)$ are often unknown. Instead, a huge number of data. $y_1, \ldots, y_N$, could be recorded and available for an online fault detection. This motivates us to give a data-driven version of the above fault detection problem.

Data-driven version of FD-P1: Given

$$y = f + \varepsilon \in \mathcal{R}^m, \varepsilon \sim \mathcal{N}(\mathcal{E}(\varepsilon), \Sigma) \tag{3.10}$$

where $\mathcal{E}(\varepsilon), \Sigma$ *are unknown, a sample of* $y, y_1, \ldots, y_N$, *and an acceptable FAR (significance level)* α. *Find a fault detection scheme that allows a reliable detection of* $f \neq 0$ *based on online data* $y_k, \ldots, y_{k+n}$ *with FAR* $\leq \alpha$.

3.2 Some Elementary Methods and Algorithms

In this section, we describe some elementary methods for solving FD-P1.

3.2.1 The Intuitive Solution

We now apply the test statistic $|\bar{y}|$ proposed in Sect. 3.1.1 to solving FD-P1 with $m = 1$. To this end, let

$$z = \frac{\sqrt{n}}{\sigma}\bar{y} \sim \mathcal{N}(0, 1)$$

for $f = 0$. Note that

$$prob\{|\bar{y}| > J_{th}\} = 2prob\{\bar{y} > J_{th}\}. \tag{3.11}$$

As a result, the following algorithm can be used for the computation of the threshold.

Algorithm 3.1 *Computing J_{th} for solving FD-P1*

S1: Determine the critical normal deviate $z_{(1-\alpha)/2}$ using the table of standard normal distribution

$$prob\left\{0 \leq z \leq z_{(1-\alpha)/2}\right\} = (1-\alpha)/2 \Longleftrightarrow prob\left\{z > z_{(1-\alpha)/2}\right\} = \alpha/2 \quad (3.12)$$

S2: Set J_{th}

$$J_{th} = z_{(1-\alpha)/2}\frac{\sigma}{\sqrt{n}}. \quad (3.13)$$

3.2.2 T^2 Test Statistic

Consider FD-P1 with model (3.9). T^2 test statistic is defined by

$$J_{T^2} = n\left(\bar{y}^T \Sigma^{-1}\bar{y}\right). \quad (3.14)$$

Since

$$\sqrt{n}\Sigma^{-1/2}\bar{y} = \sqrt{n}\Sigma^{-1/2}\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right) \sim \mathcal{N}(0, I)$$

we have

$$J_{T^2} \sim \chi^2(m) \quad (3.15)$$

where $\chi^2(m)$ is the chi-squared distribution with m degrees of freedom. T^2 test statistic is also called Hotelling's T^2 test statistic, as it was first proposed by Hotelling in 1931. For applying the T^2 test statistic to FD-P1, the threshold computation can be done using the following algorithm.

Algorithm 3.2 *Computing $J_{T^2,th}$*

S1: Determine χ_α using the table of χ^2 -distribution with m degrees of freedom

$$prob\{\chi > \chi_\alpha\} = \alpha \Longleftrightarrow prob\{\chi \leq \chi_\alpha\} = 1-\alpha$$

S2: Set $J_{T^2,th}$

$$J_{th,T^2} = \chi_\alpha. \quad (3.16)$$

3.2.3 Likelihood Ratio and Generalized Likelihood Ratio

Likelihood ratio (LR) methods are very popular in the framework of change detection. Different from the detection algorithms described in the last two subsections, the test statistic applied in the LR methods for the fault detection purpose is often the solution of an optimization problem. In this subsection, we briefly discuss the application of LR methods for the scalar fault detection. The vector-valued case will be addressed in Sect. 3.2.4.

Consider the model (3.9) with $m = 1$. The log likelihood ratio for data y_i is defined by

$$s(y_i) = 2 \ln \frac{p_f(y_i)}{p_0(y_i)} = \frac{1}{\sigma^2}\left[(y_i)^2 - (y_i - f)^2\right] \tag{3.17}$$

where

$$p_0(y_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i)^2}{2\sigma^2}}, \; p_f(y_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - f)^2}{2\sigma^2}}, \; f \neq 0$$

represent the probability density of y for $y = y_i$ in the fault-free case and faulty case, respectively, and the factor 2 in (3.17) is introduced for simple and consistent notations. The basic idea of the LR methods can be clearly seen from the decision rule

$$s(y_i) = \begin{cases} \leq J_{th} (> 0), & H_0 \; (f = 0) \text{ is accepted} \\ > J_{th}, & H_0 \text{ is rejected} \end{cases}.$$

Note that $s(y_i) > J_{th} > 0$ means $p_f(y_i) > p_0(y_i)$, and thus given y_i the probability of $f \neq 0$ is (significantly) higher than the one of $f = 0$. It is therefore reasonable to make a decision in favour of H_1.

In case that n samples of y, $y_i, i = 1, \ldots, n$, are available, the (log) LR is defined by

$$S_1^n = \sum_{i=1}^{n} s_i(y_i) = 2 \sum_{i=1}^{n} \ln \frac{p_f(y_i)}{p_0(y_i)} = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left[(y_i)^2 - (y_i - f)^2\right]. \tag{3.18}$$

Notice that in (3.18) f represents the fault to be detected and is in general unknown. In order to use the above LR as a test statistic, f should be replaced by an estimate. For this purpose, the so-called generalized likelihood ratio (GLR) method was developed, where f is replaced by its *maximum likelihood estimate*. The maximum likelihood estimate of f is an estimate achieved by maximizing the LR. Thus, the maximum LR as well as the maximum likelihood estimate of f are the solution of the following optimization problem

$$\max_f S_1^n = \max_f \frac{1}{\sigma^2} \sum_{i=1}^{n} \left[y_i^2 - (y_i - f)^2 \right]$$

$$= \max_f \frac{1}{\sigma^2} \left[\frac{1}{n} \left(\sum_{i=1}^{n} y_i \right)^2 - n \left(f - \bar{y} \right)^2 \right] \Longrightarrow \tag{3.19}$$

$$\hat{f} = \arg\max_f S_1^n = \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \max_f S_1^n = \frac{1}{\sigma^2 n} \left(\sum_{i=1}^{n} y_i \right)^2$$

$$\Longleftrightarrow \max_f S_1^n = \frac{n}{\sigma^2} \left(\bar{y} \right)^2 := J. \tag{3.20}$$

It is of practical interest to notice that

- the maximum likelihood estimate of f is the estimate of the mean value, $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$
- the maximum LR, $\frac{n}{\sigma^2} \left(\bar{y} \right)^2$, is exactly the T^2 test statistic given in (3.14) and thus
- Algorithm 3.2 can be used for the threshold setting.

It has been theoretically proven that the LR based change detection leads to a minimization of the missed detection rate for a given false alarm rate.

For the on-line implementation of the above-described LR methods, computation of J

$$J = S_1^n = \frac{1}{\sigma^2 n} \left(\sum_{i=1}^{n} y_i \right)^2$$

is needed.

3.2.4 Vector-Valued GLR

In this subsection, the vector-valued GLR test is presented. Consider again model (3.9). The probability density of Gaussian vector y is defined by

$$p_{f,\Sigma}(y) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} e^{-\frac{1}{2}(y-f)^T \Sigma^{-1} (y-f)}, \theta = 0 \text{ or } f \ (\neq 0) \,. \tag{3.21}$$

Hence, the LR for given vector y satisfies

$$s(y) = 2 \ln \frac{p_f(y)}{p_0(y)} = \left[y^T \Sigma^{-1} y - (y - f)^T \Sigma^{-1} (y - f) \right].$$

On the assumption that n (vector-valued) samples of y, $y_k, k = 1, \ldots, n$, are available, the maximum likelihood estimate of f and the maximum LR are given by

$$\max_f S_1^n = \max_f \left[\sum_{k=1}^{n} y_k^T \Sigma^{-1} y_k - \sum_{k=1}^{n} (y_k - f)^T \Sigma^{-1} (y_k - f) \right] \tag{3.22}$$

$$= \max_f \left[\sum_{k=1}^{n} y_k^T \Sigma^{-1} y_k - \sum_{k=1}^{n} y_k^T \Sigma^{-1} y_k - n \left(f^T \Sigma^{-1} f - 2 f^T \Sigma^{-1} \frac{1}{n} \sum_{k=1}^{n} y_k \right) \right]$$

$$= \max_f \left[n \bar{y}^T \Sigma^{-1} \bar{y} - n (\bar{y} - f)^T \Sigma^{-1} (\bar{y} - f) \right], \bar{y} = \frac{1}{n} \sum_{k=1}^{n} y_k$$

$$\Longrightarrow \hat{f} = \arg\max_f S_1^n = \bar{y} \Longrightarrow \max_f S_1^n = n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right). \tag{3.23}$$

Once again, we can see that also in the vector-valued case, the maximum likelihood estimate of f is $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_k$, and $\max_f S_1^n$ is the T^2 test statistic given in (3.14).

From the mathematical and computational point, $n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right)$ means a normalization of $\bar{y}^T \bar{y}$, that is

$$\sqrt{n} \Sigma^{-1/2} \bar{y} \sim \mathcal{N}(0, I) \Longrightarrow n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right) \sim \chi^2(m)$$

which allows then an easy determination of the corresponding threshold. From the fault detection viewpoint, this normalization can improve the fault detectability. In fact, for $m = 1$,

$$n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right) = \frac{n}{\sigma^2} \bar{y}^T \bar{y}.$$

There exits no difference whether $\bar{y}^T \bar{y}$ or $\frac{n}{\sigma^2} \bar{y}^T \bar{y}$ is applied as test statistic. The situation changes for $m > 1$. In that case, when $\bar{y}^T \bar{y}$ is, instead of its normalization $n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right)$, defined as the test statistic, it holds

$$\bar{y}^T \bar{y} = \frac{\left(\sqrt{n} \Sigma^{-1/2} \bar{y} \right)^T \Sigma \left(\sqrt{n} \Sigma^{-1/2} \bar{y} \right)}{n} \leq \frac{\sigma_{\max}(\Sigma)}{n} \left(n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right) \right)$$

with $\sigma_{\max}(\Sigma)$ denoting the maximal singular value of Σ. Let α be the given significance level and $J_{T^2,th}$ is determined using Algorithm 3.2. As a result, the corresponding thresholds for $n \left(\bar{y}^T \Sigma^{-1} \bar{y} \right)$ and $\bar{y}^T \bar{y}$ are respectively $J_{T^2,th}$ and $\frac{\sigma_{\max}(\Sigma)}{n} J_{T^2,th}$. Suppose that a fault f causes

$$\bar{y}^T \bar{y} - \frac{\sigma_{\max}(\Sigma)}{n} J_{th,T^2} > 0$$

that means it can be detected using the detection logic

$$\bar{y}^T \bar{y} > J_{\bar{y},th} = \frac{\sigma_{\max}(\Sigma)}{n} J_{th,T^2}.$$

Since

$$\bar{y}^T \bar{y} > \frac{\sigma_{\max}(\Sigma)}{n} J_{T^2,th} \Longleftrightarrow \frac{n}{\sigma_{\max}(\Sigma)} \bar{y}^T \bar{y} > J_{th,T^2}$$

and further

$$\frac{1}{\sigma_{\max}(\Sigma)} I = \sigma_{\min}\left(\Sigma^{-1}\right) I \leq \Sigma^{-1}$$

it yields

$$n\left(\bar{y}^T \Sigma^{-1} \bar{y}\right) \geq \frac{n}{\sigma_{\max}(\Sigma)} \bar{y}^T \bar{y} > J_{th,T^2}. \tag{3.24}$$

In other words, in this case this fault can also be detected using the test statistic $n\left(\bar{y}^T \Sigma^{-1} \bar{y}\right)$. On the other hand, the fact that

$$f^T \Sigma^{-1} f \geq \sigma_{\min}\left(\Sigma^{-1}\right) f^T f = \frac{f^T f}{\sigma_{\max}(\Sigma)} \tag{3.25}$$

tells us that there may exist faults which could be detected using the test statistic $n\left(\bar{y}^T \Sigma^{-1} \bar{y}\right)$ but not using $\bar{y}^T \bar{y}$.

3.3 The Data-Driven Solutions of the Detection Problem

In this section, we deal with the data-driven version of FD-P1 formulated in Sect. 3.1.3. Remember that the FD-P1 and its data-driven version have the (statistical) model (3.9) in common, which builds the basis for the application of the fault detection schemes presented in Sect. 3.2. On the other hand, in the context of data-driven fault detection formulation, the model parameters are unknown a prior. Instead, a huge number of historic data, $y_1, \ldots, y_N, N >> 1$, is available. This motivates us to identify the model parameters using the available data or to integrate the model parameter identification into the fault detection procedure. In this context, the fault detection problem is solved in a two-step procedure:

- identification of the model parameters using the recorded data. This is also called training phase and will run offline
- application of an existing (model-based) fault detection scheme.

3.3.1 Fault Detection with a Sufficiently Large N

It is evident that for the application of the fault detection methods introduced in Sect. 3.1.3, the mean vector and covariance matrix of y, $\mathcal{E}(y)$, $Var(y)$, are needed. It is well-known that

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i \text{ and } S_{N-1} = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y}_N)(y_i - \bar{y}_N)^T \tag{3.26}$$

are sample mean vector and covariance matrix of y and for $N \to \infty$

$$\lim_{N\to\infty} \bar{y}_N = \mathcal{E}(y) = \mu \text{ and } \lim_{N\to\infty} S_{N-1} = Var(y) = \Sigma. \tag{3.27}$$

On the assumption that N is sufficiently large so that

$$\bar{y}_N \approx \mu, S_{N-1} \approx \Sigma$$

the following procedure can be used for dealing with the fault detection problem:

- Offline training: computation of

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i, S_{N-1} = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y}_N)(y_i - \bar{y}_N)^T$$

- Offline design: setting threshold, for instance, using Algorithm 3.2 for J_{th,T^2}
- Online fault detection using the algorithm given below.

Algorithm 3.3 *Online fault detection algorithm*

S1: Collect (online) measurement data $y_{k+i}, i = 1, \ldots, n$, *and calculate*

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_{k+i}, \Delta\bar{y} = \bar{y} - \bar{y}_N$$

S2: (Online) calculate test statistic

$$J = n\left(\Delta\bar{y}^T S_{N-1}^{-1}\Delta\bar{y}\right) \tag{3.28}$$

S3: Make a decision

$$\begin{cases} J \le J_{th,T^2} \Longrightarrow \textit{fault-free} \\ J > J_{th,T^2} \Longrightarrow \textit{faulty and alarm} \end{cases}.$$

3.3.2 Fault Detection Using Hotelling's T^2 Test Statistic

Given the model (3.9) with unknown $\mathcal{E}(y)$, $Var(y) = \Sigma$. Assume that there are two groups of process data available: training data (recorded for the offline computation) $y_i, i = 1, \ldots, N$, online measurement data $y_{k+i}, i = 1, \ldots, n$, and $N >> n$. Recall that the training data should be recorded in the fault-free case, that means the data (samples) have been generated on the model assumption

$$y = \varepsilon \in \mathcal{R}^m, \mathcal{E}(y) = \mathcal{E}(\varepsilon) := \mu. \tag{3.29}$$

Differently, the online measurement data may cover both the fault-free or faulty cases. Hence, the model assumption is

$$y = f + \varepsilon \in \mathcal{R}^m, \mathcal{E}(y) = \mu + f = \mu_f, \quad f = \begin{cases} 0, & \text{fault-free} \\ \text{constant} \neq 0, & \text{faulty} \end{cases}. \tag{3.30}$$

In the context of a hypothesis test, we are now able to reformulate our original problem as

$$\begin{cases} H_0 : \mu - \mu_f = 0 \Longrightarrow & \text{fault-free} \\ H_1 : \mu - \mu_f \neq 0 \Longrightarrow & \text{faulty} \end{cases}. \tag{3.31}$$

For our purpose, the (possible) difference between the means of the two data sets should be checked. To this end, consider

$$\bar{y}_N - \bar{y} \sim \mathcal{N}\left(f, \frac{N+n}{nN}\Sigma\right), \bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i, \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_{k+i}. \tag{3.32}$$

Since Σ is unknown, it will be estimated by offline and online data sets. Let

$$S_{off} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \bar{y}_N)(y_i - \bar{y}_N)^T, S_{on} = \frac{1}{n}\sum_{i=1}^{n} (y_{k+i} - \bar{y})(y_{k+i} - \bar{y})^T$$

be sample covariance matrices from the two data sets and

$$\begin{aligned} S &= \frac{NS_{off} + nS_{on}}{n+N} \\ &= \frac{1}{n+N}\left(\sum_{i=1}^{N} (y_i - \bar{y}_N)(y_i - \bar{y}_N)^T + \sum_{i=1}^{n} (y_{k+i} - \bar{y})(y_{k+i} - \bar{y})^T\right). \end{aligned} \tag{3.33}$$

The following theorem plays a central role for the further study.

Theorem 3.1 *Let* $f, \bar{y}_N, \bar{y}, S$ *be defined in* (3.30), (3.32) *and* (3.33). *It holds*

$$\frac{nN(n+N-2)}{(n+N)^2}\,(\bar{y}_N-\bar{y})^T\,S^{-1}\,(\bar{y}_N-\bar{y}) \sim \mathcal{T}^2(m, n+N-2) \tag{3.34}$$

where $\mathcal{T}^2(m, n+N-2)$ *is Hotelling* $\mathcal{T}^2$*-distribution with* m *and* $(n+N-2)$ *degrees of freedom.*

The above theorem is a standard result on the Hotelling $\mathcal{T}^2$-distribution and its proof can be found in books on multivariate analysis. The reader is referred to the references given at the end of this chapter.

Consider further

$$\mathcal{T}^2(m, n+N-2) = \frac{m(n+N-2)}{n+N-m-1}\mathcal{F}(m, n+N-m-1)$$

where $\mathcal{F}(m, n+N-2)$ denotes $\mathcal{F}$-distribution with m and $(n+N-1)$ degrees of freedom. As a result, for the test statistic defined by

$$(\bar{y}_N-\bar{y})^T\,S^{-1}\,(\bar{y}_N-\bar{y}) \tag{3.35}$$

the corresponding threshold is set to be

$$\frac{m\,(n+N)^2}{nN\,(n+N-m-1)}\mathcal{F}_\alpha(m, n+N-m-1) \tag{3.36}$$

for a given significance level α.

Note that S consists of two terms: $\frac{NS_{off}}{n+N}, \frac{nS_{on}}{n+N}$. While $\frac{NS_{off}}{n+N}$ is available offline after the training, $\frac{nS_{on}}{n+N}$ can only be calculated during the process operation. Consequently, S^{-1} needs to be computed online, which, due to the possible high dimension of matrix S, may cause computational problem. Recall, on the other hand, that in real applications the test statistic will be checked after receiving each new measurement, that is $n = 1$. It leads to

$$S = \frac{NS_{off}}{N+1} = \frac{N-1}{N+1}\frac{1}{N-1}\sum_{i=1}^{N}(y_i-\bar{y}_N)\,(y_i-\bar{y}_N)^T := \frac{N-1}{N+1}\hat{\Sigma}$$

where $\hat{\Sigma} = \frac{1}{N-1}\sum\limits_{i=1}^{N}(y_i-\bar{y}_N)\,(y_i-\bar{y}_N)^T$ is a unbiased estimate of Σ. Finally, we define the test statistic

$$J = (\bar{y}_N - y_k)^T \hat{\Sigma}^{-1} (\bar{y}_N - y_k) = \frac{N-1}{N+1} (\bar{y}_N - y_k)^T S^{-1} (\bar{y}_N - y_k) \quad (3.37)$$

and the corresponding threshold

$$J_{T^2,th} = \frac{m\left(N^2-1\right)}{N(N-m)} \mathcal{F}_\alpha(m, N-m).$$

The following two algorithms are the summary of the above discussion.

Algorithm 3.4 *(Data-driven) offline computation of* J_{th,T^2}

S1: Collect process data, $y_1, \ldots, y_N$, *and calculate*

$$\bar{y}_N = \frac{1}{N} \sum_{i=1}^{N} y_i, \hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \bar{y}_N)(y_i - \bar{y}_N)^T, \hat{\Sigma}^{-1}$$

S2: Set J_{th,T^2}

$$J_{th,T^2} = \frac{m\left(N^2-1\right)}{N(N-m)} \mathcal{F}_\alpha(m, N-m).$$

Algorithm 3.5 *Online fault detection algorithm*

S1: Collect (online) measurement data y_k
S2: (Online) calculate test statistic

$$J = (\bar{y}_N - y_k)^T \hat{\Sigma}^{-1} (\bar{y}_N - y_k) \quad (3.38)$$

S3: Make a decision

$$\begin{cases} J \le J_{th,T^2} \Longrightarrow & \textit{fault-free} \\ J > J_{th,T^2} \Longrightarrow & \textit{faulty and alarm} \end{cases}.$$

3.3.3 Fault Detection Using Q Statistic

In the T^2 statistic, computation of the inverse matrix of $\hat{\Sigma}$ is necessary. By a high dimensional and often ill-conditional $\hat{\Sigma}$, such a computation may cause numerical trouble in practice. As an alternative statistic,

$$Q = y^T y \quad (3.39)$$

is widely accepted in practice and applied in the multivariate analysis technique. On the assumption of $y \sim \mathcal{N}(0, \Sigma_y)$, Box proved that the distribution of Q can be approximated by

$$Q \sim g\chi^2(h) \tag{3.40}$$

where $\chi^2(h)$ denotes the χ^2 distribution with h degrees of freedom, and

$$g = \frac{S}{2\mathcal{E}(Q)}, h = \frac{2\mathcal{E}^2(Q)}{S}, S = \mathcal{E}\left(\left(Q - \mathcal{E}(Q)\right)^2\right) = \mathcal{E}\left(Q^2\right) - \mathcal{E}^2(Q). \tag{3.41}$$

For our purpose of fault detection, $\mathcal{E}(Q)$, S can be estimated using the training data, and the threshold setting for the statistic (3.39), the so-called Q statistic, is summarized in the following algorithm.

Algorithm 3.6 *(Data-driven) offline computation of* $J_{th,Q}$

S1: Collect process data, $y_1, \ldots, y_N$, *and calculate*

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i, \bar{Q} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \bar{y}_N)^T (y_i - \bar{y}_N)$$

$$\overline{Q^2} = \frac{1}{N}\sum_{i=1}^{N} \left((y_i - \bar{y}_N)^T (y_i - \bar{y}_N)\right)^2, g = \frac{\overline{Q^2} - \bar{Q}^2}{2\bar{Q}}, h = \frac{2\bar{Q}^2}{\overline{Q^2} - \bar{Q}^2}$$

S2: Set $J_{th,Q}$ *for a given significance level* α

$$J_{th,Q} = g\chi^2_\alpha(h).$$

3.4 Case Example: Fault Detection in Three-Tank System

The application of Algorithms 3.3–3.6 to fault detection is illustrated on three-tank system simulation.

3.4.1 System Setup and Simulation Parameters

For our purpose, a Simulink simulation model of the laboratory system DTS200 introduced in Sect. 2.1 is applied. Considering that the fault detection schemes introduced in this chapter are dedicated to fault detection in static (steady) processes, in

our simulation study, the three-tank system is operating in the steady state around a fixed operating point. The measurement data, the liquid levels h_1, h_2, h_3 and control signal u_1, u_2, are used for the fault detection purpose. For the threshold setting, the FAR is chosen as $\alpha = 5\ \%$. In all simulations, the fault occurrence time is set to be $t = 1200\,\text{s}$.

3.4.2 Training Results and Threshold Setting

For the application of Algorithms 3.3 and 3.5, three data sets are generated independently. The simulated sampling time is 2 s.

Training results using liquid level measurements and control inputs

- training results for the application of Algorithm 3.3 with $N = 1000$

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} u_{1,i} \\ u_{2,i} \\ h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 39.0199 \\ 12.4431 \\ 45.0002 \\ 14.9999 \\ 30.3287 \end{bmatrix}$$

$$S_{N-1} = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y}_N)(y_i - \bar{y}_N)^T$$

$$= \begin{bmatrix} 0.1170 & 0.0014 & -0.0178 & 0.0004 & -0.0026 \\ 0.0014 & 0.2168 & 0.0001 & -0.0456 & -0.0017 \\ -0.0178 & 0.0001 & 0.0028 & 0 & 0 \\ 0.0004 & -0.0456 & 0 & 0.0097 & -0.0001 \\ -0.0026 & -0.0017 & 0 & -0.0001 & 0.0019 \end{bmatrix}$$

  which yields a threshold setting $J_{th,T^2} = 11.0705$.
- training results for the application of Algorithm 3.5 with $N = 300$

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} u_{1,i} \\ u_{2,i} \\ h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 39.0382 \\ 12.4372 \\ 44.9974 \\ 15.0023 \\ 30.3284 \end{bmatrix}$$

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y}_N)(y_i - \bar{y}_N)^T$$

$$= \begin{bmatrix} 0.1192 & 0.0094 & -0.0181 & -0.0012 & -0.0029 \\ 0.0094 & 0.2253 & -0.0013 & -0.0476 & -0.0010 \\ -0.0181 & -0.0013 & 0.0028 & 0.0002 & 0.0001 \\ -0.0012 & -0.0476 & 0.0002 & 0.0102 & -0.0003 \\ -0.0029 & -0.0010 & 0.0001 & -0.0003 & 0.0018 \end{bmatrix}$$

which, by means of Algorithm 3.4, gives a threshold setting $J_{th,T^2} = 11.4131$.

- training results for the application of Algorithm 3.6 with $N = 1000$

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} u_{1,i} \\ u_{2,i} \\ h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 39.0199 \\ 12.4431 \\ 45.0002 \\ 14.9999 \\ 30.3287 \end{bmatrix}$$

$$\bar{Q} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \bar{y}_N)^T (y_i - \bar{y}_N) = 0.3478$$

$$\overline{Q^2} = \frac{1}{N}\sum_{i=1}^{N} \left((y_i - \bar{y}_N)^T (y_i - \bar{y}_N) \right)^2 = 0.2583$$

$$g = 0.1975, h = 1.7609$$

which yields a threshold setting $J_{th,Q} = 1.0901$.

Training results using only liquid level measurements

- training results for the application of Algorithm 3.3 with $N = 1000$

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 45.0002 \\ 14.9999 \\ 30.3287 \end{bmatrix}$$

$$S_{N-1} = \frac{1}{N-1}\sum_{i=1}^{N} (y_i - \bar{y}_N)(y_i - \bar{y}_N)^T = \begin{bmatrix} 0.0028 & 0 & 0 \\ 0 & 0.0097 & -0.0001 \\ 0 & -0.0001 & 0.0019 \end{bmatrix}$$

which yields a threshold setting $J_{th,T^2} = 7.8147$.

- training results for the application of Algorithm 3.5 with $N = 300$

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 44.9974 \\ 15.0023 \\ 30.3284 \end{bmatrix}$$

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N} (y_i - \bar{y}_N)(y_i - \bar{y}_N)^T = \begin{bmatrix} 0.0028 & 0.0002 & 0.0001 \\ 0.0002 & 0.0102 & -0.0003 \\ 0.0001 & -0.0003 & 0.0018 \end{bmatrix}$$

which, by means of Algorithm 3.4, gives a threshold setting $J_{th,T^2} = 7.9848$.

- training results for the application of Algorithm 3.6 with $N = 1000$

$$\bar{y}_N = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 45.0002 \\ 14.9999 \\ 30.3287 \end{bmatrix}$$

$$\bar{Q} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \bar{y}_N)^T (y_i - \bar{y}_N) = 0.0144$$

$$\overline{Q^2} = \frac{1}{N}\sum_{i=1}^{N} \left((y_i - \bar{y}_N)^T (y_i - \bar{y}_N)\right)^2 = 4.2313 \times 10^{-4}$$

$$g = 0.0075, h = 1.9057$$

which yields a threshold setting $J_{th,Q} = 0.0438$.

3.4.3 Fault Detection Results

Based on the above training results, a leak fault and a sensor fault are chosen as examples to demonstrate the detection results using Algorithms 3.3–3.6, respectively. For the application of Algorithm 3.3, the parameter n for online calculation is chosen as $n = 3$.

Fault detection results using liquid level measurements and control inputs

Assume a leak fault occurs at $t = 1200\,\text{s}$ in Tank 1 with $\theta_{A_1} = 5\,\%$. Figure 3.1 gives a result of the process measurements. Using the offline training results given in Sect. 3.4.2, Figs. 3.2, 3.3 and 3.4 demonstrate the corresponding threshold and online test statistic, respectively.

Fault detection results using only liquid level measurements

Assume a sensor fault with an offset of 5 cm occurs at $t = 1200\,\text{s}$ in Tank 3. Figure 3.5 gives the result of the liquid levels. Using the offline training results given in Sect. 3.4.2, Figs. 3.6, 3.7 and 3.8 demonstrate the corresponding threshold and on-line test statistic, respectively.

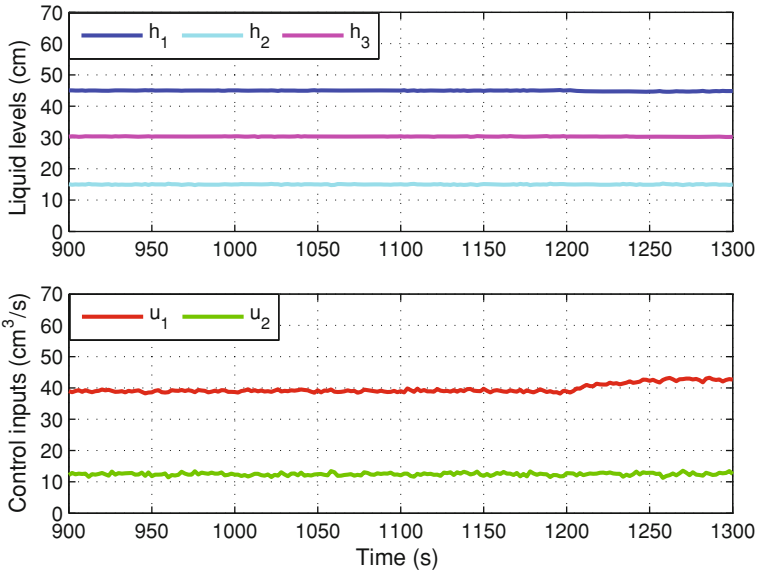


Fig. 3.1 Process measurements

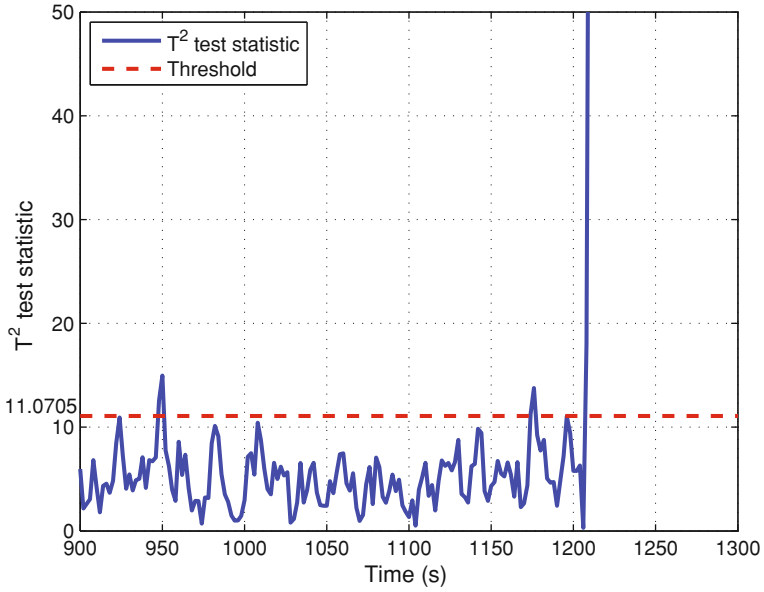


Fig. 3.2 T^2 test statistic and the threshold (Algorithm 3.3)

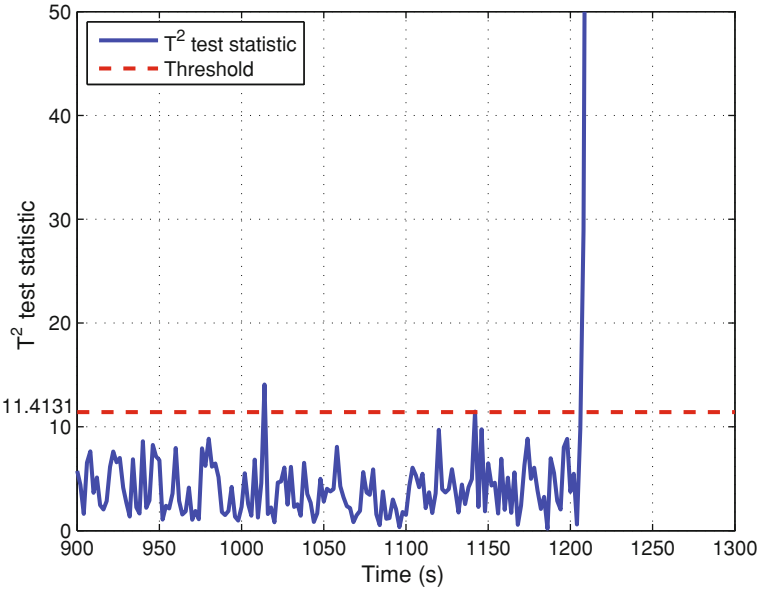


Fig. 3.3 T^2 test statistic and the threshold (Algorithm 3.5)

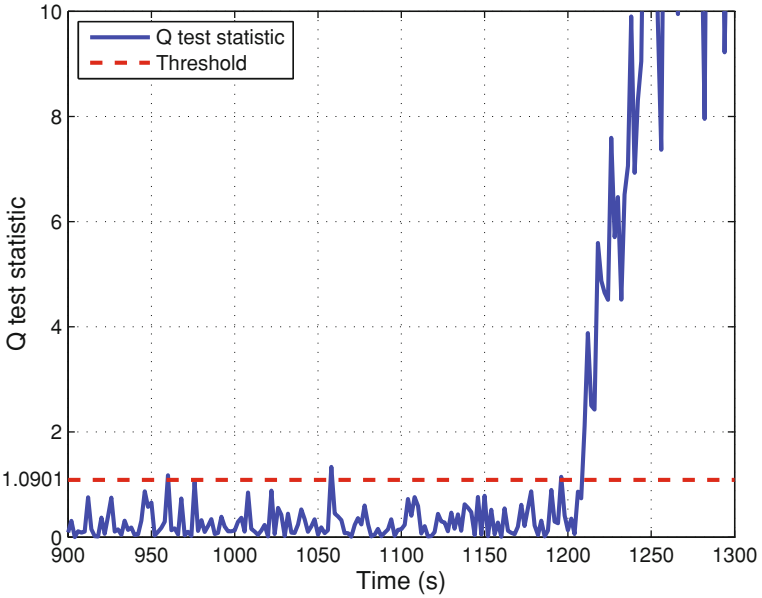


Fig. 3.4 Q test statistic and the threshold (Algorithm 3.6)

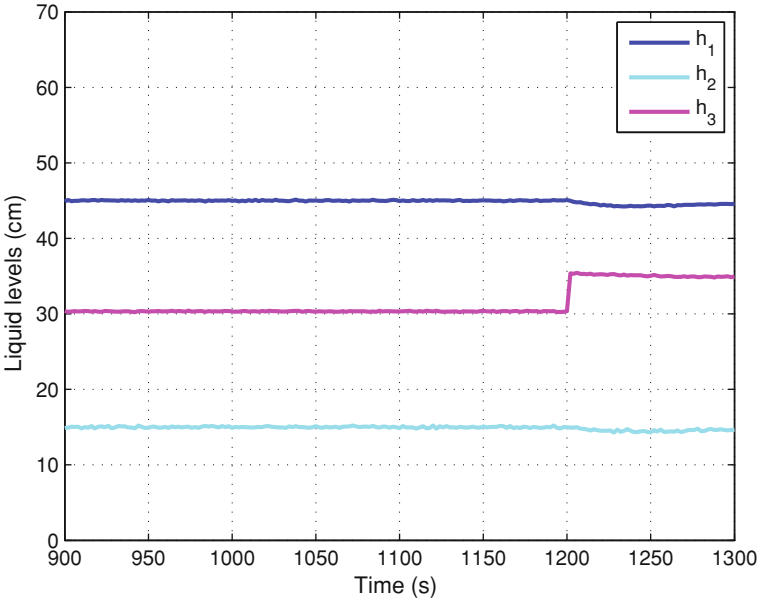


Fig. 3.5 Liquid levels

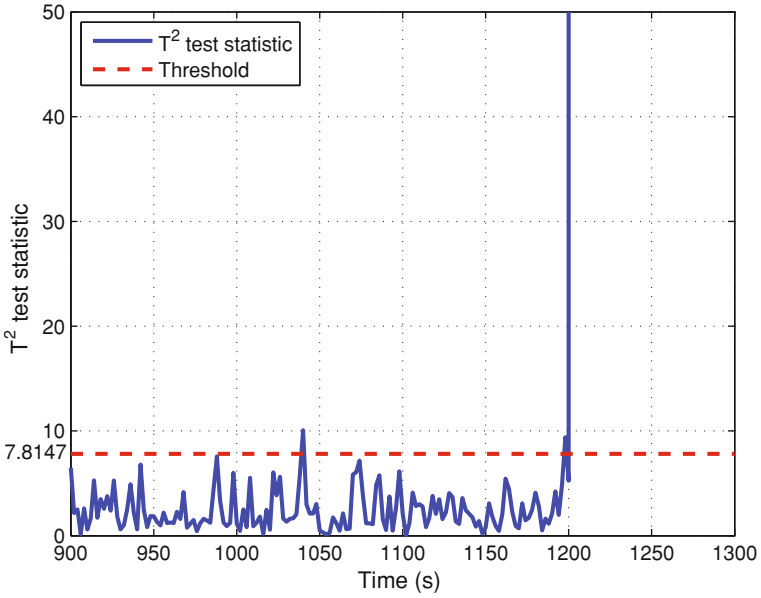


Fig. 3.6 T^2 test statistic and the threshold (Algorithm 3.3)

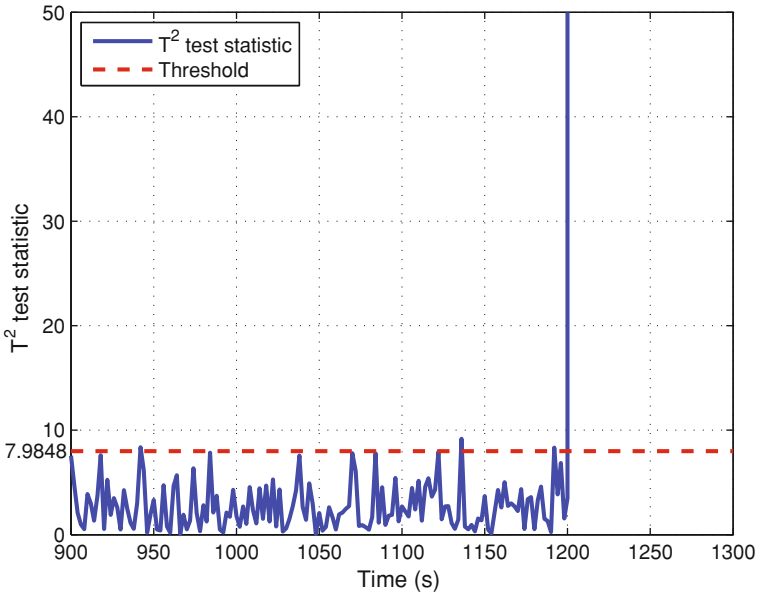


Fig. 3.7 T^2 test statistic and the threshold (Algorithm 3.5)

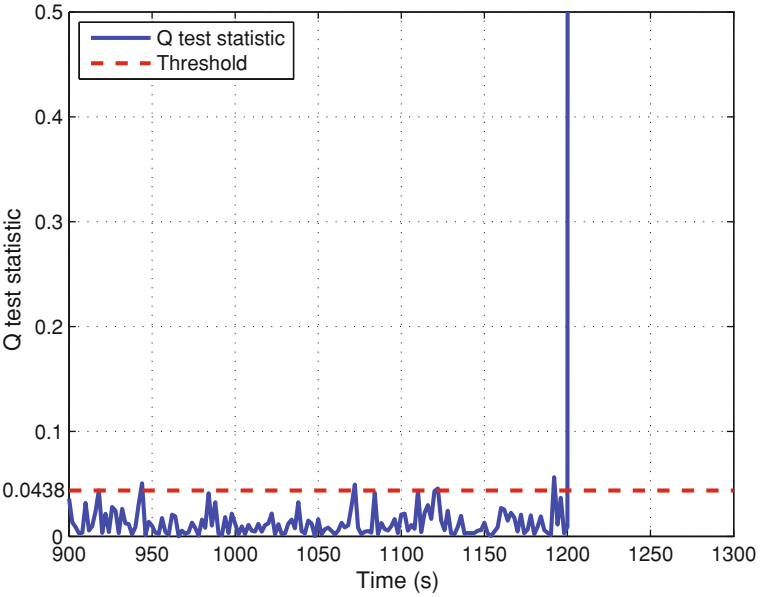


Fig. 3.8 Q test statistic and the threshold (Algorithm 3.6)

3.5 Variations of the Essential Fault Detection Problem

In this section, two basic variations of the essential fault detection FD-P1 are briefly introduced.

3.5.1 Variation I

Consider a measurement model (sensor model)

$$y = Ax + \varepsilon, \varepsilon \sim \mathcal{N}(0, \Sigma) \tag{3.42}$$

where $y \in \mathcal{R}^m$ represents the measurement vector, $x \in \mathcal{R}^n$ the internal process (state) vector which may be driven by some process input variables and is not measured. We assume that $m > n$, the covariance matrix Σ and matrix $A \in \mathcal{R}^{m \times n}$ are known. Our task is to detect faults in the measurement space, which is modeled by

$$y = Ax + f + \varepsilon, f = \begin{cases} 0, & \text{fault-free} \\ f_1 \neq 0, & \text{faulty} \end{cases}. \tag{3.43}$$

There are different ways to approach this problem. Recall that $m > n$. Then, we can find $A^{\perp} \in \mathcal{R}^{(m-n) \times n}$ so that

$$A^{\perp} A = 0 \Longrightarrow A^{\perp} y = A^{\perp} (f + \varepsilon) := \bar{f} + \bar{\varepsilon}. \tag{3.44}$$

In this way, our detection problem is reduced to the FD-P1 and can be solved using the tools introduced in Sect. 3.4. On the other hand, it is evident that by this detection scheme only those faults can be detected, which satisfy $A^{\perp} f \neq 0$. That means, the fault detectability is poor.

Next, we consider an alternative solution on the assumption that A is left-invertible. Let $\hat{x}$ be a least squares (LS) estimation, that is,

$$\hat{x} = \left(A^T A\right)^{-1} A^T y. \tag{3.45}$$

We build

$$r = y - A\hat{x} \tag{3.46}$$

in order to remove the influence of x in y, and use it for the detection purpose. Recall that the LS estimate (3.45) is unbiased. Hence, in the fault-free case

$$r \sim \mathcal{N}(0, \Sigma_r), \Sigma_r = \left(I - A\left(A^T A\right)^{-1} A^T\right) \Sigma \left(I - A\left(A^T A\right)^{-1} A^T\right)^T.$$

Again, the fault detection problem is reduced to the FD-P1. The remaining question is: is this solution an optimal one?

In our previous study we have learnt that fault detectability would be improved if the covariance matrix (of the noise) becomes smaller. Moreover, it is well-known for $W \in \mathcal{R}^{m\times m}$ and $W > 0$

$$\hat{x} = \left(A^T W A\right)^{-1} A^T W y \tag{3.47}$$

delivers a unbiased estimate for x, which is also called weighted LS, and for $W = \Sigma^{-1}$, that is,

$$\hat{x} = \left(A^T \Sigma^{-1} A\right)^{-1} A^T \Sigma^{-1} y \tag{3.48}$$

we have the minimum variance estimation, that is,

$$\mathcal{E}\left(x - \hat{x}\right)\left(x - \hat{x}\right)^T = \left(A^T \Sigma^{-1} A\right)^{-1}.$$

In this case,

$$r \sim \mathcal{N}(0, \Sigma_r),\ \Sigma_r = \Sigma - AA^T \Sigma^{-1} AA^T. \tag{3.49}$$

As a result of our discussion, it is reasonable to apply the minimum variance LS estimation (3.48) for achieving an optimal fault detection.

It is worth to notice that the solution with the minimum variance LS estimation (3.48) can be interpreted as a two-step scheme:

- normalization of the noise by $W^{1/2}$, which leads to

$$\bar{y} = W^{1/2} y = \Sigma^{-1/2} Ax + \Sigma^{-1/2}\varepsilon,\ \Sigma^{-1/2}\varepsilon \sim \mathcal{N}(0, I)$$

- LS solution

$$\hat{x} = \left(\bar{A}^T \bar{A}\right)^{-1} \bar{A}^T \bar{y} = \left(A^T \Sigma^{-1} A\right)^{-1} A^T \Sigma^{-1} y,\ \bar{A} = \Sigma^{-1/2} A.$$

3.5.2 Variation II

Different to the above variation, we now consider a monitoring or detection problem in the subspace of process internal state variables using the process measurement data. This problem is formulated as follows: Given

$$\theta = Py,\ y \sim \mathcal{N}(0, \Sigma_y) \tag{3.50}$$

where $y \in \mathcal{R}^m$ represents the measurement vector, $\theta \in \mathcal{R}^l$ the internal process state or key variable vector, our task is, on the assumption $m > l$, to detect changes or faults in the subspace of the the internal process state or key variables using process measurement data y. This is problem is in fact trivial so far Σ_y and P are known. The major focus of our study will be on the data-driven solutions with unknown Σ_y and P.

3.6 Notes and References

In this chapter, essentials of statistical methods have been briefly reviewed. For our purpose, a basic fault detection problem, FD-P1, has first been formulated. Although it is in fact a fault detection problem with model knowledge, it is useful and helpful for us to understand the basic ideas and principles of the GLR technique, the T^2 test statistic and hypothesis testing methods as well as their applications to solving fault detection problems. For more detailed and comprehensive study, we refer the reader to the excellent monographs by Basseville and Nikiforov [1] and by Gustafsson [2]. The $\mathcal{T}^2$-distribution was introduced by Hotelling in 1931 [3] and therefore the corresponding test statistic is also called Hotelling's T^2 test statistic.

The study on the data-driven solutions of FD-P1 in Sect. 3.3 is fundamental for our subsequent work. We have presented a general procedure of data-driven fault detection, which consists of two phases: (i) training with offline computation (ii) online detection. This is the fault detection practice widely accepted and adopted in real applications. This way of solving fault detection problems is intuitional and well motivated by the existing (model-based) statistical methods. The training work serves indeed for identifying (estimating) the mean value and covariance needed for fault detection. As a result, similar test statistics and threshold settings, as known in the existing statistical methods introduced in Sect. 3.2, are then adopted for the off- and online computations.

In Sect. 3.3.2, we have studied the data-driven fault detection problem in the statistical framework, in which the "training data" and online measurement data are viewed as two independent samples and the detection problem is formulated as detecting the difference (change) between the mean values of the samples. By means of the T^2 test statistic for change detection, the threshold setting is realized using $\mathcal{F}$ -distribution. We refer the reader to [4–6] for the needed mathematical knowledge and detailed discussion in this subsection, which is essential for the study in the sequel. The proof of Theorem 3.1 can be found, for instance, in Sect. 5.2 in [6].

As an alternative statistic, we have introduced Q statistic in Sect. 3.3.3. This test statistic has been, for instance, proposed by Nomikos and MacGregor for process monitoring [7], where the T^2 test statistic is, considering the involved inverse computation of the covariance matrix, replaced by the Q statistic for a reliable monitoring performance. For the purpose of threshold setting, the χ^2 approximations of the Q statistic that was proposed and proved by Box [8] has been adopted.

At the end of this chapter, we have formulated two variations of the essential fault detection problem. Different forms of the first variation will be addressed in Chaps. 7–10 and 12. For a detailed description of the LS estimation schemes, the reader is referred to [9]. The second variation will be dealt with in Chap. 6 and partly in Chaps. 9, 10.

References

1. Basseville M, Nikiforov I (1993) Detection of abrupt changes—theory and application. Prentice-Hall, New Jersey
2. Gustafsson F (2000) Adaptive filtering and change detection. Wiley, Chichester
3. Hotelling H (1931) The generalization of student's ratio. Ann Math Stat 2:360–378
4. Wadsworth HM (1997) Handbook of statistical methods for engineers and scientists, 2nd edn. McGraw-Hill, New York
5. Papoulis A, Pillai SU (2002) Probability, random variables, and stochastic process, 4th edn. McGraw-Hill, New York
6. Härdle WK, Simar L (2012) Applied multivariate statistical analysis, 3rd edn. Springer, Berlin
7. Nomikos P, Macgregor J (1995) Multivariate SPC charts for monitoring batch processes. Technometrics 37:41–59
8. Box G (1954) Some theorems on quadratic forms applied in the study of analysis of variance problems: Effect of inequality of variance in one-way classification. Ann Math Stat 25:290–302
9. Kailath T, Sayed A, Hassibi B (1999) Linear estimation. Prentice Hall, New Jersey

Chapter 4
Fault Detection in Processes with Deterministic Disturbances

In this chapter, we address fault detection in static processes with deterministic disturbances. Although this fault detection problem is often met in practice, there are only few reported results on the relevant issues. Our objective is, in comparison and relation with the study in the last chapter, to introduce some basic concepts, ideas and needed solution tools.

4.1 Problem Formulations and Some Elementary Concepts

We study, in this section, the formulation of a basic problem and, associated with it, some essential concepts.

4.1.1 A Simple Detection Problem and Its Intuitive Solution

We consider the following fault detection problem: Given a measurement model

$$y(k) = f + d(k) \in \mathcal{R} \tag{4.1}$$

where d is a unknown disturbance and bounded by

$$\forall k,\ \|d\|_{2,[k,k+n]} = \sqrt{\sum_{i=0}^{n} d^2(k+i)} \leq \delta_d \tag{4.2}$$

with a known constant $\delta_{d,n}(> 0)$ as boundedness. It is assumed that

$$f = \begin{cases} 0, & \text{fault-free} \\ \text{constant } \neq 0, & \text{faulty} \end{cases}.$$

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_4, © Springer-Verlag London 2014

The fault detection problem we now address is to detect $f \neq 0$ by means of a number of available measurement data of y, $y(1), \ldots, y(n)$.

Considering that information we have about the disturbance is the boundedness of its $l_{2,[0,n]}$-norm, we use the $l_{2,[0,n]}$-norm of the measurement y as a test (evaluation) function, that is,

$$J(k) = \|y\|^2_{2,[k,k+n]} = \sum_{i=0}^{n} y^2(k+i) \tag{4.3}$$

and set the threshold

$$J_{th} = \max_k \|d\|^2_{2,[k,k+n]} = \delta_d^2. \tag{4.4}$$

As a result, the detection logic is

$$\begin{cases} J(k) \leq J_{th} \Longrightarrow \text{fault-free } (H_0) \\ J(k) > J_{th} \Longrightarrow \text{faulty } (H_1) \end{cases}. \tag{4.5}$$

4.1.2 Some Essential Concepts

Comparing with the fault detection schemes presented in the last chapter, similarities and differences can be recognized in the essential steps of the above solution. For our detection purpose, a test function, also called evaluation function, is needed. Differently, knowledge of the unknown disturbance is expressed in form of a signal norm and its boundedness. In dealing with deterministic signals, which can be either a unknown input vector or a measurement vector, some well defined signal norms are often applied. Let

$$x(k) = \begin{bmatrix} x_1(k) \\ \vdots \\ x_m(k) \end{bmatrix} \in \mathcal{R}^m$$

be an m-dimensional vector. Below are some typical signal norms or evaluation functions widely used in the fault detection research.

- Peak value: The peak value of $x(k)$ is defined and denoted by

$$\|x\|_{peak} := \sup_{k \geq 0} \|x(k)\|_E \,, \; \|x(k)\|_E = \left(\sum_{i=1}^{m} x_i^2(k) \right)^{1/2}. \tag{4.6}$$

- $l_{2,\phi}$-norm: For $k \in \phi = [k_1, k_2]$

$$\|x\|_{2,\phi} = \left(\sum_{k=k_1}^{k_2} x^T(k) x(k) \right)^{1/2}. \tag{4.7}$$

- RMS value: For integer $n \geq 1$,

$$\|x(k)\|_{RMS} = \left(\frac{1}{n} \sum_{j=1}^{n} \|x(k+j)\|_E^2 \right)^{1/2}. \tag{4.8}$$

It is evident that a major difference between the solutions introduced in the last chapter and the above intuitive solution is that the latter guarantees no false alarm, since for $f = 0$

$$J(k) = \|y\|_{2,[k,k+n]}^2 = \|d\|_{2,[k,k+n]}^2 \leq \delta_d^2 = J_{th}. \tag{4.9}$$

This nice performance is, unfortunately, achieved often at cost of fault detectability. This point can be clearly seen from the fact that the larger δ_d^2 is, the higher the threshold becomes. Consequently, only those larger-sized faults which cause

$$\|y\|_{2,[k,k+n]}^2 = \|f + d\|_{2,[k,k+n]}^2 > J_{th}$$

can be detected.

In fact, in practice δ_d^2 is often a (very) conservative estimate of the $l_{2,[0,n]}$-norm of the disturbance, which will be approached only in some extreme situations. In order to achieve an acceptable trade-off between the false alarm rate and fault detectability, we introduce, analog to the FAR definition in the statistical framework, the following concepts.

Let $d(k) \in \mathcal{R}^{k_d}$ be a unknown input vector, assume δ_d^2 is a random variable and define $\delta_{d,\alpha}^2$ as

$$prob(\delta_d^2 > \delta_{d,\alpha}^2) = \alpha \tag{4.10}$$

for some given $\alpha > 0$. Suppose that J is the evaluation function and for some given δ_d^2

$$\max_{f=0, \|d\|^2 \leq \delta_d^2} J = \delta_d^2 \tag{4.11}$$

as applied in (4.3), (4.4), where $\|\cdot\|$ denotes some signal norm. If the threshold is now set to be

$$J_{th} = \max_{f=0, \|d\|^2 \leq \delta_{d,\alpha}^2} J = \delta_{d,\alpha}^2 \tag{4.12}$$

it turns out

$$prob\{J \leq J_{th} \mid f = 0\} \leq prob\,(\delta_d^2 \leq \delta_{d,\alpha}^2) = 1 - \alpha. \tag{4.13}$$

In other words, the probability of a false alarm is maximally equal to α. In this context, we introduce the definition of false alarm rate.

Definition 4.1 Given a random variable δ_d^2 with a known distribution and suppose that the threshold is set by (4.12). The probability

$$\text{prob}(\delta_d^2 > \delta_{d,\alpha}^2) = \alpha \tag{4.14}$$

is called false alarm rate (FAR).

We would like to comment the above FAR definition briefly. In practice, the energy level or peak value of the disturbances in and around the process continuously vary, depending on the process operation and environment conditions. The introduction of δ_d^2 as a random variable can be interpreted as modeling the variation by a statistical process. The idea behind this effort is to make use of available information to achieve an acceptable trade-off between the FAR and fault detectability.

Example 4.1 Consider the fault detection problem formulated in Sect. 4.1.1 and the solution scheme given in (4.3)–(4.5). Assume that δ_d^2 is uniformly distributed on the interval $\left[0, \delta_{d,\max}^2\right]$ with the probability density

$$f\left(\delta_d^2\right) = \frac{1}{\delta_{d,\max}^2}.$$

Since

$$prob(\delta_d^2 \le \delta_{d,\alpha}^2) = \frac{\delta_{d,\alpha}^2}{\delta_{d,\max}^2},\ \delta_{d,\alpha}^2 \le \delta_{d,\max}^2$$

for $J_{th} = \delta_{d,\alpha}^2$,

$$FAR = 1 - \frac{\delta_{d,\alpha}^2}{\delta_{d,\max}^2}.$$

On the other hand, if α is given, then we can find $\delta_{d,\alpha}^2$:

$$\delta_{d,\alpha}^2 = (1-\alpha)\,\delta_{d,\max}^2.$$

4.1.3 Problem Formulations

Next, we give a more general formulation of the previous fault detection problem.

Fault detection problem II (FD-P2): Given

$$y(k) = f + Md(k) \in \mathcal{R}^m,\ m \ge 1 \tag{4.15}$$

where $M \in R^{m\times k_d}$ is a (real) known matrix, $d(k) \in R^{k_d}$ is a unknown input vector and bounded by

$$\|d\|^2 \leq \delta_d^2$$

with $\|\cdot\|$ *denoting some signal norm, find a test function* J *and the corresponding threshold* J_{th} *such that* $f \neq 0$ *can be detected under the detection logic* (4.5)*. If, moreover,* δ_d^2 *can be represented by a random variable with a known distribution, determine the threshold* J_{th} *for given* $FAR \leq \alpha$ *, as defined in Definition 4.1.*

Again, it should be mentioned that the above formulation is a model-based fault detection problem, since we assume that M is known a prior. A data-driven version of the above fault detection problem is given below.

Data-driven version of FD-P2: Given (4.15) *with* M *being unknown, assume process data,* $y(1), \ldots, y(n)$*, are available, find a fault detection scheme that allows a reliable detection of* $f \neq 0$ *based on online data* $y(k), \ldots, y(k+n)$*.*

4.2 Some Elementary Methods and Algorithms

In this section, we are focused on solving FD-P2.

4.2.1 An Intuitive Strategy

The major idea of the intuitive solution given in Sect. 4.1.1 is to set the threshold equal to the maximal influence of the disturbance on the evaluation function, which is a norm of the measurement. In a more general form we can write it as

$$J_{th} = \max_{f=0,d} J = \max_{f=0,d} \|y\|^2 . \tag{4.16}$$

Following this idea, we define different types of evaluation functions, corresponding to the evaluation of the unknown input vector, as follows:

- $l_{2,\phi}$-norm type evaluation function: for $k \in \phi = [k_1, k_2]$

$$J_2(k_1, k_2) = \|y\|_{2,\phi}^2 = \sum_{k=k_1}^{k_2} y^T(k) y(k) \tag{4.17}$$

- RMS value evaluation function**:** for integer $n \geq 1$,

$$J_{RMS}(k, n) = \|y(k)\|_{RMS}^2 = \frac{1}{n} \sum_{j=1}^{n} y^T(k+j) y(k+j) \tag{4.18}$$

- Peak value type evaluation function:

$$J_{peak} = \|y\|^2_{peak} := \sup_{k\geq 0} \|y(k)\|^2_E \tag{4.19}$$

On the assumptions

$$\forall d, k, \|d\|^2_{2,\phi} \leq \delta^2_{d,2}, \|d(k)\|^2_{RMS} \leq \delta^2_{d,RMS}, \|d(k)\|^2_{peak} \leq \delta^2_{d,peak} \tag{4.20}$$

and considering

$$\forall d(k), \|Md(k)\|_E \leq \sigma_{\max}(M) \|d(k)\|_E$$

where $\sigma_{\max}(M)$ is the maximum singular value of M, the thresholds are set to be

$$J_{th,2} = \max_{f=0,\|d\|^2_{2,\phi}\leq\delta^2_{d,2},k_1,k_2} J_2(k_1,k_2) = \max_{\|d\|^2_{2,\phi}\leq\delta^2_{d,2}} \|Md\|^2_{2,\phi} = \sigma^2_{\max}(M)\,\delta^2_{d,2} \tag{4.21}$$

$$\begin{aligned} J_{th,RMS} &= \max_{f=0,\|d(k)\|^2_{RMS}\leq\delta^2_{d,RMS},k} J_{RMS}(k,n) = \max_{\|d(k)\|^2_{RMS}\leq\delta^2_{d,RMS}} \|Md(k)\|^2_{RMS} \\ &= \sigma^2_{\max}(M)\,\delta^2_{d,RMS} \end{aligned} \tag{4.22}$$

$$\begin{aligned} J_{th,peak} &= \max_{f=0,\|d(k)\|^2_{peak}\leq\delta^2_{d,peak}} J_{peak} = \max_{\|d(k)\|^2_{peak}\leq\delta^2_{d,peak}} \|Md(k)\|^2_{peak} \\ &= \sigma^2_{\max}(M)\,\delta^2_{d,peak}. \end{aligned} \tag{4.23}$$

It is interesting to notice that all three threshold settings have the same form and require the computation of the maximum singular value of matrix M. In this way, the fault detection problem FD-P2 can be solved.

4.2.2 An Alternative Solution

In this subsection, we first propose an alternative solution and then demonstrate that this solution delivers an improved fault detectability in comparison with the intuitive solution.

Suppose that

$$rank(M) = m \leq k_d.$$

That means, the influence of the unknown input vector spans the whole measurement subspace. Let

$$M = U \begin{bmatrix} \Sigma & 0 \end{bmatrix} V^T, \Sigma = \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_m \end{bmatrix}, \sigma_1 \geq \ldots \geq \sigma_m > 0$$

$$UU^T = I_{m\times m}, V^T V = I_{k_d \times k_d}$$

be a singular value decomposition (SVD) of M. The pseudo-inverse of M is then given by

$$M^{-} = V \begin{bmatrix} \Sigma^{-1} \\ 0 \end{bmatrix} U^{T}.$$

We now define

$$\bar{y}(k) = M^{-} y(k) \tag{4.24}$$

and, associated with it, the evaluation functions

- $l_{2,\phi}$-norm type evaluation function: for $k \in \phi = [k_1, k_2]$

$$J_2(k_1, k_2) = \|\bar{y}\|_{2,\phi}^2 = \sum_{k=k_1}^{k_2} y^T(k) U \Sigma^{-2} U^T y(k) \tag{4.25}$$

- RMS value evaluation function**:** for integer $n \geq 1$,

$$J_{RMS}(k, n) = \|\bar{y}(k)\|_{RMS}^2 = \frac{1}{n} \sum_{j=1}^{n} y^T(k+j) U \Sigma^{-2} U^T y(k+j) \tag{4.26}$$

- Peak value type evaluation function:

$$J_{peak} = \|\bar{y}\|_{peak}^2 := \sup_{k \geq 0} y^T(k) U \Sigma^{-2} U^T y(k). \tag{4.27}$$

Note that

$$\forall d(k), \left\| M^{-} M d(k) \right\|_E \leq \|d(k)\|_E .$$

As a result, the corresponding thresholds settings are

$$J_{th,2} = \max_{f=0, \|d\|_{2,\phi}^2 \leq \delta_{d,2}^2, k_1, k_2} J_2(k_1, k_2) = \max_{\|d\|_{2,\phi}^2 \leq \delta_{d,2}^2} \left\| M^{-} M d \right\|_{2,\phi}^2 = \delta_{d,2}^2 \tag{4.28}$$

$$\begin{aligned} J_{th,RMS} &= \max_{f=0, \|d(k)\|_{RMS}^2 \leq \delta_{d,RMS}^2, k} J_{RMS}(k, n) \\ &= \max_{\|d(k)\|_{RMS}^2 \leq \delta_{d,RMS}^2} \left\| M^{-} M d(k) \right\|_{RMS}^2 = \delta_{d,RMS}^2 \end{aligned} \tag{4.29}$$

$$\begin{aligned} J_{th,peak} &= \max_{f=0, \|d(k)\|_{peak}^2 \leq \delta_{d,peak}^2} J_{peak} = \max_{\|d(k)\|_{peak}^2 \leq \delta_{d,peak}^2} \left\| M^{-} M d(k) \right\|_{peak}^2 \\ &= \delta_{d,peak}^2. \end{aligned} \tag{4.30}$$

The needed off- and online computations for the application of the alternative method are summarized in the next two algorithms.

Algorithm 4.1 *Design (offline computation)*

S1: Do SVD on M and save U, Σ^{-1}
S2: Set the threshold according to (4.28)–(4.30), depending on the evaluation function.

Algorithm 4.2 *Online computation*

S1: Compute the evaluation function either (4.25) or (4.26) or (4.27)
S2: Make a decision

$$\begin{cases} J \leq J_{th} \Longrightarrow \textit{fault-free} \\ J > J_{th} \Longrightarrow \textit{faulty and alarm} \end{cases}.$$

4.2.3 A Comparison Study

We are now going to compare the two fault detection solutions to FD-P2 and demonstrate that the alternative solution delivers a better fault detection performance. To this end, we first consider the intuitive fault detection scheme proposed in Sect. 4.2.1. It follows from (4.17)–(4.23) that a fault, $f \neq 0$, can be detected if and only if

$$J > J_{th} \Longleftrightarrow \frac{\|f + Md(k)\|^2}{\sigma^2_{\max}(M)} > \delta^2_d, \sigma_{\max}(M) = \sigma_1(M) \tag{4.31}$$

where δ^2_d denotes either $\delta^2_{d,2}$ or $\delta^2_{d,RMS}$ or $\delta^2_{d,peak}$, and $\|\cdot\|$ is either $\|\cdot\|_{2,\phi}$ or $\|\cdot\|_{RMS}$ or $\|\cdot\|_{peak}$. Next, we check the condition, under which $f \neq 0$ can be detected using the alternative solution given in the last subsection. It is

$$\left\|M^-(f + Md(k))\right\|^2 > \delta^2_d. \tag{4.32}$$

It is known that

$$\left\|M^-(f + Md(k))\right\|^2 \geq \sigma^2_{\min}\left(M^-\right)\|f + Md(k)\|^2$$
$$\sigma^2_{\min}\left(M^-\right) = \frac{1}{\sigma^2_1(M)} = \frac{1}{\sigma^2_{\max}(M)}.$$

It holds

$$\left\|M^-(f + Md(k))\right\|^2 \geq \frac{\|f + Md(k)\|^2}{\sigma^2_{\max}(M)}$$

which means (4.32) is a necessary condition of (4.31). Consider further that for some M there exist f, d so that

$$\left\|M^{-}\left(f+Md(k)\right)\right\|^{2}>\sigma_{\min}^{2}\left(M^{-}\right)\|f+Md(k)\|^{2}=\frac{\|f+Md(k)\|^{2}}{\sigma_{\max}^{2}(M)}.$$

It can, thus, be concluded that in general by means of the alternative solution more faults can be detected than using the intuitive solution.

It is of interest to understand the reason for the difference in the fault detection performance. To this end, write $\bar{y}$ into

$$\bar{y}(k)=M^{-}\left(f+Md(k)\right)=V\left(\begin{bmatrix}\Sigma^{-1}\\0\end{bmatrix}U^{T}f+\begin{bmatrix}I&0\\0&0\end{bmatrix}V^{T}d(k)\right) \tag{4.33}$$

$$=V_{1}\Sigma^{-1}U^{T}f+V_{1}V_{1}^{T}d(k),\ V=\begin{bmatrix}V_{1}&V_{2}\end{bmatrix}. \tag{4.34}$$

It can be seen that $\bar{y}(k)$ consists, in part, of a projection of the fault vector onto U^{T}. Moreover, the directions of subspace spanned by U^{T} is weighted by $\sigma_{i}^{-1}(M), i=1,\ldots m$. That means, the direction with a smaller singular value will be stronger weighted. This is the explanation why the alternative solution is better than the intuitive one, by which it can be interpreted that an identical weighting, $\sigma_{\max}^{-2}(M)$, is applied to all directions.

4.2.4 Unknown Input Estimation Based Detection Scheme

Consider model (4.15) for $f=0$. It is easy to prove that $\bar{y}(k)=M^{-}y(k)$ satisfies

$$M\bar{y}(k)=Md(k)=y(k).$$

Thus, $\bar{y}(k)$ can be interpreted as an estimate of $d(k)$, that is

$$\hat{d}(k)=\bar{y}(k)=M^{-}y(k). \tag{4.35}$$

Note that $\hat{d}(k)$ is the solution of (4.15) with the minimum norm. Moreover, it holds

$$\left\|\hat{d}(k)\right\|_{E}^{2}\leq\|d(k)\|_{E}^{2}. \tag{4.36}$$

This interpretation and the associated discussion motivate us to propose a new fault detection scheme. The core of this scheme is the estimation of the unknown inputs, based on which

- the evaluation function is defined by

$$J = \left\| \hat{d}(k) \right\|^2 = \left\| M^- y(k) \right\|^2 \tag{4.37}$$

- the threshold setting

$$J_{th} = \delta_d^2. \tag{4.38}$$

We would like to emphasize that although the above evaluation function and the threshold setting are identical with the ones given in Algorithm 4.1, the ideas of these two schemes are different. Indeed, the fault detection scheme (4.37), (4.38) is a direct use of the available information about $d(k)$ and its boundedness. For this purpose, an estimate of $d(k)$ is applied. In the fault detection schemes introduced in the previous subsection, the idea is to estimate the maximum influence of $d(k)$ on the evaluation function and set it as the threshold.

4.2.5 A General Solution

We first consider the case that $rank(M) = k_d < m$, that means, the unknown input vector does not span the whole measurement subspace. Note that in this case there exists a matrix P so that $PM = 0$. If we now form a test function like

$$J = \| Py(k) \|^2$$

then J only depends on f

$$J = \| Pf \|^2 .$$

Consequently, the threshold can be set, theoretically, equal to zero. Motivated by this observation, we propose the following fault detection scheme to deal with such a case.

Let

$$M = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T, \Sigma = \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_{k_d} \end{bmatrix}, \sigma_1 \geq \ldots \geq \sigma_{k_d} > 0$$
$$U = \begin{bmatrix} U_1 \; U_2 \end{bmatrix}, U_1 \in \mathcal{R}^{m \times k_d}, UU^T = I_{m \times m}, V^T V = I_{k_d \times k_d}$$

be an SVD of M. The pseudo-inverse of M is then given by

$$M^- = V \begin{bmatrix} \Sigma^{-1} \; 0 \end{bmatrix} U^T = V \Sigma^{-1} U_1^T \in \mathcal{R}^{k_d \times m}.$$

We now define

$$\bar{y}(k) = Py(k), P = \begin{bmatrix} M^- \\ \frac{U_2^T}{\epsilon} \end{bmatrix} \in \mathcal{R}^{m \times m} \tag{4.39}$$

where $\epsilon(>0)$ is a small constant, and, associated with $\bar{y}(k)$, the evaluation functions

- $l_{2,\phi}$-norm type evaluation function: for $k \in \phi = [k_1, k_2]$

$$J_2(k_1, k_2) = \|\bar{y}\|_{2,\phi}^2 = \sum_{k=k_1}^{k_2} y^T(k) \left(U_1 \Sigma^{-2} U_1^T + \frac{U_2 U_2^T}{\epsilon^2} \right) y(k) \tag{4.40}$$

- RMS value evaluation function: for integer $n \geq 1$,

$$J_{RMS}(k, n) = \|\bar{y}(k)\|_{RMS}^2 = \frac{1}{n} \sum_{j=1}^{n} y^T(k+j) \left(U_1 \Sigma^{-2} U_1^T + \frac{U_2 U_2^T}{\epsilon^2} \right) y(k+j) \tag{4.41}$$

- Peak value type evaluation function:

$$J_{peak} = \|\bar{y}\|_{peak}^2 := \sup_{k \geq 0} \left(y^T(k) \left(U_1 \Sigma^{-2} U_1^T + \frac{U_2 U_2^T}{\epsilon^2} \right) y(k) \right). \tag{4.42}$$

Note that

$$\forall d(k), \|PMd(k)\|_E = \|d(k)\|_E .$$

As a result, the corresponding threshold settings are

$$J_{th,2} = \max_{f=0, \|d\|_{2,\phi}^2 \leq \delta_{d,2}^2, k_1, k_2} J_2(k_1, k_2) = \max_{\|d\|_{2,\phi}^2 \leq \delta_{d,2}^2} \|PMd\|_{2,\phi}^2 = \delta_{d,2}^2 \tag{4.43}$$

$$\begin{aligned} J_{th,RMS} &= \max_{f=0, \|d(k)\|_{RMS}^2 \leq \delta_{d,RMS}^2, k} J_{RMS}(k, n) \\ &= \max_{\|d(k)\|_{RMS}^2 \leq \delta_{d,RMS}^2} \|PMd(k)\|_{RMS}^2 = \delta_{d,RMS}^2 \end{aligned} \tag{4.44}$$

$$\begin{aligned} J_{th,peak} &= \max_{f=0, \|d(k)\|_{peak}^2 \leq \delta_{d,peak}^2} J_{peak} = \max_{\|d(k)\|_{peak}^2 \leq \delta_{d,peak}^2} \|PMd(k)\|_{peak}^2 \\ &= \delta_{d,peak}^2 . \end{aligned} \tag{4.45}$$

We now briefly study the above fault detection scheme. Recall that

$$U_2^T M = U_2^T \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T = 0$$

which means U_2^T spans the null subspace of the M matrix. This fact can also be interpreted as the influence of the unknown vector in the directions defined by U_2^T being zero. According to our discussion in Sect. 4.2.3, in the evaluation function these directions should be weighted by an infinitively large number. Exactly for this reason, we have introduced factor $1/\epsilon$ in P, where ϵ should be, theoretically, selected sufficiently small. It is evident that

$$J = \|\bar{y}(k)\|^2 = \left\|M^- y(k)\right\|^2 + \left\|\frac{U_2^T}{\epsilon} y(k)\right\|^2$$
$$= \left\|M^- f + d(k)\right\|^2 + \frac{1}{\epsilon^2}\left\|U_2^T f\right\|^2 \tag{4.46}$$

which tells us if the projection of f onto the subspace spanned by U_2^T is not zero, then $\frac{1}{\epsilon^2}\left\|U_2^T f\right\|^2$ can be considered as sufficiently large, which ensures that $J > J_{th}$ and thus the fault can be detected.

It is worth noting that in the fault-free case, $M^- y(k)$ delivers an exact estimation for $d(k)$. The projection of $y(k)$ onto M^- has the same interpretation introduced in the last subsection.

Below are the off- and online algorithms for solving the fault detection problem addressed in this subsection.

Algorithm 4.3 *Design (offline computation)*

S1: Do an SVD on M and save U, Σ^{-1}
S2: Set the threshold according to (4.43)–(4.45), depending on the evaluation function.

Algorithm 4.4 *Online computation*

S1: Compute the evaluation function either (4.40) or (4.41) or (4.42)
S2: Make a decision

$$\begin{cases} J \le J_{th} \Longrightarrow \textit{fault-free} \\ J > J_{th} \Longrightarrow \textit{faulty and alarm} \end{cases}.$$

4.3 A Data-Driven Solution of the Fault Detection Problem

Analog to the study on the data-driven solutions to FD-P1, we now deal with the data-driven version of FD-P2 formulated in Sect. 4.1.3. The fault detection problem is solved in the same two-step procedure:

- identification of the model parameters using the recorded data
- application of an existing (model-based) fault detection schemes introduced in the last section.

For our purpose, we only consider the evaluation functions J_2 and J_{RMS} defined in (4.17) and (4.18) respectively. Moreover, in order to cover the normal operations as much as possible, the number of the data is selected sufficiently large. The computation steps in the training phase are summarized in the following algorithm for given (fault-free) data $y(1), \ldots, y(N)$.

Algorithm 4.5 *Training*

S1: Form matrix $Y = \left[y(1) \ \ldots \ y(N) \right]$ *and compute* YY^T
S2: Do an SVD of YY^T

$$YY^T = \left[U_1 \ U_2 \right] \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}, \ \Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_r^2 \end{bmatrix} \tag{4.47}$$

$$r = \min \{m, k_d\}$$

S3: Set the threshold

$$J_{th,2} = \sum_{i=1}^{r} \sigma_i^2 \ or \ J_{th,RMS} = \frac{1}{N} \sum_{i=1}^{r} \sigma_i^2. \tag{4.48}$$

The corresponding algorithm for the online fault detection is as follows.

Algorithm 4.6 *(Data-driven) online computation*

S1: Calculate

$$\bar{y}(k) = \begin{bmatrix} U_1^T \\ \frac{U_2^T}{\epsilon} \end{bmatrix} y(k), \ J = \|\bar{y}(k)\|^2 \tag{4.49}$$

S2: Make a decision

$$\begin{cases} J \leq J_{th} \Longrightarrow \textit{fault-free} \\ J > J_{th} \Longrightarrow \textit{faulty and alarm} \end{cases}.$$

We now explain the above detection algorithms. On the assumption of model (4.15) for $f = 0$, YY^T can be written as

$$YY^T = M \left(\sum_{i=1}^{N} d(i) d^T(i) \right) M^T = \left[U_1 \ U_2 \right] \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}.$$

Note that U_2^T defines the null subspace of the subspace spanned by the unknown input vector. According to the discussion in the last subsection, the evaluation function will consist of two parts:

$$J = \left\| U_1^T y(k) \right\|^2 + \frac{1}{\epsilon^2} \left\| U_2^T y(k) \right\|^2$$

with $\left\| U_1^T y(k) \right\|^2$ representing the one in the subspace spanned by the unknown input vector (in the measurement space) and $\frac{1}{\epsilon^2} \left\| U_2^T y(k) \right\|^2$ the one in its null subspace.

Since both $M, d(k)$ are unknown, we have to use

$$\sum_{i=1}^{r} \sigma_i^2 = tr\left(\sum_{i=1}^{N} Md(i)d^T(i)M^T\right) = \sum_{i=1}^{N} d^T(i)M^T Md(i)$$

to represent the maximum influence of the unknown input on the measurement and define it as the threshold, as given in (4.48). Note that

$$J_{th,RMS} = \frac{1}{N}\sum_{i=1}^{r} \sigma_i^2 \tag{4.50}$$

is achieved by computing the average value over N samples. Thus, in the real-time application, if only n samples are adopted for the RMS computation, $J_{th,RMS}$ is interpreted as

$$\begin{aligned}
&\max_{f=0,\|d(k)\|^2_{RMS}\le\delta^2_{d,RMS}} J_{RMS}(k,n) \\
&= \max_{f=0,\|d(k)\|^2_{RMS}\le\delta^2_{d,RMS}} \frac{n}{N}\left(\begin{array}{c} \frac{1}{n}\sum_{j=1}^{n} \bar{y}^T(k+j)\bar{y}(k+j) + \\ \frac{1}{n}\sum_{j=n+1}^{2n} \bar{y}^T(k+j)\bar{y}(k+j) + \cdots \end{array}\right) \\
&= \frac{1}{N}\sum_{i=1}^{r} \sigma_i^2.
\end{aligned}$$

Correspondingly, the evaluation function should be

$$J_{RMS}(k,n) = \frac{n}{N}\left(\frac{1}{n}\sum_{j=1}^{n} \bar{y}^T(k+j)\bar{y}(k+j)\right) = \frac{1}{N}\sum_{j=1}^{n} \bar{y}^T(k+j)\bar{y}(k+j). \tag{4.51}$$

4.4 A Variation of the Essential Fault Detection Problem

In this section, we deal with a variation of FD-P2. Consider the measurement model (sensor model)

$$y(k) = Ax + Md(k) \tag{4.52}$$

where $y \in \mathcal{R}^m$ represents the measurement vector, $d(k) \in \mathcal{R}^{k_d}$ a (unknown) disturbance vector, $x \in \mathcal{R}^n$ the internal process (state) vector which is not measured. We assume that $m > n$, $k_d = m$, matrix A, M are known and (left) invertible. Our task

is to detect faults in the measurement space, which is modeled by

$$y(k) = Ax + f + Md(k),\ f = \begin{cases} 0, \text{ fault-free} \\ f_1 \neq 0, \text{ faulty} \end{cases}. \tag{4.53}$$

A standard solution is a three-step scheme that consists of

- an LS estimation of x

$$\hat{x} = \left(A^T A\right)^{-1} A^T y(k) \tag{4.54}$$

- residual building and evaluation

$$r(k) = y(k) - A\hat{x} \tag{4.55}$$

$$\begin{aligned} J_{RMS}(k, N) &= \frac{1}{N}\sum_{i=1}^{N} r^T(k+i)r(k+i) \\ &= \frac{1}{N}\sum_{i=1}^{N} d^T(k+i)M^T\left(I - A\left(A^T A\right)^{-1} A^T\right)Md(k+i) \end{aligned} \tag{4.56}$$

$$J_{peak} = r^T(k)r(k) = d^T(k)M^T\left(I - A\left(A^T A\right)^{-1} A^T\right)Md(k) \tag{4.57}$$

- threshold settings

$$J_{th,RMS} = \lambda_{\max}\delta_{d,RMS},\ J_{th,peak} = \lambda_{\max}\delta_{d,peak} \tag{4.58}$$

where $\lambda_{\max}$ is the maximum eigenvalue of $M^T\left(I - A\left(A^T A\right)^{-1} A^T\right)M$ and $\delta_{d,RMS}, \delta_{d,peak}$ denote the boundedness of the disturbance $d(k)$ defined by

$$\delta_{d,RMS} = \sup_d \frac{1}{N}\sum_{i=1}^{N} d^T(k+i)d(k+i),\ \delta_{d,peak} = \sup_{d,k} d^T(k)d(k).$$

Alternatively, we propose the following solution for this detection problem, which is analog to the study in Sects. 3.5.1 and 4.2.5. Let $\hat{x}$ be a weighted LS given by

$$\hat{x} = \left(A^T WA\right)^{-1} A^T Wy(k),\ W = M^{-T}M^{-1}. \tag{4.59}$$

We build the residual vector

$$r(k) = y(k) - A\hat{x} = \left(I - A\left(A^T W A\right)^{-1} A^T W\right) y(k)$$
$$= M\left(I - M^{-1}A\left(A^T W A\right)^{-1} A^T M^{-T}\right) d(k) \tag{4.60}$$

and define the residual evaluation functions

$$J_{RMS}(k, N) = \frac{1}{N}\sum_{i=1}^{N} \bar{r}^T(k+i)\bar{r}(k+i) \tag{4.61}$$

$$J_{peak} = \bar{r}^T(k)\bar{r}(k) \tag{4.62}$$

where $\bar{r}(k)$ is given by

$$\bar{r}(k) = \Sigma^{-1} U_1^T M^{-1} r(k)$$

with U_1, Σ from an SVD of $I - M^{-1}A\left(A^T W A\right)^{-1} A^T M^{-T}$

$$I - M^{-1}A\left(A^T W A\right)^{-1} A^T M^{-T} = U\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix} U^T,\ \Sigma > 0$$
$$\Sigma \in \mathcal{R}^{(m-n)\times(m-n)},\ U = \begin{bmatrix} U_1 & U_2 \end{bmatrix},\ U_1 \in \mathcal{R}^{m\times(m-n)}.$$

It is evident that

$$\bar{r}^T(k)\bar{r}(k) = d^T(k) U_1 U_1^T d(k).$$

As a result, the thresholds are set to be

$$J_{th,RMS} = \delta_{d,RMS},\ J_{th,peak} = \delta_{d,peak}. \tag{4.63}$$

4.5 Case Study

In our case study, the data-driven fault detection algorithms proposed in this chapter, namely Algorithms 4.5, 4.6 are applied to the laboratory system CSTH and the Simulink model of the three-tank benchmark system. It is assumed that the CSTH and the three-tank system are both working around a fixed operating point in the closed control loop.

4.5.1 Case Study on Laboratory System CSTH

In our tests on CSTH, 5 sensors are selected for consideration. They are: the water level in the stirred tank (L1), water temperature in the stirred tank (T1), water tem-

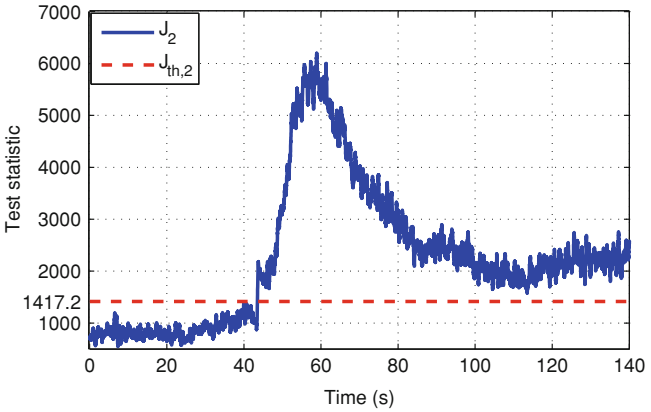


Fig. 4.1 Detection result of a fault in L1 sensor by means of J_2

perature in the heating jacket (T2), temperature of inflowing water to the stirred tank (T3) and water temperature in the inventory tank (T4). For our purpose, deterministic disturbances are injected.

Deterministic Disturbance on Sensor L1

Assume that there exists a deterministic disturbance in L1 sensor. The scenario is realized by adding a disturbance signal on the original samples of sensor L1. 5,000 samples have been collected as the training data, as the control loop works in the steady state. The step fault $f = 5$ is randomly generated and added to L1.

In the training phase, the raw data set $Y \in \mathcal{R}^{5\times 5,000}$ is centered to zero-mean firstly.

$$r = \min\{m = 5, k_d = 1\} = 1$$

is chosen a prior. After an SVD on YY^T,

$$J_{th,2} = 1.4172e + 003,\ J_{th,RMS} = 0.2834$$

are determined. Using the pre-specified $\epsilon = 0.025$, the online detection phase could be implemented as:

$$J_2(k, k+i) = \sum_{i=1}^{n} \bar{y}^T(k+i)\bar{y}(k+i)$$

$$J_{RMS}(k, n) = \frac{1}{N}\sum_{i=1}^{n} \bar{y}^T(k+i)\bar{y}(k+i).$$

Figures 4.1 and 4.2 show the detection results using J_2 and J_{RMS}, respectively.

Deterministic Disturbance on Sensor L1 and T1

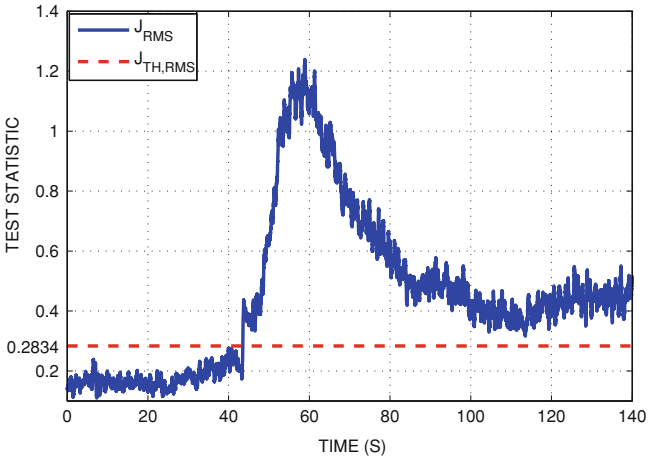


Fig. 4.2 Detection result of a fault in L1 sensor by means of J_{RMS}

For the second case, it is supposed that the deterministic disturbances exist both in T1 and L1 sensors. As the system reaches the steady state, 500 samples are collected and used as the training data. Again, a step fault f is simulated with a random generation.

Like the procedure in the previous case, the following steps are completed:

- The raw data set Y is centered to zero-mean
- $r = \min\{m = 5, K_d = 2\} = 2$ is chosen a prior
- After performing an SVD on YY^T,

$$J_{th,2} = 283.1599,\ J_{th,RMS} = 0.5652$$

are set.

The experimental results are shown in Figs. 4.3 and 4.4.

4.5.2 Case Study on Three-Tank System

The second case study is a simulation test on the three-tank system.

Deterministic Disturbance on Sensor 1

First, we consider the case when sensor 1 is disturbed by a deterministic disturbance. For the application of Algorithms 4.5, 4.6 a data set for training is formed from a simulation with a duration of 2,000 s (the system sampling time is 2 s), which is further centralized. It is followed by constructing Y and YY^T according to Algorithm 4.5. Doing an SVD on YY^T yields the threshold setting

$$J_{th,2} = 1.2946 \times 10^3,\ J_{th,RMS} = 1.2946.$$

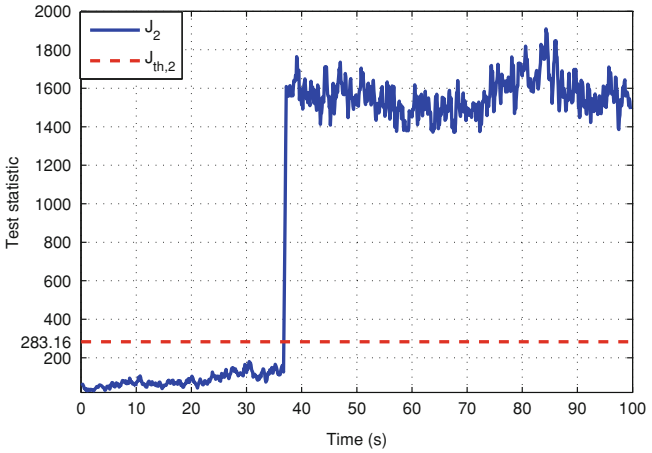


Fig. 4.3 Detection result of a fault in T1 and L1 sensors by means of J_2

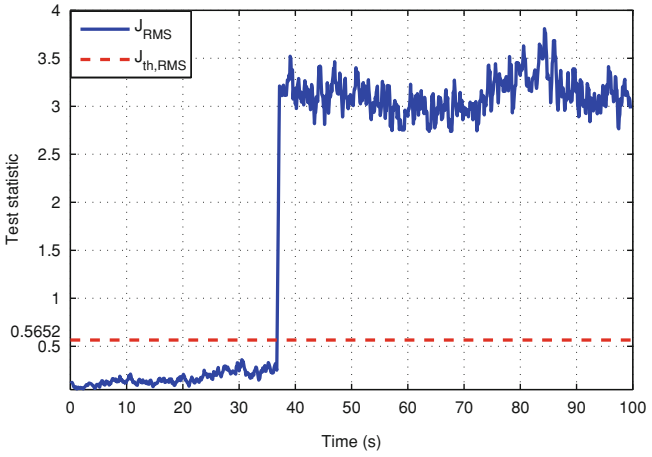


Fig. 4.4 Detection result of a fault in T1 and L1 sensors by means of J_{RMS}

For the online evaluation, the parameter ϵ is chosen as $\epsilon_{l2} = \epsilon_{RMS} = 0.04$.

In the simulation, an offset of 5 cm is added at $t = 1,200$ s as a sensor fault in Tank 2. Figures 4.5 and 4.6 are the corresponding test statistics and thresholds.

Deterministic Disturbance on Sensor 1 and Pump 2

Next, we simulate the case when sensor 1 and pump 2 are disturbed by two different deterministic disturbances respectively. Similar to the above simulation, the application of Algorithm 4.5 yields the threshold settings

$$J_{th,2} = 1.4976 \times 10^3,\ J_{th,RMS} = 1.4976.$$

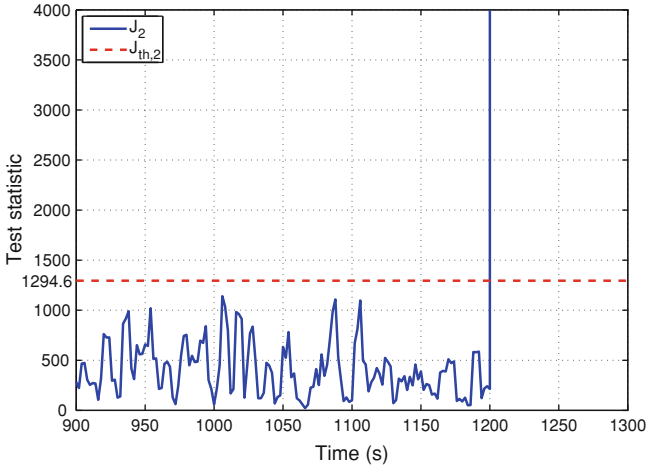


Fig. 4.5 J_2 test statistic and threshold $J_{th,2}$

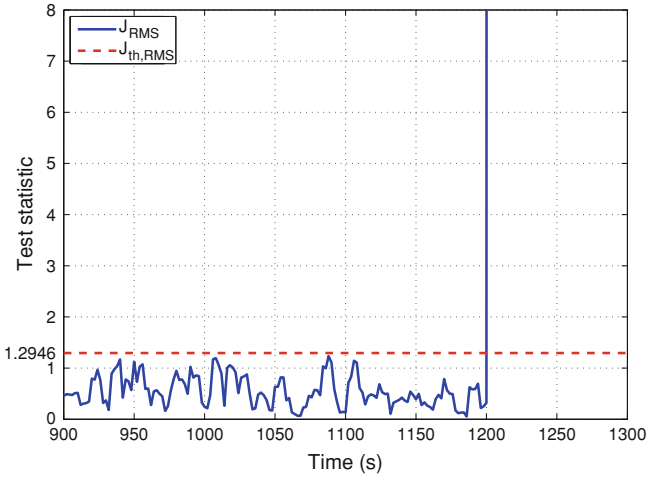


Fig. 4.6 J_{RMS} test statistic and threshold $J_{th,RMS}$

For the online evaluation, the parameter ϵ is chosen as $\epsilon_{l2} = 0.00025$ and $\epsilon_{RMS} = 0.0002$.

The same simulation with a sensor fault in Tank 2 is repeated. The achieved simulation results are given in Figs. 4.7 and 4.8.

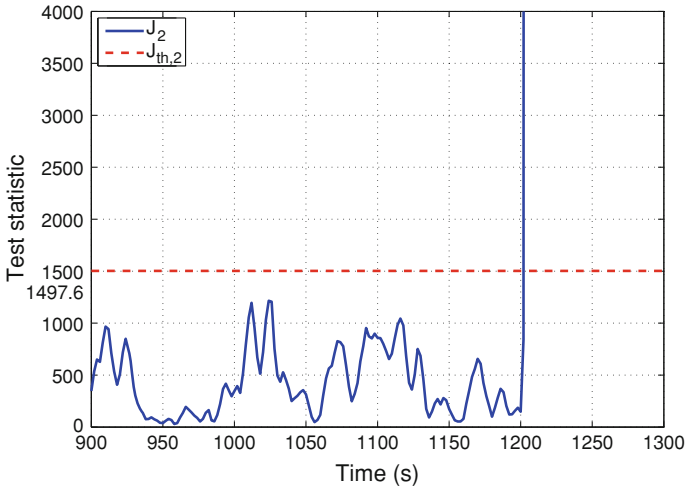


Fig. 4.7 J_2 test statistic and threshold $J_{th,2}$

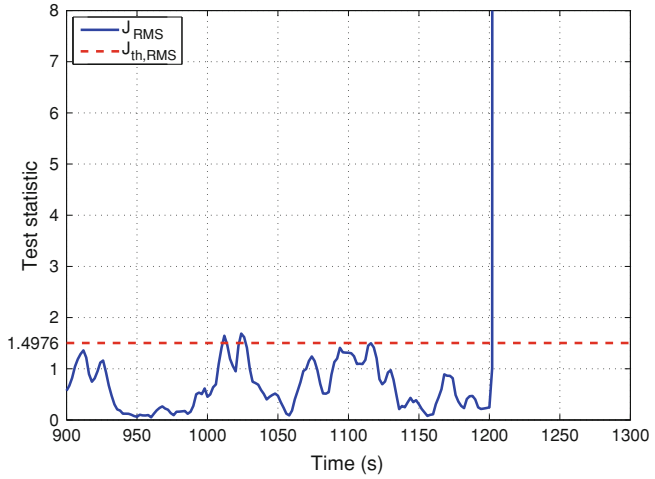


Fig. 4.8 J_{RMS} test statistic and threshold $J_{th,RMS}$

4.6 Notes and References

Study on model-based fault diagnosis in dynamic processes with deterministic disturbances builds the mainstream in the research domain and, correspondingly, there exists a great number of publications on the relevant topics [1–5]. In comparison, the topic addressed in this chapter, fault detection in static systems and processes, has received little attention. In fact, the well known and established $\mathcal{H}_\infty$ framework for threshold setting in solving fault detection problems in dynamics sys-

tems [2, 5] is a straightforward extension of the intuitive solution to the FD-P2. The study on the alternative and general solutions reveals that using the pseudo-inverse of the (disturbance) distribution matrix M as a weighting of the measurements leads to an improvement of fault detectability. The so-called unified solution scheme [5] is indeed a natural extension of this detection scheme to the case with dynamic systems. In the subsequent chapters, we mainly deal with process monitoring and diagnosis in statistic and stochastic processes. Nevertheless, considerable parts and results of our study can also be extended and applied to processes with deterministic disturbances, as illustrated in this chapter.

It is interesting to notice another interpretation of the pseudo-inverse of M, which yields an estimation of the disturbances (unknown inputs). Based on this interpretation, a new fault detection scheme, the so-called unknown input estimation based detection scheme, has been developed. It is the state of the art in the model-based fault detection framework that fault estimation is generally applied for the fault detection purpose. The idea behind the detection scheme based on unknown estimation is new, reasonable and promises an easy threshold setting.

The idea of integrating the norm-based threshold computation in a statistical framework, as shown in Definition 4.1, has been proposed in [5]. It allows to evaluate fault detection systems in a statistical framework, although the threshold computation is addressed in the norm-based framework. For this purpose, the so-called probabilistic robustness technique offers a powerful tool. The probabilistic robustness technique is a new research line that has recently emerged [6–9]. This technique allows to solve norm-based design and analysis problems in a probabilistic framework and opens a new and effective way to solve fault detection problems.

References

1. Blanke M, Kinnaert M, Lunze J, Staroswiecki M (2006) Diagnosis and fault-tolerant control, 2nd edn. Springer, Berlin Heidelberg
2. Chen J, Patton RJ (1999) Robust model-based fault diagnosis for dynamic systems. Kluwer Academic Publishers, Boston
3. Gertler JJ (1998) Fault detection and diagnosis in engineering systems. Marcel Dekker, New York, Basel
4. Patton RJ, Frank PM, Clark RN (eds) (2000) Issues of fault diagnosis for dynamic systems. Springer, London
5. Ding SX (2013) Model-based fault diagnosis techniques: design schemes, algorithms and tools, 2nd edn. Springer, London
6. Calafiore G, Dabbene F (2002) A probabilistic framework for problems with real structured uncertainty in systems and control. Automatica 38:1265–1276
7. Calafiore G, Dabbene F (2004) Control design with hard/soft performance specifications: a q-parameter randomization approach. Int J Control 77:461–471
8. Calafiore G, Polyak B (2001) Ransomized algorithms for probabilistic robustness with real and complex structured uncertainty. IEEE Trans Autom Control 45(12):2218–2235
9. Tempo R, Calafiro G, Dabbene F (2005) Randomized algorithms for analysis and control of uncertain systems. Springer, London

Part II
Application of Multivariate Analysis Methods to Fault Diagnosis in Static Processes

Chapter 5
Application of Principal Component Analysis to Fault Diagnosis

Principal component analysis (PCA) is a basic method of multivariate analysis (MVA) and plays, both in the research and application domains, an important role. It has been successfully used in numerous areas including data compression, feature extraction, image processing, pattern recognition, signal analysis and process monitoring . Thanks to its simplicity and efficiency in processing huge amount of process data, PCA is recognized as a powerful tool of statistical process monitoring and widely used in the process industry for fault detection and diagnosis. In research, PCA often serves as a basic technique for the development of advanced process monitoring and fault diagnosis techniques like recursive or adaptive PCA and kernel PCA. In this chapter, we shall first introduce the basic form of PCA applied to process monitoring and fault diagnosis as well as some of its variations. It is followed by the discussion on the relations between the PCA technique and the solutions for FD-P1 introduced in Chap. 3. We shall also remark some critical points with the application of PCA technique to fault detection.

5.1 The Basic Application Form of PCA to Fault Detection

PCA method is generally applied to solve fault detection problems in static (statistical) processes. We formulate the basic form of the saddressed problem as follows: Consider an m-dimensional process measurement vector

$$z_{obs} = \begin{bmatrix} z_{obs,1} \\ \vdots \\ z_{obs,m} \end{bmatrix} \in \mathcal{R}^m.$$

Suppose that N measurement data of this vector, $z_{obs}(1), \ldots, z_{obs}(N)$, have been collected and recorded. Find an algorithm to detect changes in the mean of the measurement vector during the online operation.

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_5, © Springer-Verlag London 2014

Since the computations in PCA are trivial and straightforward, we first give the offline (training) and online (detection) algorithms and then discuss the ideas behind them.

5.1.1 Algorithms

As a data-driven method, PCA consists of two parts: (i) data collection and training (ii) online detection. For our purpose, we summarize the computations in the first part in two algorithms, which can be, in real applications, integrated into one.

Algorithm 5.1 *Data normalization in PCA*

S1: Calculate

$$\bar{z}_{obs,i} = \frac{1}{N}\sum_{j=1}^{N} z_{obs,i}(j),\ \sigma^2_{obs,i} = \frac{1}{N-1}\sum_{j=1}^{N}\left(z_{obs,i}(j) - \bar{z}_{obs,i}\right)^2$$

and save $\bar{z}_{obs,i}, \sigma^2_{obs,i}, i = 1, \ldots, m$

S2: Form

$$z(k) = \begin{bmatrix} \frac{z_{obs,1}(k) - \bar{z}_{obs,1}}{\sigma_{obs,1}} \\ \vdots \\ \frac{z_{obs,m}(k) - \bar{z}_{obs,m}}{\sigma_{obs,m}} \end{bmatrix}, Z = \begin{bmatrix} z(1) \cdots z(N) \end{bmatrix} \in \mathcal{R}^{m\times N}.$$

Algorithm 5.2 *Threshold setting in PCA*

S1: Do an SVD on

$$\frac{1}{N-1}ZZ^T = P\Lambda P^T,\ \Lambda = diag\left(\sigma_1^2, \ldots, \sigma_m^2\right),\ \sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_m^2 \tag{5.1}$$

S2: Determine the number of principal components (PCs) l and decompose P, Λ *into*

$$\Lambda = \begin{bmatrix} \Lambda_{pc} & 0 \\ 0 & \Lambda_{res} \end{bmatrix},\ \Lambda_{pc} = diag\left(\sigma_1^2, \ldots, \sigma r_l^2\right) \tag{5.2}$$

$$\Lambda_{res} = diag\left(\sigma_{l+1}^2, \ldots, \sigma_m^2\right) \in \mathcal{R}^{(m-l)\times(m-l)},\ \sigma_l^2 >> \sigma_{l+1}^2$$

$$P = \begin{bmatrix} P_{pc}\ P_{res} \end{bmatrix} \in \mathcal{R}^{m\times m},\ P_{pc} \in \mathcal{R}^{m\times l} \tag{5.3}$$

and save Λ_{pc}, P_{pc}

S3: Set two thresholds,

$$SPE : J_{th,SPE} = \theta_1 \left(\frac{c_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_0 \left(h_0 - 1\right)}{\theta_1^2} \right)^{1/h_0} \tag{5.4}$$

$$T_{PCA}^2 : J_{th,T_{PCA}^2} = \frac{l\left(N^2 - 1\right)}{N(N-l)} F_\alpha(l, N-l) \tag{5.5}$$

$$\theta_i = \sum_{j=l+1}^{m} \left(\sigma_j^2\right)^i, i = 1, 2, 3, h_0 = 1 - \frac{2\theta_1 \theta_3}{3\theta_2^2} \tag{5.6}$$

for a (given) significance level α with c_α being the normal deviate corresponding to the upper $1 - \alpha$ percentile

---

For the (online) fault detection, the following algorithm runs after receiving a new measurement z_{obs}.

Algorithm 5.3 *PCA based fault detection*

S1: Normalize the measurement

$$z_i = \frac{z_{obs,i} - \bar{z}_{obs,i}}{\sigma_{obs,i}}, i = 1, \ldots, m$$

S2: Compute

$$SPE = \left\| \left(I - P_{pc} P_{pc}^T\right) z \right\|_E^2 = z^T \left(I - P_{pc} P_{pc}^T\right) z \tag{5.7}$$

$$T_{PCA}^2 = z^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T z \tag{5.8}$$

S3: Make a decision according to the detection logic

$$SPE \le J_{th,SPE} \text{ and } T_{PCA}^2 \le J_{th,T_{PCA}^2} \Longrightarrow \text{fault-free, otherwise faulty.}$$

5.1.2 Basic Ideas and Properties

The original idea behind the PCA is to reduce the dimension of a data set, while retaining as much as possible the variation present in the data set. The realization of this idea can be clearly seen from Algorithm 5.2. In Step S2, the decomposition of

P, Λ into P_{pc}, P_{res} as well as Λ_{pc}, Λ_{res} results in two subspaces in the m-dimensional measurement subspace. The subspace spanned by P_{pc}^T is called principal subspace, which is constructed by those eigenvectors corresponding to the larger singular values, $\sigma_1^2, \ldots, \sigma_l^2$, of the covariance matrix of the normalized measurement vector. That means the projection

$$\hat{z} = P_{pc}^T z \in \mathcal{R}^l \tag{5.9}$$

delivers a (much) lower dimensional vector $\hat{z}$ whose covariance matrix is

$$\mathcal{E}\left(P_{pc}^T z z^T P_{pc}\right) = P_{pc}^T \mathcal{E}\left(zz^T\right) P_{pc} = P_{pc}^T P \Lambda P^T P_{pc} = \Lambda_{pc} \tag{5.10}$$

that is, it retains the major (principal) variation and thus can be viewed as the information carrier. In against, the covariance matrix of the projection onto the residual subspace,

$$z_{res} = P_{res}^T z \in \mathcal{R}^{m-l},\ \mathcal{E}\left(z_{res} z_{res}\right) = P_{res}^T \mathcal{E}\left(zz^T\right) P_{res} = \Lambda_{res} \tag{5.11}$$

is (significantly) smaller and thus z_{res} can be, in view of its information content, neglected.

Let us now consider the application of the PCA technique to fault detection, as described in the above three algorithms. To this end, two test statistics, T_{PCA}^2 and *SPE* (squared prediction error), are defined, which are formed by means of the projections $\hat{z} = P_{pc}^T z$ and $z_{res} = P_{res}^T z$, respectively. In fact, in the fault detection practice *SPE* is often preferred, as will be illustrated in the subsequent discussion. This observation motivates us to review the application of the PCA technique to fault detection from the following aspects:

- Assumption and problem formulation: Although it is often not explicitly mentioned that

$$z \sim \mathcal{N}(\mathcal{E}(z),\ \Sigma_z) \tag{5.12}$$

  it becomes evident from the test statistics and their threshold setting that the application of the PCA technique to fault detection is based on the assumption (5.12). In this context, it is clear that the fault detection problem addressed by the PCA technique is exactly the one formulated as the data-driven version of FD-P1 introduced in Chap. 3. Note that on the assumption (5.12)

$$\begin{aligned}
&\hat{z} = P_{pc}^T z \sim \mathcal{N}(0,\ \Lambda_{pc}),\ z_{res} = P_{res}^T z \sim \mathcal{N}(0,\ \Lambda_{res}) \Longrightarrow \\
&\hat{z}^T \Lambda_{pc}^{-1} \hat{z} \sim \frac{l\left(N^2-1\right)}{N(N-l)} F(l, N-l) \\
&z_{res}^T \Lambda_{res}^{-1} z_{res} \sim \frac{(m-l)\left(N^2-1\right)}{N(N-m+l)} F(l, N-m+l)
\end{aligned}$$

or for a sufficiently large N

$$\hat{z}^T \Lambda_{pc}^{-1} \hat{z} \sim \mathcal{X}^2(l), z_{res}^T \Lambda_{res}^{-1} z_{res} \sim \mathcal{X}^2(m-l).$$

- Estimation of the (normalized) covariance matrix: In most studies, less attention has been paid to the data normalization, as given in Algorithm 5.1. This step delivers an estimation of the (normalized) covariance matrix Σ_z. As we have discussed in Sect. 3.3, in the data-driven fault detection framework, this step plays a central role.
- SVD and inverse of the covariance matrix: The SVD of the (estimated) covariance matrix given in *Step S2* of Algorithm 5.2 is the core of the PCA technique. On the other hand, in comparison with the data-driven solutions of the FD-P1, the SVD of the covariance matrix serves, indeed, as a numerical solution for the inverse computation of the covariance matrix, which is necessary for solving the FD-P1. This point can be seen clearly from the test statistic (5.8). In fact, the test statistic *SPE* can also be, theoretically, formulated as

$$SPE = z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z \tag{5.13}$$

  as far as Λ_{res} is regular.
- The major differences: the major differences between the PCA and the Hotelling's T^2 test statistic lie in (i) the treatment of the projection onto the residual subspace (ii) decomposition of the measurement subspaces or, equivalently, the introduction of two indices (in the PCA framework) instead of a single one (the Hotelling's T^2 test statistic).

In the next sections, we are going to address these differences and introduce some modified forms of the PCA technique applied for fault diagnosis.

5.2 The Modified Form of *SPE*: Hawkin's T_H^2 Statistic

Consider the modified form of *SPE* defined by (5.13), which is called Hawkin's T_H^2 statistic, that is

$$T_H^2 = z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z.$$

Due to possible ill-conditioning Λ_{res}, when some of the singular values of $\sigma_{l+1}, \ldots, \sigma_m$ are too close to zero, Hawkin's T_H^2 statistic is less used in practice. To avoid this difficulty, the following alternative test statistic can be used. Let

$$\Xi = diag\left(\frac{\sigma_m^2}{\sigma_{l+1}^2}, \ldots, \frac{\sigma_m^2}{\sigma_{m-1}^2}, 1\right) \in \mathcal{R}^{(m-l)\times(m-l)}$$

It turns out that

$$\bar{z}_{res} = \Xi^{1/2} z_{res} = \Xi^{1/2} P_{res}^T z \sim \mathcal{N}(0, \sigma_m^2 I_{(m-l)\times(m-l)})$$

and moreover

$$\bar{z}_{res}^T \bar{z}_{res} = z_{res}^T \Xi z_{res} = \sigma_m^2 z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z = \sigma_m^2 T_H^2.$$

Since $z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z$ is $\mathcal{X}^2$ -distributed with $m - l$ degrees of freedom, defining

$$T_{new}^2 = \bar{z}_{res}^T \bar{z}_{res} = z_{res}^T \Xi z_{res} = z^T P_{res} \Xi P_{res}^T z \tag{5.14}$$

as the test statistic results in a threshold setting, for a given significance level α, equal to

$$J_{th,T_{new}^2} = \sigma_m^2 \frac{(m-l)\left(N^2-1\right)}{N(N-m+l)} F_\alpha(l, N-m+l) \tag{5.15}$$

$$\text{or for a large } N \; J_{th,T_{new}^2} = \sigma_m^2 \mathcal{X}_\alpha^2(m-l). \tag{5.16}$$

We would like to point out that

- the new statistic T_{new}^2 is $\mathcal{X}^2$-distributed (for a sufficiently large N), and
- different from the *SPE*, the corresponding threshold can be exactly determined using the available $\mathcal{X}^2$ data table as shown in (5.15), and
- the associated computation is (considerably) simplified in comparison with the computation of $J_{th,SPE}$ given in (5.4).

5.3 Fault Sensitivity Analysis

In this section, we analyze the fault sensitivity of the test statistics introduced in the last two sections. For the purpose of simplifying the study, it is assumed that the number of the data N is sufficiently large.

Off-set and scaling (multiplicative) faults are two types of faults which are mostly considered both in the theoretic study and practical applications. Given a measurement sample z, the off-set and scaling faults are respectively defined as

$$z = z_o + f, f \neq 0 \Longrightarrow \text{off-set fault}$$
$$z = Fz_o, F \neq I \Longrightarrow \text{scaling fault}$$

where z_o represents the sample in the fault-free case and $f \in \mathcal{R}^m$ a (non-zero) constant fault vector. Next, we study the test statistics applied to detect off-set and scaling faults.

Note that the *SPE*, T_H^2 and T_{new}^2 statistics consist of (different) quadratic form computation of vector $z_{res} = P_{res}^T z$, while the T_{PCA}^2 statistic is based on $z_{pc} = P_{pc}^T z$. To simplify the notation, we shall use in the sequent study T_{res}^2 to represent the statistics associated with $z_{res} = P_{res}^T z$, that is *SPE*, T_H^2 and T_{new}^2 statistics.

5.3.1 Sensitivity to the Off-set Faults

Recall that the covariance matrices of z_{pc}, z_{res} are respectively Λ_{pc} and Λ_{res}, furthermore

$$\sigma_{\min}\left(\Lambda_{pc}\right) = \sigma_l^2 >> \sigma_{\max}\left(\Lambda_{res}\right) = \sigma_{l+1}^2.$$

As a result, the T_{res}^2 statistics can be (significantly) more sensitive to the off-set fault than the T_{PCA}^2 statistic, as pointed out in Chap. 3. Below, we demonstrate it. To this end, let us consider the Hawkin's T_H^2 statistic and T_{PCA}^2 statistic. Recall that

$$J_{th,T_H^2} = \mathcal{X}_\alpha^2(m-l), J_{th,T_{PCA}^2} = \mathcal{X}_\alpha^2(l).$$

Hence, if a fault f causes

$$\left(f^T P_{res} P_{res}^T f\right)^{1/2} > 2\sigma_{l+1} J_{th,T_H^2}^{1/2} = 2\sigma_{l+1}\left(\mathcal{X}_\alpha^2(m-l)\right)^{1/2} \tag{5.17}$$

we have

$$\begin{aligned}\left(z^T P_{res}\Lambda_{res}^{-1}P_{res}^T z\right)^{1/2} &\geq \left(f^T P_{res}\Lambda_{res}^{-1}P_{res}^T f\right)^{1/2} - \left(z_o^T P_{res}\Lambda_{res}^{-1}P_{res}^T z_o\right)^{1/2} \\ &\geq \sigma_{l+1}^{-1/2}\left(f^T P_{res}P_{res}^T f\right)^{1/2} - J_{th,T_H^2}^{1/2} > J_{th,T_H^2}^{1/2} \Longrightarrow \text{ faulty.}\end{aligned}$$

That means, this fault can be detected. Note that (5.17) is a sufficient condition that a fault can be detected. On the other hand, we have

$$\begin{aligned}\mathcal{E}\left(T_{PCA}^2\right) &= \mathcal{E}\left(z^T P_{pc}\Lambda_{pc}^{-1}P_{pc}^T z\right) = \mathcal{E}\left(z_o^T P_{pc}\Lambda_{pc}^{-1}P_{pc}^T z_o\right) + f^T P_{pc}\Lambda_{pc}^{-1}P_{pc}^T f \\ &\leq \mathcal{E}\left(z_o^T P_{pc}\Lambda_{pc}^{-1}P_{pc}^T z_o\right) + \sigma_l^{-2} f^T P_{pc}P_{pc}^T f \\ &= l + \frac{\sigma_{l+1}^2 f^T P_{pc}P_{pc}^T f \cdot \sigma_{l+1}^{-2} f^T P_{res}P_{res}^T f}{\sigma_l^2 f^T P_{res}P_{res}^T f}.\end{aligned}$$

Assume that

$$f^T P_{pc}P_{pc}^T f \approx f^T P_{res}P_{res}^T f \tag{5.18}$$

$$\sigma_{l+1}^{-2} f^T P_{res}P_{res}^T f = 4J_{th,T_H^2} + \delta \Longrightarrow \left(f^T P_{res}P_{res}^T f\right)^{1/2} > 2\sigma_{l+1} J_{th,T_H^2}^{1/2} \tag{5.19}$$

with $\delta > 0$. It turns out

$$\mathcal{E}\left(z^T P_{pc}\Lambda_{pc}^{-1}P_{pc}^T z\right) \leq l + \frac{\sigma_{l+1}^2\left(4\mathcal{X}_\alpha^2(m-l) + \delta\right)}{\sigma_l^2}$$

For

$$\frac{\sigma_{l+1}^2}{\sigma_l^2} \leq \frac{\mathcal{X}_\alpha^2(l) - l}{4\mathcal{X}_\alpha^2(m-l) + \delta} \tag{5.20}$$

we finally have

$$\mathcal{E}\left(z^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T z\right) \leq \mathcal{X}_\alpha^2(l) = J_{th,T_{PCA}^2}. \tag{5.21}$$

It follows from (5.21) that f is expectably undetected under condition (5.20), although it is detectable using the Hawkin's T_H^2 (See 5.19) and for $f^T P_{pc} P_{pc}^T f \approx f^T P_{res} P_{res}^T f$.

The previous analysis reveals that a statistic associated with $P_{res}^T z$ is more sensitive to an off-set fault than the T_{PCA}^2 statistic, when the fault has the same influence on the both test statistics, that is $f^T P_{pc} P_{pc}^T f \approx f^T P_{res} P_{res}^T f$.

5.3.2 Sensitivity to the Scaling Faults

For our purpose, we now compare the Hawkin's T_H^2 statistic and T_{PCA}^2 statistics in detecting scaling faults. To simplify the study, it is assumed that the scaling fault is modelled by

$$F = \alpha_f I, \alpha_f \geq 1, \text{ and } \alpha_f > 1 \Longrightarrow \text{faulty}.$$

It turns out

$$\begin{aligned} T_{PCA}^2 &= x^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T x = \alpha_f^2 z_o^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T z_o \\ T_H^2 &= z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z = \alpha_f^2 z_o^T P_{res} \Lambda_{res}^{-1} P_{res}^T z_o. \end{aligned}$$

Defining the fault sensitivity by

$$S_H = \frac{\max T_H^2 \,|_{\text{faulty}}}{\max T_H^2 \,|_{\text{fault-free}}}, S = \frac{\max T_{PCA}^2 \,|_{\text{faulty}}}{\max T_{PCA}^2 \,|_{\text{fault-free}}}$$

we have

$$S_H = \frac{\alpha_f^2 J_{th,T_H^2}}{J_{th,T_H^2}} = \alpha_f^2, S = \frac{\alpha_f^2 J_{th,T_{PCA}^2}}{J_{th,T_{PCA}^2}} = \alpha_f^2.$$

This analysis demonstrates that the Hawkin's T_H^2 statistic and T_{PCA}^2 statistic offer the similar performance in detecting scaling faults.

5.4 Multiple Statistical Indices and Combined Indices

Recall that one of the differences between the PCA based fault detection and the Hotelling's T^2 test statistic is the number of the test statistics. In this section, we study the use of a single or multiple statistics (indices).

Note that

$$\mathcal{X}^2_\alpha(m) > \mathcal{X}^2_\alpha(l), \mathcal{X}^2_\alpha(m) > \mathcal{X}^2_\alpha(m-l).$$

In case that the possible faults have a certain distribution in the measurement subspace a single index like the Hotelling's T^2 test statistic may be of a lower fault sensitivity than the separate use of T^2_H, T^2_{PCA}. To illustrate this, we consider those faults satisfying $P^T_{pc} f = 0$. In this case, f will cause the same change in T^2_H and T^2. Recall that the threshold corresponding to T^2 is considerably higher than the one associated with T^2_H. Thus, T^2_H is much more sensitive to these faults than the T^2 statistic.

The result may become different, when no information about possible faults is available. For this case, faults can be, for instance, described by

$$f = \gamma P \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{5.22}$$

where $\gamma > 0$ is some unknown constant and P is defined in (5.3). Expression (5.22) means that the faults may occur identically in all possible directions in the measurement subspace. It turns out

$$\begin{aligned} T^2_{PCA} &= \gamma \sum_{i=1}^{l} \sigma_i^{-2}, T^2_H = \gamma \sum_{i=l+1}^{m} \sigma_i^{-2} \\ T^2 &= \gamma \sum_{i=1}^{m} \sigma_i^{-2} = T^2_{PCA} + T^2_H. \end{aligned} \tag{5.23}$$

On the other hand, for

$$\mathcal{X}^2_\alpha(m) < \mathcal{X}^2_\alpha(l) + \mathcal{X}^2_\alpha(m-l)$$

we are able to claim that in this case the use of the single index T^2 delivers a higher fault detectability than the use of two indices T^2_{PCA} and T^2_H. This conclusion can be illustrated by the following academic example.

Example 5.1 Given $m = 10, l = 5$ and

$$\Lambda = \begin{bmatrix} \Lambda_{pc} & 0 \\ 0 & \Lambda_{res} \end{bmatrix}, \sigma_1^2 = \cdots = \sigma_5^2 = 0.8^{-1}, \sigma_6^2 = \cdots = \sigma_{10}^2 = 1.2^{-1}$$

and assume that

$$f = P \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathcal{R}^{10}.$$

Since for $\alpha = 5\,\%$,

$$\mathcal{X}^2_\alpha(10) = 18.3070,\ \mathcal{X}^2_\alpha(5) = 11.0705$$

we have

$$\mathcal{E}\left(z^T P \Lambda^{-1} P^T z\right) = 10 + \sum_{i=1}^{10} \sigma_i^{-2} = 20.0000 > \mathcal{X}^2_\alpha(10)$$

$$\mathcal{E}\left(z^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T z\right) = 5 + \sum_{i=1}^{5} \sigma_i^{-2} = 9.0000 < \mathcal{X}^2_\alpha(5)$$

$$\mathcal{E}\left(z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z\right) = 5 + \sum_{i=6}^{10} \sigma_i^{-2} = 11.0000 < \mathcal{X}^2_\alpha(5).$$

That means, using the single index T^2 a fault detection can be expected, while using two indices, T_H^2, T_{PCA}^2, as applied by the PCA technique, will expectably not lead to a detection. Indeed, the following ratios

$$\frac{f^T P \Lambda^{-1} P^T f}{\mathcal{X}^2_\alpha(10)} = \frac{10}{18.3070} = 0.5462 > \frac{f^T P_{res} \Lambda_{res}^{-1} P_{res}^T f}{\mathcal{X}^2_\alpha(5)} = \frac{6}{11.0705} = 0.5420$$
$$> \frac{f^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T f}{\mathcal{X}^2_\alpha(5)} = \frac{4}{11.0705}$$

which measure the influence of the fault on the test statistic in comparison with the threshold setting, also reveal the advantage of using the single index T^2 for enhancing the fault detectability.

The previous discussion reveals that

- the use of a single index may be of advantage when the distribution of f is nearly uniform in the measurement subspace and
- the separate use of two test statistics, as applied in the PCA fault detection technique, will improve the fault sensitivity if it can be assumed that

$$f^T P_{pc} P_{pc}^T f >> f^T P_{res} P_{res}^T f \text{ or } f^T P_{res} P_{res}^T f >> f^T P_{pc} P_{pc}^T f$$

that is if additional information about the fault or its distribution is available.

It is interesting to notice that the Hotelling's T^2 test statistic can be written in the form of a (linear) combination of T_H^2, T_{PCA}^2 statistics,

$$T^2 = z^T P \Lambda^{-1} P^T z = z^T P_{pc} \Lambda_{pc}^{-1} P_{pc}^T z + z^T P_{res} \Lambda_{res}^{-1} P_{res}^T z = T_{PCA}^2 + T_H^2.$$

Qin has introduced the concept of combined indices, which are generally formulated as

$$T_c^2 = \beta_1 T_{PCA}^2 + \beta_2 T_{res}^2 \tag{5.24}$$

with known constants $\beta_1, \beta_2 > 0$ and T_{res}^2 representing either T_H^2 or SPE or T_{new}^2. Rewriting (5.24) into

$$T_c^2 = x^T P \Phi P^T x, \Phi = \begin{bmatrix} \beta_1 \Lambda_{pc}^{-1} & 0 \\ 0 & \beta_2 Q \end{bmatrix}$$

makes it clear that a combined index is a quadratic statistic of signal $P^T z$, where

$$Q = \begin{cases} \Lambda_{res}^{-1}, T_{res}^2 = T_H^2 \\ I, T_{res}^2 = SPE \\ \Xi, T_{res}^2 = T_{new}^2 \end{cases}.$$

Since $P^T z \sim \mathcal{N}(0, \Lambda)$, $z^T P \Lambda^{-1} P^T z$ is $\mathcal{X}^2$-distributed with m degrees of freedom. This fact motivates us to introduce the following (combined) statistic.

Considering the possible numerical difficulty with the computation of Λ^{-1}, we propose

$$T_{c,new}^2 = z^T P \bar{\Xi} P^T z, \bar{\Xi} = diag\left(\frac{\sigma_m^2}{\sigma_1^2}, \ldots, \frac{\sigma_m^2}{\sigma_{m-1}^2}, 1\right) \tag{5.25}$$

which is a combined index given by

$$T_{c,new}^2 = z^T P \bar{\Xi} P^T z = \sigma_m^2 T_{PCA}^2 + T_{new}^2 = \sigma_m^2 \left(T_{PCA}^2 + T_H^2\right). \tag{5.26}$$

It is clear that the corresponding threshold is

$$J_{th,c,new} = \sigma_m^2 \mathcal{X}_\alpha^2(m)$$

for a given significance level α.

5.5 Dynamic PCA

The dynamic PCA (DPCA) is a natural extension of the PCA to dealing with those processes with (obvious) dynamic behavior, which can be roughly expressed in terms of the serial correlations between the actual observation vector with the previous, for example, h observations. To this end, let the observation vector at the time instant k be $z(k)$. Assume that for the training purpose observations in the time interval $[k-N, k]$ are available. Let

$$z_k(h) = \begin{bmatrix} z(k) \\ \vdots \\ z(k-h) \end{bmatrix} \in \mathcal{R}^{m(h+1)}.$$

The data are formed into

$$Z_k(h) = \left[z_{k-N+h}(h) \cdots z_k(h) \right] \in \mathcal{R}^{m(h+1)\times(N-h+1)}. \tag{5.27}$$

The remaining offline and online steps are then identical with the ones given in the standard PCA. Note that for the online computation of the T^2_{PCA} and SPE indices at the time instant j, the (extended) observation vector $z_j(h)$ is needed.

It should be pointed out that the DPCA is only applicable to dynamic processes in steady state or stationary stochastic processes. That means, both the mean value and the covariance matrix of the measurement vector should be constant. In fact, this is the pre-requirement for applying Algorithm 5.1. In Part III, we shall introduce fault detection schemes for dynamics processes with transient behavior.

5.6 Fault Identification

Below, we shall first study the problems of identifying off-set and scaling faults respectively, and then present a general procedure for fault identification.

5.6.1 Identification of Off-set Faults

Recall that given

$$z = \varepsilon + f \in \mathcal{R}^m, \varepsilon \sim \mathcal{N}(0, \Sigma)$$

with known Σ and unknown constant vector f, an LR estimate for f is delivered by

$$\hat{f} = \frac{1}{M}\sum_{i=1}^{M} z(i)$$

where $z(i), i = 1, \ldots, M$, are the M data collected from the process. Now, suppose that using the standard PCA method a fault is detected with sample $z(k)$. We propose to use the following algorithm for identifying the detected fault f.

Algorithm 5.4 *Identification of off-set faults*

S1: Collect further M data and calculate

$$z(k) = z_{obs}(k) - \bar{z}_{obs}, \ldots, z(k+M) = z_{obs}(k+M) - \bar{z}_{obs} \tag{5.28}$$

where $z_{obs}(k), \ldots, z_{obs}(k+M)$ denote the measurements and $\bar{z}_{obs}$ is the mean value achieved using the training data, as described in Algorithm 5.1

S2: Compute the estimate of f

$$\hat{f} = \frac{1}{M+1}\sum_{i=k}^{k+M} z(i). \tag{5.29}$$

It is worth to mention that the above algorithm can also be realized in a recursive manner.

5.6.2 Identification of Scaling Faults

Consider

$$z = Fz_o, F \neq I \Longrightarrow \text{scaling fault}$$

and suppose that a fault has been detected using the standard PCA method. Let $z(k), \ldots, z(k+M)$ be $M+1$ centered data collected after the fault has been detected. We then have

$$\frac{1}{M}\left[z^T(k) \cdots z^T(k+M)\right]\begin{bmatrix} z(k) \\ \vdots \\ z(k+M) \end{bmatrix} \approx F\Sigma F^T = FP\Lambda P^T F^T$$

An SVD of

$$\frac{1}{M}\left[z^T(k) \cdots z^T(k+M)\right]\begin{bmatrix} z(k) \\ \vdots \\ z(k+M) \end{bmatrix} = V\Pi V^T, \Pi = diag\left(\bar{\sigma}_1^2, \ldots, \bar{\sigma}_m^2\right)$$

leads to

$$V\Pi^{1/2} = FP\Lambda^{1/2}$$

On the assumption that $\Lambda_{res} \approx 0$, the pseudo-inverse of $P\Lambda^{1/2}$ is given by

$$\left(P\Lambda^{1/2}\right)^{-} \approx \begin{bmatrix} \Lambda_{pc}^{-1/2} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P_{pc}^T \\ P_{res}^T \end{bmatrix}.$$

Therefore,

$$\hat{F} = V_1 \Pi_{pc}^{1/2} \Lambda_{pc}^{-1/2} P_{pc}^T \tag{5.30}$$

with

$$\Pi_{pc}^{1/2} = diag\left(\bar{\sigma}_1, \ldots, \bar{\sigma}_l\right), V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}, V_1 \in \mathcal{R}^{m \times l}$$

delivers a reasonable estimate for F.

5.6.3 A Fault Identification Procedure

In this subsection, we summarize the major results achieved in the previous subsections into the following algorithm aiming at identifying the faults. We assume that a fault has been detected using T_{PCA}^2 and T_{res}^2 test statistics.

Algorithm 5.5 *Identification of off-set and scaling faults*

S1: Run Algorithm 5.4 for $\hat{f}$
S2: Re-center the data

$$\bar{z}(k) = z(k) - \hat{f}, \ldots, \bar{z}(k+M) = z(k+M) - \hat{f} \tag{5.31}$$

S3: Do an SVD

$$\frac{1}{M} \begin{bmatrix} \bar{z}^T(k) & \cdots & \bar{z}^T(k+M) \end{bmatrix} \begin{bmatrix} \bar{z}(k) \\ \vdots \\ \bar{z}(k+M) \end{bmatrix} = V\Pi V^T$$

S4: Computing $\hat{F}$ *using (5.30).*

5.7 Application to TEP

We now demonstrate the application of PCA technique to fault diagnosis on the TEP simulator. There are 6 defined operating modes for the TEP. In this application study, the process is running under mode number 1 and the decentralized control strategy is adopted. The detailed simulation conditions are given in Sect. 2.3. In the following, we will first apply PCA based fault diagnosis technique to the fault scenario 4 aiming at providing the reader with the basic procedure of applying PCA technique to fault diagnosis in an industrial process. It is followed by fault detection charts for some other fault scenarios.

5.7.1 Case Study on Fault Scenario 4

Before applying PCA for fault diagnosis, the process variables must be defined. In this case study, the involved process variables are listed in Table 5.1. The manipulated variables $XMV(5)$, $XMV(9)$ and $XMV(12)$ are excluded, since they are constant for the current setup and will provide no useful information for fault diagnosis and, in addtion, may cause numerical difficulties. Moreover, the process measurements XMEAS(1)–XMEAS(22) are included, which contain operational information about the process. The intermediate and final quality measurements XMEAS(23)–XMEAS(41), which have much larger sampling interval, are not included.

Offline Training

In the normal operating condition, 960 samples of data are collected for training. This data set is denoted by $Z_{obs,offline} \in \mathcal{R}^{m\times N}$, where each row represents one variable and each column one sample. By applying Algorithm 5.1, the normalized data set $Z_{offline} \in \mathcal{R}^{m\times N}$ can be obtained. Based on it, the training procedure given in Algorithm 5.2 is carried out to extract the needed parameters. For PCA based fault diagnosis, selection of the number of PCs plays an important role in practice. In this case study 16 PCs are retained, which contains 81.95 % of the total variance. Based on a significance level of $\alpha = 0.05$, the thresholds for T^2 and SPE test statistics are 26.9162 and 10.5770, respectively.

Validation of Offline Training Result

Before using the offline trained parameters for the online detection purpose, a validation procedure is necessary. Here we choose the significance level α for the validation. The theoretical value is specified by users to be 0.05. One experiment is carried out in the fault-free scenario. Based on the 960 samples of process data, the calculated significance level for T^2 and SPE statistics are 0.0417 and 0.0407, respectively.

Online Running

For the fault scenario, the length of test is 48 h and the fault happens at the 8 h. Denote the online data as $z_{obs,online}(k) \in \mathcal{R}^m, k = 1, \ldots, 960$, where

Table 5.1 Process variables involved in the PCA model

Tag	Description
XMV(1)	D feed flow
XMV(2)	E feed flow
XMV(3)	A feed flow
XMV(4)	A and C feed flow
XMV(6)	Purge valve
XMV(7)	Separator pot liquid flow
XMV(8)	Stripper liquid product flow
XMV(10)	Reactor cooling water flow
XMV(11)	Condenser cooling water flow
XMEAS(1)	A feed (stream 1)
XMEAS(2)	D feed (stream 2)
XMEAS(3)	E feed (stream 3)
XMEAS(4)	A and C feed
XMEAS(5)	Recycle flow
XMEAS(6)	Reactor feed rate
XMEAS(7)	Reactor pressure
XMEAS(8)	Reactor level
XMEAS(9)	Reactor temperature
XMEAS(10)	Purge rate
XMEAS(11)	Separator temperature
XMEAS(12)	Separator level
XMEAS(13)	Separator pressure
XMEAS(14)	Separator underflow
XMEAS(15)	Stripper level
XMEAS(16)	Stripper pressure
XMEAS(17)	Stripper underflow
XMEAS(18)	Stripper temperature
XMEAS(19)	Stripper steam flow
XMEAS(20)	Compressor work
XMEAS(21)	Reactor water temperature
XMEAS(22)	Separator water temperature

$z_{obs,online} = [z_1, \ldots, z_m]^T$. By applying Algorithm 5.3, the fault detection results are obtained and plotted in Fig. 5.1. Both the T^2 and *SPE* statistics have successfully detected the fault after its occurrence.

Analysis of the Online Running Results

Fault scenario 4 is caused by a step change in the reactor cooling water inlet temperature. Since the TEP is running in the closed-loop configuration, a direct influence of the fault is on the 10th manipulated variable $XMV(10)$, which is the reactor cooling water flow. From Fig. 5.2 we can see that there is an increase in the flow after the fault happens at the 8 h. In addition, this fault has influenced the process variables $XMEAS(9)$ (reactor temperature) and $XMEAS(21)$ (reactor cooling water outlet

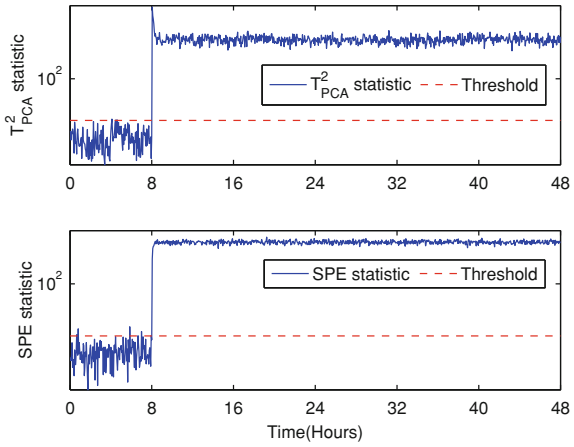


Fig. 5.1 PCA based fault detection results for fault scenario 4

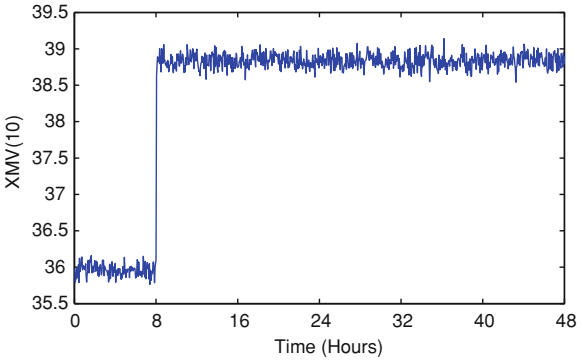


Fig. 5.2 Faulty manipulated variable: XMV(10)

temperature) for a short time, which are depicted in Figs. 5.3 and 5.4. According to the detection results, the PCA based technique has successfully extracted these fault symptoms and raised an alarm in time.

5.7.2 Case Study Results for the Other Fault Scenarios

This subsection includes the plots of detection results for the fault scenarios 1, 6, 11, 20. It is remarkable that for fault scenario 6, the process is shut down after about 15 h because of low stripper liquid level (Figs. 5.5, 5.6).

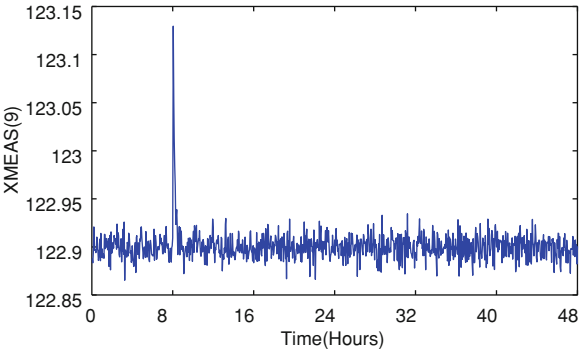


Fig. 5.3 Faulty process variable: XMEAS(9)

Fig. 5.4 Faulty process variable: XMEAS(21)

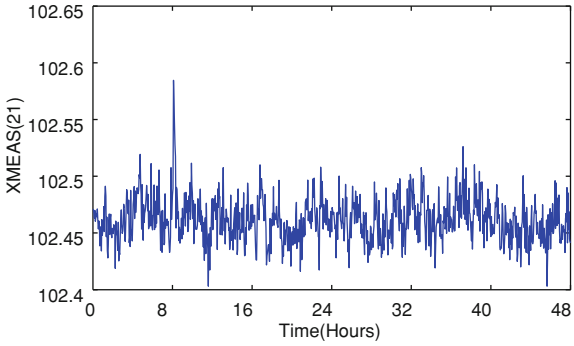
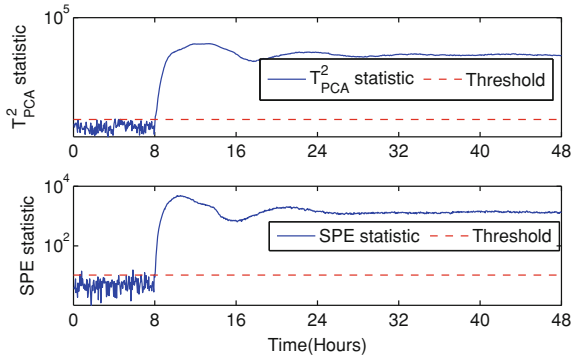


Fig. 5.5 PCA based fault detection results for fault scenario 1



5.7.3 Comparison of Multiple Indices with Combined Indices

This case study is carried out for the fault scenario 12, where a random variation in the condenser cooling water inlet temperature has happened (Figs. 5.7, 5.8).

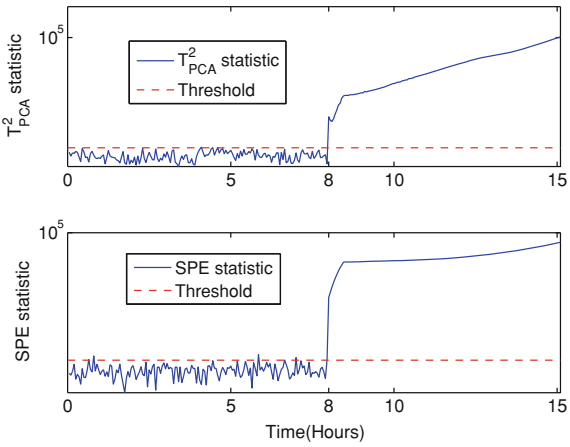


Fig. 5.6 PCA based fault detection results for fault scenario 6

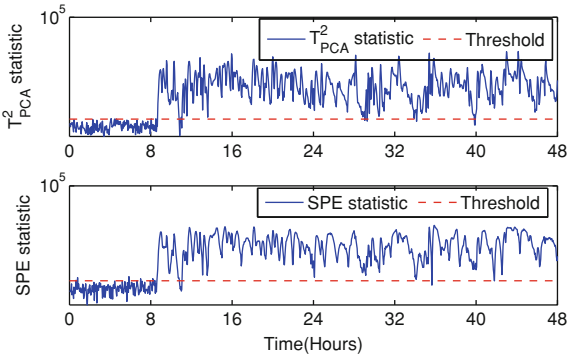


Fig. 5.7 PCA based fault detection results for fault scenario 11

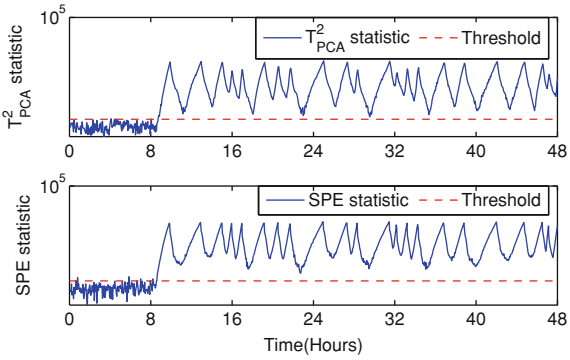


Fig. 5.8 PCA based fault detection results for fault scenario 20

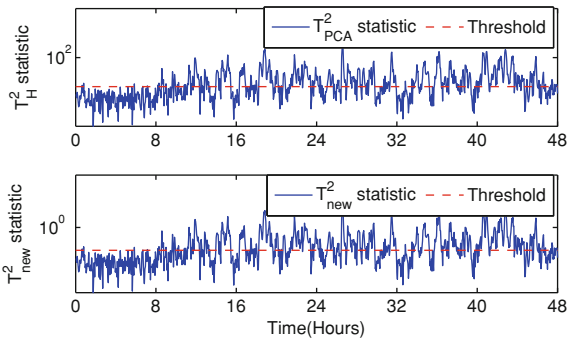


Fig. 5.9 Fault scenario 12 (T^2_{PCA}:$FAR = 0.0187$, $FDR = 0.4412$; T^2_{new}: $FAR = 0.0313$, $FDR = 0.6175$)

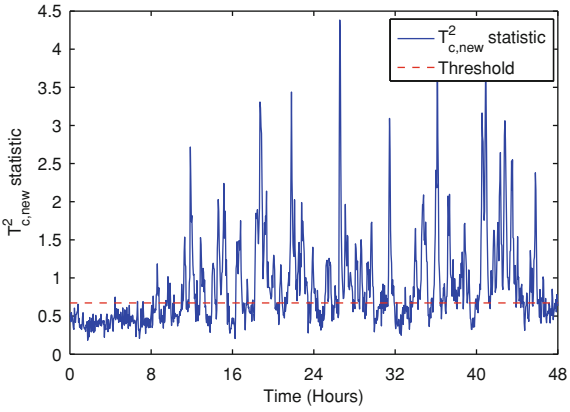


Fig. 5.10 Fault scenario 12 ($T^2_{c,new}$:$FAR = 0.0187$, $FDR = 0.6362$)

As described in Sect. 5.4, the threshold values are set as follows:

- Threshold for the T^2_{PCA} statistic: 26.9162
- Threshold for the T^2_{new} statistic: 0.3661;
- Threshold for the T^2_H statistic: 25.5539;
- Threshold for the $T^2_{c,new}$ statistic: 0.6718, which is smaller than

$$\sigma_m^2 J_{th,T^2_{PCA}} + J_{th,T^2_{new}} = \sigma_m^2 (J_{th,T^2_{PCA}} + J_{th,T^2_H}) = 0.7518.$$

The detection results using T^2_{PCA} and T^2_{new} statistics are plotted in Fig. 5.9 and the results for the new combined statistic is given in Fig. 5.10. It can be seen that the $T^2_{c,new}$ index of the combined statistic is higher than either the T^2_{PCA} statistic or the T^2_{new} statistic.

5.8 Notes and References

As a basic method of multivariate analysis, PCA has been widely and successfully applied to data compression, feature extraction, image processing, pattern recognition, signal analysis and process monitoring [1]. The application of PCA methods to fault detection with the offline training and online detection algorithms introduced in Sect. 5.1 can be found, for example, in [2–4]. In Algorithm 5.2, threshold settings for T^2_{PCA} and *SPE* are given. While $J_{th,T^2_{PCA}}$ given in (5.5) is evidently identical with J_{th,T^2} used in solving FD-P1 addressed in Chap. 3, the derivation of threshold setting for *SPE* test statistic, $J_{th,SPE}$ in (5.4), is more complicated. The reader is referred to [5] for a detailed study. It should be mentioned that the step, *Determine the number of principal components l*, in Algorithm 5.2 plays an important role in the PCA application, where *l* often serves as a tuning parameter. For this purpose, there exists a number of algorithms [6]. On the other hand, for the application to fault detection, the determination of *l* is less critical, since both test statistics, T^2_{PCA} and *SPE* (or T^2_{res}), are used and thus there is no loss of information. See also the following remarks.

We would like to call reader's attention to the ideas behind the PCA fault detection methods and make the following remarks:

- Although the SVD is the core of the PCA technique, it only serves, by its application to dealing with fault detection, as a numerical solution for computing the inverse of the covariance matrix of the measurements. It does not lead to improvement of fault detection performance.
- Instead, the data normalization given in Algorithm 5.1 plays a central role, since it provides us with an estimation of the (measurement) covariance matrix.
- The PCA method can be considered as an alternative solution to the problem FD-P1 introduced in Chap. 3.

In Sects. 5.2, 5.3 5.4, we have introduced the modified forms of the PCA technique aiming at a reliable fault detection, and discussed the properties of different test statistics and indices, which are also helpful to gain a deeper insight into the application of PCA. Study on the concept of combined indices has been reported by Qin in [4] .

Having understood the basic principle of the PCA for fault detection, it is clear that the application of DPCA [3] is limited to the processes in the steady operation.

Recent development in applying the PCA technique to fault diagnosis is focused on achieving adaptive process monitoring using for instance recursive implementation of PCA [7], fast moving window PCA [8] or kernel PCA [9], just mentioning some representative results.

References

1. Jolliffe I (1986) Principal component analysis. Springer, New York
2. MacGregor JF, Kourti T (1995) Statistical process control of multivariate processes. Contr Eng Pract 3:403–414

3. Chiang LH, Russell EL, Braatz RD (2001) Fault detection and diagnosis in industrial systems. Springer, London
4. Qin SJ (2003) Statistical process monitoring: basics and beyond. J Chemometr 17:480–502
5. Jackson JE, Mudholkar GS (1979) Control procedures for residuals associated with principal component analysis. Technometrics 21:341–349
6. Valle S, Li W, Qin SJ (1999) Selection of the number of principal components. Ind Eng Chem Res 38:4389–4401
7. Li W, Yue HH, Valle-Cervantes S, Qin SJ (2000) Recursive PCA for adaptive process monitoring. J Process Control 10:471–486
8. Wang X, Kruger U, Irwin GW (2005) Process monitoring approach using fast moving window PCA. Ind Eng Chem Res 44:5691–5702
9. Liu X, Kruger U, Lttler T, Xie L, Wang S (2009) Moving window kernel PCA for adaptive monitoring of nonlinear processes. Chemometr Intell Lab Syst 96:132–143

Chapter 6
Application of Partial Least Squares Regression to Fault Diagnosis

Roughly speaking, partial least squares (PLS) regression is a multivariate analysis method that constructs a linear regression between two data sets expressed in form of data matrices. In typical fault diagnosis and process monitoring applications, one of these two data sets represents the data from the process measurement variables. Another data set includes the data of those process variables, which represent, for instance, process operation performance or product quality. These process variables are often online unavailable or sampled with a long sampling period. Also, the number of the process measurement variables is significantly larger than the number of the process variables under monitoring. By means of PLS regression, the process variables are predicted using the online process measurements and, moreover, by T^2 and SPE test statistics changes (faults) in the process variables can be detected.

In this chapter, we shall first introduce the standard PLS algorithms. It is followed by the discussion on the application of the PLS regression method to fault diagnosis and interpretation of PLS algorithms.

6.1 Partial Least Squares Algorithms

Suppose that the process under consideration has a measurement (observation) vector $y_{obs} \in \mathcal{R}^m$ and the process variables under monitoring build vector $\theta_{obs} \in \mathcal{R}^\kappa$, $1 \leq \kappa < m$. As a data-driven method, the application of PLS to fault diagnosis also consists of two phases: (i) offline training (ii) online detection. Below, we first introduce the offline algorithm. It is supposed that both for y_{obs} and θ_{obs}, N data, $y_{obs}(1), \ldots, y_{obs}(N)$, $\theta_{obs}(1), \ldots, \theta_{obs}(N)$, are collected and recorded. There are numerous variations of PLS algorithms. The following algorithm is analog to the one provided by Dayal and MacGregor.

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_6, © Springer-Verlag London 2014

Algorithm 6.1 *PLS regression*

S1: Data normalization:

$$\bar{y}_{obs,i} = \frac{1}{N}\sum_{j=1}^{N} y_{obs,i}(j),\ \sigma^2_{y,obs,i} = \frac{1}{N-1}\sum_{j=1}^{N}\left(y_{obs,i}(j) - \bar{y}_{obs,i}\right)^2,\ i = 1,\ldots,m$$

$$y(k) = \begin{bmatrix} \frac{y_{obs,1}(k)-\bar{y}_{obs,1}}{\sigma_{y,obs,1}} \\ \vdots \\ \frac{y_{obs,m}(k)-\bar{y}_{obs,m}}{\sigma_{y,obs,m}} \end{bmatrix},\ Y = \left[\, y(1) \cdots y(N) \,\right] \in \mathcal{R}^{m\times N}$$

$$\bar{\theta}_{obs,i} = \frac{1}{N}\sum_{j=1}^{N} \theta_{obs,i}(j),\ \sigma^2_{\theta,obs,i} = \frac{1}{N-1}\sum_{j=1}^{N}\left(\theta_{obs,i}(j) - \bar{\theta}_{obs,i}\right)^2,\ i = 1,\ldots,\kappa$$

$$\theta(k) = \begin{bmatrix} \frac{\theta_{obs,1}(k)-\bar{\theta}_{obs,1}}{\sigma_{\theta,obs,1}} \\ \vdots \\ \frac{\theta_{obs,\kappa}(k)-\bar{\theta}_{obs,\kappa}}{\sigma_{\theta,obs,\kappa}} \end{bmatrix},\ \Theta = \left[\, \theta(1) \cdots \theta(N) \,\right] \in \mathcal{R}^{\kappa\times N}$$

S2: Set $Y_1 = Y$ and recursively compute: for $i = 1,\ldots,\gamma$

$$q_i^* = \arg\max_{\|q_i\|=1} \left\| Y_i \Theta^T q_i \right\|_E \tag{6.1}$$

$$w_i = \frac{Y_i \Theta^T q_i^*}{\left\| Y_i \Theta^T q_i^* \right\|_E},\ t_i = Y_i^T w_i,\ p_i = \frac{Y_i t_i}{\|t_i\|_E^2} \tag{6.2}$$

$$r_i = \begin{cases} w_1, i = 1 \\ \prod_{j=1}^{i-1}\left(I - w_j p_j^T\right) w_i, i > 1, \end{cases}\ q_i = \frac{\Theta Y_i^T w_i}{\|t_i\|_E^2} \tag{6.3}$$

$$Y_{i+1} = Y_i - p_i t_i^T \tag{6.4}$$

where γ is given or determined by applying a known stopping criterion
S3: Form matrices P, Q, R and T:

$$P = \left[\, p_1 \cdots p_\gamma \,\right] \in \mathcal{R}^{m\times\gamma},\ T = \left[\, t_1 \cdots t_\gamma \,\right] \in \mathcal{R}^{N\times\gamma} \tag{6.5}$$

$$Q = \left[\, q_1 \cdots q_\gamma \,\right] \in \mathcal{R}^{\kappa\times\gamma},\ R = \left[\, r_1 \cdots r_\gamma \,\right] \in \mathcal{R}^{m\times\gamma} \tag{6.6}$$

S4: Threshold settings:

$$SPE_{PLS} : J_{th,SPE_{PLS}} = g\chi^2_\alpha(h) \tag{6.7}$$

$$T^2_{PLS} : J_{th,T^2_{PLS}} = \frac{\gamma\left(N^2-1\right)}{N(N-\gamma)} F_\alpha(\gamma, N-\gamma) \tag{6.8}$$

where α is the significance level, $g\chi^2_\alpha(h)$ is the χ^2 distribution with scaling factor $g = \frac{S}{2\mu_{SPE}}$ and $h = \frac{2\mu^2_{SPE}}{S}$ degree of freedom, μ_{SPE}, S are respectively the mean and variance of SPE_{PLS}.

For the solution of the optimization problem (6.1), the following algorithm is available.

Algorithm 6.2 *Solution of optimization problem (6.1)*

S1: Solve eigenvalue-eigenvector problem

$$\left(\Theta Y_i^T Y_i \Theta^T - \lambda_i I\right) q_i = 0, i = 1, \ldots, \kappa$$

S2: If $\lambda_{\max} = \max\{\lambda_i, i = 1, \ldots, \kappa\} > 0$, *set*

$$q_i^* = q_{\max}, q_{\max}^T q_{\max} = 1 \tag{6.9}$$

otherwise stop the algorithm, where $q_{\max}$ is the (normalized) eigenvector corresponding to the maximum eigenvalue $\lambda_{\max}$.

Below is the (online) fault detection algorithm running after receiving a new measurement y_{obs}.

Algorithm 6.3 *PLS based fault detection and process monitoring*

S1: Normalization of the measurement

$$y_i = \frac{y_{obs,i} - \bar{y}_{obs,i}}{\sigma_{y,obs,i}}, i = 1, \ldots, m$$

S2: Online computation of SPE and T^2 statistic, prediction of θ:

$$SPE_{PLS} = \left\| \left(I - PR^T\right) y \right\|_E^2, T^2_{PLS} = y^T R \left(\frac{T^T T}{N-1}\right)^{-1} R^T y \tag{6.10}$$

$$\hat{\theta} = QR^T y \tag{6.11}$$

S3: Decision making according to the detection logic

$$T^2_{PLS} > J_{th,T^2_{PLS}} \text{ and } SPE_{PLS} \le J_{th,SPE_{PLS}}$$
$$\Longrightarrow \text{faulty in } \theta, \text{ otherwise fault-free in } \theta.$$

6.2 On the PLS Regression Algorithms

We discuss about the essential properties of the PLS algorithms and the ideas behind them.

6.2.1 Basic Ideas and Properties

Roughly speaking, PLS regression Algorithm 6.1 decomposes the data spaces spanned by Y, Θ respectively into two subspaces. Two of them are identified to be "mostly" correlated expressed in terms of the covariance and used for the regression purpose, based on which, for instance, the process variables are predicted.

The core of Algorithm 6.1 is the solution of optimization problem (6.1). In order to understand the idea behind it, we first consider the case with $i = 1$. It holds

$$q_1^* = \arg \max_{\|q_1\|=1} \left\| Y\Theta^T q_1 \right\|_E . \tag{6.12}$$

Moreover, it follows from Algorithm 6.2 and (6.2), (6.3) that

$$q_1 = cq_1^*, c = \frac{\lambda_{\max}}{\|t_1\|_E^2 \|w_1\|_E} = \frac{\lambda_{\max}^{1/2}}{\|t_1\|_E^2}.$$

Recall that

$$\frac{1}{N-1} Y\Theta^T \approx \mathcal{E}\left(y\theta^T\right) \Longrightarrow$$
$$\max_{\|q_1\|=1} \left\| Y\Theta^T q_1 \right\|_E = \sigma_{\max}\left(Y\Theta^T\right) \approx (N-1)\sigma_{\max}\left(\mathcal{E}\left(y\theta^T\right)\right).$$

Thus, the optimization problem (6.1) leads to finding the maximum correlation between Y and Θ (or y and θ), expressed in terms of the covariance, and its direction defined by q_1^* (or equivalently q_1). In this sense, the matrix $p_1 t_1^T$ with

$$t_1 = Y^T w_1, p_1 = \frac{Yt_1}{\|t_1\|_E^2} \Longrightarrow p_1 t_1^T = \frac{Yt_1 t_1^T}{\|t_1\|_E^2}$$

defines a subspace in Y, which is one dimensional, scaled and "mostly" correlated with Θ in sense of covariance. This subspace can be considered as a projection of Y with projection matrix $p_1 r_1^T$:

$$p_1 r_1^T = \frac{Yt_1 w_1^T}{\|t_1\|_E^2} \Longrightarrow p_1 r_1^T Y = \frac{Yt_1 w_1^T}{\|t_1\|_E^2} Y = p_1 t_1^T .$$

Analog to it, $q_1 w_1^T = q_1 r_1^T$ defines a mapping that maps Y onto Θ

$$q_1 r_1^T Y = \frac{\Theta t_1 w_1^T}{\|t_1\|_E^2} Y$$

which can then be considered as an estimate for Θ, denoted by $\hat{\Theta}_1$. Note that

$$q_1 r_1^T Y = \frac{\Theta t_1 w_1^T}{\|t_1\|_E^2} Y = \Theta \frac{t_1 t_1^T}{\|t_1\|_E^2}$$

is an one-dimensional subspace in Θ, which is "mostly" correlated with $\frac{Y t_1 t_1^T}{\|t_1\|_E^2}$. For $\gamma > 1$, we now consider $i = 2$ and let

$$Y_2 = Y_1 - p_1 t_1^T = Y \left(I - \frac{t_1 t_1^T}{\|t_1\|_E^2} \right).$$

It is evident that Y_2 is orthogonal to $\frac{Y t_1 t_1^T}{\|t_1\|_E^2}$, that is

$$Y_2 \left(\frac{Y t_1 t_1^T}{\|t_1\|_E^2} \right)^T = 0.$$

It is worth to note that Y_2 is a subspace of Y and achieved by a deflation of Y in terms of $\frac{Y t_1 t_1^T}{\|t_1\|_E^2}$. The optimization problem

$$\max_{\|q_2\|=1} \left\| Y_2 \Theta^T q_2 \right\|_E$$

and (6.2), (6.3) deliver w_2 and q_2 that gives the direction of the maximum correlation between Y_2 and Θ. The further computations,

$$\begin{aligned} t_2 = Y_2^T w_2,\ p_2 = \frac{Y_2 t_2}{\|t_2\|_E^2} \Longrightarrow p_2 t_2^T = \frac{Y_2 t_2 t_2^T}{\|t_2\|_E^2} \\ q_2 \left(Y_2^T w_2 \right)^T = q_2 \left(\left(I - \frac{t_1 t_1^T}{\|t_1\|_E^2} \right) Y^T w_2 \right)^T \\ = q_2 \left(Y^T \left(I - w_1 p_1^T \right) w_2 \right)^T = q_2 r_2^T Y \end{aligned}$$

repeat the same procedure for $i = 1$ and yield a further estimate of Θ by $q_2 r_2^T Y := \hat{\Theta}_2$. At the end of the recursive computations in Algorithm 6.1, we receive

$$\hat{\Theta}_i = q_i r_i^T Y, i = 1, \ldots \gamma. \tag{6.13}$$

We now study some interesting properties of $p_i t_i^T, q_i r_i^T Y, i = 1, \ldots \gamma$, which are helpful to gain a deeper insight into the PLS algorithm. Note that

$$Y^T r_1 = Y_1^T w_1 = t_1, Y^T r_i = Y_i^T w_i = t_i, i = 2, \ldots, \gamma \tag{6.14}$$

and moreover for $i > j$

$$\begin{aligned} p_i t_i^T &= p_i w_i^T Y_i = p_i w_i^T Y_{i-1} \left(I - \frac{t_{i-1} t_{i-1}^T}{\|t_{i-1}\| E^2} \right) \\ &= p_i w_i^T \left(I - Y_{i-1} \frac{t_{i-1} w_{i-1}^T}{\|t_{i-1}\|_E^2} \right) Y_{i-1} = p_i w_i^T \Psi Y_j \left(I - \frac{t_j t_j^T}{\|t_j\|_E^2} \right) \end{aligned}$$

$$r_i^T Y = t_i^T = w_i^T Y_i = w_i^T \Psi Y_j \left(I - \frac{t_j t_j^T}{\|t_j\|_E^2} \right)$$

$$\Psi = \begin{cases} I, i = j+1 \\ \prod_{k=1}^{i-j-1} \left(I - Y_{i-k} \frac{t_{i-k} w_{i-k}^T}{\|t_{i-k}\|_E^2} \right), i > j+1 \end{cases}.$$

It turns out for $i \neq j$

$$t_i^T t_j = r_i^T Y \left(r_j^T Y \right)^T = 0, p_i t_i^T \left(p_j t_j^T \right)^T = 0. \tag{6.15}$$

The second equation in (6.15) tells us that $p_i t_i^T, i = 1, \ldots, \gamma$, span γ orthogonal subspaces in Y which are "mostly correlated" with Θ in terms of covariance. Notice that

$$E := Y_{\gamma+1} = Y - \sum_{i=1}^{\gamma} p_i t_i^T = Y_j - \sum_{i=j}^{\gamma} p_i t_i^T, j = 1, \ldots, \gamma$$

$$\left(Y_j - \sum_{i=j}^{\gamma} p_i t_i^T \right) \left(p_j t_j^T \right)^T = Y_j \left(I - \frac{t_j t_j^T}{\|t_j\|_E^2} \right) t_j p_j^T + \sum_{i=j+1}^{\gamma} p_i t_i^T \left(p_j t_j^T \right)^T = 0.$$

Thus, we can claim that the data space spanned by Y can be decomposed into $\gamma + 1$ orthogonal subspaces, respectively spanned by $p_i t_i^T, i = 1, \ldots, \gamma + 1$.

Write Y as

$$Y = P T^T + E.$$

One of the interpretations of the above statement is that the deflation of Y is realized recursively and, in each step, Y is deflated by subtracting an one-dimensional subspace. It follows from (6.14) that

$$T = Y^T R \Longrightarrow Y = PR^T Y + E. \tag{6.16}$$

Recall that PR^T projects the observations onto a subspace of Y that is "mostly" correlated with the process variables under monitoring. For our purpose, we define

$$\hat{y}_{y2\theta} = PR^T y \tag{6.17}$$

which gives an estimate of the component in an (online) observation that is "mostly" correlated with θ.

In order to estimate Θ with the aid of the data set Y, it follows (6.13) that

$$\hat{\Theta} = \sum_{i=1}^{\gamma} q_i \left(Y_i^T w_i\right)^T = \sum_{i=1}^{\gamma} q_i r_i^T Y = QR^T Y \Longrightarrow \hat{\Psi} = QR^T \tag{6.18}$$

provides a reasonable estimation for $\hat{\Theta}$, where $\hat{\Psi} = QR^T$ defines the mapping from the process observation variables to the process variables under monitoring. As a result, an estimation/prediction of $\theta(k)$ can be achieved by

$$\hat{\theta} = QR^T y \tag{6.19}$$

which is often applied for predicting/estimating the process variables in terms of the (online) process observations.

In summary, we would like to emphasize that

- by means of recursive deflations the data space spanned by Y is decomposed into two orthogonal subspaces: $PT^T = \sum_{i=1}^{\gamma} p_i t_i^T$ and E $(= p_{\gamma+1} t_{\gamma+1}^T)$
- PT^T is a γ-dimensional subspace that is "mostly" correlated with Θ in sense of covariance. It is constructed by γ one-dimensional orthogonal subspaces
- corresponding to the decomposition in Y, an estimate for θ is achieved by $QT^T = \sum_{i=1}^{\gamma} q_i^T t_i^T$, where QT^T is "mostly" correlated with PT^T in Y, expressed in terms of covariance
- E may be also (weakly) correlated with Θ and in analog
- QT^T can also contain components of Y which are uncorrelated with Θ.

6.2.2 Application to Fault Detection and Process Monitoring

On the assumption that $y \sim \mathcal{N}(0, \Sigma_y)$, the PLS regression algorithm has been widely applied for the fault detection and process monitoring purposes. The typical

test statistics for detecting faults in θ and y are T^2_{PLS}, SPE_{PLS} defined in Algorithm 6.3. Recall that

$$\begin{aligned} r_i^T p_j &= r_i^T \frac{Y_i t_i}{\|t_i\|_E^2} = \frac{r_i^T Y^T t_j}{\|t_i\|_E^2} = \begin{cases} = 1, i = j \\ 0, i \neq j \end{cases}, i, j = 1, \ldots, \gamma \\ \Longrightarrow R^T P &= I \Longleftrightarrow \begin{bmatrix} R^T \\ R^{\perp T} \end{bmatrix} P = \begin{bmatrix} I \\ 0 \end{bmatrix}. \end{aligned} \tag{6.20}$$

It turns out

$$\begin{bmatrix} R^T \\ R^{\perp T} \end{bmatrix} \hat{y}_{y2\theta} = \begin{bmatrix} R^T \hat{y}_{y2\theta} \\ 0 \end{bmatrix} = \begin{bmatrix} R^T y \\ 0 \end{bmatrix}.$$

Thus, for the detection purpose, $R^T y$ is used for building the test statistic. Note that $R^T y \sim \mathcal{N}(0, R^T \Sigma_y R)$ and $R^T \Sigma_y R$ can be, using the training data, estimated by

$$R^T \Sigma_y R \approx \frac{R^T Y Y^T R}{N-1} = \frac{T^T T}{N-1}.$$

As a result, T^2_{PLS} is a T^2 test statistic for $R^T y$, which can be realized as follows

$$T^2_{PLS} = y^T R \left(R^T \Sigma_y R \right)^{-1} R^T y = y^T R \left(\frac{T^T T}{N-1} \right)^{-1} R^T y$$

with the corresponding threshold setting for $J_{th,T^2_{PLS}}$ given in (6.8), as discussed in Sect. 3.3.2.

Similar to the PCA based fault detection, also $y - \hat{y}_{y2\theta}$ is applied for the fault detection purpose. On the other hand, it should be noticed that, different to the PCA application, the vector $y - \hat{y}_{y2\theta}$ represents the component (sub-vector) in the observation y which is weakly correlated with θ. Thus, it is often applied for constructing an additional index and checking, in combination with T^2_{PLS}, a possible change in θ.

Considering that the quadratic form of $y - \hat{y}_{y2\theta}$ can be rewritten into

$$\left(y - \hat{y}_{y2\theta} \right)^T \left(y - \hat{y}_{y2\theta} \right) = z^T z, z \in \mathcal{R}^{(m-\gamma)}$$

with $z \sim \mathcal{N}(0, \Sigma_z)$, $\Sigma_z \in \mathcal{R}^{(m-\gamma)\times(m-\gamma)}$ and moreover $m >> \gamma$, Q statistic, instead of T^2, is generally applied, which leads to the threshold setting $J_{th,SPE_{PLS}}$ given in (6.7). To this end, Algorithm 3.6 given in Sect. 3.3.3 is applied with

$$g = \frac{S}{2\mu_{SPE}}, h = \frac{2\mu^2_{SPE}}{S}, \hat{y}_{y2\theta}(k) = P R^T y(k)$$

$$\mu_{SPE} = \frac{1}{N}\sum_{i=1}^{N}\left(y(k) - \hat{y}_{y2\theta}(k)\right)^T\left(y(k) - \hat{y}_{y2\theta}(k)\right)$$

$$S = \frac{1}{N}\sum_{i=1}^{N}\left(\left(y(k) - \hat{y}_{y2\theta}(k)\right)^T\left(y(k) - \hat{y}_{y2\theta}(k)\right)\right)^2 - \mu_{SPE}^2.$$

6.3 Relations Between LS and PLS

In this section, we first re-study the PLS algorithm in the context of least squares (LS) estimation. This is helpful for us to view the PLS-based fault detection scheme from a different viewpoint.

6.3.1 LS Estimation

Suppose that $y \sim \mathcal{N}(0, \Sigma_y)$, $\theta \sim \mathcal{N}(0, \Sigma_\theta)$, and there exists the following regression between y and θ

$$\theta = \Psi y + \varepsilon_\theta \in \mathcal{R}^\kappa \tag{6.21}$$

with $rank\,(\Psi) = \kappa$, where ε_θ is the part in θ which is uncorrelated with y, that is $\mathcal{E}(\varepsilon_\theta y^T) = 0$. On the assumption that data sets Y, Θ with a large N are available and YY^T is invertible,

$$\hat{\Psi} = \Theta Y^T\left(YY^T\right)^{-1}, \hat{\theta} = \hat{\Psi} y \tag{6.22}$$

deliver an LS solution for Ψ and θ estimations. Noting that

$$\frac{1}{N-1}\Theta Y^T \approx \mathcal{E}(\theta y^T), \frac{1}{N-1}YY^T \approx \mathcal{E}(yy^T) \tag{6.23}$$

$\hat{\Psi}$ can also be written into

$$\hat{\Psi} = \frac{1}{N-1}\Theta Y^T\left(\frac{1}{N-1}YY^T\right)^{-1} \approx \mathcal{E}(\theta y^T)\left(\mathcal{E}(yy^T)\right)^{-1}.$$

In case that

$$rank\left(YY^T\right) = l < m$$

the LS estimation for Ψ is given by

$$\hat{\Psi} = \Theta Y^T \left(YY^T\right)^+ \tag{6.24}$$

with $\left(YY^T\right)^+$ being the pseudo-inverse of YY^T which can be expressed in terms of the SVD of YY^T as follows

$$\begin{aligned} YY^T &= U\Sigma U^T, \Sigma = \begin{bmatrix} diag(\sigma_1, \ldots, \sigma_l)\ 0 \\ 0 \qquad\qquad 0 \end{bmatrix}, U = \begin{bmatrix} U_1\ U_2 \end{bmatrix} \\ &\Longrightarrow \left(YY^T\right)^+ = U\Sigma^+ U^T = U_1 diag(\sigma_1^{-1}, \ldots, \sigma_l^{-1}) U_1^T. \end{aligned}$$

We now address the LS estimation scheme from the viewpoint of the correlation between θ and y or Θ and Y. Let

$$\mathcal{E}(\theta y^T) = V\Sigma U^T \tag{6.25}$$

be an SVD of $\mathcal{E}(\theta y^T)$ with

$$\begin{aligned} \Sigma = \begin{bmatrix} \Sigma_{\theta y}\ 0 \end{bmatrix}, \Sigma_{\theta y} = diag\left(\sigma_{\theta y,1}, \ldots, \sigma_{\theta y,\kappa}\right), \sigma_{\theta y,1} \geq \cdots \geq \sigma_{\theta y,\kappa} > 0 \\ VV^T = I, UU^T = I, U = \begin{bmatrix} U_{\theta y}\ U_{\theta y}^{\perp} \end{bmatrix} \in \mathcal{R}^{m\times m}, U_{\theta y} \in \mathcal{R}^{m\times\kappa} \end{aligned}$$

which can be further written as

$$\mathcal{E}(\theta y^T) = V\Sigma U^T = V\Sigma_{\theta y} U_{\theta y}^T.$$

For our purpose, let us introduce the notation

$$\mathcal{E}(U^T y y^T U) = \bar{\Sigma}_y = \begin{bmatrix} \Sigma_{11}\ \Sigma_{12} \\ \Sigma_{12}^T\ \Sigma_{22} \end{bmatrix} \in \mathcal{R}^{m\times m}, \bar{\Sigma}_y > 0, \Sigma_{11} \in \mathcal{R}^{\kappa\times\kappa} \tag{6.26}$$

and consider $\mathcal{E}(\theta y^T U)\mathcal{E}(U^T y y^T U)^{-1}$, which is written into

$$\mathcal{E}(\theta y^T U)\mathcal{E}(U^T y y^T U)^{-1} = V\Sigma U^T U\bar{\Sigma}_y^{-1} = V\Sigma_{\theta y}\Pi \tag{6.27}$$

$$\Pi = \begin{bmatrix} \Sigma_{11}^{-1} + \Sigma_{11}^{-1}\Sigma_{12}\Delta^{-1}\Sigma_{12}^T\Sigma_{11}^{-1} & -\Sigma_{11}^{-1}\Sigma_{12}\Delta^{-1} \end{bmatrix} := \begin{bmatrix} \Pi_{11}\ \Pi_{12} \end{bmatrix} \tag{6.28}$$

$$\Delta = \Sigma_{22} - \Sigma_{12}^T\Sigma_{11}^{-1}\Sigma_{12}.$$

The LS estimate for Ψ is then given by

$$\hat{\Psi} = \mathcal{E}(\theta y^T U)\mathcal{E}(U^T y y^T U)^{-1} U^T = V\Sigma U^T U\bar{\Sigma}_y^{-1} = V\Sigma_{\theta y}\Pi U^T. \tag{6.29}$$

We now rewrite (6.29) in terms of the data sets Y, Θ and relations (6.22), (6.23). It holds

$$\frac{1}{N-1}\Theta Y^T = V\Sigma U^T = V\Sigma_{\theta y}U_{\theta y}^T \tag{6.30}$$

$$\Theta Y^T\left(YY^T\right)^{-1} = \frac{1}{N-1}\Theta Y^T\left(\frac{1}{N-1}YY^T\right)^{-1} = \Theta Y^T U_{\theta y}\Pi U^T. \tag{6.31}$$

Noting that

$$\mathcal{E}(U^T yy^T U) = U^T\mathcal{E}(yy^T)U \Longleftrightarrow \left(\mathcal{E}(U^T yy^T U)\right)^{-1} = U^T\left(\mathcal{E}(yy^T)\right)^{-1}U \tag{6.32}$$

$$\Longrightarrow \Pi_{11} = U_{\theta y}^T\left(\mathcal{E}(yy^T)\right)^{-1}U_{\theta y},\ \Pi_{12} = U_{\theta y}^T\left(\mathcal{E}(yy^T)\right)^{-1}U_{\theta y}^{\perp} \tag{6.33}$$

$\hat{\Psi}$ can be further written into

$$\hat{\Psi} = \Theta Y^T U_{\theta y}\left[U_{\theta y}^T\left(YY^T\right)^{-1}U_{\theta y}\ \ U_{\theta y}^T\left(YY^T\right)^{-1}U_{\theta y}^{\perp}\right]U^T. \tag{6.34}$$

It is of interest to notice that, in case of $\Sigma_{12} = U_{\theta y}^T YY^T U_{\theta y}^{\perp} = 0$, it holds

$$\hat{\Psi} = \Theta Y^T U_{\theta y}U_{\theta y}^T\left(YY^T\right)^{-1}U_{\theta y}U_{\theta y}^T = \Theta Y^T U_{\theta y}\Sigma_{11}^{-1}U_{\theta y}^T. \tag{6.35}$$

6.3.2 LS Interpretation of the PLS Regression Algorithm

We are now in a position to give an alternative interpretation of the PLS regression algorithm in terms of its relation to the LS algorithm. We first consider the special case that YY^T is invertible and $\gamma = m$. It holds

$$Y = PT^T,\ YY^T = PT^TTP^T.$$

Recall that

$$R^T P = I,\ R \in \mathcal{R}^{m\times m},\ P \in \mathcal{R}^{m\times m} \Longleftrightarrow R^T = P^{-1}$$

which yields

$$\hat{\Psi} = \Theta Y^T\left(YY^T\right)^{-1} = \Theta T\left(T^TT\right)^{-1}R^T$$

$$\left(T^TT\right)^{-1} = diag\left(\frac{1}{\|t_1\|_E^2},\ldots,\frac{1}{\|t_m\|_E^2}\right).$$

On the other hand, we know that the PLS regression delivers

$$\hat{\Psi} = QR^T = \Theta \left[\frac{Y_1^T w_1}{\|t_1\|_E^2} \cdots \frac{Y_m^T w_m}{\|t_m\|_E^2} \right] R^T = \Theta T \left(T^T T \right)^{-1} R^T.$$

As a result, it becomes clear that the PLS regression is an LS solution.

Next, we consider the case that $\gamma = rank\left(YY^T\right) = l < m$. Since

$$YY^T = PT^T TP^T$$

it holds

$$\hat{\Psi} = \Theta Y^T \left(YY^T \right)^- = \Theta T P^T \left(PT^T TP^T \right)^-.$$

where $(\cdot)^-$ denotes the pseudo-inverse of a matrix. Let

$$P = U \Sigma V^T, \Sigma = \begin{bmatrix} diag(\sigma_1, \ldots, \sigma_l) \\ 0 \end{bmatrix}$$

be an SVD of P. It is easy to demonstrate that

$$\begin{aligned} &\left(PT^T TP^T \right)^- \\ &= U \begin{bmatrix} diag(\sigma_1^{-1}, \ldots, \sigma_l^{-1}) V^T \left(T^T T \right)^{-1} V diag(\sigma_1^{-1}, \ldots, \sigma_l^{-1}) & 0 \\ 0 & 0 \end{bmatrix} U^T. \end{aligned}$$

Moreover,

$$R^T P = I \Longrightarrow R^T = V \left[diag(\sigma_1^{-1}, \ldots, \sigma_l^{-1}) \; 0 \right] U^T.$$

It yields

$$\hat{\Psi} = \Theta Y^T \left(YY^T \right)^- = \Theta T \left(T^T T \right)^{-1} R^T$$

which demonstrates that, also in this case, the PLS regression is an LS solution.

We now consider a more general case with $\gamma < rank\left(YY^T\right) = m$. For our purpose, we first give an alternative interpretation of Y_i. Let

$$\Theta Y^T = \Theta Y_1^T = V_1 \Sigma_1 U_1^T, V_1 \in \mathcal{R}^{\kappa \times \kappa}, U_1 \in \mathcal{R}^{m \times m}$$

be an SVD of ΘY^T and denote

$$V_1 = \left[v_1 \; V_1^\perp \right], U_1 = \left[u_1 \; U_1^\perp \right].$$

It is evident that the first columns of V_1 and U_1 are v_1 and w_1, respectively, as delivered by Algorithm 6.1. Note further that

$$U_1^T Y_2 = U_1^T Y \left(I - \frac{t_1 t_1^T}{\|t_1\|_E^2} \right) = U_1^T Y - \frac{U_1^T Y Y^T w_1 w_1^T Y}{\|t_1\|_E^2}$$
$$= \begin{bmatrix} 0 \\ U_1^{\perp T} Y \left(I - \frac{t_1 t_1^T}{\|t_1\|_E^2} \right) \end{bmatrix} = \begin{bmatrix} 0 \\ U_1^{\perp T} Y_2 \end{bmatrix} \tag{6.36}$$

which leads to

$$U_1^T Y_2 Y_2^T U_1 = \begin{bmatrix} 0 & 0 \\ 0 & U_1^{\perp T} Y_2 Y_2^T U_1^{\perp} \end{bmatrix}$$
$$U_1^T p_1 t_1^T t_1 p_1^T U_1 = \begin{bmatrix} u_1^T Y Y^T u_1 & u_1^T Y Y^T U_1^{\perp} \\ U_1^{\perp T} Y Y^T u_1 & U_1^{\perp T} Y \frac{t_1 t_1^T}{\|t_1\|_E^2} Y^T U_1^{\perp} \end{bmatrix}.$$

Now, consider

$$\Theta Y_2^T U_1 = \begin{bmatrix} 0 & \Theta Y_2^T U_1^{\perp} \end{bmatrix}$$

and let

$$\Theta Y_2^T U_1^{\perp} = V_2 \Sigma_2 \bar{U}_2^T, \bar{U}_2 = \begin{bmatrix} \bar{u}_2 & \bar{U}_2^{\perp} \end{bmatrix}$$

be an SVD of $\Theta Y_2^T U_1^{\perp}$. It is clear that

$$\Theta Y_2^T U_1 = V_2 \Sigma_2 \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \bar{U}_2^T \end{bmatrix} \Longrightarrow \Theta Y_2^T = V_2 \Sigma_2 \bar{U}_2^T U_1^{\perp T}$$
$$\Longrightarrow w_2 = U_1^{\perp} \bar{u}_2 := u_2, U_1^{\perp} \bar{U}_2^{\perp} := U_2^{\perp}.$$

Define

$$U_2^T = \begin{bmatrix} 1 & 0 \\ 0 & \bar{U}_2^T \end{bmatrix} \Longrightarrow U_2 U_2^T = I.$$

In a similar way, it can be then demonstrated that

$$U_2^T U_1^T Y_3 = U_2^T U_1^T Y_2 \left(I - \frac{t_2 t_2^T}{\|t_2\|_E^2} \right) = \begin{bmatrix} 0 \\ \bar{U}_2^{\perp T} U_1^{\perp T} Y_3 \end{bmatrix}$$
$$= \begin{bmatrix} 0 \\ 0 \\ U_2^{\perp T} Y_3 \end{bmatrix}, U_2^T U_1^T Y_3 Y_3^T U_1 U_2 = \begin{bmatrix} 0\,0 & 0 \\ 0\,0 & 0 \\ 0\,0 & U_2^{\perp T} Y_3 Y_3^T U_2^{\perp} \end{bmatrix} \Longleftrightarrow$$
$$U_2^T U_1^T \left(p_1 t_1^T t_1 p_1^T + p_2 t_2^T t_2 p_2^T \right) U_1 U_2 =$$
$$\begin{bmatrix} u_1^T Y Y^T u_1 & u_1^T Y^T U_1^{\perp} \\ U_1^{\perp T} Y u_1 & u_2^T \Pi_2 u_2 \quad u_2^T \Pi_2 U_2^{\perp T} \\ & U_2^{\perp T} \Pi_2 u_2 \quad U_2^{\perp T} \Pi_2 U_2^{\perp} \end{bmatrix}, \Pi_2 = \frac{Y_1 t_1 t_1^T Y_1^T}{\|t_1\|_E^2} + \frac{Y_2 t_2 t_2^T Y_2^T}{\|t_2\|_E^2}.$$

Introducing the following notations, for $i = 1, \ldots, \gamma$

$$\Theta Y_i^T U_{i-1}^{\perp} = V_i \Sigma_i \bar{U}_i^T, U_0^{\perp} = I, U_i^T = \begin{bmatrix} I & 0 \\ 0 & \bar{U}_i^T \end{bmatrix}, U_i U_i^T = I, U_1 = \bar{U}_1$$
$$\bar{U}_i = \begin{bmatrix} \bar{u}_i & \bar{U}_i^{\perp} \end{bmatrix}, u_i = w_i = U_{i-1}^{\perp} \bar{u}_i, U_i^{\perp} = U_{i-1}^{\perp} \bar{U}_i^{\perp}$$
$$\Pi_i = \sum_{j=1}^{i} \frac{Y_j t_j t_j^T Y_j^T}{\|t_j\|_E^2}, 1 = 2, \ldots, \gamma,$$

by induction it can be proven that

$$\left(\prod_{i=1}^{\gamma} U_i \right)^T Y_{\gamma+1} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ U_{\gamma}^{\perp T} Y_{\gamma+1} \end{bmatrix}. \tag{6.37}$$

It follows from (6.37) that the deflation of Y is equivalent to remove those rows of Y, which are correlated with Θ, one by one and according to their correlation degree. In this manner, the role of the deflation becomes evident. We are now going to discuss different cases. First, assume that

$$\sigma_{\max}\left(Y_{\gamma+1}\right) = \sigma_{\max}\left(\left(\prod_{i=1}^{\gamma} U_i \right)^T Y_{\gamma+1} \right) \approx 0.$$

It is clear that under this (hard) assumption it holds

$$\gamma = rank\left(YY^T\right) \leq m$$

and hence, according to the previous discussion, the PLS estimation for Ψ is a good approximation of the LS estimation. A more realistic and reasonable assumption, considering the above mentioned deflation mechanism, is that

$$\Theta Y_{\gamma+1}^T = 0. \tag{6.38}$$

If, in addition, assume that

$$\Xi\left(\gamma + 1 : m, 1 : N\right) \Xi^T \left(1 : \gamma, 1 : N\right) = 0 \tag{6.39}$$
$$\text{with } \Xi = \left(\prod_{i=1}^{\gamma} U_i \right)^T \left(\sum_{i=1}^{\gamma} p_i t_i^T \right)$$

it holds,

$$U^T Y Y^T U = U^T \left(P T^T T P^T + Y_{\gamma+1} Y_{\gamma+1}^T \right) U =$$
$$\begin{bmatrix} \hat{\Xi}\,(1:\gamma, 1:\gamma) & 0 \\ 0 & \hat{\Xi}\,(\gamma+1:m, \gamma+1:m) + U_\gamma^{\perp T} Y_{\gamma+1} Y_{\gamma+1}^T U_\gamma^\perp \end{bmatrix}$$
$$\hat{\Xi} = \Xi \Xi^T$$

with $U = \prod_{i=1}^{\gamma} U_i$, $\hat{\Xi} = \Xi \Xi^T \in \mathcal{R}^{m \times m}$. It leads to an LS estimate for $\hat{\Psi}$

$$\hat{\Psi} = \Theta Y^T U U^T \left(Y Y^T \right)^{-1} U U^T = \Theta \left(\sum_{i=1}^{\gamma} t_i p_i^T \right) U \Phi^{-1} U^T \tag{6.40}$$

$$\Phi^{-1} = \begin{bmatrix} \hat{\Xi}^{-1}\,(1:\gamma, 1:\gamma) & 0 \\ 0 & \left(\hat{\Xi}\,(\gamma+1:m, \gamma+1:m) + U_\gamma^{\perp T} Y_{\gamma+1} Y_{\gamma+1}^T U_\gamma^\perp \right)^{-1} \end{bmatrix}.$$

When, moreover,

$$\Theta \Xi^T\,(\gamma+1:m, 1:N) = 0 \tag{6.41}$$

we have

$$\hat{\Psi} = \Theta \left(\sum_{i=1}^{\gamma} t_i p_i^T \right) U \Phi^{-1} U^T = \Theta \left[\Xi^T\,(1:\gamma, 1:N)\, \hat{\Xi}^{-1}\,(1:\gamma, 1:\gamma)\ 0 \right] U^T \tag{6.42}$$

which means the equivalence between the PLS and LS estimations

$$\begin{aligned} \hat{\Psi} &= \Theta Y^T \left(Y Y^T \right)^{-1} = \Theta Y^T U U^T \left(Y Y^T \right)^{-1} U U^T \\ &= \Theta \Xi^T\,(1:\gamma, 1:N)\, \hat{\Xi}^{-1}\,(1:\gamma, 1:\gamma)\,(U\,(1:m, 1:\gamma))^T \\ &= \Theta \left(\sum_{i=1}^{\gamma} t_i p_i^T \right) \left(\sum_{i=1}^{\gamma} p_i t_i^T t_i p_i^T \right)^{-} = Q R^T. \end{aligned}$$

It should be emphasized that this equivalence is achieved under the conditions (6.38), (6.39) and (6.41). These conditions mean that the first γ rows of $U^T Y$, $\Xi\,(1:\gamma, 1:N)$, are correlated with Θ, while the $\gamma+1$ to m rows, $\Xi\,(\gamma+1:m, 1:N)$, are uncorrelated with Θ. Notice that this is exactly the situation given in (6.34).

In general, if one of the conditions given in (6.38), (6.39) and (6.41) is not satisfied, the equivalence between the PLS and LS estimations will not hold. As a result, the estimation performance of the PLS regression algorithm becomes poorer than the one delivered by the LS solution.

6.4 Remarks on PLS Based Fault Diagnosis

From the viewpoint of fault diagnosis, the application of PLS regression technique serves establishing a linear regression between two groups of variables using the collected (training) data. It is also a significant feature by applying PLS based fault diagnosis methods that one of these two variable groups includes a great number of measurement variables and the number of the (process) variables in another group is considerably less. The main goal is to predict the process variables based on the established linear regression model and, using the predicted value, to realize process monitoring and to detect changes in the process variables.

Identifying a linear regression model using process data is a trivial engineering problem. There are numerous mathematical and engineering methods for solving this problem. The questions may arise: why and under which conditions should the PLS based methods be applied? In order to answer these questions, we first summarize the major features of the PLS regression in the form of Algorithm 6.1:

- low computation demands: it is evident that the major computation in Algorithm 6.1 is the solution of the optimization problem (6.1). As given in Algorithm 6.2, it deals with an eigenvalue-eigenvector computation of a $\kappa \times \kappa$-dimensional matrix. Since $\kappa << m$, the computation in each iteration is numerically highly reliable and its cost is low
- recursive computation: the overall solution is approached step by step. In this manner, numerical stability is achieved and unnecessary computations can be avoided
- suboptimal linear regression: As discussed in the last section, PLS regression technique leads to a linear regression which is, in general, suboptimal in the sense of least squares prediction error.

In this context, the basic criterion for selecting the PLS regression for fault diagnosis purpose are:

- a great number of measurement variables is available
- the application goal is to find out a limited number of combinations of the measurement variables, which deliver sufficient information for a reliable prediction of the process variables under monitoring
- the search procedure can deliver alternative solutions for testing or simulations
- the computation capacity and resources are limited and there may exist numerical stability and reliability concerns.

It should be pointed out that the last point is nowadays often less critical, although it was, a couple of decades ago, a convincing argument for the application of PLS instead of, for instance, LS method.

6.5 Case Study on TEP

The application of PLS to fault diagnosis is now demonstrated on the TEP simulator introduced in Sect. 2.3. Among the six operating modes, the mode number 1 and the decentralized control strategy are adopted in our case study. The detailed simulation conditions are given in Sect. 2.3.

6.5.1 Test Setup

As listed in Table 6.1, 31 process variables and 1 product quality variable are considered for testing the PLS based fault detection approach. In the process variables group, the manipulated variables $XMV(5)$, $XMV(9)$ and $XMV(12)$ are excluded, since they are constant for the considered setup, do not provide useful information for fault diagnosis and can, on the other hand, cause numerical difficulties. The process measurements $XMEAS(1)$–$XMEAS(22)$ are included, which contain operational information of the process. Considering that component G is the main product of this process, the product quality variable is set as $XMEAS(35)$, namely the composition of G.

6.5.2 Offline Training

In the offline training phase, $N = 960$ samples are collected. The well-accepted leave-one-out cross-validation based approach results in the setting $\gamma = 6$. It follows the establishment of the PLS model with the needed matrices for online monitoring. The threshold settings are

- T^2 statistic: $J_{th,T^2_{PLS}} = 12.5916$
- SPE statistic: $J_{th,SPE_{PLS}} = 29.313$.

For both of these two statistics, the significance level is set as $\alpha = 0.05$.

6.5.3 Online Running

In the online stage, process operation is simulated for 48 h, and the faults are added in the 8 h. The collected online measurement data are first centered likewise in training phase. To illustrate the test, we describe, as a representative example, the procedure of detecting fault scenario 1. As the fault occurs, the A/C feed ratio is changed in a step form. This decreases the composition of A ($XMEAS23$). As a reaction to it, a control loop increases the A feed ($XMEAS1$). The variations in

Table 6.1 Variables involved in the PLS model

Variable	Tag	Number
Process variables	XMV(1–4;6–9;10–11), XMEAS(1–22)	31
Product quality variable	XMEAS (35)	1

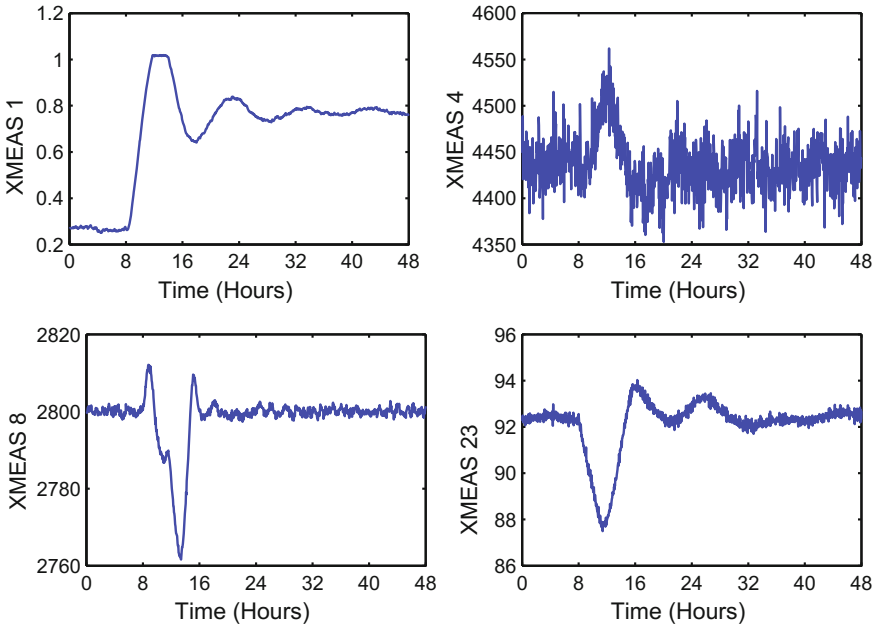


Fig. 6.1 Changes in the relevant process variables caused by fault scenario 1

the flow rate and compositions to the reactor cause the variations in the reactor level ($XMEAS8$), which affects the flow rate ($XMEAS4$) through a cascade control loop. Furthermore, the variables associated with reaction such as pressure and composition of reactants are also affected correspondingly. Figure 6.1 shows the affected variables. This fault will ultimately influence the composition of G ($XMEAS35$), namely the product quality index concerned. In Fig. 6.2, the simulation results of the composition of G in the fault-free and faulty cases are compared. It can be seen clearly that the composition of G is significantly affected by the fault. Figure 6.3 presents the detection results by PLS based approach. A reliable and early detection of the fault can be observed. Additionally, four other faults are selected for the demonstration purpose. These are faults 4, 6, 11 and 20. The detection results are given in Figs. 6.4, 6.5, 6.6, 6.7.

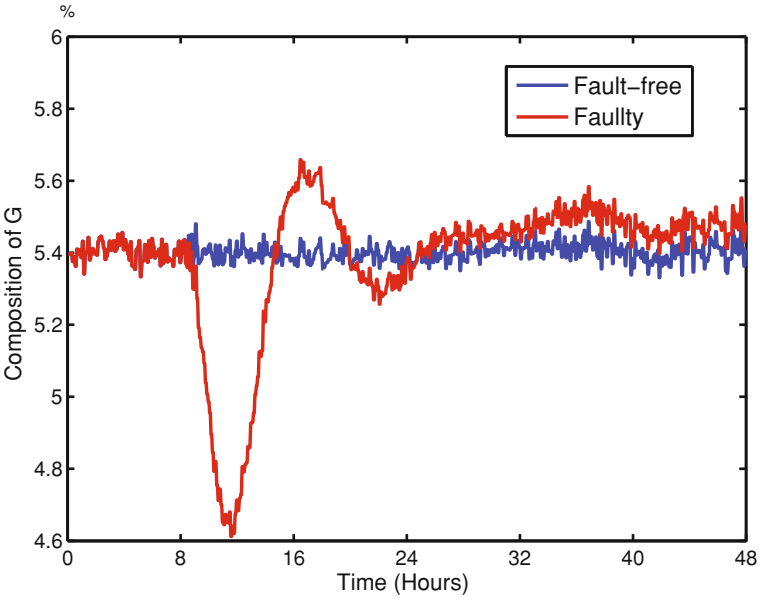


Fig. 6.2 Comparison of the product quality variable: fault-free versus faulty

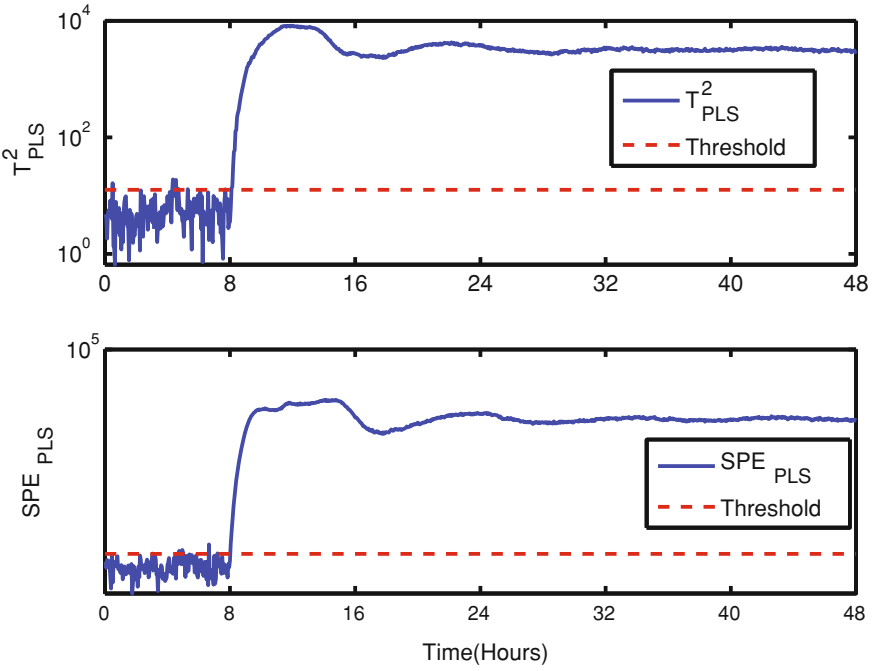


Fig. 6.3 PLS based fault detection results for fault scenario 1

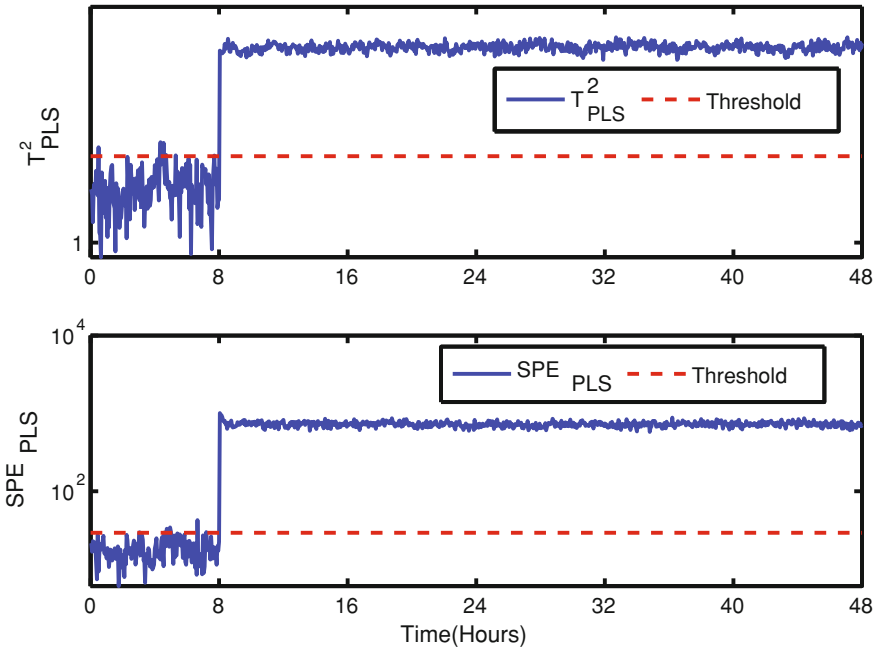


Fig. 6.4 PLS based fault detection results for fault scenario 4

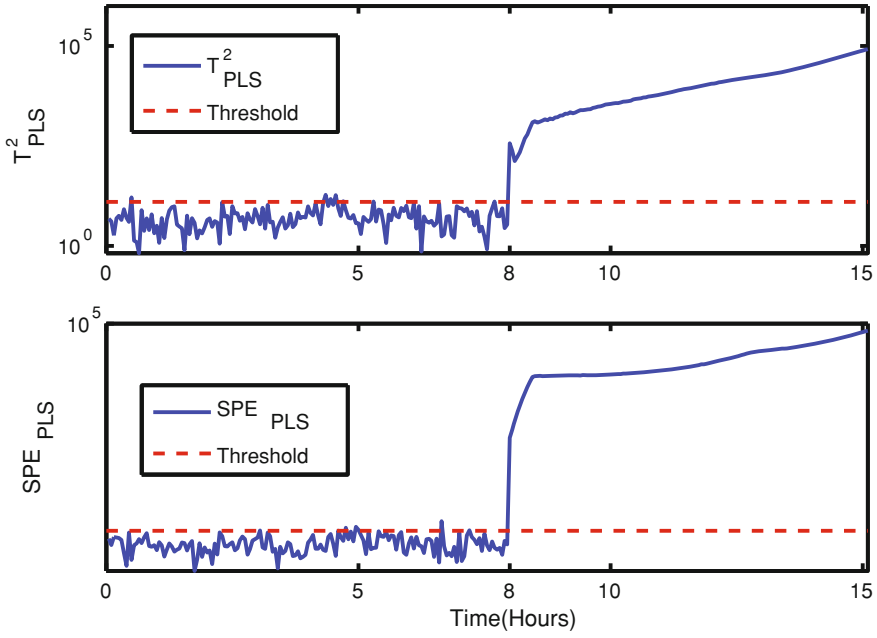


Fig. 6.5 PLS based fault detection results for fault scenario 6

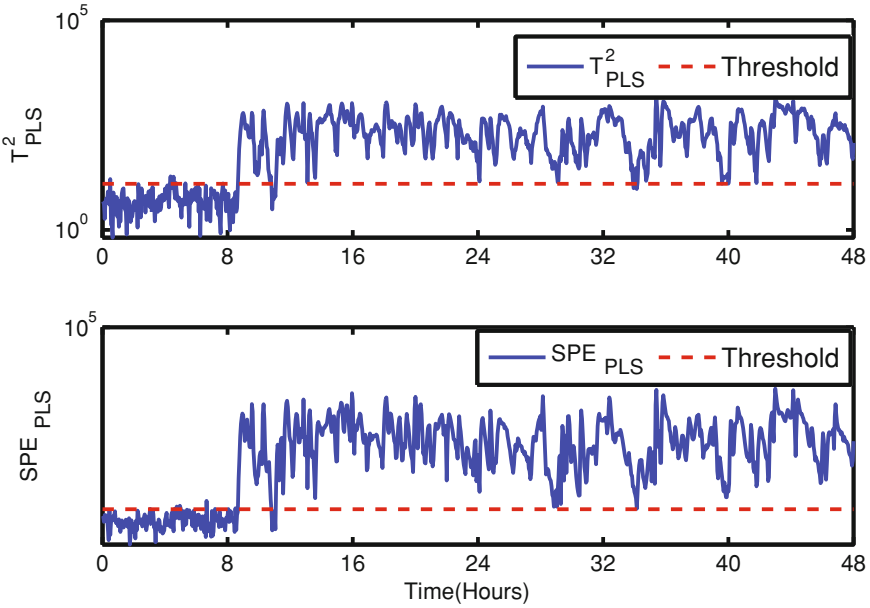


Fig. 6.6 PLS based fault detection results for fault scenario 11

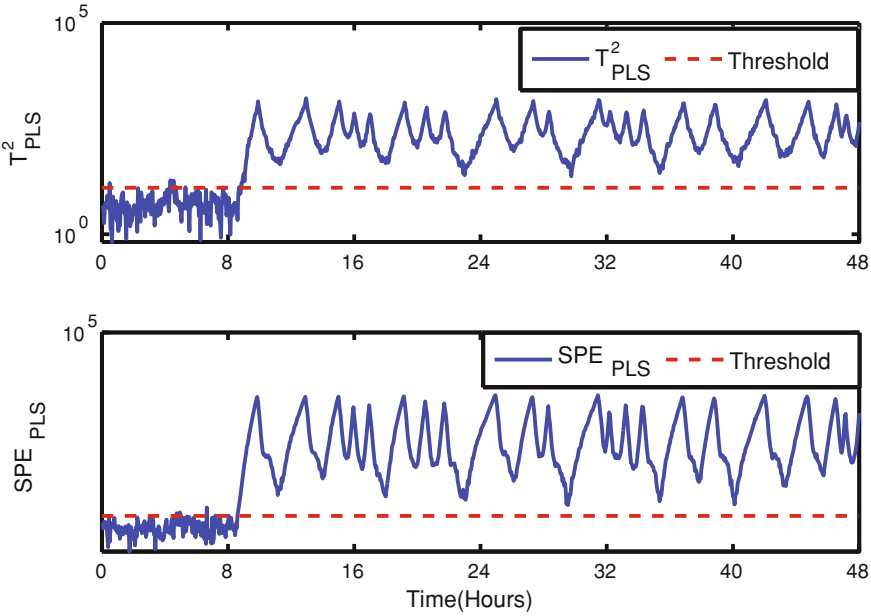


Fig. 6.7 PLS based fault detection results for fault scenario 20

6.6 Notes and References

PLS regression technique is a standard tool and widely used in chemometrics [12]. The first successful applications to fault diagnosis and process monitoring have been reported in [5, 8, 11]. There exists a number of variations of PLS regression algorithms [2–4]. In Algorithm 6.1, we have adopted the one given by Dayal and MacGregor in [2] for our discussion and study.

In recent years, remarkable efforts have been made to study PLS regression technique [6] and to improve its application to process monitoring and fault diagnosis. In [10], recursive PLS (RPLS) has been proposed. Li et al. have studied the geometric properties of PLS for process monitoring [6]. Zhou et al. have developed the so-called T-PLS (total PLS) approach [14] and demonstrated its excellence in the application to process monitoring and fault detection. It is worth to mention that in the T-PLS algorithm, in addition to running a standard PLS algorithm, SVDs (as the key computation of PCA) of some $(\kappa \times N)$- and $(m \times N)$-dimensional matrices are embedded.

Benchmark and comparison studies on PLS and successful application of PLS have been reported in [1, 7, 9, 13].

References

1. Chiang LH, Russell EL, Braatz RD (2001) Fault detection and diagnosis in industrial systems. Springer, London
2. Dayal B, MacGregor JF (1997) Improved PLS algorithms. J Chemometr 11:73–85
3. Helland IS (1988) On the structure of partial least squares regression. Commun stat simul comput 17:581–607
4. Höskuldsson A (1988) PLS regression methods. J Chemomet 2:211–228
5. Kresta J, MacGregor J, Marlin T (1991) Multivariate statistical monitoring of process operating performance. Canadian J Chem Eng 69:35–47
6. Li G, Qin SJ, Zhou D (2010) Geometric properties of partial least squares for process monitoring. Automatica 46(1):204–210
7. Li G, Alcala CF, Qin SJ, Zhou D (2011) Generalized reconstruction-baed contributions for output-relevant fault diagnosis with application to the tennessee eastman process. IEEE Trans Contr Syst Tech 19:1114–1127
8. MacGregor JF, Kourti T (1995) Statistical process control of multivariate processes. Contr Eng Practice 3:403–414
9. Peng KX, Zhang K, Li G, Zhou D (2013) Contribution rate plot for nonlinear quality-related fault diagnosis with application to the hot strip mill process. Contr Eng Practice 21:360–369
10. Qin SJ (1998) Recursive PLS algorithms for adaptive data modeling. Computers Chem Eng 22:503–514
11. Wise B, Gallagher N (1996) The process chemometrics approach to process monitoring and fault detection. J Process Control 6:329–348
12. Wold S, Sjöström M, Eriksson L (2001) PLS-regression: a basic tool of chemometrics. Chemomet Int Lab Syst 58:109–130
13. Yin S, Ding SX, Haghani A, Hao H, Zhang P (2012) A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. J Process Control 22:1567–1581
14. Zhou D, Li G, Qin SJ (2010) Total projection to latent structures for process monitoring. AIChE J 56:168–178

Chapter 7
Canonical Variate Analysis Based Process Monitoring and Fault Diagnosis

In the previous chapters, we have studied two standard MVA methods, PCA and PLS, and their applications to process monitoring and fault diagnosis. Canonical correlation analysis (CCA) is a further MVA method, which is also, in particular in the control community, known as canonical variate analysis (CVA) and recognized as a standard method for system identification. In this chapter, we first briefly introduce the essential ideas and computations of CCA and then discuss about their applications in process monitoring and fault diagnosis.

7.1 Introduction to CCA

CCA is a statistical method to analyze correlation relations between two random vectors. Suppose that $y \in \mathcal{R}^m$, $u \in \mathcal{R}^l$ are two random vectors satisfying

$$\begin{bmatrix} u \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathcal{E}(u) \\ \mathcal{E}(y) \end{bmatrix}, \begin{bmatrix} \Sigma_u & \Sigma_{uy} \\ \Sigma_{yu} & \Sigma_y \end{bmatrix} \right). \tag{7.1}$$

Let J, L define linear combinations of u and y. Roughly speaking, CCA deals with finding those linear combinations J, L that deliver the most closed correlations between u and y. As the basis for the correlation evaluation, matrix

$$\mathcal{K} = \Sigma_u^{-1/2} \Sigma_{uy} \Sigma_y^{-1/2}$$

is taken into account. Assume that

$$rank\left(\Sigma_{uy}\right) = rank\left(\mathcal{K}\right) = \kappa.$$

By an SVD of $\mathcal{K}$ we have

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_7, © Springer-Verlag London 2014

$$\mathcal{K} = R\Sigma V^T, \Sigma = \begin{bmatrix} diag(\sigma_1, \ldots, \sigma_\kappa) & 0 \\ 0 & 0 \end{bmatrix}, \sigma_1 \geq \cdots \geq \sigma_\kappa > 0 \tag{7.2}$$
$$R = \begin{bmatrix} r_1 \ \ldots \ r_l \end{bmatrix}, RR^T = I, V = \begin{bmatrix} v_1 \ \ldots \ v_m \end{bmatrix}, V^T V = I.$$

Let

$$J = \Sigma_u^{-1/2} R, J = \begin{bmatrix} J_1 \ \ldots \ J_l \end{bmatrix}, L = \Sigma_y^{-1/2} V, L = \begin{bmatrix} L_1 \ \ldots \ L_m \end{bmatrix}. \tag{7.3}$$

It is evident that

$$J^T \Sigma_u J = I, L^T \Sigma_y L = I \tag{7.4}$$
$$J^T \Sigma_{uy} L = \Sigma = \begin{bmatrix} diag(\sigma_1, \ldots, \sigma_\kappa) & 0 \\ 0 & 0 \end{bmatrix}. \tag{7.5}$$

Definition 7.1 Consider random vectors $y \in \mathcal{R}^m$, $u \in \mathcal{R}^l$ satisfying (7.1) and let $\mathcal{K} = R\Sigma V^T$, J, L be defined in (7.2) and (7.3), respectively. Then,

$$J_i = \Sigma_u^{-1/2} r_i, L_i = \Sigma_y^{-1/2} v_i, i = 1, \ldots, \kappa \tag{7.6}$$

are called canonical correlation vectors,

$$\eta_i = J_i^T u, \varphi_i = L_i^T y, i = 1, \ldots, \kappa \tag{7.7}$$

canonical correlation variables, and $\sigma_1, \ldots, \sigma_\kappa$ are called canonical correlation coefficients.

It is clear that J_1, L_1 define the mostly correlated linear combinations. The computation of J, L and the canonical correlation variables is summarized in the following algorithm.

Algorithm 7.1 *Computation of canonical correlation vectors and variables*

S1: Form $\mathcal{K} = \Sigma_u^{-1/2} \Sigma_{uy} \Sigma_y^{-1/2}$
S2: Do an SVD of $\mathcal{K} = U\Sigma V^T$
S3: Compute J, L *according to (7.3).*

In the practical applications, the covariance matrices given in (7.1) are estimated using the sampled data from u and y. A typical application of CCA technique is to check the correlation amongst two data sets.

7.2 CVA-Based System Identification

The application of the CVA to system identification was proposed by Larimore in 1983. In the next two decades since then, this topic drew strong research attention. In his pioneering work, Larimore proposed to apply CVA (CCA) technique to studying the correlations between the past and future process data sets and, based on them, to identifying a state space model of a (linear) dynamic system. In this section, we are going to describe the application of CCA (CVA) technique to the identification of a dynamic process in the steady state.

Let $Z_p \in \mathcal{R}^{\xi \times N}$, $Z_f \in \mathcal{R}^{\varsigma \times N}$ be two process data sets, collected from the process output variables (sensors) and input variables (actuators) in a time interval which is divided into the "past" and "future" periods. Z_p represents the past data set and Z_f the future one. We assume that there exist mappings P, M so that

$$X = P^T Z_p \tag{7.8}$$

$$MZ_f = QX + E \tag{7.9}$$

where E denotes the sample set of the noise and X the sample set of the process canonical correlation variables (state variables). Model (7.8), (7.9) and the associated data structure are motivated by the following observation, which will be dealt in the next chapter in more details.

Consider the state space representation of a process

$$x(k+1) = Ax(k) + Bu(k) + w(k), x(0) = x_0 \tag{7.10}$$

$$y(k) = Cx(k) + v(k) \tag{7.11}$$

where $x \in \mathcal{R}^n$ is the so-called state vector, x_0 the initial condition of the system, $u \in \mathcal{R}^l$ the input vector and $y \in \mathcal{R}^m$ the output vector, and $w \in \mathcal{R}^n$ and $v \in \mathcal{R}^m$ denote noise sequences. Matrices A, B, C are appropriately dimensioned real constant matrices. Model (7.10), (7.11) can also be equivalently written as

$$x(k+1) = A_K x(k) + Bu(k) + Ky(k), A_K = A - KC \tag{7.12}$$

$$y(k) = Cx(k) + e(k) \tag{7.13}$$

for Kalman filter gain matrix K that ensures that the eigenvalues of A_K are all located in the unit circle, where $e(k)$ is the so-called innovation sequence. It is straightforward from (7.12) that

$$x(k) = A_K^s x(k-s) + \sum_{i=1}^{s} A_K^{i-1} Bu(k-i) + \sum_{i=1}^{s} A_K^{i-1} Ky(k-i) \tag{7.14}$$

and thus for a large s

$$x(k) \approx P^T z_p, z_p(k) = \begin{bmatrix} y(k-s) \\ \vdots \\ y(k-1) \\ u(k-s) \\ \vdots \\ u(k-1) \end{bmatrix}, P^T = \begin{bmatrix} P_y & P_u \end{bmatrix} \tag{7.15}$$

$$P_y = \begin{bmatrix} A_K^{s-1} K & \ldots & A_K K & K \end{bmatrix}, P_u = \begin{bmatrix} A_K^{s-1} B & \ldots & A_K B & B \end{bmatrix}.$$

On the other hand, we have

$$\left(I - H_{K,y,s_f}\right) \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+s_f) \end{bmatrix} = \Gamma_{K,s_f} x(k) + H_{K,u,s_f} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+s_f) \end{bmatrix} + \begin{bmatrix} e(k) \\ e(k+1) \\ \vdots \\ e(k+s_f) \end{bmatrix},$$

$$H_{K,u,s_f} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ CB & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA_K^{s_f-1} B & \cdots & CB & 0 \end{bmatrix}$$

$$H_{K,y,s_f} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ CK & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA_K^{s_f-1} K & \cdots & CK & 0 \end{bmatrix}, \Gamma_{K,s_f} = \begin{bmatrix} C \\ CA_K \\ \vdots \\ CA_K^{s_f} \end{bmatrix}. \tag{7.16}$$

which is further written as

$$Mz_f(k) = Qx(k) + e_f(k), z_f(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k+s_f) \\ u(k) \\ \vdots \\ u(k+s_f) \end{bmatrix}, e_f(k) = \begin{bmatrix} e(k) \\ e(k+1) \\ \vdots \\ e(k+s_f) \end{bmatrix} \tag{7.17}$$

$$M = \begin{bmatrix} I - H_{K,y,s_f} & -H_{K,u,s_f} \end{bmatrix}, Q = \Gamma_{K,s_f}.$$

where s_f is some (large) integer. We see that $z_p(k)$ includes the process input and output data in the time period $[k-s, k-1]$ and is, thus, denoted by the "past" process measurements, while $z_f(k)$ is composed of the "future" process data in the

time period $[k, k+s_f]$. We suppose that $Z_p \in \mathcal{R}^{\xi\times N}$, $Z_f \in \mathcal{R}^{\varsigma\times N}$ are data sets that include respectively N samples generated from $z_p(k)$ and $z_f(k)$.

Next, we apply the CCA method to the identification of the mappings P, Q using the process data Z_p, Z_f. To this end, we assume that

$$\begin{bmatrix} z_p \\ z_f \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathcal{E}(z_p) \\ \mathcal{E}(z_f) \end{bmatrix}, \begin{bmatrix} \Sigma_{z_p} & \Sigma_{z_p z_f} \\ \Sigma_{z_f z_p} & \Sigma_{z_f} \end{bmatrix}\right)$$

and Z_p, Z_f are centered so that

$$\begin{bmatrix} \Sigma_{z_p} & \Sigma_{z_p z_f} \\ \Sigma_{z_f z_p} & \Sigma_{z_f} \end{bmatrix} \approx \frac{1}{N-1}\begin{bmatrix} Z_p Z_p^T & Z_p Z_f^T \\ Z_p^T Z_f & Z_f Z_f^T \end{bmatrix}.$$

By the CCA algorithm introduced in the last section, it turns out

$$P = \left(Z_p Z_p^T\right)^{-1/2} U(:, 1:n) \tag{7.18}$$

$$\begin{aligned}\left(\frac{Z_p Z_p^T}{N-1}\right)^{-1/2}\left(\frac{Z_p Z_f^T}{N-1}\right)\left(\frac{Z_f Z_f^T}{N-1}\right)^{-1/2} &= \left(Z_p Z_p^T\right)^{-1/2}\left(Z_p Z_f^T\right)\left(Z_f Z_f^T\right)^{-1/2} \\ &= U\Sigma V^T, \Sigma = \begin{bmatrix} diag(\sigma_1, \ldots, \sigma_n) & 0 \\ 0 & 0 \end{bmatrix}.\end{aligned} \tag{7.19}$$

Recall that $P^T z_p$ builds the vector of the canonical correlation variables, which is then defined as the state variable vector, that is,

$$x = P^T z_p. \tag{7.20}$$

As a result, an estimate for X is given by

$$X = P^T Z_p. \tag{7.21}$$

Note that the first m rows of Z_f satisfy

$$Z_f(1:m, :) = Q(1:m, :)X + E(1:m, :).$$

Hence, an LS estimate of $Q(1:m, :)$ is given by

$$Q(1:m, :) = Z_f(1:m, :)X^T\left(XX^T\right)^{-1}. \tag{7.22}$$

As a result,

$$C = Q(1:m, :)$$

and H_{K,u,s_f}, H_{K,y,s_f} can be determined by operations of $Q(1:m,:)$ and the sub-matrices in the identified matrix P. Finally, we are able to identify remaining sub-matrices of Q using an LS estimate based on the equation

$$MZ_f = QX + E.$$

We would like to make the following remarks:

- In the context of state space representations, the state vector given by (7.20) can be different from the one defined in (7.10). In that case, there exists a regular state transformation.
- The existence of noise E in model (7.9) can yield difficulty to determine the system order n, that is, the number of non-zero singular values in the SVD (7.19) is larger than n. In this case, several methods are available for determining the right system order, among them the so-called AIC index (Akaike information criterion) is frequently used.
- In practice, model (7.9) is mostly presented in a simple form

$$\bar{Y}_f = QX + E \tag{7.23}$$

after a data normalization, where $\bar{Y}_f \in \mathcal{R}^{\varsigma \times N}$ represents N normalized samples of the output data

$$\bar{y}_f(k) = \begin{bmatrix} \bar{y}(k) \\ \vdots \\ \bar{y}(k+s_f) \end{bmatrix} \in \mathcal{R}^{\varsigma}, \varsigma = \left(s_f+1\right) m.$$

A widely adopted normalization is

$$\bar{y}_f(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k+s_f) \end{bmatrix} - \Sigma_{y_f u_f} \Sigma_{u_f u_f}^{-1} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+s_f) \end{bmatrix}$$

where $\Sigma_{y_f u_f} \Sigma_{u_f u_f}^{-1}$ is approximated by

$$\Sigma_{y_f u_f} \Sigma_{u_f u_f}^{-1} \approx \left(Y_f U_f^T\right) \left(U_f U_f^T\right)^{-1}$$

and Y_f, U_f are the sub-matrices of Z_f including N samples from

$$z_f(k) = \begin{bmatrix} y_f(k) \\ u_f(k) \end{bmatrix}, y_f(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k+s_f) \end{bmatrix}, u_f(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+s_f) \end{bmatrix}.$$

Thus,

$$\bar{Y}_f = Y_f - \left(Y_f U_f^T\right)\left(U_f U_f^T\right)^{-1} U_f. \tag{7.24}$$

On the assumption of model (7.8) and (7.23), the following algorithm can be used for the system identification.

Algorithm 7.2 *CVA-based system identification*

S1: Normalize the process data, which delivers Z_p, $\bar{Y}_f$
S2: Do an SVD

$$\left(Z_p Z_p^T\right)^{-1/2}\left(Z_p \bar{Y}_f^T\right)\left(\bar{Y}_f \bar{Y}_f^T\right)^{-1/2} = U \Sigma V^T \tag{7.25}$$

S3: Determine the system order n
S4: Form P according to (7.18)
S5: Compute X according to (7.21)
S6: Find an LS estimate of Q based on (7.23).

7.3 Applications to Process Monitoring and Fault Detection

Having introduced the theoretical background of the CCA/CVA technique, we now address the potential applications of the CCA/CVA technique to process monitoring and fault diagnosis.

7.3.1 Process Monitoring

It is well-known that the state variables reflect the process operation and hence provide process engineers with real-time and important information for process monitoring. On the assumption that the process (input and output) variables and noises are normally distributed, a CVA-based T^2 statistic can be used for the process monitoring purpose.

Let

$$z_{s-1}(k) = \begin{bmatrix} y(k-s+1) \\ \vdots \\ y(k) \\ u(k-s+1) \\ \vdots \\ u(k) \end{bmatrix}$$

be normalized, that is, $z_{s-1}(k) \sim \mathcal{N}(0, \Sigma_z)$. Remember that by Algorithm 7.2

$$P^T \Sigma_z P \approx P^T \left(Z_p Z_p^T \right) P = U^T(:, 1:n) U(:, 1:n) = I_{n \times n}.$$

It follows from (7.20) and the discussion in Sect. 3.3.2 that we can apply T^2 statistic

$$T^2 = z_{s-1}^T(k) P P^T z_{s-1}(k) \tag{7.26}$$

and a threshold

$$J_{th,T^2} = \frac{n\left(N^2-1\right)}{N(N-n)} \mathcal{F}_\alpha(n, N-n) \tag{7.27}$$

with a given significance level α to detect changes (faults) in the state space.

7.3.2 Fault Detection Schemes

The first fault detection scheme is based on an immediate utilization of the identified model (7.8) and (7.23), which builds a (steady) input and output relation described by

$$\bar{y}_{s_f}(k+s_f+1) = Q P^T z_{s-1}(k) + e_{s_f}(k+s_f+1) \tag{7.28}$$

$$\bar{y}_{s_f}(k+s_f+1) = \begin{bmatrix} \bar{y}(k+1) \\ \vdots \\ \bar{y}(k+s_f+1) \end{bmatrix}, e_{s_f}(k+s_f+1) = \begin{bmatrix} e(k+1) \\ \vdots \\ e(k+s_f+1) \end{bmatrix}.$$

Therefore, it is reasonable to define a so-called residual vector $r(k)$,

$$r(k) = \bar{y}_{s_f}(k+s_f+1) - Q P^T z_{s-1}(k) \in \mathcal{R}^{(s_f+1)m} \tag{7.29}$$

for the detection purpose. In order to apply T^2 or Q statistic of $r(k)$, it is necessary to estimate the covariance matrix of the process noise $e_{s_f}(k+s_f+1)$. This can be realized as follows:

$$\begin{aligned} &E = \bar{Y}_f - QX, X = P^T Z_p \Longrightarrow \\ &\Sigma_e \approx \frac{1}{N-1} \left(\bar{Y}_f - Q P^T Z_p \right) \left(\bar{Y}_f - Q P^T Z_p \right)^T. \end{aligned} \tag{7.30}$$

As a result, we are able, for instance, to use T^2 statistic defined by

$$J_{T^2} = r^T(k) \Sigma_e^{-1} r(k). \tag{7.31}$$

The idea behind the second fault detection scheme is the utilization of the relations amongst the canonical correlation vectors. Recall that by CCA P, L can be found:

$$P = \left(Z_p Z_p^T\right)^{-1/2} U(:, 1:n), L = \left(\bar{Y}_f \bar{Y}_f^T\right)^{-1/2} V(:, 1:n) \tag{7.32}$$

$$\begin{aligned} \left(Z_p Z_p^T\right)^{-1/2} \left(Z_p \bar{Y}_f^T\right) \left(\bar{Y}_f \bar{Y}_f^T\right)^{-1/2} &= U \Sigma V^T \\ U^T(:, 1:n) \left(Z_p Z_p^T\right)^{-1/2} \left(Z_p \bar{Y}_f^T\right) \left(\bar{Y}_f \bar{Y}_f^T\right)^{-1/2} V(:, 1:n) &= diag(\sigma_1, \ldots, \sigma_n) \\ \Longleftrightarrow P^T Z_p \bar{Y}_f^T L &= diag(\sigma_1, \ldots, \sigma_n) \end{aligned} \tag{7.33}$$

and moreover

$$P^T Z_p Z_p^T P = I, L^T \bar{Y}_f \bar{Y}_f^T L = I.$$

Hence, (7.33) leads to

$$\Omega P^T Z_p = L^T \bar{Y}_f, \Omega = diag(\sigma_1, \ldots, \sigma_n). \tag{7.34}$$

Based on this relation, we define a residual vector

$$r(k) = L^T \bar{y}_{s_f}(k + s_f + 1) - \Omega P^T z_{s-1}(k) \tag{7.35}$$

for building a test statistic. Note that the covariance matrix of $r(k)$ can be estimated by

$$\begin{aligned} &\left(L^T \bar{Y}_f - \Omega P^T Z_p\right) \left(L^T \bar{Y}_f - \Omega P^T Z_p\right)^T \\ &= L^T \bar{Y}_f \bar{Y}_f^T L + \Omega^2 P^T Z_p Z_p^T P - 2\Omega P^T Z_p \bar{Y}_f^T L = I - \Omega^2. \end{aligned} \tag{7.36}$$

As a result, the T^2 statistic can be formed by

$$J_{T^2} = (N-1) r^T(k) \left(I - \Omega^2\right)^{-1} r(k). \tag{7.37}$$

The design steps of the second fault detection scheme is summarized in the following algorithm.

Table 7.1 Selected outputs

Block name	Variable name	Tag
Separator	Separator temperature	XMEAS(11)
	Separator level	XMEAS(12)
	Separator pressure	XMEAS(13)
	Separator underflow	XMEAS(14)
Stripper	Stripper level	XMEAS(15)
	Stripper pressure	XMEAS(16)
	Stripper underflow	XMEAS(17)
	Stripper temperature	XMEAS(18)
	Stripper steam flow	XMEAS(19)

Algorithm 7.3 *A CVA-based fault detection scheme*

S1: Normalize the process data, which delivers $Z_p, \bar{Y}_f$
S2: Do an SVD

$$\left(Z_p Z_p^T\right)^{-1/2} \left(Z_p \bar{Y}_f^T\right) \left(\bar{Y}_f \bar{Y}_f^T\right)^{-1/2} = U \Sigma V^T$$

S3: Determine the system order n
S4: Form P, L *according to (7.32)*
S5: Compute $\left(I - \Omega^2\right)^{-1}$ *and build the test statistic (7.37)*
S6: Determine the threshold.

7.4 Case Study: Application to TEP

This section demonstrates the application of the CVA technique to fault detection on the TEP simulator. As remarked in Sect. 2.3, the whole data set consisting of 52 variables with 960 samples may cause computation overhead and memory problem for testing the CVA based detection approach. This motivates the effort to divide the process variables into several groups. As shown in Table 2.4, explicit input-output information is given in each group.

7.4.1 Test Setup and Training

In our study, the manipulated variables (XMV 1–11) are applied as the inputs, two blocks of outputs are selected, which are listed in Table 7.1. The first block includes

Table 7.2 Parameters in the training phase

Block number	m	l	s_f	s_p	N	n
1	4	11	7	7	473	23
2	5	11	7	7	473	28

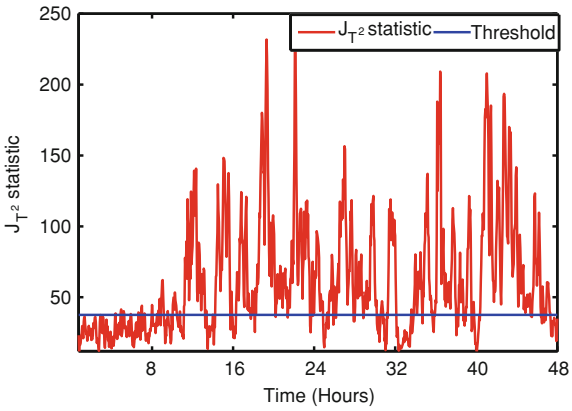


Fig. 7.1 Detection result of fault scenario 12 with CVA (FDR = 0.7653)

the measurements in the separator and the second one the measurements in the stripper.

In the training phase, two types of input-output patterns are used for training with CVA. The key training parameters are given Table 7.2. Referring to a work by Odiowei and Cao, $s_f = s_p$ is predefined and the process order is achieved in a PCA-like way. For the applied T^2 statistics with a significance level 0.05, the thresholds for the two measurement groups are set to be 37.5522 and 44.6938, respectively.

7.4.2 Test Results and a Comparison Study

In order to illustrate the application of CVA to fault diagnosis, the test results are evaluated in comparison with the PCA and PLS methods introduced in the previous chapters. As a representative example, detection of fault scenario 12 is demonstrated in details. For the block 1 output variables, the result of CVA based fault detection is presented in Fig. 7.1, while the results achieved by PCA and PLS are shown in Figs. 7.2 and 7.3, respectively. It is clear that the application of CVA based fault detection leads to a higher fault detection rate (FDR), which is computed as follows:

$$FDR = \frac{\text{number of those samplings, at which } J > J_{th} \text{ as } f \neq 0}{\text{total number of the samplings}}.$$

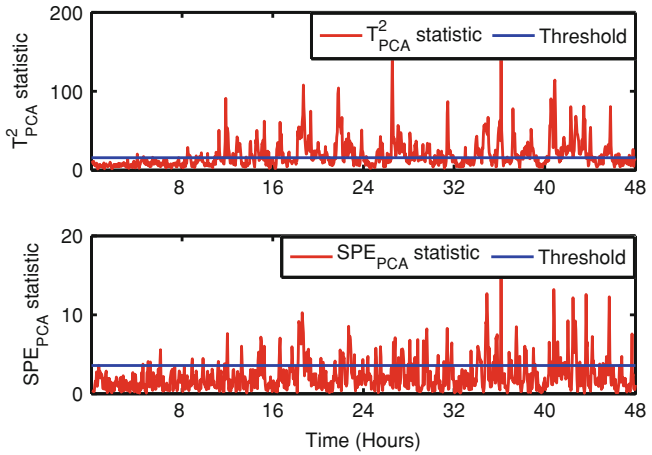


Fig. 7.2 Detection result of fault scenario 12 with PCA (FDR = 0.5655)

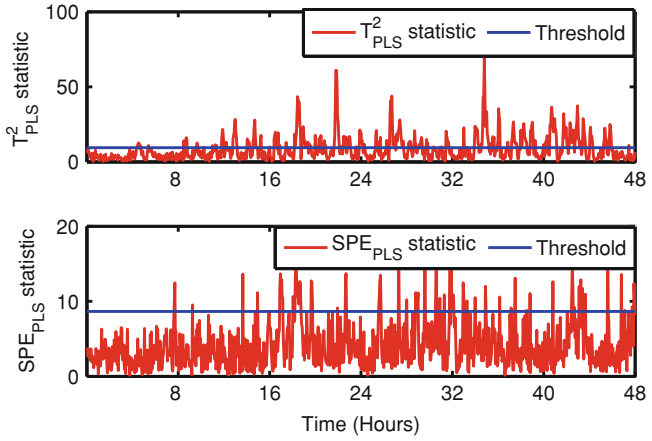


Fig. 7.3 Detection result of fault scenario 12 with PLS (FDR = 0.4494)

The test results for the block 2 output variables are respectively shown in Figs. 7.4, 7.5 and 7.6. In Tables 7.3 and 7.4, the results of detecting further faults by CVA, PCA and PLS are summarized and compared.

7.5 Notes and References

CCA is a standard MVA method. The brief introduction to CCA in the first section of this chapter follows from a summary of Chap. 15 in [1].

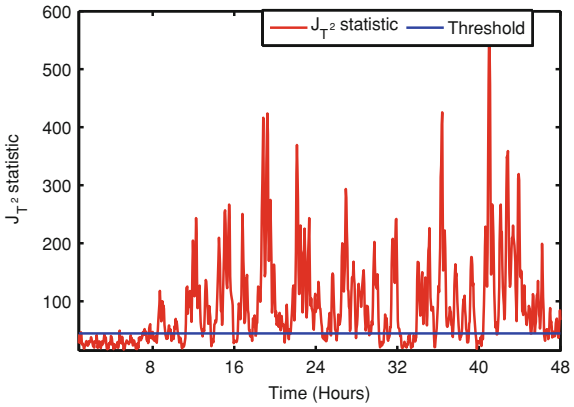


Fig. 7.4 Detection result of fault scenario 12 with CVA (FDR = 0.7853)

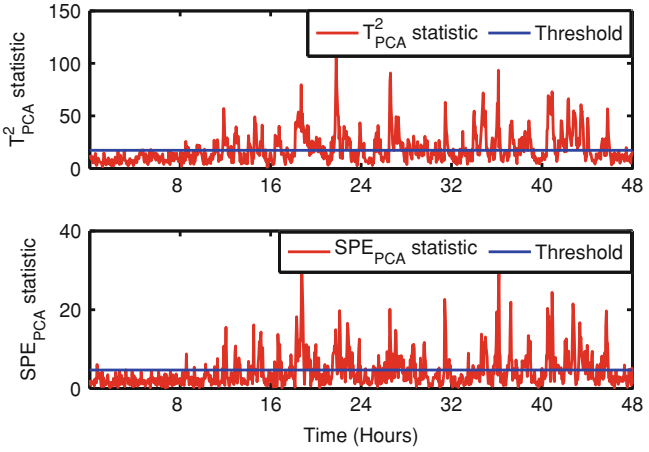


Fig. 7.5 Detection result of fault scenario 12 with PCA (FDR = 0.5605)

In his pioneering work in 1983 [2], Larimore introduced CCA technique to the identification of state space system models, and called this technique CVA. Nowadays CVA is widely accepted as a standard system identification method [3, 4]. The CVA-based system identification presented in Sect. 7.2 is standard. For more details, the reader is referred to, for instance, [4, 5]. Larimore proposed model (7.23) with the normalization computation (7.24) and further to apply the AIC to the determination of the system order [6]. Since then, the AIC and model (7.23) have been widely adopted in the theoretical study and real applications [4, 7].

The first applications of CVA technique to process monitoring, fault detection and diagnosis have been reported in [5, 8–10]. An extended and detailed description of

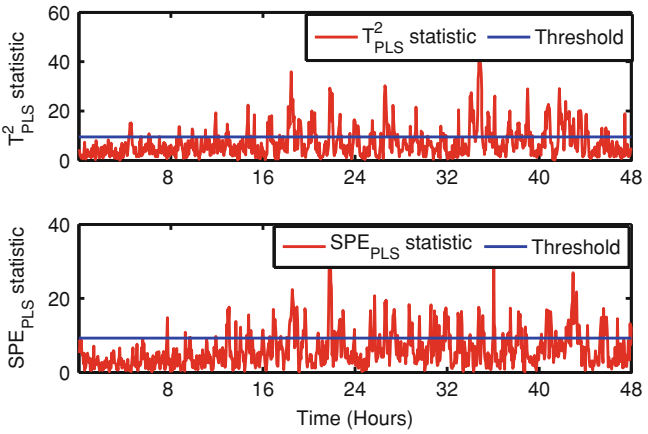


Fig. 7.6 Detection result of fault scenario 12 with PLS (FDR = 0.4020)

Table 7.3 FDR comparison between CVA, PCA and PLS: output block 1

Fault number	CVA	PCA	PLS
1	1	0.9975	0.9988
2	0.9975	0.9925	0.9900
4	1	0.9988	0.9988
6	1	0.9930	0.9930
7	1	0.9988	0.9988
8	0.9925	0.9913	0.9913
10	0.9629	0.8527	0.7953
11	0.9838	0.9773	0.9638
13	0.9925	0.9888	0.9863
14	0.9975	0.8989	0.8964
17	0.9238	0.9536	0.9213
18	0.3496	0.3027	0.3271
19	0.9975	0.9975	0.9963
20	0.9913	0.9838	0.9768

the CVA technique to process monitoring, as presented in Sect. 7.3, can be found in [5, 8], while the second fault detection scheme has been reported in [10].

In our case study in Sect. 7.4, the selection of s_f, s_p and the determination of the process order have been realized using the method given by Odiowei and Cao [11]. It should be pointed out that the FDR computation adopted in the case study is, thanks to its simplicity, popular in practice [12], although it does not follow the classic FDR definition in the statistic framework.

Table 7.4 FDR comparison between CVA, PCA and PLS: output block 2

Fault number	CVA	PCA	PLS
1	0.9975	0.9988	0.9988
2	0.9963	0.9950	0.9900
4	1	0.9988	0.9988
6	1	0.9930	0.9930
7	1	0.9988	0.9988
8	0.9888	0.9913	0.9913
10	0.9975	0.9588	0.8290
11	0.9838	0.9713	0.9613
13	0.9888	0.9900	0.9863
14	0.9950	0.8959	0.9001
17	0.9513	0.9638	0.9039
18	0.3720	0.3059	0.2896
19	0.9975	0.9975	0.9775
20	0.9900	0.9838	0.9763

References

1. Härdle WK, Simar L (2012) Applied multivariate statistical analysis, 3rd edn. Springer, Berlin Heidelberg
2. Larimore WE (1983) System identification reduced-order filtering and modeling via canonical variate analysis. In: Proceedings of the American control conference (ACC), pp 445–451
3. Katayama T (2005) Subspace methods for system identification. Springer-Verlag, London
4. Huang B, Kadali R (2008) Dynamic modelling, predictive control and performance monitoring, a data-driven subspace approach. Springer-Verlag, London
5. Russell EL, Chiang L, Braatz RD (2000) Data-driven techniques for fault detection and diagnosis in chemical processes. Springer-Verlag, London
6. Larimore W (1990) Canonical variate analysis in identification, filtering, and adaptive control. In: Proceedings of the 29th IEEE CDC, pp 596–604
7. Wang J, Qin SJ (2002) A new subspace identification approach based on principle component analysis. J Process Control 12:841–855
8. Russell EL, Chiang LH, Braatz RD (2000) Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Chemom Intell Lab Syst 51: 81–93
9. Simoglou A, Martin E, Morris A (2000) Multivariate statistical process control of an industrial fluidised-bed reactor. Control Eng Pract 8:893–909
10. Juricek BC, Seborg DE, Larimore WE (2004) Fault detection using canonical variate analysis. Ind Eng Chem Res 43:458–474
11. Odiowei P, Cao Y (2010) Nonlinear dynamic process monitoring using canonical variate analysis and kernel density estimations. IEEE Trans Ind Inform 6:36–45
12. Yin S, Ding SX, Haghani A, Hao H, Zhang P (2012) A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. J Process Control 22:1567–1581

Part III
Data-driven Design of Fault Diagnosis Systems for Dynamic Processes

Chapter 8
Introduction, Preliminaries and I/O Data Set Models

8.1 Introduction

Process monitoring and fault diagnosis for dynamic processes are currently receiving considerably increasing attention in the application and research domains. Thanks to their intimate relations to automatic control systems, model-based schemes are widely accepted as a powerful technology in dealing with process monitoring and fault diagnosis for dynamic processes. In comparison, the major focus in the field of data-driven process monitoring and fault diagnosis is on the extension of standard MVA schemes like PCA and PLS. In order to deal with the existing dynamics in the process under consideration, dynamic PCA/PLS, recursive implementation of PCA/PLS, fast moving window PCA and multiple-mode PCA have been developed in recent years. The MVA technique is generally applied to static or dynamic processes in the steady state and delivers optimal performance for high level process monitoring and diagnosis in large-scale systems. Differently, the model-based fault detection and isolation (FDI) technique, thanks to the application of advanced system and control theory, provides a more efficient and powerful tool to investigate FDI issues in highly dynamic systems and control loops, which are generally located at the process level. The possible high FDI performance is often achieved at the cost of a highly complex process modelling and, based on it, a sophisticated FDI system design procedure.

Recently, study on subspace identification methods (SIM) based FDI system design draws remarkable research attention, in which different FDI schemes have been proposed. Due to their similarity to the SIM in collecting, structuring and processing process data, these schemes are viewed as SIM-based methods. On the other hand, the basic idea behind these schemes is a *direct construction* of an FDI system utilizing the collected process data *without* explicitly identifying a system model. This is the major difference to a standard SIM whose task is to identify the system model and the associated system matrices. In this context, these schemes are called data-driven FDI system design and mainly characterized by

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_8, © Springer-Verlag London 2014

- their significantly simplified design procedure
  Without a sophisticated system design, FDI performance similar to the one delivered by a model-based FDI method can be achieved
- their ability in dealing with dynamic processes
  These schemes are powerful in addressing FDI in dynamic systems and devices with embedded control loops, which are typically located at the lower process level.

Although the major research activities in this research field have begun first in recent years, numerous design schemes and successful applications have been reported. Rapid development in research and wide applications in practice are also expected in this thematic field in the next years. In this and the next two chapters, we are going to

- introduce the basic ideas and major schemes of the data-driven design of FDI systems that are applicable for dynamic processes
- sketch a framework of SIM based data-driven design of process monitoring and fault diagnosis systems and
- present some recent results in this framework.

For our purpose, we shall, in this chapter, first review some essential ideas and schemes in the model-based FDI framework, which is helpful and, in part, necessary for our study on the data-driven design of process monitoring and fault diagnosis systems by means of the SIM technique. It will be then followed by a study on the different forms of input–output (I/O) data models.

8.2 Preliminaries and Review of Model-Based FDI Schemes

Roughly speaking, the major differences between the model-based FDI and MVA methods lie in

- the representation forms of the process knowledge and
- treating the nominal system dynamics with respect to the process inputs.

In the model-based FDI framework, mathematical models are the essential representation form of the process knowledge, and the model-based residual generation builds the major work to address the process dynamics. These two topics are the focus of our study in the sequel.

8.2.1 System Models

8.2.1.1 Description of Nominal System Behavior

A dynamic system can be described in different ways. The so-called linear time invariant (LTI) system model offers the simplest from and thus widely used in research

and application domains. We call disturbance-free and fault-free systems nominal and suppose that the nominal systems are LTI. There are two standard mathematical model forms for LTI systems: the transfer matrix and the state space representation. Below, they will be briefly introduced.

Roughly speaking, a transfer matrix is an input–output description of the dynamic behavior of an LTI system in the frequency domain. Throughout this book, notation $G_{yu}(z)$ is used for presenting a transfer matrix from the input vector $u \in \mathcal{R}^l$ to the output vector $y \in \mathcal{R}^m$, that is,

$$y(z) = G_{yu}(z)u(z). \tag{8.1}$$

It is assumed that $G_{yu}(z)$ is a proper real-rational matrix. We use z to denote the complex variable of z-transform for discrete-time signals.

The standard form of the state space representation of a discrete-time LTI system is

$$x(k+1) = Ax(k) + Bu(k), x(0) = x_0 \tag{8.2}$$

$$y(k) = Cx(k) + Du(k) \tag{8.3}$$

where $x \in \mathcal{R}^n$ is called the state vector, x_0 the initial condition of the system, $u \in \mathcal{R}^l$ the input vector and $y \in \mathcal{R}^m$ the output vector. Matrices A, B, C, D are appropriately dimensioned real constant matrices.

State space models can be either directly achieved by modelling or derived based on a transfer matrix. The latter is called a state space realization of $G_{yu}(z) = C(zI - A)^{-1}B + D$ and denoted by

$$G_{yu}(z) = (A, B, C, D) \text{ or } G_{yu}(z) = \begin{bmatrix} A & B \\ C & D \end{bmatrix}. \tag{8.4}$$

In general, we assume that (A, B, C, D) is a minimal realization of $G_{yu}(z)$.

8.2.1.2 Coprime Factorization Technique

Coprime factorization of a transfer function (matrix) gives a further system representation form which will be intensively used in our subsequent study. Roughly speaking, a coprime factorization over $\mathcal{RH}_\infty$ is to factorize a transfer matrix into two stable and coprime transfer matrices.

Definition 8.1 Two stable transfer matrices $\hat{M}(z)$, $\hat{N}(z)$ are called left coprime if there exist two stable transfer matrices $\hat{X}(z)$ and $\hat{Y}(z)$ such that

$$\begin{bmatrix} \hat{M}(z) & \hat{N}(z) \end{bmatrix} \begin{bmatrix} \hat{X}(z) \\ \hat{Y}(z) \end{bmatrix} = I. \tag{8.5}$$

Similarly, two stable transfer matrices $M(z), N(z)$ are right coprime if there exist two stable matrices $Y(z), X(z)$ such that

$$\begin{bmatrix} X(z) & Y(z) \end{bmatrix} \begin{bmatrix} M(z) \\ N(z) \end{bmatrix} = I. \tag{8.6}$$

Let $G(z)$ be a proper real-rational transfer matrix. The left coprime factorization (LCF) of $G(z)$ is a factorization of $G(z)$ into two stable and coprime matrices which will play a key role in designing the so-called residual generator. To complete the notation, we also introduce the right coprime factorization (RCF), which is however only occasionally applied in our study.

Definition 8.2 $G(z) = \hat{M}^{-1}(z)\hat{N}(z)$ with the left coprime pair $\left(\hat{M}(z), \hat{N}(z)\right)$ is called LCF of $G(z)$. Similarly, RCF of $G(z)$ is defined by $G(z) = N(z)M^{-1}(z)$ with the right coprime pair $(M(z), N(z))$.

Below, we present a lemma that provides us with a state space computation algorithm of $\left(\hat{M}(z), \hat{N}(z)\right)$, $(M(z), N(z))$ and the associated pairs $\left(\hat{X}(z), \hat{Y}(z)\right)$ and $(X(z), Y(z))$.

Lemma 8.1 *Suppose $G(z)$ is a proper real-rational transfer matrix with a state space realization (A, B, C, D), and it is stabilizable and detectable. Let F and L be so that $A + BF$ and $A - LC$ are Schur matrices (that is, their eigenvalues are inside the unit circle on the complex plane), and define*

$$\hat{M}(z) = (A - LC, -L, C, I), \hat{N}(z) = (A - LC, B - LD, C, D) \tag{8.7}$$

$$M(z) = (A + BF, B, F, I), N(z) = (A + BF, B, C + DF, D) \tag{8.8}$$

$$\hat{X}(z) = (A + BF, L, C + DF, I), \hat{Y}(z) = (A + BF, -L, F, 0) \tag{8.9}$$

$$X(z) = (A - LC, -(B - LD), F, I), Y(z) = (A - LC, -L, F, 0). \tag{8.10}$$

Then

$$G(z) = \hat{M}^{-1}(z)\hat{N}(z) = N(z)M^{-1}(z) \tag{8.11}$$

are the LCF and RCF of $G(z)$, respectively. Moreover, the so-called Bezout identity holds

$$\begin{bmatrix} X(z) & Y(z) \\ -\hat{N}(z) & \hat{M}(z) \end{bmatrix} \begin{bmatrix} M(z) & -\hat{Y}(z) \\ N(z)) & \hat{X}(z) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{8.12}$$

Note that

$$r(z) = \begin{bmatrix} -\hat{N}(z) & \hat{M}(z) \end{bmatrix} \begin{bmatrix} u(z) \\ y(z) \end{bmatrix} \tag{8.13}$$

is a dynamic system with the process input and output vectors u, y as its inputs and the residual vector r as its output. It is called residual generator. In some literature, $\begin{bmatrix} -\hat{N}(z) & \hat{M}(z) \end{bmatrix}$ is also called kernel representation of system (8.2), (8.3).

8.2.1.3 Representations of Systems With Disturbances

Disturbances around the process under consideration, unexpected changes within the technical process as well as measurement and process noises are often modeled as unknown input vectors. We denote them by d, ν or η and integrate them into the state space model (8.2), (8.3) or input–output model (8.1) as follows:

- state space representation

$$x(k+1) = Ax(k) + Bu(k) + E_d d(k) + \eta(k) \tag{8.14}$$
$$y(k) = Cx(k) + Du(k) + F_d d(k) + \nu(k) \tag{8.15}$$

  with E_d, F_d being constant matrices of compatible dimensions, $d \in \mathcal{R}^{k_d}$ is a deterministic unknown input vector, $\eta \in \mathcal{R}^{k_\eta}$, $\nu \in \mathcal{R}^{k_\nu}$ are, if no additional remark is made, white, normal distributed noise vectors with $\eta \sim \mathcal{N}(0, \Sigma_\eta)$, $\nu \sim \mathcal{N}(0, \Sigma_\nu)$.
- input–output model

$$y(z) = G_{yu}(z)u(z) + G_{yd}(z)d(z) + G_{y\nu}(z)\nu(z) \tag{8.16}$$

  where $G_{yd}(z)$, $G_{y\nu}(z)$ are known.

8.2.1.4 Modeling of Faults

There exists a number of ways of modeling faults. Extending model (8.16) to

$$y(z) = G_{yu}(z)u(z) + G_{yd}(z)d(z) + G_{yf}(z)f(z) \tag{8.17}$$

is a widely adopted one, where $f \in \mathcal{R}^{k_f}$ is a unknown vector that represents all possible faults and will be zero in the fault-free case, $G_{yf}(z) \in \mathcal{LH}_\infty$ is a known transfer matrix. Throughout this book, f is assumed to be a deterministic time function. No further assumption on it is made, provided that the type of the fault is not specified.

Suppose that a minimal state space realization of (8.17) is given by

$$x(k+1) = Ax(k) + Bu(k) + E_d d(k) + E_f f(k) \tag{8.18}$$
$$y(k) = Cx(k) + Du(k) + F_d d(k) + F_f f(k) \tag{8.19}$$

with known matrices E_f, F_f. Then we have

$$G_{yf}(z) = F_f + C\left(zI - A\right)^{-1} E_f. \tag{8.20}$$

It becomes evident that E_f, F_f indicate the place where a fault occurs and its influence on the system dynamics. It is the state of the art that faults are divided into three categories:

- sensor faults f_S: these are faults that directly act on the process measurement
- actuator faults f_A: these faults cause changes in the actuator
- process faults f_P: they are used to indicate malfunctions within the process.

A sensor fault is often modeled by setting $F_f = I$, that is,

$$y(k) = Cx(k) + Du(k) + F_d d(k) + f_S(k) \tag{8.21}$$

while an actuator fault by setting $E_f = B$, $F_f = D$, which leads to

$$x(k+1) = Ax(k) + B\,(u(k) + f_A) + E_d d(k) \tag{8.22}$$

$$y(k) = Cx(k) + D\,(u(k) + f_A) + F_d d(k). \tag{8.23}$$

Depending on their type and location, process faults can be modeled by $E_f = E_P$ and $F_f = F_P$ for some E_P, F_P. For a system with sensor, actuator and process faults, we define

$$f = \begin{bmatrix} f_A \\ f_P \\ f_S \end{bmatrix}, E_f = \begin{bmatrix} B & E_P & 0 \end{bmatrix}, F_f = \begin{bmatrix} D & F_P & I \end{bmatrix} \tag{8.24}$$

and apply (8.18), (8.19) to represent the system dynamics.

Due to the way how they affect the system dynamics, the faults described by (8.18), (8.19) are called additive faults. It is very important to note that the occurrence of an additive fault will not affect the system stability, independent of the system configuration. Typical additive faults met in practice are, for instance, an offset in sensors and actuators or a drift in sensors. The former can be described by a constant, while the latter by a ramp.

In practice, malfunctions in the process or in the sensors and actuators often cause changes in the model parameters. They are called multiplicative faults and generally modeled in terms of parameter changes.

8.2.2 Model-Based Residual Generation Schemes

Next, we introduce some standard model- and observer-based residual generation schemes.

8.2.2.1 Fault Detection Filter

Fault detection filter (FDF) is the first type of observer-based residual generators proposed by Beard and Jones in the early 1970s. Their work marked the beginning of a stormy development of model-based FDI techniques.

Core of an FDF is a full-order state observer

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L\left(y(k) - C\hat{x}(k) - Du(k)\right) \tag{8.25}$$

which is constructed on the basis of the nominal system model $G_{yu}(z) = C(zI - A)^{-1}B + D$. Built upon (8.25), the residual is simply defined by

$$r(k) = y(k) - \hat{y}(k) = y(k) - C\hat{x}(k) - Du(k). \tag{8.26}$$

Introducing variable

$$e(k) = x(k) - \hat{x}(k)$$

yields, on the assumption of process model (8.18), (8.19),

$$e(k+1) = (A - LC)e(k) + (E_d - LF_d)\,d(k) + \left(E_f - LF_f\right)f(k) \tag{8.27}$$

$$r(k) = Ce(k) + F_d d(k) + F_f f(k). \tag{8.28}$$

It is evident that $r(k)$ has the characteristic features of a residual when the observer gain matrix L is chosen so that $A - LC$ is stable.

The advantages of an FDF lie in its simple construction form and, for the reader who is familiar with the modern control theory, in its intimate relationship with the state observer design and especially with the well-established robust control theory by designing robust residual generators.

We see that the design of an FDF is in fact the determination of the observer gain matrix L. To increase the degree of design freedom, we can switch a matrix to the output estimation error $y(z) - \hat{y}(z)$, that is

$$r(z) = V\left(y(z) - \hat{y}(z)\right). \tag{8.29}$$

A disadvantage of FDF scheme is the online implementation of the full-order state observer, since in many practical cases a reduced order observer can provide us with the same or similar performance but with less online computation. This is one of the motivations for the development of Luenberger type residual generators, also called diagnostic observers.

8.2.2.2 Diagnostic Observer Scheme

The diagnostic observer (DO) is, thanks to its flexible structure and similarity to the Luenberger type observer, one of the mostly investigated model-based residual generator forms.

The core of a DO is a Luenberger type (output) observer described by

$$z(k+1) = Gz(k) + Hu(k) + Ly(k) \tag{8.30}$$

$$r(k) = Vy(k) - Wz(k) - Qu(k) \tag{8.31}$$

where $z \in \mathcal{R}^s$, s denotes the observer order and can be equal to or lower or higher than the system order n. Although most contributions to the Luenberger type observer are focused on the first case aiming at getting a reduced order observer, higher order observers will play an important role in the optimization of FDI systems.

Assume $G_{yu}(z) = C(zI - A)^{-1}B + D$, then matrices G, H, L, Q, V and W together with a matrix $T \in \mathcal{R}^{s\times n}$ have to satisfy the so-called Luenberger conditions,

$$I.\ G \text{ is stable} \tag{8.32}$$

$$II.\ TA - GT = LC,\ H = TB - LD \tag{8.33}$$

$$III.\ VC - WT = 0,\ Q = VD \tag{8.34}$$

under which system (8.30), (8.31) delivers a residual vector, that is

$$\forall u, x(0),\ \lim_{k\to\infty} r(k) = 0.$$

Let $e(k) = Tx(k) - z(k)$, it is straightforward to show that the system dynamics of DO is, on the assumption of process model (8.18), (8.19), governed by

$$e(k+1) = Ge(k) + (TE_d - LF_d)\,d(k) + \left(TE_f - LF_f\right) f(k) \tag{8.35}$$

$$r(k) = Ve(k) + VF_d d(k) + VF_f f(k). \tag{8.36}$$

Remember that in the last section it has been claimed all residual generator design schemes can be formulated as the search for an observer gain matrix and a post-filter. It is therefore of practical and theoretical interest to reveal the relationships between matrices G, L, T, V and W solving Luenberger equations (8.32)–(8.34) and observer gain matrix as well as post-filter.

A comparison with the FDF scheme makes it clear that

- the diagnostic observer scheme may lead to a reduced order residual generator, which is desirable and useful for online implementation,
- we have more degree of design freedom but, on the other hand,
- more involved design.

8.2.2.3 Kalman Filter Based Residual Generation

Consider (8.14), (8.15). Assume that $\eta(k), \nu(k)$ are white Gaussian processes and independent of initial state vector $x(0), u(k)$ with

$$\mathcal{E}\begin{bmatrix}\eta(i)\eta^T(j) & \eta(i)\nu^T(j)\\ \nu(i)\eta^T(j) & \nu(i)\nu^T(j)\end{bmatrix} = \begin{bmatrix}\Sigma_\eta & S_{\eta\nu}\\ S_{\nu\eta} & \Sigma_\nu\end{bmatrix}\delta_{ij}, \delta_{ij} = \begin{cases}1, & i = j\\ 0, & i \neq j\end{cases} \tag{8.37}$$

$$\Sigma_\nu > 0, \Sigma_\eta \geq 0, \mathcal{E}[\eta(k)] = 0, \mathcal{E}[\nu(k)] = 0 \tag{8.38}$$

$$\mathcal{E}[x(0)] = \bar{x}, \mathcal{E}[(x(0) - \bar{x})(x(0) - \bar{x})^T] = P_o. \tag{8.39}$$

A Kalman filter is, although structured similar to an observer of the full-order, a time-varying system given by the following recursions:
recursive computation for optimal state estimation:

$$\hat{x}(0) = \bar{x} \tag{8.40}$$

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K(k)\left(y(k) - C\hat{x}(k) - Du(k)\right) \tag{8.41}$$

recursive computation for Kalman filter gain:

$$P(0) = P_o \tag{8.42}$$

$$P(k+1) = AP(k)A^T - K(k)R_e(k)K^T(k) + \Sigma_\eta \tag{8.43}$$

$$K(k) = \left(AP(k)C^T + S_{\eta\nu}\right)R_e^{-1}(k) \tag{8.44}$$

$$R_e(k) = \Sigma_\nu + CP(k)C^T \tag{8.45}$$

where $\hat{x}(k)$ denotes the estimation of $x(k)$ and

$$P(k) = \mathcal{E}\left[\left(x(k) - \hat{x}(k)\right)(x(k) - \hat{x}(k))^T\right] \tag{8.46}$$

is the associated estimation error covariance.

The significant characteristics of Kalman filter is

- the state estimation is optimal in the sense of

$$P(k) = \mathcal{E}\left[\left(x(k) - \hat{x}(k)\right)(x(k) - \hat{x}(k))^T\right] \Longrightarrow min$$

- the so-called innovation process $e(k) = y(k) - C\hat{x}(k) - Du(k)$ is a white Gaussian process with covariance

$$\mathcal{E}\left(e(k)e^T(k)\right) = R_e(k) = \Sigma_\nu + CP(k)C^T.$$

Below is an algorithm for the *online* implementation of the Kalman filter algorithm given by (8.40)–(8.45).

Algorithm 8.1 *On-line implementation of Kalman filter*

S0: Set $\hat{x}(0)$, $P(0)$ *as given in (8.40) and (8.42)*
S1: Calculate $R_e(k)$, $K(k)$, $\hat{x}(k)$, *according to (8.45), (8.44) and (8.41)*
S2: Increase k and calculate $P(k+1)$ *according to (8.43)*
S3: Go S1.

Suppose the process under consideration is stationary, then

$$\lim_{k\to\infty} K(k) = K = \text{constant matrix}$$

which is subject to

$$K = \left(APC^T + S_{\eta v^T}\right) R_e^{-1} \tag{8.47}$$

with

$$P = \lim_{k\to\infty} P(k),\ R_e = \Sigma_\nu + CPC^T. \tag{8.48}$$

It holds

$$P = APA^T - KR_eK^T + \Sigma_\eta. \tag{8.49}$$

Equation (8.49) is an algebraic Riccati equation whose solution P is positive definite under certain conditions. It thus becomes evident that given system model the gain matrix K can be calculated offline by solving Riccati equation (8.49). The corresponding residual generator is then given by

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K\left(y(k) - C\hat{x}(k) - Du(k)\right) \tag{8.50}$$

$$r(k) = y(k) - C\hat{x}(k) - Du(k). \tag{8.51}$$

Note that we now have in fact an observer of the full-order.

Remark 8.1 The offline set up (S0) in the above algorithm is needed only for one time, but S1–S3 have to be repeated at each time instant. Thus, the online implementation, compared with the steady-state Kalman filter, is computationally consuming. For the FDI purpose, we can generally assume that the system under consideration is operating in its steady state before a fault occurs. Therefore, the use of the steady-state type residual generator (8.50), (8.51) is advantageous. In this case, the most involved computation is finding a solution for Riccati equation (8.49), which, nevertheless, is carried out offline, and moreover for which there exists a number of numerically reliable methods and CAD programs.

8.2.2.4 Parity Space Approach

The parity space approach was initiated by Chow and Willsky in their pioneering work in the early 1980s. Although a state space model is used for the purpose of residual generation, the so-called parity relation, instead of an observer, builds the core of this approach. The parity space approach is generally recognized as one of the important model-based residual generation approaches, parallel to the observer-based and the parameter estimation schemes.

We consider in the following the state space model (8.18), (8.19) and, without loss of generality, assume that $rank(C) = m$. For the purpose of constructing a residual generator, we first suppose $f(k) = 0, d(k) = 0$. Following (8.18), (8.19), $y(k-s), s > 0$, can be expressed in terms of $x(k-s)$, $u(k-s)$, and $y(k-s+1)$ in terms of $x(k-s)$, $u(k-s+1)$, $u(k-s)$ as follows

$$\begin{aligned} y(k-s) &= Cx(k-s) + Du(k-s) \\ y(k-s+1) &= Cx(k-s+1) + Du(k-s+1) \\ &= CAx(k-s) + CBu(k-s) + Du(k-s+1). \end{aligned}$$

Repeating this procedure yields

$$\begin{aligned} y(k-s+2) &= CA^2x(k-s) + CABu(k-s) + CBu(k-s+1) \\ &\quad + Du(k-s+2), \ldots, \\ y(k) &= CA^sx(k-s) + CA^{s-1}Bu(k-s) + \cdots + CBu(k+1) + Du(k). \end{aligned}$$

Introducing the notations

$$y_s(k) = \begin{bmatrix} y(k-s) \\ y(k-s+1) \\ \vdots \\ y(k) \end{bmatrix}, u_s(k) = \begin{bmatrix} u(k-s) \\ u(k-s+1) \\ \vdots \\ u(k) \end{bmatrix} \tag{8.52}$$

$$H_{o,s} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^s \end{bmatrix}, H_{u,s} = \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}B & \cdots & CB & D \end{bmatrix} \tag{8.53}$$

leads to the following compact model form

$$y_s(k) = H_{o,s}x(k-s) + H_{u,s}u_s(k). \tag{8.54}$$

Note that (8.54) describes the input and output relationship in dependence on the past state vector $x(k-s)$. It is expressed in an explicit form, in which

- $y_s(k)$ and $u_s(k)$ consist of the temporal and past outputs and inputs respectively and are known
- matrices $H_{o,s}$ and $H_{u,s}$ are composite of system matrices A, B, C, D and also known
- the only unknown variable is $x(k-s)$.

The underlying idea of the parity relation based residual generation lies in the utilization of the fact, known from the linear control theory, that for $s \geq n$ the following rank condition holds:

$$rank\left(H_{o,s}\right) \leq n < \text{the row number of matrix } H_{o,s} = (s+1)m.$$

This ensures that for $s \geq n$ there exists at least a (row) vector $v_s(\neq 0) \in \mathcal{R}^{(s+1)m}$ such that

$$v_s H_{o,s} = 0. \tag{8.55}$$

Hence, a parity relation based residual generator is constructed by

$$r(k) = v_s\left(y_s(k-s) - H_{u,s}u_s(k-s)\right) \tag{8.56}$$

whose dynamics is governed by, in case of $f(k) = 0, d(k) = 0$,

$$r(k) = v_s\left(y_s(k) - H_{u,s}u_s(k)\right) = v_s H_{o,s}x(k-s) = 0.$$

Vectors satisfying (8.55) are called parity vectors, the set of which,

$$P_s = \{v_s \mid v_s H_{o,s} = 0\} \tag{8.57}$$

is called the parity space of the sth order.

In order to study the influence of f, d on residual generator (8.56), let $f(k) \neq 0, d(k) \neq 0$. It is straightforward that

$$y_s(k) = H_{o,s}x(k-s) + H_{u,s}u_s(k) + H_{f,s}f_s(k) + H_{d,s}d_s(k)$$

where

$$f_s(k) = \begin{bmatrix} f(k-s) \\ f(k-s+1) \\ \vdots \\ f(k) \end{bmatrix}, H_{f,s} = \begin{bmatrix} F_f & 0 & \cdots & 0 \\ CE_f & F_f & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}E_f & \cdots & CE_f & F_f \end{bmatrix} \tag{8.58}$$

$$d_s(k) = \begin{bmatrix} d(k-s) \\ d(k-s+1) \\ \vdots \\ d(k) \end{bmatrix}, H_{d,s} = \begin{bmatrix} F_d & 0 & \cdots & 0 \\ CE_d & F_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}E_d & \cdots & CE_d & F_d \end{bmatrix}. \quad (8.59)$$

Constructing a residual generator according to (8.56) finally results in

$$r(k) = v_s \left(H_{f,s} f_s(k) + H_{d,s} d_s(k) \right), v_s \in P_s. \quad (8.60)$$

We see that the design parameter of the parity relation based residual generator is the parity vector whose selection has decisive impact on the performance of the residual generator.

Remark 8.2 One of the significant properties of parity relation based residual generators, also widely viewed as the main advantage over the observer-based approaches, is that the design can be carried out in a straightforward manner. In fact, it only deals with solutions of linear equations or linear optimization problems. In against, the implementation form (8.56) is surely not ideal for an online realization, since it is presented in an explicit form, and thus not only the temporal but also the past measurement and input data are needed and have to be recorded.

8.2.2.5 Kernel Representation and Parameterization of Residual Generators

In the model-based FD study, the FDF, DO and Kalman filter based residual generators are called closed-loop configurated, since a feedback of the residual signal is embedded in the system configuration and the computation is realized in a recursive form. Differently, the parity space residual generator is open-loop structured. In fact, it is an FIR (finite impulse response) filter. Below, we introduce a general form for all types of LTI residual generators, which is also called parameterization of residual generators.

A fundamental property of the LCF is that in the fault- and noise-free case

$$\forall u, \begin{bmatrix} -\hat{N}(z) & \hat{M}(z) \end{bmatrix} \begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = 0. \quad (8.61)$$

Equation (8.61) is called kernel representation (KR) of the system under consideration and useful in parameterizing all residual generators. For our purpose, we introduce below a more general definition of kernel representation.

Definition 8.3 Given system (8.2), (8.3), a stable linear system $\mathcal{K}$ driven by $u(z)$, $y(z)$ and satisfying

$$\forall u(z), r(z) = \mathcal{K} \begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = 0 \quad (8.62)$$

is called stable kernel representation (SKR) of (8.2), (8.3).

It is clear that system $\left[-\hat{N}(z)\ \hat{M}(z)\right]$ is an SKR. Now, consider the process model (8.18), (8.19) with unknown input vectors. The parameterization forms of all LTI residual generators and their dynamics are described by

$$r(z) = R(z)\left[-\hat{N}(z)\ \hat{M}(z)\right]\begin{bmatrix} u(z) \\ y(z) \end{bmatrix} \tag{8.63}$$

$$= R(z)\left(\hat{N}_d(z)d(z) + \hat{N}_f(z)f(z)\right) \tag{8.64}$$

where

$$\hat{N}_d(z) = (A - LC, E_d - LF_d, C, F_d), \hat{N}_f(z) = \left(A - LC, E_f - LF_f, C, F_f\right)$$

$R(z)\ (\neq 0)$ is a stable parameterization matrix and called post-filter. Moreover, in order to avoid loss of information about faults to be detected, the condition $rank(R(z)) = m$ is to be satisfied.

It has been demonstrated that all LTI residual generators can be expressed by (8.63), while their dynamics with respect to the unknown inputs and faults are parameterized by (8.64). Moreover, it holds

$$\left[-\hat{N}(z)\ \hat{M}(z)\right]\begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = y(z) - \hat{y}(z)$$

with $\hat{y}$ delivered by a full order observer as an estimate of y, we can apply an FDF,

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L\left(y(k) - \hat{y}(k)\right), \hat{y}(k) = C\hat{x}(k) + Du(k)$$

for the online realization of (8.63).

As a summary, we present a theorem which immediately follows from Definition 8.3 and the residual generator parametrization.

Theorem 8.1 *Given process model* (8.18), (8.19)*, an LTI dynamic system is a residual generator if and only if it is an SKR of* (8.2), (8.3).

8.3 I/O Data Models

In order to connect analytical models and process data, we now introduce different I/O data models. They are essential in our subsequent study and build the link between the model-based FD schemes introduced in the previous sections and the data-driven design methods to be introduced below. For our purpose, the following LTI process model is assumed to be the underlying model form adopted in our study

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{8.65}$$

$$y(k) = Cx(k) + Du(k) + v(k) \tag{8.66}$$

where $u \in \mathcal{R}^l$, $y \in \mathcal{R}^m$ and $x \in \mathcal{R}^n$, $w \in \mathcal{R}^n$ and $v \in \mathcal{R}^m$ denote noise sequences that are normally distributed and statistically independent of u and $x(0)$.

Let $\omega(k) \in \mathcal{R}^\xi$ be a data vector. We introduce the following notations

$$\omega_s(k) = \begin{bmatrix} \omega(k-s) \\ \vdots \\ \omega(k) \end{bmatrix} \in \mathcal{R}^{(s+1)\xi}, \Omega_k = \begin{bmatrix} \omega(k) \cdots \omega(k+N-1) \end{bmatrix} \in \mathcal{R}^{\xi \times N} \tag{8.67}$$

$$\Omega_{k,s} = \begin{bmatrix} \omega_s(k) \cdots \omega_s(k+N-1) \end{bmatrix} = \begin{bmatrix} \Omega_{k-s} \\ \vdots \\ \Omega_k \end{bmatrix} \in \mathcal{R}^{(s+1)\xi \times N} \tag{8.68}$$

where s, N are some (large) integers. In our study, $\omega(k)$ can be $y(k)$, $u(k)$, $x(k)$, and ξ represents m or l or n given in (8.65), (8.66).

The first I/O data model described by

$$Y_{k,s} = \Gamma_s X_{k-s} + H_{u,s} U_{k,s} + H_{w,s} W_{k,s} + V_{k,s} \in \mathcal{R}^{(s+1)m \times N} \tag{8.69}$$

$$\Gamma_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^s \end{bmatrix} \in \mathcal{R}^{(s+1)m \times n}, H_{u,s} = \begin{bmatrix} D & 0 & & \\ CB & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}B & \cdots & CB & D \end{bmatrix}$$

follows directly from the I/O model (8.54) introduced in the study on parity space scheme, where $H_{u,s} \in \mathcal{R}^{(s+1)m \times (s+1)l}$, $H_{w,s} W_{k,s} + V_{k,s}$ represents the influence of the noise vectors on the process output with $H_{w,s}$ having the same structure like $H_{u,s}$ and $W_{k,s}$, $V_{k,s}$ as defined in (8.67), (8.68).

In the SIM framework, the so-called innovation form, instead of (8.69), is often applied to build an I/O model. The core of the innovation form is a (steady) Kalman filter, which is written as

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K\left(y(k) - \hat{y}(k)\right), \hat{y}(k) = C\hat{x}(k) + Du(k) \tag{8.70}$$

with the innovation sequence $y(k) - \hat{y}(k) := e(k)$ being a white noise sequence and K the Kalman filter gain matrix. Based on (8.70), the I/O relation of the process can be alternatively written into

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + Ke(k) = A_K\hat{x}(k) + B_K u(k) + Ky(k) \tag{8.71}$$

$$y(k) = C\hat{x}(k) + Du(k) + e(k), A_K = A - KC, B_K = B - KD. \tag{8.72}$$

The following two I/O data models follow from (8.71), (8.72):

$$Y_{k,s} = \Gamma_s \hat{X}_{k-s} + H_{u,s} U_{k,s} + H_{e,s} E_{k,s},\ H_{e,s} = \begin{bmatrix} I & 0 & & \\ CK & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}K & \cdots & CK & I \end{bmatrix} \tag{8.73}$$

$$\left(I - H^K_{y,s}\right) Y_{k,s} = \Gamma^K_s \hat{X}_{k-s} + H^K_{u,s} U_{k,s} + E_{k,s},\ \Gamma^K_s = \begin{bmatrix} C \\ CA_K \\ \vdots \\ CA^s_K \end{bmatrix} \tag{8.74}$$

$$H^K_{y,s} = \begin{bmatrix} 0 & 0 & & \\ CK & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}_K K & \cdots & CK & 0 \end{bmatrix},\ H^K_{u,s} = \begin{bmatrix} D & 0 & & \\ CB_K & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}_K B_K & \cdots & CB_K & D \end{bmatrix}$$

with $H^K_{y,s} \in \mathcal{R}^{(s+1)m\times(s+1)m}$, $H_{e,s} \in \mathcal{R}^{(s+1)m\times(s+1)m}$, $E_{k,s} \in \mathcal{R}^{(s+1)m\times N}$ having the same structure as given by (8.67), (8.68).

8.4 Notes and References

In order to apply the MVA technique to solving FD problems in dynamic processes, numerous methods have been developed in the last two decades. Among them, dynamic PCA/PLS [1–3], recursive implementation of PCA/PLS [4, 5], fast moving window PCA [6] and multiple-mode PCA [7] are widely used in the research and applications in recent years.

SIM is a well-established technique and widely applied in process identification [8–11]. The application of the SIM technique to FDI study is new and has been first proposed by [12–15].

In Sect. 8.2, the basics of the model-based FDI framework have been reviewed. They can be found in the monographs [1, 16–25] and in the survey papers [26–29]. The first work on FDF has been reported by Beard and Jones in [30, 31], and Chow and Willsky have proposed the first optimal FDI solution using the parity space scheme [32]. The reader is referred to Chaps. 3 and 5 in [25] for a systematic handling of the issues on process and fault modeling and model-based residual generation schemes.

The concept SKR will play an important role in our subsequent studies. In fact, residual generator design is to find an SKR, as given in Theorem 8.1. The SKR definition given in Definition 8.3 is similar to the one introduced in [33] for nonlinear systems.

The I/O data models (8.69), (8.73) and (8.74) are essential in the SIM framework and thus can be found in the monographs and survey papers on the SIM technique. The reader is referred to e.g. [10, 11] for a more comprehensive description.

References

1. Russell EL, Chiang L, Braatz RD (2000) Data-driven techniques for fault detection and diagnosis in chemical processes. Springer-Verlag, London
2. Russell EL, Chiang LH, Braatz RD (2000) Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Chemom Intell Lab Syst 51:81–93
3. Li W, Qin SJ (2001) Consistent dynamic PCA based on errors-in-variables subspace identification. J Process Control 11:661–678
4. Li W, Yue HH, Valle-Cervantes S, Qin SJ (2000) Recursive PCA for adaptive process monitoring. J Process Control 10:471–486
5. Qin SJ (1998) Recursive PLS algorithms for adaptive data modeling. Comput Chem Eng 22:503–514
6. Wang X, Kruger U, Irwin GW (2005) Process monitoring approach using fast moving window PCA. Ind Eng Chem Res 44:5691–5702
7. Garcia-Alvares D, Fuente MJ, Sainz GJ (2012) Fault detection and isolation in transient states using principal component analysis. J Process Control 22:551–563
8. Favoreel W, Moor BD, Overschee PV (2000) Subspace state space system identification for industrial processes. J Process Control 10:149–155
9. Overschee PV, Moor BD (1996) Subspace identification for linear systems. Kluwer Academic Publishers, USA
10. Qin SJ (2006) An overview of subspace identification. Comp Chem Eng 30:1502–1513
11. Huang B, Kadali R (2008) Dynamic modelling, predictive control and performance monitoring, a data-driven subspace approach. Springer-Verlag, London
12. Qin SJ, Li W (2001) Detection and identification of faulty sensors in dynamic processes. AIChE J 47:1581–1593
13. Ding SX, Zhang P, Naik A, Ding E, Huang B (2009) Subspace method aided data-driven design of fault detection and isolation systems. J Process Control 19:1496–1510
14. Dong J, Verhaegen M (2009) Subspace based fault detection and identification for LTI systems. In: Proceedings of the 7th IFAC symposium SAFEPROCESS, pp 330–335
15. Dong J (2009) Data driven fault tolerant control: a subspace approach. PhD dissertation, Technische Universiteit Delft
16. Gertler JJ (1998) Fault detection and diagnosis in engineering systems. Marcel Dekker, New York, Basel, Hong Kong
17. Mangoubi R (1998) Robust estimation and failure detection. Springer, London
18. Chen J, Patton RJ (1999) Robust model-based fault diagnosis for dynamic systems. Kluwer Academic Publishers, Boston
19. Patton RJ, Frank PM, Clark RN (eds) (2000) Issues of fault diagnosis for dynamic systems. Springer, London
20. Gustafsson F (2000) Adaptive filtering and change detection. Wiley, Chichester
21. Chiang LH, Russell EL, Braatz RD (2001) Fault detection and diagnosis in industrial systems. Springer, London
22. Blanke M, Kinnaert M, Lunze J, Staroswiecki M (2006) Diagnosis and fault-tolerant control, 2nd edn. Springer, Berlin Heidelberg
23. Simani S, Fantuzzi S, Patton RJ (2003) Model-based fault diagnosis in dynamic systems using identification techniques. Springer-Verlag, London

24. Isermann R (2006) Fault diagnosis systems. Springer-Verlag, Berlin Heidelberg
25. Ding SX (2013) Model-based fault diagnosis techniques—design schemes, algorithms and tools, 2nd edn. Springer-Verlag, London
26. Frank PM, Ding X (1997) Survey of robust residual generation and evaluation methods in observer-based fault detection systems. J Process Control 7(6):403–424
27. Zhang P, Ding SX (2008) On fault detection in linear discrete-time, periodic, and sampled-data systems (survey). J Control Sci Eng 2008:1–18
28. Mangoubi R, Desai M, Edelmayer A, Sammak P (2009) Robust detection and estimation in dynamic systems and statistical signal processing: intersection, parallel paths and applications. Eur J Control 15:348–369
29. Hwang I, Kim S, Kim Y, Seah C (2010) A survey of fault detection, isolation, and reconfiguration methods. IEEE Trans Control Syst Tech 18:636–653
30. Beard R (1971) Failure accomondation in linear systems through self-reorganization. PhD dissertation, MIT
31. Jones H (1973) Failure detection in linear systems. PhD dissertation, MIT
32. Chow EY, Willsky AS (1984) Analytical redundancy and the design of robust failure detection systems. IEEE Trans Autom Control 29:603–614
33. Van der Schaft A (2000) L2—gain and passivity techniques in nonlinear control. Springer, London

Chapter 9
Data-Driven Diagnosis Schemes

In this chapter, we present the first design schemes and algorithms for fault diagnosis in dynamic processes. They are similar to the multivariate analysis methods addressed in Chaps. 5– 7 and different from the diagnosis schemes to be investigated in the next chapter. For our purpose, we first introduce some basic concepts and design issues in fault diagnosis and then present fault detection, isolation and identification schemes step by step and in details.

9.1 Basic Concepts and Design Issues of Fault Diagnosis in Dynamic Processes

Recall that the core of the application of the MVA methods to fault detection is to run a T^2 or some similar test statistics to detect a change in the mean of the statistic process under monitoring. To this end, a normalization (centralization) of the collected data is needed so that in the fault-free case a zero mean process is available for the construction and computation of the test statistics. In dealing with dynamic processes, a similar step is also necessary. It is called residual generation, which has been briefly introduced in Chap. 8 in the model-based fault detection framework. Next, we define residual generation in the data-driven framework.

Remember that the major feature of a residual signal is that it is (almost) decoupled from the process inputs. As described in Chap. 8, residual generation issues can be addressed in the context of the SKR. Next, we introduce a data-driven form of the SKR, which plays a central role in the sequel.

Definition 9.1 Given system (8.2), (8.3), matrix $\mathcal{K}_{d,s}$ is called a data-driven realization of the SKR, if for some integer s it holds

$$\forall u_s(k), x(0), r(k) = \mathcal{K}_{d,s} \begin{bmatrix} u_s(k) \\ y_s(k) \end{bmatrix} = 0. \tag{9.1}$$

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_9, © Springer-Verlag London 2014

Let U, Y denote the (recorded training) process input and output data sets. For the residual generation purpose in the data-driven design framework, a data-driven realization of the SKR can be found by solving an (algebraic) equation of the form

$$\forall U, \mathcal{K}_{d,s} \begin{bmatrix} U \\ Y \end{bmatrix} = 0 \tag{9.2}$$

in the fault- and noise-free case.

On the assumption that process and measurement noises are zero-mean and normally distributed, relation (9.2) can also be schematically written as

$$\forall U, \mathcal{K}_{d,s} \begin{bmatrix} U \\ Y \end{bmatrix} \sim \mathcal{N}(0, \Sigma_{res}). \tag{9.3}$$

Thus, the data-driven design of fault detection is formulated as finding $\mathcal{K}_{d,s}$ and Σ_{res} using the available process data.

Once $\mathcal{K}_{d,s}$, Σ_{res} are identified, an (online) residual generation can be realized in the form of

$$r(k) = \mathcal{K}_{d,s} \begin{bmatrix} u_s(k) \\ y_s(k) \end{bmatrix} \tag{9.4}$$

with $u_s(k), y_s(k)$ denoting the process input and output data in the time interval $[k - s, k]$. Moreover, a test statistic and the corresponding threshold can be defined, which allows a fault detection.

In the next sections, we shall address these issues with a focus on residual generation.

9.2 Data-Driven Design Schemes of Residual Generators

In this section, we derive three residual generation schemes corresponding to the I/O data models (8.69), (8.73) and (8.74). Note that due to the unmeasurable X_k or $\hat{X}_k$, these data models cannot be directly identified and then applied for the residual generation purpose. Hence, the key issue in our study is to eliminate the influence of X_k or $\hat{X}_k$.

9.2.1 Scheme I

We first consider (8.71), from which we have

$$\hat{x}(k) = A_K^{\rho} \hat{x}(k - \rho) + \sum_{i=1}^{\rho} A_K^{i-1} \begin{bmatrix} B_K & K \end{bmatrix} \begin{bmatrix} u(k - i) \\ y(k - i) \end{bmatrix}.$$

Since the eigenvalues of A_K are all located within the unit circle, it holds, for a large integer ρ,

$$A_K^{\rho} \approx 0 \Longrightarrow \hat{x}(k) \approx \sum_{i=1}^{\rho} A_K^{i-1} \begin{bmatrix} B_K & K \end{bmatrix} \begin{bmatrix} u(k-i) \\ y(k-i) \end{bmatrix}. \tag{9.5}$$

As a result, (8.73) is rewritten into

$$\begin{aligned} Y_{k,s} &\approx \Upsilon_{s,\rho-1} Z_{k-s-1,\rho-1} + H_{u,s} U_{k,s} + H_{e,s} E_{k,s}, \rho >> 1 \\ Z_{k-s-1,\rho-1} &= \begin{bmatrix} U_{k-s-1,\rho-1} \\ Y_{k-s-1,\rho-1} \end{bmatrix}, \Upsilon_{s,\rho-1} = \begin{bmatrix} \Upsilon_{s,\rho-1,u} & \Upsilon_{s,\rho-1,y} \end{bmatrix} \\ \Upsilon_{s,\rho-1,y} &= \Gamma_s \begin{bmatrix} A_K^{\rho-1} K & \dots & K \end{bmatrix}, \Upsilon_{s,\rho-1,u} = \Gamma_s \begin{bmatrix} A_K^{\rho-1} B_K & \dots & B_K \end{bmatrix}. \end{aligned} \tag{9.6}$$

Now, identifying $\Upsilon_{s,\rho-1}$, $H_{u,s}$ using process data $Y_{k,s}$, $U_{k,s}$, $Z_{k-s-1,\rho-1}$ allows the construction of a residual generator, as summarized in the following algorithm.

Algorithm 9.1 *Residual generation: Scheme I*

S1: Collect process data and form $Z_{k-s-1,\rho-1}$, $U_{k,s}$, $Y_{k,s}$
S2: Solve the least squares problem for $\Upsilon_{s,\rho-1}$, $H_{u,s}$

$$\min_{\Upsilon_{s,\rho-1}, H_{u,s}} \left\| Y_{k,s} - \Upsilon_{s,\rho-1} Z_{k-s-1,\rho-1} - H_{u,s} U_{k,s} \right\|_F.$$

By means of the identified $\Upsilon_{s,\rho-1}$, $H_{u,s}$, we are able to compute

$$\begin{aligned} H_{e,s} E_{k,s} &= Y_{k,s} - \Upsilon_{s,\rho-1} Z_{k-s-1,\rho-1} - H_{u,s} U_{k,s} \Longrightarrow \\ \Sigma_{res} &\approx \frac{1}{N-1} \left(H_{e,s} E_{k,s} E_{k,s}^T H_{e,s}^T \right) \end{aligned} \tag{9.7}$$

for an estimation of the covariance matrix of the residual vector and, moreover, to construct the residual generator for the (online) residual generation as

$$r(k) = y_s(k) - \Upsilon_{s,\rho-1} z_{\rho-1}(k-s-1) - H_{u,s} u_s(k). \tag{9.8}$$

9.2.2 Scheme II

This scheme has been proposed by Dong. It is based on the I/O data model (8.74). Using (9.5), this data model can be further re-written into

$$
\begin{aligned}
Y_{k,s} &= \Upsilon^K_{s,\rho-1} Z_{k-s-1,\rho-1} + H^K_{y,s} Y_{k,s} + H^K_{u,s} U_{k,s} + E_{k,s} \\
\Upsilon^K_{s,\rho-1} &= \left[\Upsilon^K_{s,\rho-1,u} \ \Upsilon^K_{s,\rho-1,y} \right], \Upsilon^K_{s,\rho-1,u} = \Gamma^K_s \left[A^{\rho-1}_K B_K \ \ldots \ B_K \right] \\
\Upsilon^K_{s,\rho-1,y} &= \Gamma^K_s \left[A^{\rho-1}_K K \ \ldots \ K \right].
\end{aligned} \tag{9.9}
$$

Note that $\Upsilon^K_{s,\rho-1}, H^K_{y,s}, H^K_{u,s}$ can be fully constructed in terms of the Markov parameters

$$
CA^{\kappa-1}_K B, \ldots, CB, CA^{\kappa-1}_K K, \ldots, CK, D, \kappa = s + \rho.
$$

Hence, when these Markov parameters are identified, we are able to apply model (9.9) for the residual generation purpose. To this end, consider

$$
\begin{aligned}
Y_k &\approx \sum_{i=1}^{\kappa} CA^{i-1}_K \left[B_K \ K \right] \begin{bmatrix} U_{k-i} \\ Y_{k-i} \end{bmatrix} + DU_k + E_k \\
&:= \Pi \begin{bmatrix} Z_{k-s-1,\kappa-1} \\ U_k \end{bmatrix}, Z_{k-s-1,\kappa-1} = \begin{bmatrix} U_{k-s-1,\kappa-1} \\ Y_{k-s-1,\kappa-1} \end{bmatrix} \\
E_k &= \left[e(k) \ \ldots \ e(k+N-1) \right] \\
\Pi &= \left[CA^{\kappa-1}_K B \ \ldots\ldots, CB \ CA^{\kappa-1}_K K \ \ldots \ CK \ D \right]
\end{aligned}
$$

where $Y_k, U_k, E_k, U_{k-i}, Y_{k-i}$ are as defined in (8.67). Assume that N is sufficiently large and the data matrix $\begin{bmatrix} Z_{k-s-1,\kappa-1} \\ U_k \end{bmatrix}$ is of full row rank, then using data $Y_{k-\kappa}, \ldots Y_k, U_{k-\kappa}, \ldots U_k$ the Markov parameters can be estimated by

$$
\hat{\Pi} = Y_k \begin{bmatrix} Z_{k-s-1,\kappa-1} \\ U_k \end{bmatrix}^- \tag{9.10}
$$

where for a full row rank matrix T, T^- denotes the right inverse given by

$$
T^- = T^T \left(TT^T \right)^{-1} \Longrightarrow TT^- = I.
$$

Finally, substituting the Markov parameters in (9.9) by their estimates allows

- an estimation of the covariance matrix of the residual vector

$$
\begin{aligned}
E_{k,s} &= Y_{k,s} - \Upsilon^K_{s,\rho-1} Z_{k-s-1,\rho-1} - H^K_{y,s} Y_{k,s} - H^K_{u,s} U_{k,s} \Longrightarrow \\
\Sigma_{res} &= \frac{1}{N-1} E_{k,s} E^T_{k,s}
\end{aligned} \tag{9.11}
$$

- and an online residual generation

$$r(k) = \left(I - H_{y,s}^K\right) y_s(k) - \Upsilon_{s,\rho-1}^K z_{\rho-1}(k-s-1) - H_{u,s}^K u_s(k). \tag{9.12}$$

Below is the residual generation algorithm.

Algorithm 9.2 *Residual generation: Scheme II*

S1: Collect process data and form $Z_{k-s-1,\kappa-1}$, U_k, Y_k
S2: Solve the least squares estimation problem for Π

$$\min_{\Pi} \left\| Y_k - \Pi \begin{bmatrix} Z_{k-s-1,\kappa-1} \\ U_k \end{bmatrix} \right\|_F.$$

S3: Form $H_{y,s}^K$, $\Upsilon_{s,\rho-1}^K$, $H_{u,s}^K$ *using the estimated* Π.

9.2.3 Scheme III

Note that in the previous two schemes X_k or $\hat{X}_k$ are approximated by the process input and output data. The next residual generation scheme is based on a direct identification of the data-driven form of the kernel representation as described in (9.2) or (9.3). To this end, rewrite (8.69) into

$$\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \Psi_s \begin{bmatrix} U_{k,s} \\ X_{k-s} \end{bmatrix} + \begin{bmatrix} 0 \\ H_{w,s} W_{k,s} + V_{k,s} \end{bmatrix}, \Psi_s = \begin{bmatrix} I & 0 \\ H_{u,s} & \Gamma_s \end{bmatrix}. \tag{9.13}$$

Since $\Psi_s \in \mathcal{R}^{(s+1)(m+l)\times(n+(s+1)l)}$ and for $s \geq n$, $(s+1)(m+l) > n + (s+1)l$, there exists $\Psi_s^\perp$ so that

$$\begin{aligned} &\Psi_s^\perp \Psi_s = 0, \Psi_s^\perp \in \mathcal{R}^{((s+1)m-n)\times(s+1)(m+l)} \Longrightarrow \\ &\Psi_s^\perp \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \Psi_s^\perp \begin{bmatrix} 0 \\ H_{w,s} W_{k,s} + V_{k,s} \end{bmatrix}. \end{aligned}$$

In other words, $\Psi_s^\perp$ is a data-driven realization of the SKR $\mathcal{K}_{d,s}$ and thus builds a residual generator. Next, we study the issue of identifying

$$\mathcal{K}_{d,s} = \Psi_s^\perp.$$

Let

$$Z_{k-s-1,s} = \begin{bmatrix} U_{k-s-1,s} \\ Y_{k-s-1,s} \end{bmatrix}.$$

Notice that $Z_{k-s-1,s}$ is uncorrelated with $W_{k,s}$, $V_{k,s}$ in the sense that

$$\frac{1}{N-1} W_{k,s} Z_{k-s-1,s}^T \approx 0, \frac{1}{N-1} V_{k,s} Z_{k-s-1,s}^T \approx 0.$$

Hence, it holds

$$\frac{1}{N-1} \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} Z_{k-s-1,s}^T \approx \Psi_s \begin{bmatrix} U_{k,s} \\ X_{k-s} \end{bmatrix} Z_{k-s-1,s}^T.$$

Assume that

$$\begin{bmatrix} U_{k,s} \\ X_k \end{bmatrix} Z_{k-s-1,s}^T \text{ is of full row rank.}$$

It yields

$$\Psi_s^\perp \Psi_s = 0 \Longleftrightarrow \Psi_s^\perp \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} \frac{Z_{k-s-1,s}^T}{N-1} = 0. \tag{9.14}$$

An SVD

$$\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} \frac{Z_{k-s-1,s}^T}{N-1} = \begin{bmatrix} U_1 \; U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{9.15}$$

leads to

$$\Sigma_2 \approx 0, \Psi_s^\perp = U_2^T \in \mathcal{R}^{((s+1)m-n)\times(s+1)(l+m)}. \tag{9.16}$$

Let $\mathcal{K}_{d,s} = \Psi_s^\perp = \begin{bmatrix} \Psi_{s,u}^\perp \; \Psi_{s,y}^\perp \end{bmatrix}, \Psi_{s,y}^\perp \in \mathcal{R}^{((s+1)m-n)\times(s+1)m}$. It is evident that

$$\Psi_{s,y}^\perp \Gamma_s = 0, \Psi_{s,u}^\perp = -\Psi_{s,y}^\perp H_{u,s}. \tag{9.17}$$

Thus, $\Psi_{s,y}^\perp = \Gamma_s^\perp$ is the so-called parity subspace. By means of the identified kernel representation $\Psi_s^\perp$, a residual vector can be generated as follows:

$$r(k) = \Psi_{s,y}^\perp y_s(k) + \Psi_{s,u}^\perp u_s(k) = \Psi_s^\perp \begin{bmatrix} u_s(k) \\ y_s(k) \end{bmatrix}. \tag{9.18}$$

Using the process (training) data $U_{k,s}$, $Y_{k,s}$, we are also able to estimate the covariance matrix as follows

$$\Psi_s^{\perp}\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \Psi_{s,y}^{\perp}\left(H_{w,s}W_{k,s} + V_{k,s}\right) \Longrightarrow$$
$$\Sigma_{res} = \frac{1}{N-1}\Psi_s^{\perp}\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix}\left(\Psi_s^{\perp}\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix}\right)^T. \tag{9.19}$$

9.2.4 A Numerically Reliable Realization Algorithm

It can be observed that in the first two algorithms the least squares estimations using process data sets result in the major computations. It is known that by means of a QR-decomposition of the process data these computations will run in a numerically robust and reliable way. Next, we introduce an algorithm for this purpose. Without loss of generality, let the data sets be U_p, Y_p, U_f, Y_f and assume that

$$rank\begin{bmatrix} Z_p \\ U_f \end{bmatrix} = \text{number of the row}, Z_p = \begin{bmatrix} U_p \\ Y_p \end{bmatrix}$$

where Z_p represents $Z_{k-s-1,\rho-1}$ in Scheme I and $Z_{k-s-1,\kappa-1}$ in Scheme II respectively, and U_f, Y_f stand for $U_{k,s}$, $Y_{k,s}$ and U_k, Y_k respectively. By a QR decomposition of the form

$$\begin{bmatrix} Z_p \\ U_f \\ Y_f \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{bmatrix}\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{9.20}$$

we have

$$Y_f = \begin{bmatrix} R_{31} & R_{32} \end{bmatrix}\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} + R_{33}Q_3$$
$$\begin{bmatrix} Z_p \\ U_f \end{bmatrix}^{-} = \begin{bmatrix} Z_p \\ U_f \end{bmatrix}^T\left(\begin{bmatrix} Z_p \\ U_f \end{bmatrix}\begin{bmatrix} Z_p \\ U_f \end{bmatrix}^T\right)^{-1} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}^T\begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}^{-} \Longrightarrow$$
$$Y_f\begin{bmatrix} Z_p \\ U_f \end{bmatrix}^{-} = \begin{bmatrix} R_{31} & R_{32} \end{bmatrix}\begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}^{-} + R_{33}Q_3\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}^T\begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}^{-}$$
$$= \begin{bmatrix} R_{31} & R_{32} \end{bmatrix}\begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}^{-}. \tag{9.21}$$

It follows from (9.21) that the solutions of the estimation problems in Algorithms 9.1 and 9.2 can be respectively given by

$$\begin{bmatrix} \Upsilon_{s,\rho-1} & H_{u,s} \end{bmatrix} = \begin{bmatrix} R_{31} & R_{32} \end{bmatrix} \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}^{-} \tag{9.22}$$

$$\Pi = \begin{bmatrix} R_{31} & R_{32} \end{bmatrix} \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix}^{-}. \tag{9.23}$$

Moreover, it becomes clear that $R_{33}Q_3$ represents the noise in Y_f . For instance, in Scheme I

$$R_{33}Q_3 = H_{e,s}E_{k,s}.$$

Next, we are going to propose an alternative algorithm for the realization of the data-driven KR and, associated with it, for residual generation. To this end, we consider $U_{k,s}$, $Y_{k,s}$ in Scheme III

$$\begin{aligned} \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} &= \Psi_s \begin{bmatrix} U_{k,s} \\ X_{k-s} \end{bmatrix} + \begin{bmatrix} 0 \\ H_{w,s}W_{k,s} + V_{k,s} \end{bmatrix} \\ &= \begin{bmatrix} 0 & I \\ \Upsilon_{s,\rho-1}Z_{k-\rho,\rho-1} & H_{w,s} \end{bmatrix} \begin{bmatrix} Z_{k-s-1,\rho-1} \\ U_{k,s} \end{bmatrix} + \begin{bmatrix} 0 \\ H_{w,s}W_{k,s} + V_{k,s} \end{bmatrix}. \end{aligned}$$

By a QR decomposition

$$\begin{bmatrix} Z_{k-s-1,\rho-1} \\ U_{k,s} \\ Y_{k,s} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{9.24}$$

it holds

$$\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \begin{bmatrix} R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} + \begin{bmatrix} 0 \\ R_{33}Q_3 \end{bmatrix}.$$

Recall moreover

$$R_{33}Q_3 = H_{w,s}W_{k,s} + V_{k,s}.$$

We finally have

$$\Psi_s^{\perp} \begin{bmatrix} R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix} = 0 \Longleftrightarrow \Psi_s^{\perp}\Psi_s = 0 \tag{9.25}$$

$$\Longrightarrow \Psi_s^{\perp} \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \Psi_{s,y}^{\perp} \left(H_{w,s}W_{k,s} + V_{k,s} \right) = \Psi_{s,y}^{\perp} R_{33}Q_3. \tag{9.26}$$

This result allows us to propose the following algorithm for a numerically reliable computation of the data-driven realization of the kernel representation given in Scheme III.

Algorithm 9.3 *Residual generation (identification of data-driven SKR)*

S1: Collect process data and form $Z_{k-s-1,\rho-1}$, $U_{k,s}$, $Y_{k,s}$
S2: Do QR-decomposition (9.24)
S3: Do SVD (9.15) and set $\mathcal{K}_{d,s} = \Psi_s^{\perp} = U_2^T$
S4: Compute

$$\Sigma_{res} = \frac{\Psi_{s,y}^{\perp} R_{33} Q_3 \left(\Psi_{s,y}^{\perp} R_{33} Q_3\right)^T}{N-1} = \frac{\Psi_{s,y}^{\perp} R_{33} R_{33}^T \left(\Psi_{s,y}^{\perp}\right)^T}{N-1}. \tag{9.27}$$

9.2.5 Comparison and Discussion

We now briefly compare the above three schemes in view of the needed off- and online computations.

For the design purpose, all these three schemes need sufficient data for training (identification). While the core of the first two schemes is a least squares estimation, which, roughly speaking, deals with a (right) inverse computation as given in (9.22) and (9.23), the key in the third scheme is the computation of a null-subspace. From the computational viewpoint, the needed computation efforts in these three schemes are similar.

The online realization of the above introduced residual generation systems can be described in the following general form

$$r(k) = \Psi \begin{bmatrix} u_\gamma(k-\gamma) \\ y_\gamma(k-\gamma) \end{bmatrix} \tag{9.28}$$

where $\gamma = s + \rho$ for the first two schemes and $\gamma = s$ for the third one. This means, the third residual generation scheme demands less online computation capacity (including working memory) than the first two schemes. Moreover, it is a (data-driven) realization of the kernel representation of the process. Thus, in ideal case, it delivers a residual vector that is independent of the process inputs. This property is of practical interests in process automatic control systems. Differently, in the first two schemes, the generated residual signal may be a function of the process inputs, depending on the selection of ρ and the process dynamics.

It is clear that residual generator (9.28) is an FIR (finite impulse response) filter and open-loop configurated. With the measurement data in the time interval $[k-\gamma, k]$ as its input, (9.28) delivers a residual vector that may contain redundant information and is less robust against disturbances. Also, for the online implementation of the residual generator (9.28), matrix Ψ demands high working memory.

9.3 Test Statistics, Threshold Settings and Fault Detection

Once Ψ, Σ_{res} have been identified, test statistics and thresholds can be defined for the detection purpose. On the assumption that the process and measurement noises are normally distributed and the identified Σ_{res} is (well) invertible, T^2 test statistic can be adopted for the detection purpose, that is

$$J = r^T(k)\Sigma_{res}^{-1}r(k).$$

Below is a summary of the threshold settings for the three residual generation schemes presented in the last section.

$$\text{Scheme I:} \quad J_{th} = \chi_\alpha^2((s+1)m) \tag{9.29}$$

$$\text{Scheme II:} \quad J_{th} = \chi_\alpha^2((s+1)m) \tag{9.30}$$

$$\text{Scheme III:} \quad J_{th} = \chi_\alpha^2((s+1)m - n). \tag{9.31}$$

It should be pointed out that $r(k)$ is a high-dimensional vector and may contain redundant signals. As a result, the covariance matrix Σ_{res} may become ill-conditional. In this case, the PCA technique introduced in Chap. 5 provides us with a powerful and useful solution.

Alternatively, the recursive and observer-like realization of the proposed residual generators, which will be presented in the next chapter, can solve this problem in a more efficient manner.

9.4 Fault Isolation and Identification Schemes

It is remarkable that both in the model-based and data-driven framework most fault isolation and identification approaches have been developed on the assumption of an available fault model. In general, a successful data-driven fault isolation and identification demands process data which cover all operation conditions, also including the faulty operations. In practice, data from faulty process operations are seldom available or at least often not available in a well synchronized form with a detailed fault description. In fact, it is the nature of technical processes that the quality of the data collected in the faulty process operation may be poor. Moreover, the faults (functions) cannot be recorded during their presence in the way like we do with the process input variables. These facts make a data-driven (fault) modeling and data-driven fault isolation and identification very difficult.

Considering that sensor and actuator behavior is implicitly embedded in the I/O data model of a process, the data-driven realization of the process kernel representation can also be used to isolate and identify typical sensor or actuator faults like off-sets or drifts. This is also the basic idea behind the FDI schemes which we briefly introduce in this section.

9.4.1 Problem Formulation

Consider the state space representation with sensor and actuator faults (8.21), (8.22). Correspondingly, the I/O data model and $x(k)$ can be expressed in terms of the innovation form respectively by

$$\hat{x}(k+1) = A_K \hat{x}(k) + B_K\,(u(k) + f_A(k)) + K\,(y(k) - f_S(k)) \tag{9.32}$$

$$y(k) = C\hat{x}(k) + Du(k) + e(k) + Df_A(k) + f_S(k) \tag{9.33}$$

$$\begin{aligned} x(k) \approx & \sum_{i=1}^{\rho} A_K^{i-1} \begin{bmatrix} B_K & K \end{bmatrix} \begin{bmatrix} u(k-i) \\ y(k-i) \end{bmatrix} + \sum_{i=1}^{\rho} A_K^{i-1} B_K f_A(k-i) \\ & - \sum_{i=1}^{\rho} A_K^{i-1} K f_S(k-i). \end{aligned} \tag{9.34}$$

Note that here $e(k)$ denotes the innovation (white) sequence. As a result, the dynamics of the three types of residual generators are respectively:

- for residual generator (9.8)

$$\begin{aligned} r(k) &= y_s(k) - \hat{\Upsilon}_{s,\rho-1} z_{\rho-1}(k-s-1) - \hat{H}_{u,s} u_s(k) \\ &= H_{e,s} e_s(k) + H_{f_A} f_{A,s+\rho}(k) + H_{f_S} f_{S,s+\rho}(k) \\ H_{f_A} &= \begin{bmatrix} -\Upsilon_{s,\rho-1,u} & H_{u,s} \end{bmatrix}, H_{f_S} = \begin{bmatrix} \Upsilon_{s,\rho-1,y} & I \end{bmatrix} \end{aligned} \tag{9.35}$$

- for residual generator (9.12)

$$\begin{aligned} r(k) &= \left(I - H_{y,s}^K\right) y_s(k) - \Upsilon_{s,\rho-1}^K z_{\rho-1}(k-s-1) - H_{u,s}^K u_s(k) \\ &= e_s(k) + H_{f^A}^K f_{A,s+\rho}(k) + H_{f^S}^K f_{S,s+\rho}(k) \\ H_{f^A}^K &= \begin{bmatrix} -\Upsilon_{s,\rho-1,u}^K & H_{u,s}^K \end{bmatrix}, H_{f^S}^K = \begin{bmatrix} \Upsilon_{s,\rho-1,y}^K & H_{f^S,s}^K \end{bmatrix} \\ H_{f^S,s}^K &= \begin{bmatrix} I & 0 & & \\ -CK & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ -CA_K^{s-1}K & \cdots & -CK & I \end{bmatrix} \end{aligned} \tag{9.36}$$

- for residual generator (9.18)

$$\begin{aligned} r(k) &= \Psi_{s,y}^{\perp} y_s(k) + \Psi_{s,u}^{\perp} u_s(k) \\ &= \Psi_{s,y}^{\perp} H_{e,s} e_s(k) + \Psi_{s,y}^{\perp} H_{u,s} f_{A,s}(k) + \Psi_{s,y}^{\perp} f_{A,s}(k) \end{aligned} \tag{9.37}$$

where

$$f_{A,\xi}(k) = \begin{bmatrix} f_A(k-\xi) \\ \vdots \\ f_A(k) \end{bmatrix}, \; f_{S,\xi}(k) = \begin{bmatrix} f_S(k-\xi) \\ \vdots \\ f_S(k) \end{bmatrix}.$$

Writing (9.35)–(9.37), without loss of generality, in a unified form

$$r_\varsigma(k) = e_\varsigma(k) + H_{f_A,\varsigma} f_{A,\varsigma}(k) + H_{f_S,\varsigma} f_{S,\varsigma}(k) \tag{9.38}$$

the fault isolation and identification problems can then be formulated as

- finding mappings M_A, M_S so that

actuator fault isolation:

$$M_A H_{f_A,\varsigma} f_{A,\varsigma}(k) = \begin{bmatrix} \mathcal{A}_1 f_{A,1,\varsigma}(k) \\ \vdots \\ \mathcal{A}_l f_{A,l,\varsigma}(k) \end{bmatrix} \tag{9.39}$$

$$f_A(k) = \begin{bmatrix} f_{A,1}(k) \\ \vdots \\ f_{A,l}(k) \end{bmatrix}, \; f_{A,i,\varsigma}(k) = \begin{bmatrix} f_{A,i}(k-\xi) \\ \vdots \\ f_{A,i}(k) \end{bmatrix}, \; i = 1, \ldots, l (= k_f)$$

sensor fault isolation:

$$M_S H_{f_S,\varsigma} = \begin{bmatrix} \mathcal{S}_1 f_{S,1,\varsigma}(k) \\ \vdots \\ \mathcal{S}_m f_{S,m,\varsigma}(k) \end{bmatrix} \tag{9.40}$$

$$f_S(k) = \begin{bmatrix} f_{S,1}(k) \\ \vdots \\ f_{S,l}(k) \end{bmatrix}, \; f_{S,i,\varsigma}(k) = \begin{bmatrix} f_{S,i}(k-\xi) \\ \vdots \\ f_{S,i}(k) \end{bmatrix}, \; i = 1, \ldots, m \; (= k_f)$$

where $\mathcal{A}_1, \ldots, \mathcal{A}_l, \mathcal{S}_1, \ldots, \mathcal{S}_m$ denote some (non-zero) matrices.

- finding mappings $H^+_{f_A,\varsigma}$, $H^+_{f_S,\varsigma}$ so that

$$\text{actuator fault identification:} \quad \hat{f}_{A,s}(k) = H^+_{f_A,\varsigma} r_\varsigma(k) \tag{9.41}$$

$$\text{sensor fault identification:} \quad \hat{f}_{S,s}(k) = H^+_{f_S,\varsigma} r_\varsigma(k). \tag{9.42}$$

9.4.2 Fault Isolation Schemes

Next, we introduce a standard approach for the above-defined fault isolation problem. Let

$$H_{f_{A,\varsigma}} f_{A,\varsigma}(k) = \bar{H}^i_{f_{A,\varsigma}} \bar{f}^i_{A,\varsigma}(k) + H^i_{f_{A,\varsigma}} f_{A,i,\varsigma}(k), i = 1, \ldots, l$$

$$\bar{f}^i_{A,\varsigma}(k) = \begin{bmatrix} \bar{f}^i_A(k-\varsigma) \\ \vdots \\ \bar{f}^i_A(k) \end{bmatrix}, \bar{f}^i_A(k-j) = \begin{bmatrix} f_{A,1}(k-j) \\ \vdots \\ f_{A,i-1}(k-j) \\ f_{A,i+1}(k-j) \\ \vdots \\ f_{A,l}(k-j) \end{bmatrix}, j = 0, \ldots, \varsigma$$

$$H_{f_{S,\varsigma}} f_{S,\varsigma}(k) = \bar{H}^i_{f_{S,\varsigma}} \bar{f}^i_{S,\varsigma}(k) + H^i_{f_{S,\varsigma}} f_{S,i,\varsigma}(k), i = 1, \ldots, m$$

$$\bar{f}^i_{S,\varsigma}(k) = \begin{bmatrix} \bar{f}^i_S(k-\varsigma) \\ \vdots \\ \bar{f}^i_S(k) \end{bmatrix}, \bar{f}^i_S(k-j) = \begin{bmatrix} f_{S,1}(k-j) \\ \vdots \\ f_{S,i-1}(k-j) \\ f_{S,i+1}(k-j) \\ \vdots \\ f_{S,l}(k-j) \end{bmatrix}, j = 0, \ldots, \varsigma.$$

The actuator and sensor fault isolation problems (9.39) and (9.40) can be solved by finding $\bar{\mathcal{A}}_i, i = 1, \ldots, l, \bar{\mathcal{S}}_i, i = 1, \ldots, m$,

$$\bar{\mathcal{A}}_i \begin{bmatrix} \bar{H}^i_{f_{A,\varsigma}} & H^i_{f_{A,\varsigma}} \end{bmatrix} = \begin{bmatrix} 0 & \mathcal{A}_i \end{bmatrix} \tag{9.43}$$

$$\bar{\mathcal{S}}_i \begin{bmatrix} \bar{H}^i_{f_{S,\varsigma}} & H^i_{f_{S,\varsigma}} \end{bmatrix} = \begin{bmatrix} 0 & \mathcal{S}_i \end{bmatrix} \tag{9.44}$$

where $\mathcal{A}_i, \mathcal{S}_i$ can be any matrix (vector) different from zero when no additional demand on them is made. In this case, a necessary and sufficient condition is

$$rank \begin{bmatrix} \bar{H}^i_{f_{A,\varsigma}} & H^i_{f_{A,\varsigma}} \end{bmatrix} > rank \left(\bar{H}^i_{f_{A,\varsigma}} \right), i = 1, \ldots, l \tag{9.45}$$

$$rank \begin{bmatrix} \bar{H}^i_{f_{S,\varsigma}} & H^i_{f_{S,\varsigma}} \end{bmatrix} > rank \left(\bar{H}^i_{f_{S,\varsigma}} \right), i = 1, \ldots, m. \tag{9.46}$$

Note that conditions (9.45) and (9.46) are satisfied as long as there respectively exists a row in $H^i_{f_{A,\varsigma}}$ and $H^i_{f_{S,\varsigma}}$ that is linearly independent of the rows in $\bar{H}^i_{f_{A,\varsigma}}$ and $\bar{H}^i_{f_{S,\varsigma}}$. On the other hand, if, for instance, $\mathcal{A}_i$ is a vector, fault diagnosis performance may become (very) poor, since some type of $f_{A,i,\varsigma}(k)$ may result in

$$\bar{\mathcal{A}}_i H^i_{f_{A,\varsigma}} f_{A,i,\varsigma}(k) = \mathcal{A}_i f_{A,i,\varsigma}(k) = 0$$

and thus is undetectable or at least undetectable in some time interval. Therefore, it is reasonable to find those $\bar{\mathcal{A}}_i, \bar{\mathcal{S}}_i$ so that $rank(\mathcal{A}_i), rank(\mathcal{S}_i)$ are as large as possible. The ideal case can be expressed by

$$rank\left(H_{f_A,\varsigma}\right) = rank\left[\begin{array}{ccc} H^1_{f_A,\varsigma} & \cdots & H^l_{f_A,\varsigma} \end{array}\right] = \sum_{i=1}^{l} rank\left(H^i_{f_A,\varsigma}\right)$$

$$rank\left(H_{f_S,\varsigma}\right) = rank\left[\begin{array}{ccc} H^1_{f_S,\varsigma} & \cdots & H^m_{f_S,\varsigma} \end{array}\right] = \sum_{i=1}^{m} rank\left(H^i_{f_S,\varsigma}\right).$$

which are the existence conditions for a total decoupling of these faults.

9.4.3 Fault Identification Schemes

Roughly speaking, fault identification is an inversion problem. General solutions for fault identification problems (9.41) and (9.42) can be given as follows. Let

$$H_{f_A,\varsigma} = U_A \left[\begin{array}{cc} \Sigma_A & 0 \\ 0 & 0 \end{array}\right] V_A^T, H_{f_S,\varsigma} = U_S \left[\begin{array}{cc} \Sigma_S & 0 \\ 0 & 0 \end{array}\right] V_S^T$$

be the SVDs of $H_{f_A,\varsigma}$ and $H_{f_S,\varsigma}$. It turns out the pseudo-inverses of $H_{f_A,\varsigma}$ and $H_{f_S,\varsigma}$ are given by

$$H^-_{f_A,\varsigma} = V_A \left[\begin{array}{cc} \Sigma_A^{-1} & 0 \\ 0 & 0 \end{array}\right] U_A^T, H^-_{f_S,\varsigma} = V_S \left[\begin{array}{cc} \Sigma_S^{-1} & 0 \\ 0 & 0 \end{array}\right] U_S^T. \tag{9.47}$$

It should be mentioned that in the previous study no knowledge of faults is assumed. In practice, types of the faults are often known. For instance, offset and drift are typical forms of sensors faults. In this case,

$$f_{S,\varsigma}(k) = \Phi_S \left[\begin{array}{c} f_o \\ \Delta_S \end{array}\right], \Phi_S = \left[\begin{array}{cc} I & I \\ I & 2I \\ \vdots & \vdots \\ I & (\varsigma+1)I \end{array}\right], k \geq k_0 + \varsigma$$

where f_o, Δ_S are unknown vectors representing offset and drift changes and k_0 denotes the time instance at which the faults occur. As a result, fault identification problem can be formulated as estimating f_o, Δ_S with

$$H_{f_S,\varsigma} f_{S,\varsigma}(k+1) = H_{f_S,\varsigma} \Phi_S \left[\begin{array}{c} f_o \\ \Delta_S \end{array}\right]. \tag{9.48}$$

On the assumption that $H_{fS,\varsigma}\Phi_S$ is of full column rank, the sensor fault identification is achievable by means of an LS estimation

$$\begin{bmatrix} \hat{f}_o(k) \\ \hat{\Delta}_S(k) \end{bmatrix} = \left(\Phi_S^T H_{fS,\varsigma}^T H_{fS,\varsigma}\Phi_S\right)^{-1} \left(H_{fS,\varsigma}\Phi_S\right)^T r_\varsigma(k) \Longrightarrow \tag{9.49}$$

$$\hat{f}_{S,\varsigma}(k) = \Phi_S \left(\Phi_S^T H_{fS,\varsigma}^T H_{fS,\varsigma}\Phi_S\right)^{-1} \left(H_{fS,\varsigma}\Phi_S\right)^T r_\varsigma(k). \tag{9.50}$$

This idea can be slightly extended to a more general case. Assume that a sensor or an actuator fault is described by a dynamic process

$$x_f(k+1) = A_f x_f(k) + u_f,\ f(k) = C_f x_f(k) + f_o \tag{9.51}$$

where A_f, C_f are some known matrices and $x_f(0), u_f, f_o$ are unknown constant vectors. It leads to

$$f_\varsigma(k) = \begin{bmatrix} f(k-\varsigma) \\ \vdots \\ f(k) \end{bmatrix} = \Phi \begin{bmatrix} f_o \\ x_f(0) \\ u_f \end{bmatrix}, \Phi = \begin{bmatrix} I & C_f & 0 \\ I & C_f A_f & C_f \\ \vdots & \vdots & \vdots \\ I & C_f A_f^\varsigma & C_f A_f^{\varsigma-1} \end{bmatrix}, k \geq k_0 + \varsigma.$$

As a result, the fault identification problem is solved by estimating $x_f(0), u_f, f_o$ as follows

$$\begin{bmatrix} \hat{f}_o \\ \hat{x}_f(0) \\ \hat{u}_f \end{bmatrix} = \left(\Phi^T H_{f,\varsigma}^T H_{f,\varsigma}\Phi\right)^{-1} \left(H_{f,\varsigma}\Phi_S\right)^T r_\varsigma(k) \Longrightarrow \tag{9.52}$$

$$\hat{f}_\varsigma(k) = \Phi \left(\Phi^T H_{f,\varsigma}^T H_{f,\varsigma}\Phi\right)^{-1} \left(H_{f,\varsigma}\Phi_S\right)^T r_\varsigma(k) \tag{9.53}$$

where $H_{f,\varsigma}$ stands either for $H_{fS,\varsigma}$ or for $H_{fA,\varsigma}$.

9.5 Case Study: Fault Detection in Three-Tank System

In this section, the application of data-driven fault detection scheme proposed in this chapter, namely Algorithm 9.3, is illustrated on the simulation model of the three-tank system.

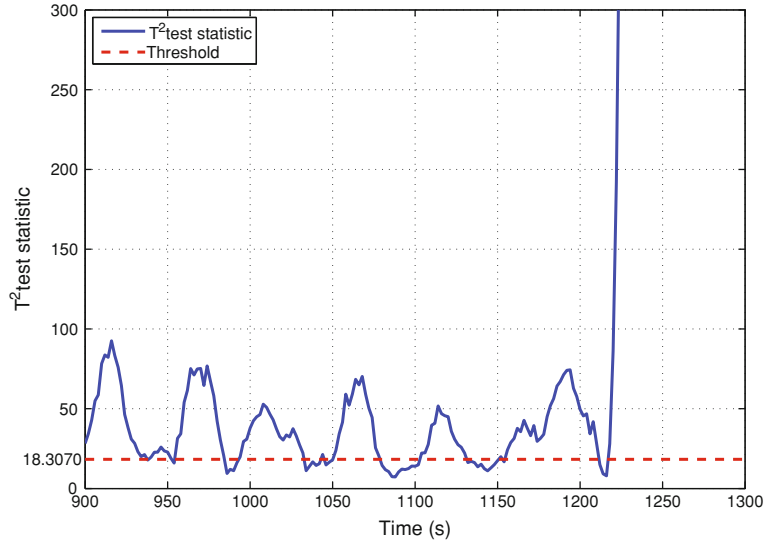


Fig. 9.1 Detection of a 5 % leakage in tank 1

9.5.1 System and Test Setup

The basic system and test settings are identical with the ones given in Sects. 2.1 and 3.4. The control signals sent to the two pumps are considered as the input variables, while the sensor signals for the water level in the three tanks are the output variables.

In the offline training phase, data are collected for a duration of 2,000 s, as the system works in the steady operation state. They are used for the identification of the kernel representation of the system. For online monitoring purpose, settings are:

- $s = 10$
- the significance level: $\alpha = 5\,\%$ and
- the corresponding threshold: $J_{th} = 18.3070$.

9.5.2 Test Results

Eleven fault scenarios, including three sensor faults, two actuator faults, three leakages in the three tanks and three plugging faults between the tanks are simulated. In all simulations, the fault occurrence time is set to be $t = 1{,}200$ s. In Figs. 9.1, 9.2, 9.3 and 9.4, representative results are shown. Although all simulated faults can be successfully detected, it is clear that the FAR is too high. A reason for it ill-conditioning Σ_{res}.

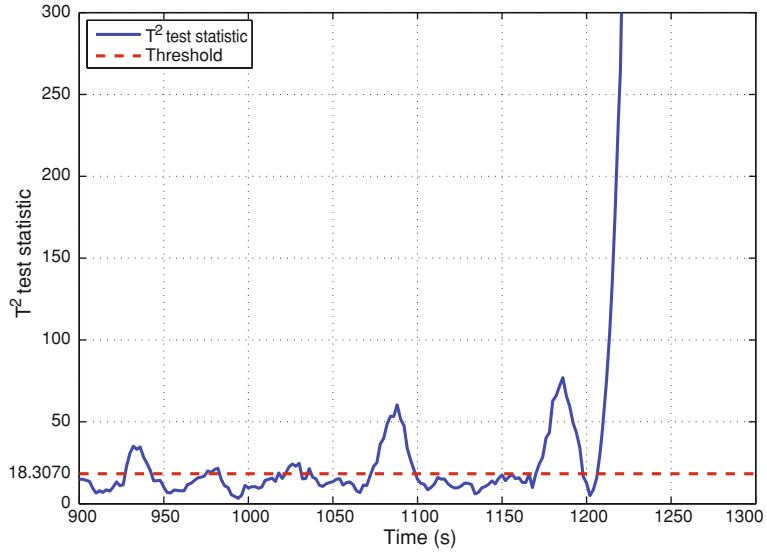


Fig. 9.2 Detection of a 5 % plugging between tank 1 and tank 3

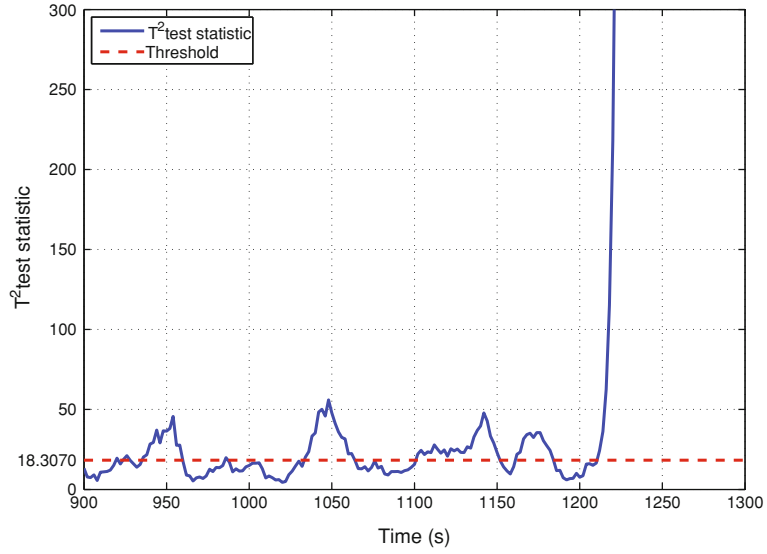


Fig. 9.3 Detection of a 10 % power reduction in pump 1

9.5.3 Handling of Ill-Conditioning Σ_{res}

To avoid possible ill-conditioning Σ_{res}, the PCA technique introduced in Chap. 5 is tested in our case study. Following the threshold setting given in Chap. 5, the

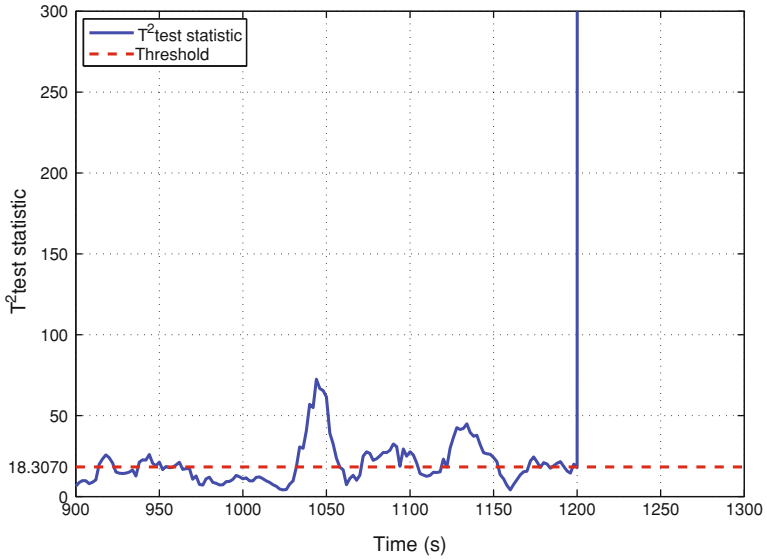


Fig. 9.4 Detection of a 5 cm offset in the level sensor 3

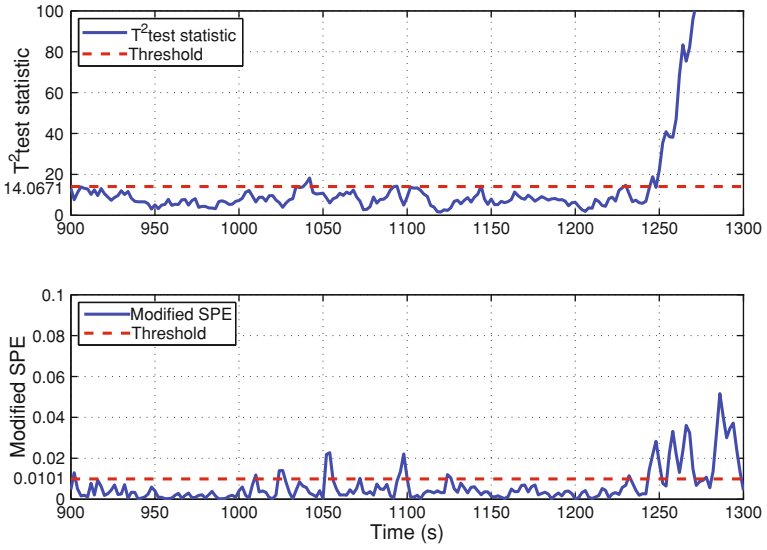


Fig. 9.5 Detection of a 5 % leakage in tank 1

thresholds of T^2 test statistic, T^2_{PCA}, and the modified form of SPE, $J_{T^2_{new}}$, are respectively set to be

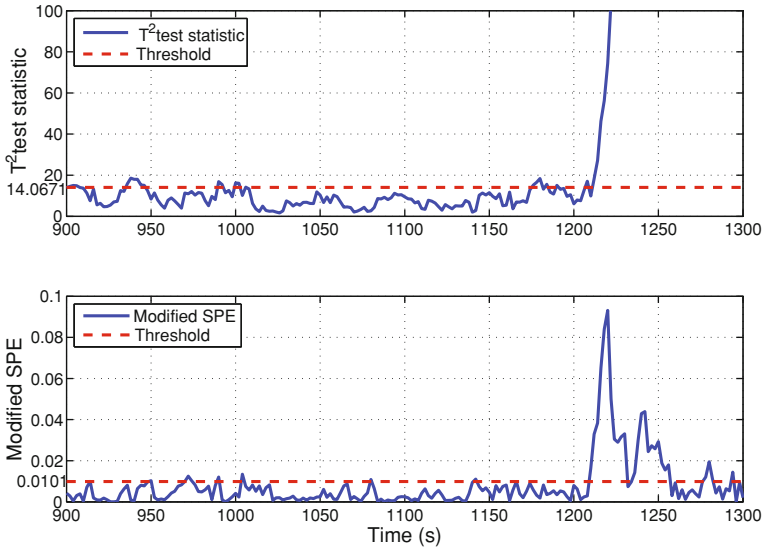


Fig. 9.6 Detection of a 5 % plugging between tank 1 and tank 3

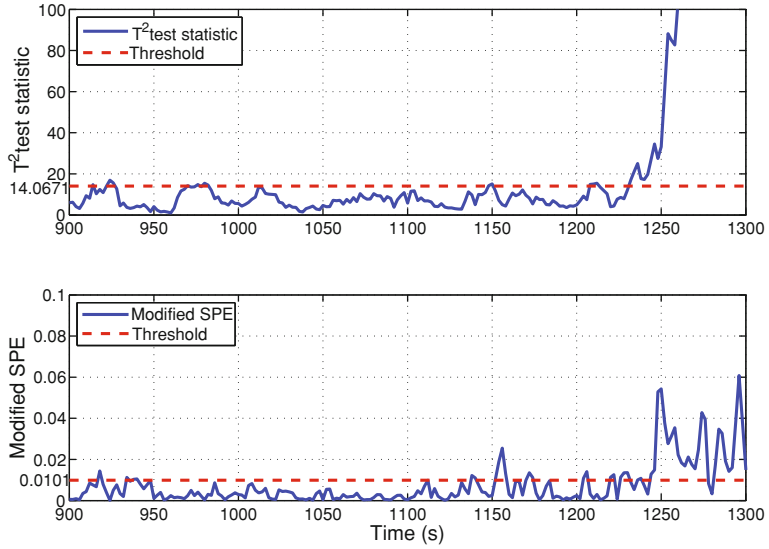


Fig. 9.7 Detection of a 10 % power reduction in pump 1

$$J_{th,T^2_{PCA}} = 14.0671,\ J_{th,T^2_{new}} = 0.0101.$$

The representative and comparable fault detection results are given in Figs. 9.5, 9.6, 9.7 and 9.8. It can be seen that the false alarm rate becomes lower.

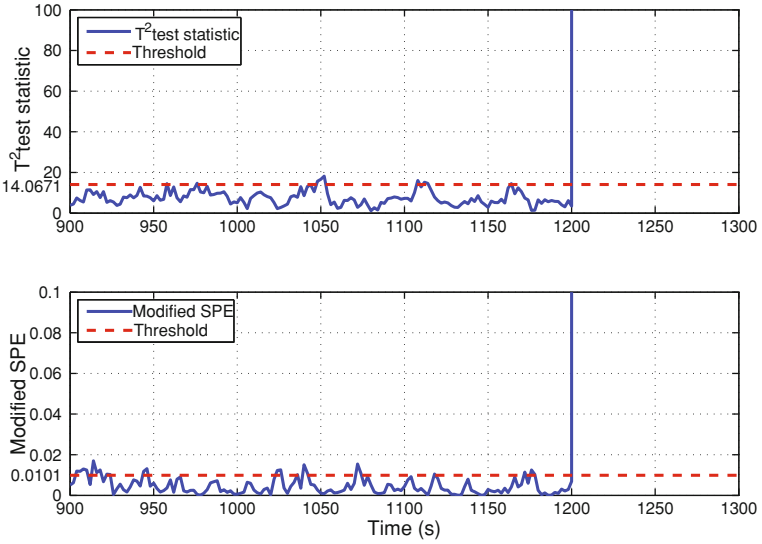


Fig. 9.8 Detection of a 5 cm offset in the level sensor 3

9.6 Notes and References

The I/O data models and the major computations involved in the identification procedure in Schemes I and II are well-known in the SIM framework [1–4]. The application of QR decomposition technique for numerically reliable identification and estimation is well described, for instance, in [3, 4].

The residual generation Scheme II has been proposed by Dong [5] and applied in further studies [6, 7].

The development of Scheme III has been inspired by the work in [8–10]. The idea behind this scheme is a direct identification of the data-driven form of the kernel representation. The application of the QR decomposition results in a numerically reliable identification of the KR.

In practice of (statistical) process monitoring, contribution plots are widely applied as a standard PCA-based fault isolation and identification technique [11]. Recently [12–14], have proposed to estimate/identify the faults in a statistical process by means of reconstruction technique. In [5, 6], data-driven approaches have been developed to identify faults in a dynamic process. It is remarkable that these approaches have been developed on the assumption of an available fault model. Considering that sensor and actuator behavior is implicitly embedded in the I/O data model of a process, we have addressed sensor and actuator fault isolation and identification issues without an explicit fault model. For our purpose, the isolation and identification problems have been formulated as trivial linear equation problems. In the context of parity space based fault isolation and identification, these issues are addressed and numerous solutions are presented in [15].

At the end of this chapter, we would like to emphasize that the three types of residual generators introduce in this chapter are FIR filter and open-loop configurated. In comparison with observer-based fault systems as briefly introduced in the last chapter, these fault detections systems are less robust against disturbances and uncertainties and less attractive for the online implementation.

References

1. Favoreel W, Moor BD, Overschee PV (2000) Subspace state space system identification for industrial processes. J Process Control 10:149–155
2. Overschee PV, Moor BD (1996) Subspace identification for linear systems. Kluwer Academic Publishers, Dordrecht
3. Qin SJ (2006) An overview of subspace identification. Comput Chem Eng 30:1502–1513
4. Huang B, Kadali R (2008) Dynamic modelling, predictive control and performance monitoring, a data-driven subspace approach. Springer, London
5. Dong J (2009) Data driven fault tolerant control: a subspace approach. Ph.D. dissertation, Technische Universiteit Delft
6. Dong J, Verhaegen M (2012) Identification of fault estimation filter from I/O data for systems with stable inversion. IEEE Trans Atom Control 57:1347–1362
7. Dong J, Verhaegen M, Gustafsson F (2012) Robust fault detection with statistical uncertainty in identified parameters. IEEE Trans Signal Process 60:5064–5076
8. Qin SJ, Li W (2001) Detection and identification of faulty sensors in dynamic processes. AIChE J 47:1581–1593
9. Wang J, Qin SJ (2002) A new subspace identification approach based on principle component analysis. J Process Control 12:841–855
10. Ding SX, Zhang P, Naik A, Ding E, Huang B (2009) Subspace method aided data-driven design of fault detection and isolation systems. J Process Control 19:1496–1510
11. Chiang LH, Russell EL, Braatz RD (2001) Fault detection and diagnosis in industrial systems. Springer, London
12. Qin SJ (2009) Data-driven fault detection and diagnosis for complex industrial processes. In: Proceedings of IFAC SAFEPROCESS Symposium, pp 1115–1125
13. Alcala CF, Qin SJ (2009) Reconstruction-based contribution for process monitoring. Automatica 45:1593–1600
14. Li G, Qin SJ, Ji Y, Zhou D (2010) Reconstruction based fault pronosis for continous processes. Control Eng Pract 18:1211–1219
15. Ding SX (2013) Model-based fault diagnosis techniques: design schemes, algorithms and tools, 2nd edn. Springer, London

Chapter 10
Data-Driven Design of Observer-Based Fault Diagnosis Systems

10.1 Motivation and Problem Formulation

Motivated by the fact that the residual generators introduced in the previous chapter are open-loop configurated and thus less robust and involved for the online computation, we present, in this chapter, schemes for the construction/realization of observer-based fault diagnosis systems with the aid of the data-driven realization of kernel representation $\Psi_s^{\perp}$ given in Algorithm 9.3. As known, observer-based residual generators are closed-loop configurated with a feedback of the residual signals and the involved computations are realized in a recursive form. By a suitable design, they are of high robustness and demand less online computational capacity.

For our purpose, we begin with the generation of a scalar residual signal, extend the study to the multiple case, and finally introduce the design of a Kalman filter based fault diagnosis system. The study on the multiple residual generation will also lead to the construction of a full-order observer, which can be used for the process monitoring and soft-sensor purpose.

10.2 Parity Vectors Based Construction of Observer-Based Residual Generators

The results given in this section are essential for the subsequent study and link the data-driven realization of the kernel representation and the observer-based residual generation.

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_10, © Springer-Verlag London 2014

10.2.1 Generation of a Scalar Residual Signal

Consider process model (8.2), (8.3) and a parity vector

$$\alpha_s = \left[\alpha_{s,0}\ \alpha_{s,1}\ \cdots\ \alpha_{s,s}\right] \in \mathcal{R}^{(s+1)m},\ \alpha_s \begin{bmatrix} C \\ CA \\ \vdots \\ CA^s \end{bmatrix} = 0$$

$\alpha_{s,i} \in \mathcal{R}^m$, $i = 0, 1, \ldots, s$. It has been demonstrated that matrices

$$A_z = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \in \mathcal{R}^{s\times s},\ L_z = -\begin{bmatrix} \alpha_{s,0} \\ \alpha_{s,1} \\ \vdots \\ \alpha_{s,s-1} \end{bmatrix} \tag{10.1}$$

$$T = \begin{bmatrix} \alpha_{s,1} & \alpha_{s,2} & \cdots & \alpha_{s,s-1} & \alpha_{s,s} \\ \alpha_{s,2} & \cdots & \cdots & \alpha_{s,s} & 0 \\ \vdots & \cdots & \cdots & \vdots & \vdots \\ \alpha_{s,s} & 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix} \tag{10.2}$$

solve the Luenberger equations

$$TA - A_z T = L_z C,\ c_z T = gC,\ c_z = \left[0 \cdots 0\ 1\right],\ g = \alpha_{s,s}. \tag{10.3}$$

A direct application of this result is the construction of an observer-based residual generator for the given parity vector α_s as follows

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) \in \mathcal{R}^s,\ B_z = TB - L_z D \tag{10.4}$$

$$r(k) = gy(k) - c_z z(k) - d_z u(k) \in \mathcal{R},\ d_z = gD. \tag{10.5}$$

Note that

$$\begin{bmatrix} B_z \\ d_z \end{bmatrix} = \begin{bmatrix} \alpha_{s,0} & \alpha_{s,1} & \cdots & \alpha_{s,s-1} & \alpha_{s,s} \\ \alpha_{s,1} & \cdots & \cdots & \alpha_{s,s} & 0 \\ \vdots & \cdots & \cdots & \vdots & \vdots \\ \alpha_{s,s} & 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} D \\ CB \\ CAB \\ \vdots \\ CA^{s-1}B \end{bmatrix} \tag{10.6}$$

$$= \begin{bmatrix} \alpha_s H_{u,s}(:, 1:l) \\ \alpha_s H_{u,s}(:, l+1:2l) \\ \vdots \\ \alpha_s H_{u,s}(:, sl+1:(s+1)l) \end{bmatrix}.$$

Equations (10.1), (10.3) and (10.6) allow a direct construction of a residual generator given by (10.4), (10.5) using a row of $\Psi_s^\perp$ and without any additional design effort. Concretely, let $\psi_s^\perp$ be a row of $\Psi_s^\perp$ and of the form

$$\psi_s^\perp = \left[\psi_{s,u}^\perp \ \psi_{s,y}^\perp \right].$$

Since $\psi_{s,y}^\perp$ is a parity vector and

$$\psi_{s,u}^\perp = -\psi_{s,y}^\perp H_{u,s}$$

we have, besides of A_z, c_z given in (10.1) and (10.3), B_z, d_z, g, L_z formed in terms of $\psi_{s,u}^\perp, \psi_{s,y}^\perp$:

$$L_z = -\begin{bmatrix} \psi_{s,y}^\perp(1:m) \\ \vdots \\ \psi_{s,y}^\perp((s-1)m+1:sm) \end{bmatrix}, g = \psi_{s,y}^\perp((sm+1:(s+1)m) \tag{10.7}$$

$$B_z = -\begin{bmatrix} \psi_{s,u}^\perp(1:l) \\ \vdots \\ \psi_{s,u}^\perp((s-1)l+1:sl) \end{bmatrix}, d_z = -\psi_{s,u}^\perp(sl+1:(s+1)l). \tag{10.8}$$

Remember that s is generally selected sufficiently large (typically much larger than n) during the identification/training phase. From the real-time computational viewpoint, it is of practical advantage to reduce the order of the residual generator as much as possible. For this purpose, the following algorithm is useful.

Algorithm 10.1 *Order reduction*

S1: Do a QR decomposition of $\Gamma_s^\perp V = \left[Q_1 R_1 \ \ Q_2 \right]$ *with*

$$V = \begin{bmatrix} 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ \vdots & \vdots & \iddots & \iddots & \vdots \\ 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathcal{R}^{m(s+1)\times m(s+1)}$$

R_1 *being a* $\eta \times \eta$ *upper triangular,* Q_1 *a* $\eta \times \eta$ *with orthonormal columns and* Q_2 *a* $\eta \times n$ *matrix*
S2: Compute

$$Q_1^{-1}\Gamma_s^\perp = \left[R_1 \ \ Q_1^{-1}Q_2 \right] V^{-1}.$$

In the above algorithm, $\Gamma_s^{\perp}$ denotes the null matrix of Γ_s, that is $\Gamma_s^{\perp}\Gamma_s = 0$. As a result,

$$Q_1^{-1}\Gamma_s^{\perp} = \begin{bmatrix} Q_3 & \Psi \end{bmatrix} \tag{10.9}$$

where Q_3 is a $\eta \times n$ matrix, $\eta = m(s+1) - n$, and Ψ is of the form

$$\Psi = \begin{bmatrix} \psi_{1,1} & \cdots & \psi_{1,\eta-2} & \psi_{1,\eta-1} & \psi_{1,\eta} \\ \psi_{2,1} & \cdots & \psi_{2,\eta-2} & \psi_{2,\eta-1} & 0 \\ \vdots & \vdots & \iddots & \iddots & \vdots \\ \psi_{\eta-1,1} & \psi_{\eta-1,2} & 0 & \cdots & 0 \\ \psi_{\eta,1} & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathcal{R}^{\eta\times\eta}. \tag{10.10}$$

It is evident that the last row of $Q_1^{-1}\Gamma_s^{\perp}$ is of the form

$$\begin{bmatrix} \alpha_n & 0 & \cdots & 0 \end{bmatrix} \in \mathcal{R}^{(s+1)m}$$

and thus $\alpha_n \in \mathcal{R}^{n+1}$ is a parity vector whose order is n.

Recall that $\Psi_s^{\perp}$ delivered by Algorithm 9.3 is composed of $\Psi_{s,y}^{\perp}$, $\Psi_{s,u}^{\perp}$ with $\Psi_{s,y}^{\perp} = \Gamma_s^{\perp}$. Applying Algorithm 9.3 to $\Psi_{s,y}^{\perp}$ yields Q_1^{-1}, α_n. Now, multiply Q_1^{-1} to $-\Psi_{s,u}^{\perp} = \Psi_{s,y}^{\perp}H_{u,s}$ and denote the last row of $-Q_1^{-1}\Psi_{s,u}^{\perp}$ by $\alpha_{n,u}$. It holds

$$\alpha_{n,u} = \alpha_n H_{u,s} \Longrightarrow \begin{bmatrix} B_z \\ d_z \end{bmatrix} = \begin{bmatrix} \alpha_{n,u}(1:l) \\ \alpha_{n,u}(l+1:2l) \\ \vdots \\ \alpha_{n,u}(nl+1:(n+1)l) \end{bmatrix}. \tag{10.11}$$

In summary, we have

Algorithm 10.2 *Observer-based residual generation system design*

S1: Run Algorithm 9.3 with outputs $\Psi_{s,y}^{\perp}$, $\Psi_{s,u}^{\perp}$
S2: Run Algorithm 10.1
S3: Compute $-Q_1^{-1}\Psi_{s,u}^{\perp}$ for $\alpha_{n,u}$
S4: Form B_z, d_z, g, L_z according to (10.7)–(10.8)
S5: Construct residual generator (10.4)–(10.5).

10.2.2 Generation of m-Dimensional Residual Vectors

It is well-known that an m-dimensional residual vector is necessary for a reliable fault detection and isolation in the framework of observer-based fault diagnosis systems. This motivates the study on extending Algorithm 10.2 to the multiple case. For our

purpose, we are going to present two schemes. The first one is an algorithm whose basic idea is to design a full order observer using a single parity vector and then generate an m-dimensional residual vector. To this end, the following theorem given by Mercere and Bako plays a central role.

Theorem 10.1 *Given an observable system*

$$x(k+1) = Ax(k) + Bu(k) \in \mathcal{R}^n, y(k) = Cx(k) + Du(k) \in \mathcal{R}^m$$

and assume that the system matrix A is non-derogatory. Let

$$\kappa_o = \left[\kappa_{o,1} \cdots \kappa_{o,m} \right] \in \mathcal{R}^m$$

be a vector generated randomly from a uniform distribution, then it holds with probability one that

$$rank \begin{bmatrix} \kappa_o C \\ \kappa_o CA \\ \vdots \\ \kappa_o CA^{n-1} \end{bmatrix} = n. \tag{10.12}$$

Suppose that a vector κ_o satisfying (10.12) is found. Without loss of generality, assume $\kappa_{o,1} \neq 0$. Let

$$\bar{y}_1(k) = \kappa_o y(k) = \kappa_o \begin{bmatrix} y_1(k) \\ \vdots \\ y_m(k) \end{bmatrix} := \bar{c}_1 x(k) + \bar{d}_1 u(k)$$

$$\bar{y}_2(k) = \begin{bmatrix} y_2(k) \\ \vdots \\ y_m(k) \end{bmatrix} := \bar{C}_2 x(k) + \bar{D}_2 u(k).$$

In order to generate an m-dimensional residual vector, we now re-order the output data set into

$$\bar{Y}_{k,s} = \begin{bmatrix} \bar{Y}^1_{k,s} \\ \bar{Y}^2_{k,s} \end{bmatrix} \in \mathcal{R}^{(s+1)m \times N}, \bar{Y}^1_{k,s} = \begin{bmatrix} \bar{y}_1(k-s) & \cdots & \bar{y}_1(k-s+N-1) \\ \vdots & \vdots & \vdots \\ \bar{y}_1(k) & \cdots & \bar{y}_1(k+N-1) \end{bmatrix}$$

$$\bar{Y}^2_{k,s} = \begin{bmatrix} \bar{y}_2(k-s) & \cdots & \bar{y}_2(k-s+N-1) \\ \vdots & \vdots & \vdots \\ \bar{y}_2(k) & \cdots & \bar{y}_2(k+N-1) \end{bmatrix} \in \mathcal{R}^{(s+1)(m-1) \times N} \tag{10.13}$$

which leads to, without considering the process and measurement noises, the following I/O data model

$$\bar{Y}_{k,s} = \Gamma_s X_{k-s} + H_{u,s} U_{k,s} \in \mathcal{R}^{(s+1)m \times N},\ \Gamma_s = \begin{bmatrix} \bar{c}_1 \\ \vdots \\ \bar{c}_1 A^s \\ \bar{C}_2 \\ \vdots \\ \bar{C}_2 A^s \end{bmatrix},\ H_{u,s} = \begin{bmatrix} H^1_{u,s} \\ H^2_{u,s} \end{bmatrix}$$

$$H^1_{u,s} = \begin{bmatrix} \bar{d}_1 & 0 & & \\ \bar{c}_1 B & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ \bar{c}_1 A^{s-1} B & \cdots & \bar{c}_1 B & \bar{d}_1 \end{bmatrix},\ H^2_{u,s} = \begin{bmatrix} \bar{D}_2 & 0 & & \\ \bar{C}_2 B & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ \bar{C}_2 A^{s-1} B & \cdots & \bar{C}_2 B & \bar{D}_2 \end{bmatrix}$$

and form the data sets for the identification as

$$Z_{k,s} = \begin{bmatrix} U_{k,s} \\ \bar{Y}_{k,s} \end{bmatrix},\ Z_{k-s-1,s} = \begin{bmatrix} U_{k-s-1,s} \\ \bar{Y}_{k-s-1,s} \end{bmatrix}. \tag{10.14}$$

Now, apply Algorithm 10.2 to these data sets and select the last row of the identified $Q_1^{-1}\Psi^{\perp}_{s,y} = Q_1^{-1}\Gamma_s^{\perp}$ given in Algorithm 10.1, which is of the form

$$Q_y := Q_1^{-1}\Psi^{\perp}_{s,y},\ Q_y(\eta, 1 : m(s+1)) = \begin{bmatrix} \alpha_n & 0 & \cdots & 0 \end{bmatrix} \in \mathcal{R}^{m(s+1)} \tag{10.15}$$

where α_n is a parity vector satisfying

$$\alpha_n \begin{bmatrix} \kappa_o C \\ \kappa_o C A \\ \vdots \\ \kappa_o C A^n \end{bmatrix} = \alpha_n \begin{bmatrix} \bar{c}_1 \\ \bar{c}_1 A \\ \vdots \\ \bar{c}_1 A^n \end{bmatrix} = 0.$$

As a result, $A_z \in \mathcal{R}^{n \times n}$, B_z, c_z, d_z, L_z can be formed. Note that (c_z, A_z) builds an observability canonical form and $z(k) = Tx(k)$ delivers an estimation of the full state vector, where $T \in \mathcal{R}^{n \times n}$ is the (regular) state transformation as defined in (10.2). Moreover,

$$\bar{C}_2 = \Gamma_s\left((s+1)+1 : (s+1)+m-1\right),\ Q_u := -Q_1^{-1}\Psi^{\perp}_{s,u} \tag{10.16}$$

$$Q_u(1 : \eta, sl+1 : (s+1)l) = Q_y(:, s+1)\bar{d}_1 + Q_y(:, sm+2 : (s+1)m)\bar{D}_2 \tag{10.17}$$

$$\bar{d}_1 = d_z/g. \tag{10.18}$$

Thus, an m-dimensional residual vector can be generated by

$$z(k+1) = A_z z(k) + B_z u(k) + L_z \kappa_o y(k) \tag{10.19}$$

$$r(k) = Gy(k) - C_z z(k) - D_z u(k) \tag{10.20}$$

$$G = \begin{bmatrix} g\kappa_o \\ 0\ I_{(m-1)\times(m-1)} \end{bmatrix}, C_z = \begin{bmatrix} c_z \\ \bar{C}_2 T^{-1} \end{bmatrix}, D_z = \begin{bmatrix} d_z \\ \bar{D}_2 \end{bmatrix}.$$

We call (C_z, A_z) quasi observability canonical form (QOCF).

Algorithm 10.3 *QOCF-based residual generation*

S1: Find κ_o using the algorithm given by Mercere and Bako
S2: Re-order the data sets according to (10.13)–(10.14)
S3: Run Algorithm 10.2 for $Q_1^{-1}\Psi_{s,y}^{\perp}$ and form A_z, B_z, c_z, d_z, L_z
S4: Compute $\bar{C}_2$ and $\bar{D}_2$ using (10.16)–(10.17).

The basic idea of the second scheme consists in constructing m residual generators, and each of them is driven by a single sensor signal and the process input variables. Let

$$C = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix}, c_i \in \mathcal{R}^n, i = 1, \ldots, m$$

and $\alpha_i, i = 1, \ldots, m$, be parity vectors satisfying

$$\alpha_i \begin{bmatrix} c_i \\ c_i A \\ \vdots \\ c_i A^{s_i} \end{bmatrix} = 0, \alpha_i = \begin{bmatrix} \alpha_{i,0} \cdots \alpha_{i,s_i} \end{bmatrix}, \alpha_{i,s_i} \neq 0.$$

Without loss of generality, we assume below $\alpha_{i,s_i} = 1$. Considering that

$$rank \begin{bmatrix} \Gamma_{s_1} \\ \vdots \\ \Gamma_{s_m} \end{bmatrix} = rank\,(\Gamma_s)\,, \Gamma_{s_i} = \begin{bmatrix} c_i \\ c_i A \\ \vdots \\ c_i A^{s_i} \end{bmatrix}, i = 1, \ldots, m \tag{10.21}$$

where $s_i, i = 1, \ldots, m$, are not smaller than the largest observability index and $s \geq \max\{s_i, i = 1, \ldots, m\}$, and moreover

$$rank(T) = n, T = \begin{bmatrix} T_1 \\ \vdots \\ T_m \end{bmatrix} \tag{10.22}$$

$$T_i = \begin{bmatrix} \alpha_{i,1} & \alpha_{i,2} & \cdots & \alpha_{i,s_i-1} & \alpha_{i,s_i} \\ \alpha_{2,2} & \cdots & \cdots & \alpha_{i,s_i} & 0 \\ \vdots & \cdots & \cdots & \vdots & \vdots \\ \alpha_{i,s_i} & 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} c_i \\ c_i A \\ \vdots \\ c_i A^{s_i-1} \end{bmatrix}, i = 1, \ldots, m$$

the residual generator bank,

$$z_i(k+1) = A_{z,i} z_i(k) + B_{z,i} u(k) + l_{z,i} y_i(k) \in \mathcal{R}^{s_i} \tag{10.23}$$

$$r_i(k) = y_i(k) - c_{z,i} z(k) - d_{z,i} u(k) \in \mathcal{R}, i = 1, \ldots, m \tag{10.24}$$

with $A_{z,i}, B_{z,i}, l_{z,i}, c_{z,i}, d_{z,i}$ being defined in (10.1)–(10.3), delivers an m-dimensional residual vector that contains all information about the system disturbances and faults like an FDF. Note that $g_i = \alpha_{i,s_i} = 1$. This observation motivates us to apply the following algorithm for generating an m-dimensional residual vector.

Algorithm 10.4 *Generation of an m-dimensional residual vector*

S1: For $i = 1 : m$
S2: Run Algorithm 10.2 with the data from y_i and all process input variables
S3: End.

10.2.3 Data-Driven Design of Kalman Filter Based Residual Generators

As introduced in Chap. 8, the steady Kalman filter (8.50), (8.51) delivers an innovation $e(k) = y(k) - \hat{y}(k)$ with

$$\mathcal{E}\left(e(i)e^T(j)\right) = \Sigma_e \delta_{ij} \tag{10.25}$$

and can be applied to build, together with the T^2 statistic $J = e^T(k)\Sigma_e^{-1}e(k)$ and the threshold $J_{th} = \chi_\alpha^2(m)$, a fault detection system with the optimal trade-off between the false alarm rate and fault detectability. This subsection addresses the identification of the Kalman filter gain matrix K and the covariance matrix Σ_e with the aid of Algorithm 9.3 aiming at designing a Kalman filter based fault detection system.

Recalling that

$$R_{33} Q_3 = H_{w,s} W_{k,s} + V_{k,s} = H_{e,s} E_{k,s}, \frac{1}{N-1} E_{k,s} Y_{k-s}^T \approx \begin{bmatrix} \Sigma_e \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$Y_{k-s} = \left[\, y(k-s) \cdots y(k-s+N-1) \,\right] \in \mathcal{R}^{m\times N}$$

it turns out

$$\frac{1}{N-1}\Psi_s^{\perp}\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} Y_{k-s}^T = \frac{\Psi_{s,y}^{\perp} R_{33} Q_3 Y_{k-s}^T}{N-1} = \frac{\Psi_{s,y}^{\perp} H_{e,s} E_{k,s} Y_{k-s}^T}{N-1} \tag{10.26}$$

$$\approx \Psi_{s,y}^{\perp}\begin{bmatrix} \Sigma_e \\ CK\Sigma_e \\ \vdots \\ CA^{s-1}K\Sigma_e \end{bmatrix} \tag{10.27}$$

where $\Psi_s^{\perp} = \left[\, \Psi_{s,u}^{\perp}\ \Psi_{s,y}^{\perp} \,\right]$ is delivered by Algorithm 9.3. Now, solve

$$\Psi_{s,y}^{\perp}\Gamma_s = 0 \tag{10.28}$$

for Γ_s and further compute

$$\begin{bmatrix} \Sigma_e \\ K\Sigma_e \end{bmatrix} = \Xi^{-}\frac{1}{N-1}\Psi_s^{\perp}\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} Y_{k-s}^T,\ \Xi = \Psi_{s,y}^{\perp}\begin{bmatrix} I & 0 \\ 0 & \Gamma_s(1:sm,:) \end{bmatrix} \tag{10.29}$$

$$\Xi^{-} = \left(\Xi^T\Xi\right)^{-1}\Xi^T,\ \Xi^{-}\Xi = I \tag{10.30}$$

for Σ_e, K. Next, we deal with the construction of a Kalman filter based residual generator.

Without loss of generality, suppose that by Algorithm 10.3 an n-dimensional state vector $z(k)$ is built which satisfies

$$z(k) = Tx(k),\ C_z T = C,\ rank(T) = n.$$

To determine T, (10.2) is available with the identified Γ_s and the parity vector adopted in Algorithm 10.3. Note that although Γ_s identified by solving (10.28) is not unique (up to a regular transformation), the product TK is unique with K identified based on (10.29). Moreover, it follows from Luenberger Eq. (8.33) that

$$TA = A_z T + L_z C \Longleftrightarrow TAT^{-1} = A_z + L_z C_z.$$

As a result, we are able to construct a Kalman filter based residual generator as follows

$$z(k+1) = \bar{A}_z z(k) + \bar{B}_z u(k) + TKe(k) \tag{10.31}$$

$$r(k) = e(k) = y(k) - C_z z(k) - D_z u(k) \tag{10.32}$$

$$\bar{A}_z = A_z + L_z C_z,\ \bar{B}_z = B_z + L_z D_z$$

with B_z, D_z being determined by means of $\Psi_{s,u}^{\perp}$. It is straightforward to prove that the dynamics of the residual generator is governed by

$$\begin{aligned} \Delta x(k+1) &= \bar{A}_K \Delta x(k) + T w(k) - T K v(k), \, r(k) = C \Delta x(k) + v(k) \\ \Delta x(k) &= T x(k) - z(k), \, \bar{A}_K = T (A - KC) T^{-1}. \end{aligned} \tag{10.33}$$

Algorithm 10.5 *Data-driven design of a Kalman filter based residual generator*

S1: Run Algorithm 10.3
S2: Compute $\frac{1}{N-1} \Psi_{s,y}^{\perp} R_{33} Q_3 Y_{k-s}^T$
S3: Solve (10.28) and (10.29)
S4: Form $\bar{A}_z$, $\bar{B}_z$, T
S5: Construct Kalman filter (10.31)–(10.32).

10.3 Fault Detection, Isolation and Identification

In the last section, a number of observer-based residual generation schemes have been presented, which allow us to implement fault diagnosis systems in a recursive form and with, thanks to their closed-loop configuration, high robustness. A further benefit for the observer-based realization is that it links the data-driven and model-based FDI schemes and serves as a platform for the application of the powerful model-based methods to address the FDI issues. In this section, we briefly address FDI issues using the observer-based residual generators.

10.3.1 On Fault Detection

Suppose that a Kalman filter based residual generator is designed with the aid of Algorithm 10.5. Considering that $r(k) = e(k) \sim \mathcal{N}(0, \Sigma_e)$ and $e(k)$ is white, it is reasonable to apply T^2 statistic

$$J = r^T(k) \Sigma_e^{-1} r(k) \tag{10.34}$$

for the fault detection purpose. Correspondingly, the threshold is set to be

$$J_{th} = \chi_{\alpha}^2(m). \tag{10.35}$$

In general, if residual generator (10.19), (10.20) is applied for fault isolation, it holds, on the assumption of sensor and actuator faults defined in (8.21), (8.23),

$$\Delta x(k+1) = A_z \Delta x(k) + (B_z - L_z D)\, f_A(k) - L_z f_S(k)$$
$$r(k) = C_z \Delta x(k) + G f_S(k) + D_z f_A(k),\ \Delta x(k) = T x(k) - z(k).$$

Note that in the above system, no design freedom is available for approaching fault isolation problems. For this reason, the observer form

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) + L r_o(k) \tag{10.36}$$
$$r(k) = V r_o(k),\ r_o(k) = G y(k) - C_z z(k) - D_z u(k) \tag{10.37}$$

is adopted, which leads to

$$\begin{aligned}\Delta x(k+1) &= (A_z - L C_z)\, \Delta x(k) + \bar{B}_A f_A(k) - (L_z + LG)\, f_S(k) \\ &= (A_z - L C_z)\, \Delta x(k) + \left(E_f - L F_f\right) f(k),\ \bar{B}_A = B_z - L_z D - L D_z\end{aligned} \tag{10.38}$$

$$r(k) = V\left(C_z \Delta x(k) + G f_S(k) + D_z f_A(k)\right) = V C_z \Delta x(k) + V F_f f(k) \tag{10.39}$$

$$E_f = \left[\, B_z - L_z D \ -L_z \,\right],\ F_f = \left[\, D_z \ G \,\right],\ f(k) = \begin{bmatrix} f_A(k) \\ f_S(k) \end{bmatrix}.$$

This is the standard form of the dynamics of FDF with L, V as design parameters. There exists a number of design approaches for the FDF based fault detection. The reader is referred to the literature given at the end of this chapter.

10.3.2 Fault Isolation Schemes

Consider the FDF dynamics given in (10.38), (10.39). We first present an algorithm for isolating actuator faults on the assumption $D = 0 \Longrightarrow D_z = 0$, that is

$$\Delta x(k+1) = (A_z - L C_z)\, \Delta x(k) + B_z f_A(k),\ r(k) = V C_z \Delta x(k). \tag{10.40}$$

Let

$$B_z = \left[\, b_{z1} \cdots b_{zl} \,\right],\ \rho_i = \min\{j : C_z A_z^{j-1} b_{zi} \neq 0,\ j = 1, 2, \ldots\},\ i = 1, \ldots, l$$

and form

$$F_{iso} = \left[\, C_z A_z^{\rho_1 - 1} b_{z1} \cdots C_z A_z^{\rho_l - 1} b_{zl} \,\right].$$

Assume that F_{iso} is left invertible and

$$F_{iso}^{-} = \left(F_{iso}^T F_{iso}\right)^{-1} F_{iso}^T.$$

It is known that

$$L = \left[A_z^{\rho_1} b_{z1} \cdots A_z^{\rho_l} b_{zl} \right] F_{iso}^{-}, V = F_{iso}^{-} \tag{10.41}$$

delivers an FDF which ensures a stable fault isolation as

$$r(z) = VC_z \left(zI - A_z + LC_z\right)^{-1} B_z f_A(z) = diag\left(\frac{1}{z^{\rho_1}}, \ldots, \frac{1}{z^{\rho_l}}\right) f_A(z)$$

$$\Longleftrightarrow \begin{bmatrix} r_1(z) \\ \vdots \\ r_l(z) \end{bmatrix} = \begin{bmatrix} \frac{1}{z^{\rho_1}} f_{A,1}(z) \\ \vdots \\ \frac{1}{z^{\rho_l}} f_{A,l}(z) \end{bmatrix}.$$

Algorithm 10.6 *Isolation of actuator faults*

S1: Determine $\rho_i, i = 1, \ldots, l$
S2: Form F_{iso} *and compute* F_{iso}^{-}
S3: Set L, V *according to (10.41).*

Next, we deal with sensor fault isolation. To this end, we propose to apply Algorithm 10.4 to generate a bank of residual signals. Note that each of the resulted m residual generators described by (10.23), (10.24) is driven by a single sensor. This is in fact a data-driven realization of the well-known DOS (dedicated observer scheme) for isolating sensor fault. The dynamics of the residual generator with respect to the sensor fault can be expressed by

$$\Delta x_i(k+1) = A_{z,i} \Delta x_i(k) + l_{z,i} f_{S,i}(k) \tag{10.42}$$

$$r_i(k) = c_{z,i} \Delta x_i(k) + f_{S,i}(k), \Delta x_i(k) = T_i x(k) - z_i(k). \tag{10.43}$$

10.3.3 A Fault Identification Scheme

In this subsection, we present a simple but practical fault identification scheme. Consider residual generator (10.36), (10.37) whose dynamics is governed by (10.38), (10.39). Assume that the fault to be identified is (nearly) constant, that is

$$f(k) = f = const. \Longleftrightarrow f(k+1) - f(k) = 0. \tag{10.44}$$

Now, for our purpose, define

$$\begin{bmatrix} z(k) \\ \hat{f}(k) \end{bmatrix} \in \mathcal{R}^{s+k_f}, k_f = l + m$$

and extend the residual generator to

$$\begin{bmatrix} z(k+1) \\ \hat{f}(k+1) \end{bmatrix} = \begin{bmatrix} A_z & E_f \\ 0 & I \end{bmatrix} \begin{bmatrix} z(k) \\ \hat{f}(k) \end{bmatrix} + \begin{bmatrix} B_z \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} L_z \\ 0 \end{bmatrix} y(k) + \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} r_o(k) \tag{10.45}$$

$$r_o(k) = Gy(k) - \begin{bmatrix} C_z & F_f \end{bmatrix} \begin{bmatrix} z(k) \\ \hat{f}(k) \end{bmatrix} - D_z u(k) \tag{10.46}$$

$$E_f = \begin{bmatrix} B_z & -L_z \end{bmatrix}, F_f = \begin{bmatrix} D_z & G \end{bmatrix} \tag{10.47}$$

which is an augmented observer. It turns out

$$\begin{bmatrix} \Delta x(k+1) \\ \Delta f(k+1) \end{bmatrix} = \begin{bmatrix} A_z - L_1 C_z & E_f - L_1 F_f \\ -L_2 C_z & I - L_2 F_f \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \Delta f(k) \end{bmatrix}, \Delta f(k) = f(k) - \hat{f}(k). \tag{10.48}$$

Thus, if the augmented system $\left(\begin{bmatrix} C_z & F_f \end{bmatrix}, \begin{bmatrix} A_z & E_f \\ 0 & I \end{bmatrix} \right)$ is detectable,

$$L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}$$

can be selected so that

$$\lim_{k \to \infty} \Delta f(k) = 0 \Longrightarrow \hat{f}(k) = f.$$

In this way, a successful fault identification is achieved.

Algorithm 10.7 *Fault identification*

S1: Check the detectability of the system pair $\left(\begin{bmatrix} C_z & F_f \end{bmatrix}, \begin{bmatrix} A_z & E_f \\ 0 & I \end{bmatrix} \right)$
S2: Design L
S3: Construct residual generator (10.45)–(10.46).

10.4 Observer-Based Process Monitoring

A by-product of Algorithm 10.3 is the realization of a state observer of the general form

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) \tag{10.49}$$

which can be applied, for instance, for the process monitoring or control purposes. Since this observer is a deadbeat observer, in case that different dynamics is wished, it can be extended to

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) + Lr(k) \tag{10.50}$$

$$r(k) = gy(k) - c_z z(k) - d_z u(k). \tag{10.51}$$

L is an additional observer gain matrix and selected by considering the eigenvalues of matrix $A_z - Lc_z$.

Algorithm 10.8 *QOCF-based observer design*

S1: Find κ_o using the algorithm given by Mercere and Bako
S2: Re-order the data sets according to (10.13)-(10.14)
S3: Run Algorithm 10.2 for $Q_1^{-1}\Psi_{s,y}^{\perp}$ and determine the system order n
S4: Form A_z, B_z, c_z, d_z, L_z and select L if needed
S5: Construct the state observer (10.50)-(10.51).

The state observer (10.49) can be directly applied as the dynamic core of a soft-sensor for the estimation of some key variable ς

$$\varsigma(k) = C_\varsigma z(k) + D_\varsigma u(k)$$

where C_ς, D_ς are system matrices in the output model and have to be identified using collected data and as a part of the kernel representation.

10.5 Case Study on CSTH

In our case study, Algorithms 10.2–10.5 are implemented for residual generation on the continuous stirred tank heater (CSTH) process introduced in Sect. 2.2. This case study shall demonstrate the construction/realization of observer-based fault diagnosis systems in the data-driven manner.

10.5.1 System Setup

As described in Sect. 2.2, water is continuously fed and conducted as the medium in the laboratory setup CSTH process. The water level h_T and temperature T_T in the stirred tank are two variables that can be controlled respectively. During the experiment, the operating point is selected as $h_T = 15\,\text{cm}$ and $T_T = 30\,^\circ\text{C}$. The system sampling time is 0.01 s. It is noteworthy that the water temperature T_T is under default PID control, and on the other hand, simulated sensor faults are inserted in the water level part for the purpose of fault detection.

10.5.2 Towards the Kalman Filter-Based Residual Generator

In the fault-free case, a Kalman filter based fault diagnosis system generates the residual signal such that

$$r(k) = e(k) \sim \mathcal{N}(0, \Sigma_e).$$

For the sake of simplicity, we concentrate on the part of water level as an application case. Based on Algorithm 10.2, a parity vector can be obtained as

$$\alpha_n = \begin{bmatrix} 0.8635 & -0.8127 & -1.0507 & 1.0000 \end{bmatrix}$$

and correspondingly, the last row of $-Q_1^{-1}\Psi_{s,u}^{\perp}$, that is, the vector $\alpha_{n,u}$, is

$$\alpha_{n,u} = \begin{bmatrix} 0.0013 & -0.0028 & -0.0059 & 0.0078 \end{bmatrix}.$$

It follows from Algorithm 10.5 that

$$\frac{1}{N-1}\Psi_{s,y}^{\perp} R_{33} Q_3 Y_k^T = 0.0295$$

by solving (10.29) and (10.28), we have $\Sigma_e = 0.0145$ and the Kalman gain matrix identified as

$$K = \begin{bmatrix} -1.6611 \\ 0.5835 \\ 0.5468 \end{bmatrix}.$$

As a result, the Kalman filter based residual generator is constructed in the form of

$$\begin{aligned} z(k+1) &= \bar{A}_z z(k) + \bar{B}_z u(k) + TK e(k) \\ r(k) &= e(k) = y(k) - c_z z(k) - d_z u(k) \end{aligned}$$

where

$$\bar{A}_z = \begin{bmatrix} 0 & 0 & -0.8635 \\ 1 & 0 & 0.8127 \\ 0 & 1 & 1.0507 \end{bmatrix}, \bar{B}_z = \begin{bmatrix} -0.0054 \\ 0.0035 \\ 0.0023 \end{bmatrix}, TK = \begin{bmatrix} 2.9350 \\ -0.7378 \\ -0.6851 \end{bmatrix}$$
$$c_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, d_z = 0.0078.$$

It then turns out that the residual signal, in the fault-free case, can be successfully generated as depicted in Fig. 10.1.

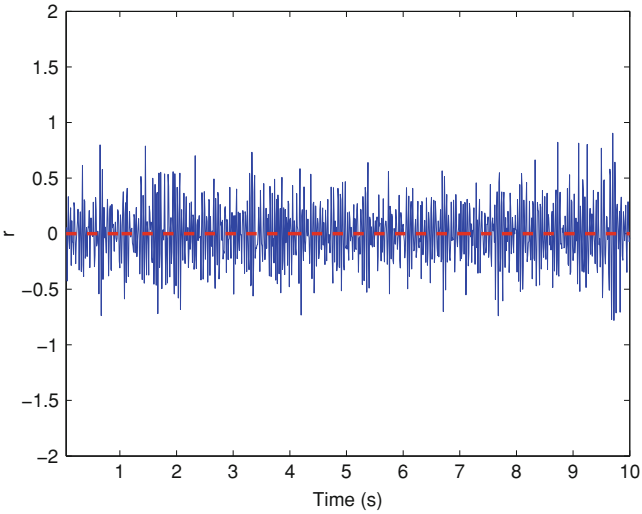


Fig. 10.1 Residual signal in the fault-free case

10.5.3 Towards the Generation of m-Dimensional Residual Vectors

In the following, the extension of Algorithm 10.2 to the multiple case is carried out. In our test study, sensor faults occurred in the part of water level are added.

- *QOCF-based residual generation (Algorithm 10.3)*
  The main idea of this scheme is to design a full order observer based on a single parity vector and then generate an m-dimensional residual vector. To this end, the vector κ_o is first obtained as

$$\kappa_o = \begin{bmatrix} 2.7148 & 1.8605 \end{bmatrix}$$

  with the algorithm given by Mercere and Bako.
  By means of Algorithm 10.3, a parity vector can be identified as

$$\alpha_n = \begin{bmatrix} 0.1278 & -0.9756 & -0.1526 & 1.0000 \end{bmatrix}$$

  and its associated vector

$$\alpha_{n,u} = \begin{bmatrix} \alpha_{n,u,1} & \alpha_{n,u,2} \end{bmatrix}, \alpha_{n,u,1} = \begin{bmatrix} -0.0063 & -0.0066 & -0.0120 & -0.0076 \end{bmatrix}$$
$$\alpha_{n,u,1} = \begin{bmatrix} -0.0078 & 0.0202 & 0.0164 & -0.0030 \end{bmatrix}$$

  which leads to the construction of A_z, B_z, c_z, d_z, L_z as follows

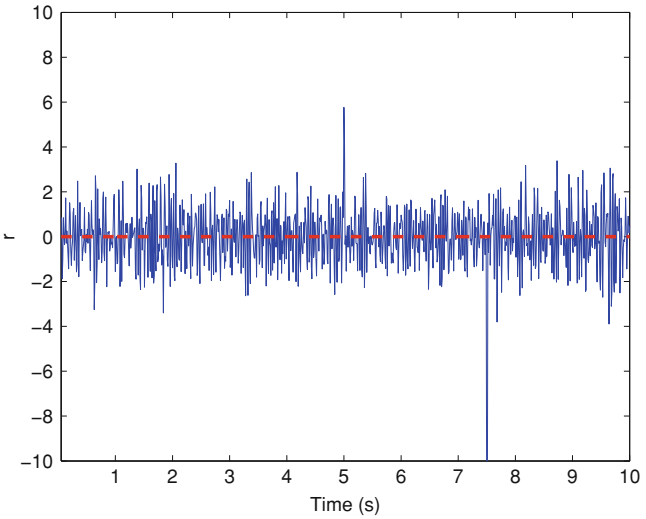


Fig. 10.2 Residual signal (level part) in the fault case

$$A_z = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, B_z = \begin{bmatrix} -0.0063 & -0.0066 \\ -0.0120 & -0.0076 \\ -0.0078 & 0.0202 \end{bmatrix}, L_z = \begin{bmatrix} -0.1278 \\ 0.9756 \\ 0.1526 \end{bmatrix}$$
$$c_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, d_z = \begin{bmatrix} 0.0164 & -0.0030 \end{bmatrix}.$$

Meanwhile, the (regular) state transformation matrix turns out to be

$$T = \begin{bmatrix} 0.2717 & 0.3472 & -0.3775 \\ 0.7215 & 0.4087 & -0.5312 \\ 0.5482 & -0.1200 & -0.1053 \end{bmatrix}$$

and the matrices $\bar{C}_2$ and $\bar{D}_2$ from (10.16), (10.17)

$$\bar{C}_2 = \begin{bmatrix} 0.0378 & 0.0263 & 0.0718 \end{bmatrix}, \bar{D}_2 = \begin{bmatrix} 0.0005 & -0.0018 \end{bmatrix}$$

then an m-dimensional residual generator can be built according to (10.19), (10.20), and as a result, the sensor fault is successfully detected as shown in Figs. 10.2 and 10.3.

- *Construction of m residual generators (Algorithm 10.4)*
  The second scheme here involves the construction of two residual generators for water level h_T and water temperature T_T, respectively. Each of them is driven by a single sensor signal and the process input variables. More specifically, the procedures of Algorithms 10.2 and 10.4 lead to

  – residual generator for the part of water level h_T

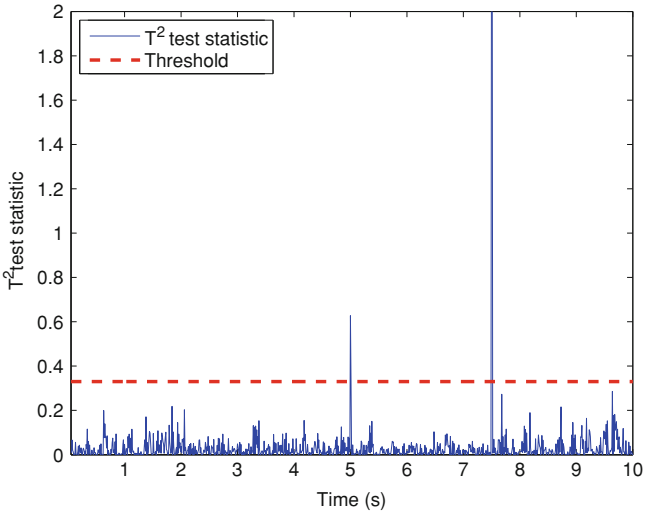


Fig. 10.3 Residual evaluation for fault detection

$$z_1(k+1) = A_{z,1}z_1(k) + B_{z,1}u(k) + l_{z,1}y_1(k)$$
$$r_1(k) = y_1(k) - c_{z,1}z_1(k) - d_{z,1}u(k)$$

where

$$A_{z,1} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, B_{z,1} = \begin{bmatrix} -0.0017 & -0.0032 \\ -0.0042 & -0.0019 \\ -0.0024 & 0.0072 \end{bmatrix}, l_{z,1} = \begin{bmatrix} -0.1950 \\ 0.8252 \\ 0.3722 \end{bmatrix}$$
$$c_{z,1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, d_{z,1} = \begin{bmatrix} 0.0053 & 0.0007 \end{bmatrix}.$$

Recall that sensor faults have been inserted in the level part. This is indicated by the residual signal and also by the estimated states, as shown in Figs. 10.4, 10.5. In other words, the fault has been successfully detected.

– Residual generator for the part of water temperature T_T

$$z_2(k+1) = A_{z,2}z_2(k) + B_{z,2}u(k) + l_{z,2}y_2(k)$$
$$r_2(k) = y_2(k) - c_{z,2}z_2(k) - d_{z,2}u(k)$$

where

$$A_{z,2} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, B_{z,2} = \begin{bmatrix} -0.0023 & -0.0038 \\ 0.0012 & 0.0043 \\ 0.0006 & -0.0053 \end{bmatrix}, l_{z,2} = \begin{bmatrix} 1.4268 \\ -2.1849 \\ 1.7598 \end{bmatrix}$$
$$c_{z,2} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, d_{z,2} = \begin{bmatrix} -0.0008 & 0.0032 \end{bmatrix}.$$

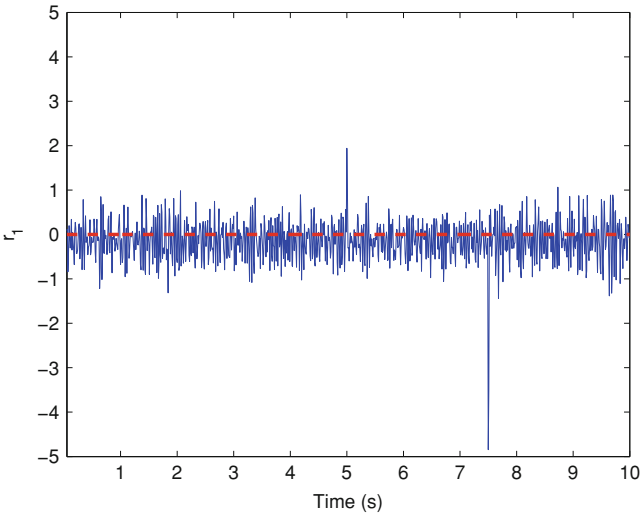


Fig. 10.4 Residual signal (level part)

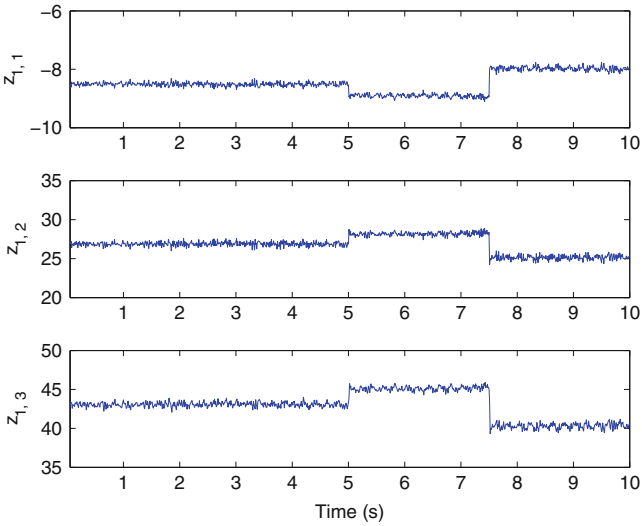


Fig. 10.5 State estimation (level part)

Compared with the level part, the water temperature is under fault-free conditions, and Figs. 10.6, 10.7 present the results of residual generation and state estimation. In this way, a successful fault isolation is also demonstrated.

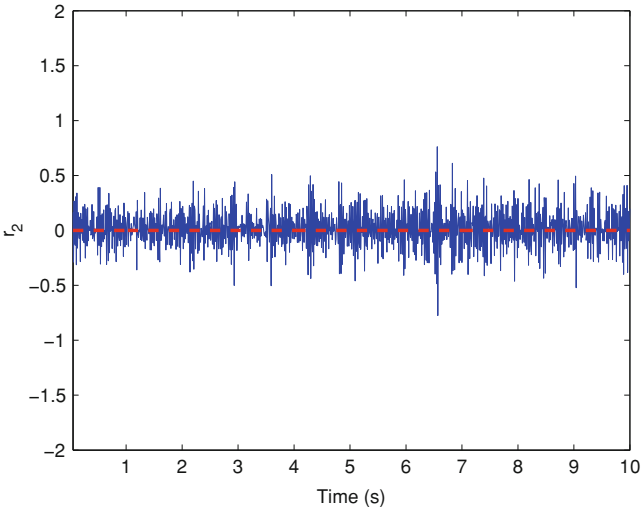


Fig. 10.6 Residual signal (temperature part) in the fault-free case

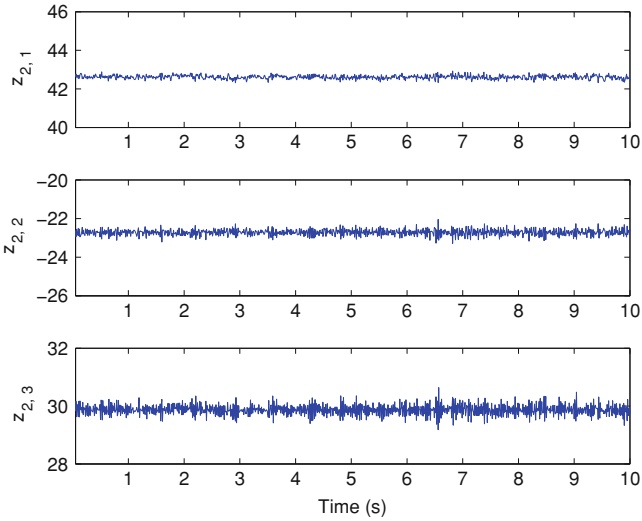


Fig. 10.7 State estimation (temperature part) in the fault-free case

10.6 Case Study on TEP

In this section, we briefly present some representative results of our benchmark study on TEP. The data-driven design of data-driven observer-based fault detection is illustrated by an example of the input feed block. The block uses 9 manipulated variables as inputs (the others are zeros for the current simulation setting) and feed

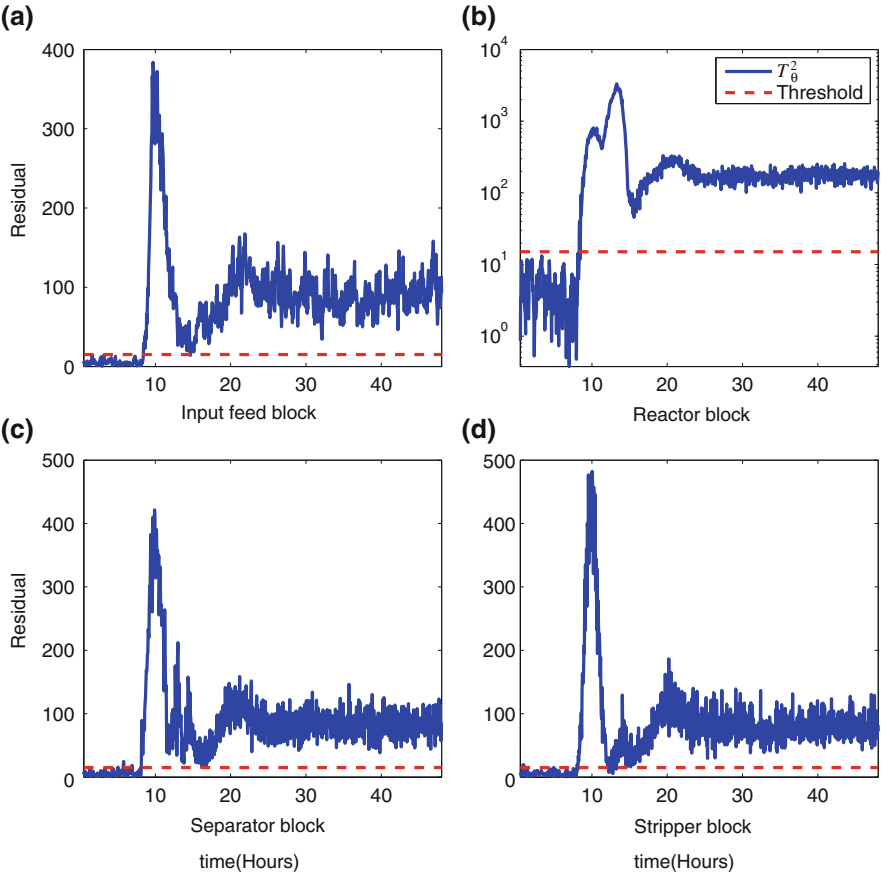


Fig. 10.8 Detection results in case of IDV1

flows of A, B, D and A/C as outputs. The length of parity vector is set to be 5. The fault-free covariance matrix of the residual signals is directly identified when calculating the parameters of residual generator via the QR decomposition. We have adopted the well-known T^2 test statistic for residual evaluation. The threshold is selected by choosing the false alarm rate of 1 % with the χ^2-distribution.

In our study, two representative faults have been selected. The first fault is a step type change in the feed ratio of A/C components (IDV1), which occurs after 8 h of normal operation. This fault affects material balance equations of the plant leaving almost half of the monitored variables to change through various control loops and recycle streams. Figure 10.8 shows the observer-based detection results for the input feed, reactor, separator and stripper blocks. As expected, we can see that the residual is below threshold before fault occurs. After 8 h the residual goes above the threshold, thus indicating the presence of a fault.

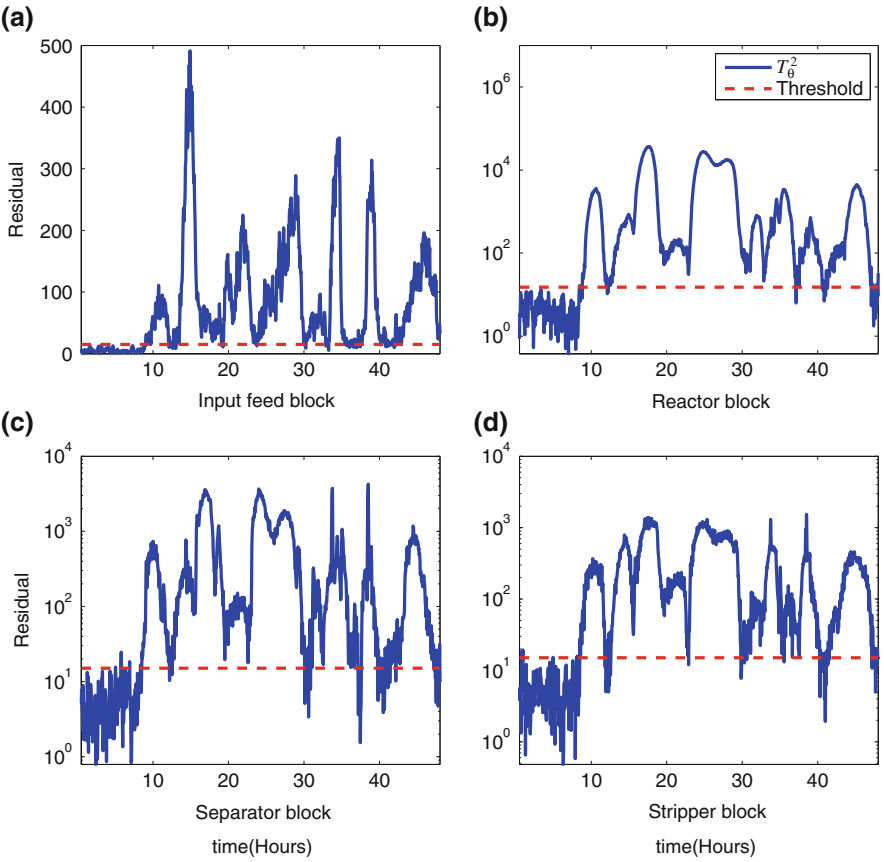


Fig. 10.9 Detection results in case of IDV13

Different from the first one, the second fault is a random variation in the reaction kinetics (IDV13). It belongs to the process fault and generally takes very complex form and all the four process variables are influenced by this fault after different operating time. Figure 10.9 shows the monitoring results for the fault mode IDV13. The test statistic crosses the threshold after the occurrence of fault. Except for the reactor block, the other three blocks have detected the fault as well. This is because of the feedback control actions during the faulty operation, which is explicitly implemented in the used simulator. The fault has propagated to other blocks among the whole process, which is always the case in practice.

10.7 Remarks on the Application of the Data-Driven FDI Systems

In comparison with the development and applications of the SIM technology, the data-driven process monitoring and diagnosis technique introduced in this and last chapters is still in its early development. It is therefore reasonable that there are limited experiences with its applications in real industrial processes. This fact motivates us to make remarks on some key issues by the application of these data-driven schemes.

It is the state of the art in the process industry that advanced process monitoring and diagnosis modules/functionalities are often installed or in operation and optimized after automatic control systems work successfully. In fact, it is the major task of a monitoring and diagnosis system to supervise the process operation and the performance of the automatic control system. A logic consequence is that the process data used for the data-driven design of the diagnosis system have been collected during the process operation. That means, these data are generally the so-called "closed-loop" data. Consequently, the I/O data sets used for training (identification) are the data sets of the process references and measurement signals.

It is well-known in the SIM framework that a successful system identification depends strongly on the quality of the available excitations and correlations between the process and measurement noises and process input and output variables. For the controller design based on an SIM-based model, these issues are of a special importance. Differently, in the process monitoring and diagnosis framework, the identification procedure serves residual generation, which is used to recover the process I/O relation in the normal (fault-free) process operation. Therefore, similar to the PCA/PLS approaches, a successful residual generation can be achieved as far as the collected process data do cover the major process operation scenarios. In fact, it is well known that the order of a residual generator whose core is an output observer or a parity vector can be (much) lower than the system order. On the other hand, it should be kept in mind that technical processes are mainly nonlinear. It is recommended to design a bank of residual generators corresponding to different process operation points.

It is clear that due to the embedded control loops biases in the estimations are inevitable. Different from the application of an SIM model for the controller design, in the process monitoring and diagnosis framework biases in the I/O model and thus in the residual signals are addressed in terms of threshold setting. Unfortunately, in the theoretical study less attention has been devoted to this issue. It is well-known that the theoretical setting based on advanced control theory may lead to (too) conservative thresholds which result in poor fault detectability. It is the state of the art in real applications that optimal threshold settings are often achieved on the basis of great number of tests in the real application environment. In this context, the threshold setting methods introduced in this and last chapters can be interpreted as a systematic realization of the test-based threshold setting, which provides, as mentioned, a more reasonable solution than the theoretical methods.

The core of the FDI schemes presented in this chapter is residual generation. In principle, this idea can also be adopted and applied to detect changes in process control performance. On the other hand, in real applications it is often more effective to construct a soft-sensor for prediction and detection of well-defined process key performance variables. The methods presented in this chapter can be applied as a design tool for this purpose.

Finally, we would like to remind the difficulties with collecting data from faulty process operations. Data-driven fault isolation and identification is a real challenge and demands for more research activities.

10.8 Notes and References

In this chapter, we have addressed recursive and closed-loop configurated realization of the data-driven kernel representation. We would like to emphasize the needs for this work in order to achieve high real-time ability and robustness, and to link the model-based FDI methods to the data-driven framework.

The control theoretical basis for our study is the relationships between the kernel representation, parity vector/matrix and the diagnostic observer realization, as presented in Sect. 10.2.1. They are described in details in Chap. 5 in [1].

The data-driven design scheme introduced in Sect. 10.2.1 has been proposed in [2], in which also Algorithm 10.1 has been proven.

Theorem 10.1 is a result given in [3], which plays a central role in constructing an m-dimensional residual vector using Algorithm 10.3. The resulted residual generator (10.36), (10.37) is a standard form of the FDF, whose dynamics given in (10.38), (10.39) allows the application of existing model-based fault detection, isolation and identification schemes. The reader is referred to [1] and references therein.

Algorithm 10.6 for the actuator fault isolation is a special case of Theorem 13.4 in [1], which can also be applied, for instance, to the cases with both sensor and actuator faults. Moreover, further algorithms are also available, as described in [1] and in [2, 4, 5]. The solution using a bank of residual generators, the DOS, was proposed by Clark [6] and has been widely accepted as a standard sensor fault isolation method.

Fault identification is a complex and difficult issue, both in applications and research. Algorithm 10.7 is a simple solution that is widely accepted in practice. In fact, the application of the augmented observer schemes to fault identification has received increasing attention in recent years. As briefly introduced in this chapter, the underlying idea of the augmented observer schemes lies in addressing the faults as additional state variables, which are then reconstructed by an augmented observer. The well-known PI-observer is a special kind of such observers [7, 8]. Often, such observers/estimators are designed based on certain assumption on the faults, for instance the boundedness on the derivative. We refer the reader to [9–13] for some recent publications on this topic.

It is worth to mention that the benchmark study on TEP has been reported in [2, 14] in details. Yin et al. [14] has compared the data-driven observer-based FDI

schemes with some standard MVA methods on the TEP benchmark and demonstrated the power of the observer-based FDI schemes in dealing with dynamic processes. In [15], a successful application of an observer-based soft-sensor scheme to an industrial hot strip mill has been reported. In this work, the finishing mill exit strip thickness is estimated/predicted using a soft-sensor and abnormal changes are successfully detected.

Comparing with the well-established MVA and model-based process monitoring and fault diagnosis techniques, the data-driven technique presented in this and last chapters is in the early stage of its development. Great efforts have to be made to establish a framework and to significantly improve its acceptance in industry. To this end, studies on the following issues would be helpful:

- enhancing the system robustness against (deterministic) disturbances: Different from the well-established robust control theory [16] and model-based FDI technique [1], the presented data-driven methods have been developed in the framework of stochastic systems. In the industrial environment, disturbances (unknown inputs) are unavoidable. Consequently, methods are needed for the data-driven design of process monitoring and fault diagnosis systems with high robustness against disturbances.
- process monitoring and fault diagnosis methods for nonlinear and non-Gaussian processes: LPV (linear parameter varying) technique is an efficient tool for dealing with nonlinear issues and widely accepted in industry. Pioneering work in this area has been reported in [17]. The successful methods based on e.g. particle filter technique [18], the multiway discrete hidden Markov model-based approach [19] and machine learning technique [20] also demonstrate promising results for addressing nonlinear and non-Gaussian problems.
- fault isolation and identification issues: As mentioned, data-driven design of FDI systems with incomplete data in faulty operation modes is a challenging topic. In [21], a promising approach has been proposed.
- integration of data-driven and model-based methods: one of the major advantages of the data-driven observer-based methods consists in the established framework, in which advanced model-based methods can be implemented based on I/O data models or the data-driven realization of the kernel representation.

References

1. Ding SX (2013) Model-based fault diagnosis techniques: design schemes, algorithms and tools, 2nd edn. Springer, London
2. Ding SX, Zhang P, Naik A, Ding E, Huang B (2009) Subspace method aided data-driven design of fault detection and isolation systems. J Process Control 19:1496–1510
3. Mercere G, Bako L (2011) Parameterization and identification of multivariable state-space systems: a canonical approach. Automatica 47:1547–1555
4. Wang Y, Ma G, Ding SX, Li C (2011) Subspace aided data-driven design of robust fault detection and isolation systems. Automatica 47:2474–2480

5. Wan Y, Ye H (2012) Data-driven diagnosis of sensor precision degradation in the presence of control. J Process Control 22:26–40
6. Clark RN (1978) Instrument fault detection. IEEE Trans Aerosp Electron Syst 14:456–465
7. Busawon K, Kabore P (2001) Disturbance attenuation using proportional integral observers. Int J Control 74:618–627
8. Saif M (1993) Reduced-order proportional integral observer with application. J Guidance Control Dyn 16:985–988
9. Gao Z, Ho D (2004) Proportional multiple-integral observer design for descriptor systems with measurement output disturbances. IEE Proc Control Theory Appl 151(3):279–288
10. Gao Z, Ho DWC (2006) State/noise estimator for descriptor systems with application to sensor fault diagnosis. IEEE Trans Signal Process 54:1316–1326
11. Ha QP, Trinh H (2004) State and input simultaneous estimation for a class of nonlinear systems. Automatica 40:1779–1785
12. Gao Z, Ding SX (2007) Actuator fault robust estimation and fault-tolerant control for a class of nonlinear descriptor systems. Automatica 43:912–920
13. Gao Z, Shi X, Ding S (2008) Fuzzy state/disturbance observer design for T-S fuzzy systems with application to sensor fault estimation. IEEE Trans Syst Man Cybern B Cybern 38:875–880
14. Yin S, Ding SX, Haghani A, Hao H, Zhang P (2012) A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. J Process Control 22:1567–1581
15. Ding SX, Yin S, Peng K, Hao H, Shen B (2013) A novel scheme for key performance indicator prediction and diagnosis with application to an industrial hot strip mill. IEEE Trans Ind Inform 9:2239–2247
16. Zhou K, Doyle J, Glover K (1996) Robust and optimal control. Prentice-Hall, Upper Saddle River
17. Dong J (2009) Data driven fault tolerant control: a subspace approach. Ph.D. dissertation, Technische Universiteit Delft
18. Yu J (2012) A particle filter driven dynamic gaussian mixture model approach for complex process monitoring and fault diagnosis. J Process Control 22:778–788
19. Yu J (2012) Multiway discrete hidden markov model-based approach for dynamic batch process monitoring and fault classification. AIChE J 58:2714–2725
20. Rashid MM, Yu J (2012) A new dissimilarity method integrating multidimensional mutual information and independent component analysis for non-aussian dynamic process monitoring. Chemometr Intell Lab Syst 115:44–58
21. Yu J (2013) A support vector clustering-based probabilistic method for unsupervised fault detection and classification of complex chemical processes using unlabeled data. AIChE J 59:407–419

Part IV
Adaptive and Iterative Optimization Techniques for Data-driven Fault Diagnosis

Chapter 11
Adaptive Fault Diagnosis Schemes

The key of data-driven fault diagnosis schemes consists in a direct identification of key fault detection statistics like covariance matrices applied in the MVA or SIM based methods. An online update of these statistics during process operation is of considerable practical interests for two good reasons:

- the process operation conditions and the environment around the process under monitoring may change with time
- the process data collected during process operation should be archived online and compressed.

Adaptation of covariance matrices by means of online data serves for this purpose. Among the existing adaptive schemes, the recursive technique has been well developed and widely applied to the standard MVA methods like PCA, PLS. Since the pioneering work by Helland et al. and Qin, numerous recursive schemes have been reported. The recent research focus is mainly on the recursive computation of SVD, which is used both in the PCA and PLS.

In this chapter, we shall study three adaptive schemes. The first two approaches deal with the adaptation of covariance matrices, while the third one leads to an adaptive SKR. We assume that the process under monitoring and its statistic properties vary with time slowly.

11.1 OI-based Recursive SVD Computation and Its Application

In this section, we first introduce the so-called Orthogonal Iteration (OI) based methods for a recursive computation of SVD, and then study their application to fault detection.

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_11, © Springer-Verlag London 2014

11.1.1 Problem Formulation

Recall that SVD of an observation covariance matrix is a key step in many MVA methods like PCA, PLS and CCA. The first problem to be addressed in this section is formulated as follows: given a covariance matrix $\Sigma_z \geq 0$ of the observation vector $z \in \mathcal{R}^n$ with

$$\Sigma_z = Udiag(\sigma_1, \ldots \sigma_l, \sigma_{l+1}, \ldots \sigma_n)U^T \in \mathcal{R}^{n\times n} \tag{11.1}$$

$$\sigma_1 \geq \cdots \geq \sigma_l > \sigma_{l+1} \geq \cdots \geq \sigma_n, UU^T = I, U = \left[u_1 \ \ldots \ u_n \right] \tag{11.2}$$

where $\sigma_i, u_i, i = 1, \ldots, n$, denote the singular values and vectors respectively, we are interested in an update of the singular vectors $u_i, i = 1, \ldots, l$, by a (new) observation.

11.1.2 DPM: An Adaptive Algorithm

The OI is an iterative procedure that can be applied to compute those singular vectors corresponding to the first largest singular values of a symmetric and positive definite or semi-definite matrix. Below, some standard results and algorithms are introduced, for which the reader is referred to the references given in the end of this chapter.

The basic idea and result of the OI is well described in the following lemma.

Lemma 11.1 *Given $\Sigma_z \in \mathcal{R}^{n\times n}$ and $\Sigma_z \geq 0$ with singular values $\sigma_1 \geq \cdots \geq \sigma_l > \sigma_{l+1} \geq \cdots \geq \sigma_n$ and corresponding singular vectors $u_i, i = 1, \ldots, n$, define the iteration sequence*

$$U_k = orthnorm\left\{\Sigma_z U_{k-1}\right\}, k = 1, 2, \ldots,$$

where $U_k \in \mathcal{R}^{n\times l}$ and orthnorm stands for orthonormalization using QR-decomposition or the modified Gram–Schmidt procedure, then

$$\lim_{k\to\infty} U_k = \left[u_1 \ \ldots \ u_l \right] \tag{11.3}$$

provided that matrix $U_0^T \left[u_1 \ \ldots \ u_l \right]$ is not singular.

Now, replacing Σ_z by $\left(I + \mu z_k z_k^T\right)$ yields

$$U_k = orthnorm\left\{\left(I + \mu z_k z_k^T\right) U_{k-1}\right\}, k = 1, 2, \ldots, \tag{11.4}$$

which allows us to track $\left[u_1 \ \ldots \ u_l \right]$. In (11.4), z_k denotes the new data and μ can be, for instance, determined by

$$\mu = \frac{1}{\sigma_z^2(k)}, \sigma_z^2(k) = \eta\sigma_z^2(k-1) + (1-\eta)z_k^T z_k \tag{11.5}$$

with $0 < \eta < 1$ as an exponential forgetting factor.

Recall that, for instance, in the PCA study, we are also interested in the residual subspace, that is, the subspace spanned by the singular vectors corresponding to the smallest singular values. It is proven that using

$$U_k = orthnorm\left\{\left(I - \mu z_k z_k^T\right)U_{k-1}\right\}, k = 1, 2, \ldots, \tag{11.6}$$

we are able to track

$$\left[u_{n-l+1} \ldots u_n\right].$$

The following algorithm can be applied to tracking

$$\left[u_1 \ldots u_l\right] \text{ and } \left[u_{n-l+1} \ldots u_n\right].$$

It is called data projection method (DPM).

Algorithm 11.1 *DPM: adaptive update of covariance matrix*

S1: Set initial conditions: $U_0, \sigma_z^2(0)$

S2: Compute μ *and*

$$r_k = U_{k-1}^T z_k$$

S3: Compute

$$T_{k,+} = U_{k-1} + \mu z_k r_k^T, T_{k,-} = U_{k-1} - \mu z_k r_k^T$$

S4: Compute

$$U_{k,+} = orthnorm\left\{T_{k,+}\right\}, U_{k,-} = orthnorm\left\{T_{k,-}\right\}.$$

As a result, $U_{k,+}$ tracks $\left[u_1 \ldots u_l\right]$, while $U_{k,-}$ approaches $\left[u_{n-l+1} \ldots u_n\right]$.

Due to its computational complexity, different variants of DPM have been proposed in recent years. The reader is referred to the literatures for more details.

11.1.3 Applications to Fault Detection

An immediate application of the DPM and Algorithm 11.1 to fault detection is the realization of a recursive SVD in the PCA method. In that case, both the principal component and residual sub-spaces can be recursively computed to fit the changes in the process. The following algorithm is proposed for this purpose.

Algorithm 11.2 *Recursive PCA*

S1: Run Algorithm 5.2 with P_{pc}, P_{res} as output
S2: (Online) run Algorithm 11.1 with the initial setting

$$U_{0,+} = P_{pc} \in \mathcal{R}^{m\times l}$$

S3: (Online) run Algorithm 11.1 with the initial setting

$$U_{0,-} = P_{res} \in \mathcal{R}^{m\times(m-l)}.$$

As a result, $U_{k,+}$ tracks the principal subspace P_{pc} and $U_{k,-}$ the residual subspace P_{res}.

11.2 An Adaptive SVD Algorithm and Its Applications

The DPM introduced in the last section leads to an adaptive computation of a group of singular vectors of a symmetric covariance matrix. In the data-driven fault detection study presented in the previous chapters, we often deal with an SVD of $\mathcal{E}\left(yx^T\right)$ with x, y representing two random vectors. As an example, remember the fault detection scheme III presented in Sect. 9.2.3 whose core is an SVD of a covariance matrix formed by the process data sets.

11.2.1 The Adaptive SVD Algorithm

Let y_k, x_k be two random vectors. Assume that

$$\mathcal{E}\left(y_k x_k^T\right) = H_k = U_k \Sigma_k V_k^T \in \mathcal{R}^{m\times n}$$

changes slowly with time. Here, $U_k \Sigma_k V_k$ denotes the SVD of H_k with

$$\Sigma_k = \begin{bmatrix} diag\left(\sigma_{1,k}, \ldots, \sigma_{l,k}\right) & 0 \\ 0 & 0 \end{bmatrix}, U_k = \left[\, u_{1,k} \ \ldots \ u_{m,k} \,\right], U_k U_k^T = I_{m\times m}$$
$$V_k = \left[\, v_{1,k} \ \ldots \ v_{n,k} \,\right], V_k^T V_k = I_{n\times n}.$$

In order to track the change in H_k, the following recursive update of H_k is widely adopted

$$H_k = \alpha H_{k-1} + (1-\alpha)\, y_k x_k^T, 0 < \alpha \leq 1. \tag{11.7}$$

Rewrite the above equation into

$$H_k = H_{k-1} + E, E = (1-\alpha)\left(y_k x_k^T - H_{k-1}\right).$$

On the assumption that E is small and $U_{k-1}, \Sigma_{k-1}, V_{k-1}$ (or their estimates) are available with

$$H_{k-1} = U_{k-1}\Sigma_{k-1}V_{k-1}^T$$

we would like to find estimates for U_k, Σ_k, V_k based on $U_{k-1}, \Sigma_{k-1}, V_{k-1}$. Note that H_k can be written as

$$H_k = U_{k-1}\left(\Sigma_{k-1} + F\right)V_{k-1}^T, F = U_{k-1}^T E V_{k-1} \tag{11.8}$$

which motivates the utilization of the small perturbation theory for determining U_k, Σ_k, V_k recursively. The results are summarized in the following lemma.

Lemma 11.2 *Consider* (11.8) *and assume that the singular values of H_k are simple. Then, an SVD of*

$$H_k = U_k \Sigma_k V_k^T$$

can be computed as

$$U_k = U_{k-1}(I + A), V_k = V_{k-1}(I + B)$$

where the (i, j)-th elements of A and B are given by

$$\begin{aligned}
a_{ii} &= 0, b_{ii} = 0 \\
for\ j > i, a_{ji} &= \frac{\sigma_{i,k-1} f_{ji} + \sigma_{j,k-1} f_{ij}}{\sigma_{i,k-1}^2 - \sigma_{j,k-1}^2} + O\left(\|E\|^2\right) \\
b_{ji} &= \frac{\sigma_{i,k-1} f_{ij} + \sigma_{j,k-1} f_{ji}}{\sigma_{i,k-1}^2 - \sigma_{j,k-1}^2} + O\left(\|E\|^2\right) \\
for\ j < i, a_{ji} &= -a_{ij}, b_{ji} = -b_{ij}.
\end{aligned}$$

with

$$f_{ij} = (1-\alpha)\, u_{i,k-1}^T y_k x_k^T v_{j,k-1}$$

denoting the (i, j)-th element of matrix F. Moreover, it holds

$$\begin{aligned}
\sigma_{i,k} &= \sigma_{i,k-1} + f_{ii} + O\left(\|E\|^2\right) \\
f_{ii} &= (1-\alpha)\left(u_{i,k-1}^T y_k x_k^T v_{i,k-1} - \sigma_{i,k-1}\right).
\end{aligned}$$

In the above equations, $O\left(\|E\|^2\right)$ is used to express a first-order approximation.

Based on these results, Willink has proposed the following algorithm for an adaptive (recursive) computation of the SVD.

Algorithm 11.3 *Adaptive computation of an SVD*

S1: Update of H_k

$$H_k = \alpha H_{k-1} + (1-\alpha)\, y_k x_k^T, \, 0 < \alpha \leq 1$$

S2: Update of singular vectors

$$u_{i,k} = u_{i,k-1} + \sum_{j=1, j\neq i}^{m} a_{ji} u_{j,k-1}, \, a_{ji} = \frac{(1-\alpha)\, u_{j,k-1}^T y_k x_k^T v_{i,k-1}}{\sigma_{i,k-1}}$$

$$v_{i,k} = v_{i,k-1} + \sum_{j=1, j\neq i}^{n} b_{ji} v_{j,k-1}, \, b_{ji} = \frac{(1-\alpha)\, v_{j,k-1}^T x_k y_k^T u_{i,k-1}}{\sigma_{i,k-1}}$$

S3: Update of singular values

$$\sigma_{i,k} = \alpha \sigma_{i,k-1} + (1-\alpha)\, u_{i,k-1}^T y_k x_k^T v_{i,k-1}.$$

11.2.2 Applications to Fault Detection

It is remarkable that Algorithm 11.3 delivers an update of all (or selected) singular vectors and singular values of a matrix by new data. It can thus be used for achieving an adaptive or a recursive realization of all those fault detection schemes, whose core consists of an SVD. For instance, such an application leads to

- adaptive GLR or T^2 test statistic based fault detection, where the inverse of the covariance matrix is realized by means of the SVD of this matrix
- adaptive PCA
- adaptive parity space based fault detection, the fault detection scheme III introduced in Sect. 9.2.3.

11.3 Adaptive SKR Based Residual Generation Method

It is worth to mention that in the existing recursive and adaptive MVA methods no attention has been paid to the convergence or stability issues. In particular, the latter plays a central role when a dynamic process is under consideration and a diagnostic

observer is applied. The well-established adaptive control and observer theory offers an alternative and powerful solution. In this section, we present an approach to the design and implementation of an adaptive residual generator. Different from the methods studied in the previous sections, the basic idea of this approach is the application of the well-established adaptive control methods to the observer-based residual generator.

11.3.1 Problem Formulation

For the simplicity of our discussion, at first we only consider the residual generator (10.4), (10.5), which is designed using Algorithm 10.2 and expressed by

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) \in \mathcal{R}^s,\ B_z = TB - L_z D \tag{11.9}$$

$$r(k) = gy(k) - c_z z(k) - d_z u(k) \in \mathcal{R},\ d_z = gD. \tag{11.10}$$

Recall that by means of Algorithm 10.2 all system matrices of the above residual generator are determined by a row of the kernel representation $\Psi_s^{\perp}$. Since any change in the process under consideration can be represented by the changes in its kernel representation, adaptation of the system matrices B_z, L_z, g, d_z in (11.9), (11.10) will deliver an adaptive residual signal. It is further to remark that for $s = n$ the number of the parameters in the data-driven realization of the kernel representation is

$$q = m(n+1) + nl$$

which can be significantly smaller than the parameter number of the original process.

In the sequel, we assume that there exists no structural change, that is, changes in the observability, and the changes in parameters are slow and can be considered nearly constant in a (large) time interval. For our purpose, we now extend the residual generator (11.9), (11.10) to

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) + L_0 r(k) \tag{11.11}$$

$$r(k) = gy(k) - c_z z(k) - d_z u(k) \tag{11.12}$$

where L_0 provides additional degree of design freedom and should ensure the stability of $\bar{A}_z = A_z - L_0 c_z$ (that is, it is a Schur matrix). Let

$$\theta = \begin{bmatrix} \theta_u \\ \theta_y \end{bmatrix} \in \mathcal{R}^{(s+1)(m+l)},\ \theta_u = col\begin{bmatrix} B_z \\ d_z \end{bmatrix},\ \theta_y = col\begin{bmatrix} L_z \\ g \end{bmatrix}$$

$$Q\left(u(k), y(k)\right) = \begin{bmatrix} \mathcal{U}(k) & -L_0 u^T(k) & \mathcal{Y}(k) & L_0 y^T(k) \end{bmatrix}$$

$$\mathcal{U}(k) = \begin{bmatrix} u_1(k) \times I_{s\times s} & \ldots & u_l(k) \times I_{s\times s} \end{bmatrix}$$

$$\mathcal{Y}(k) = \begin{bmatrix} y_1(k) \times I_{s\times s} & \ldots & y_m(k) \times I_{s\times s} \end{bmatrix}$$

Eqs. (11.11), (11.12) can be rewritten into

$$z(k+1) = \bar{A}_z z(k) + Q\left(u(k), y(k)\right)\theta \tag{11.13}$$

$$r(k) = \left[\, -u^T(k)\ y^T(k) \,\right] \bar{I}\theta - c_z z(k) \tag{11.14}$$

$$\bar{I} = \begin{bmatrix} 0 \ldots & I & \ldots 0 \ldots 0 \\ 0 & \ldots\ 0 & 0 \ldots I \end{bmatrix}$$

$$\Longrightarrow \left[\, -u^T(k)\ y^T(k) \,\right] \bar{I}\theta = \left[\, 0 \ldots -u^T(k)\ 0 \ldots y^T(k) \,\right]\theta = \left[\, -u^T(k)\ y^T(k) \,\right] \begin{bmatrix} d_z^T \\ g^T \end{bmatrix}.$$

It is remarkable that the parameter vector θ is identical with the data-driven realization of an SKR, as identified by Algorithm 9.3. The task consists in designing a residual generator which is adaptive to θ and delivers a residual signal $r(k)$ satisfying $\lim_{k\to\infty} r(k) = 0$ and, if possible, with an exponential converging speed independent of a constant change in θ.

11.3.2 The Adaptive Residual Generation Algorithm

The adaptive residual generator scheme given below is inspired by the known adaptive observer schemes given in the literature listed in the end of this chapter. It consists of three subsystems:
Residual generator

$$\hat{z}(k+1) = \bar{A}_z \hat{z}(k) + Q\left(u(k), y(k)\right)\hat{\theta}(k) + V(k+1)\left(\hat{\theta}(k+1) - \hat{\theta}(k)\right) \tag{11.15}$$

$$\begin{aligned} r(k) &= \left[\, -u^T(k)\ y^T(k) \,\right] \bar{I}\hat{\theta}(k) - c_z \hat{z}(k) \\ &= \left[\, -u^T(k)\ y^T(k) \,\right] \begin{bmatrix} \hat{d}_z^T(k) \\ \hat{g}^T(k) \end{bmatrix} - c_z \hat{z}(k). \end{aligned} \tag{11.16}$$

Auxiliary filter

$$V(k+1) = \bar{A}_z V(k) + Q(u(k), y(k)) \tag{11.17}$$

$$\begin{aligned} \varphi(k) &= c_z V(k) - \left[\, -u^T(k)\ y^T(k) \,\right] \bar{I} \\ &= c_z V(k) - \left[\, 0 \ldots -u^T(k)\ 0 \ldots y^T(k) \,\right]. \end{aligned} \tag{11.18}$$

Parameter estimator

$$\hat{\theta}(k+1) = \gamma(k)\varphi^T(k) r(k) + \hat{\theta}(k), \gamma(k) = \frac{\mu}{\delta + \varphi(k)\varphi^T(k)}, \delta \geq 0, 0 < \mu < 2. \tag{11.19}$$

The online implementation of the above adaptive residual generator is summarized in the following algorithm.

Algorithm 11.4 *Adaptive residual generation*

S0: Set the initial values $k = 0, \hat{z}(0), \hat{\theta}(0), V(0) = 0, \varphi(0) = 0$ *and*

$$r(0) = \hat{g}(0)y(0) - c_z\hat{z}(0) - \hat{d}_z(0)u(0)$$

S1: Compute $V(k+1)$ *according to (11.17)*
S2: Compute $\hat{\theta}(k+1)$ *according to (11.19)*
S3: Compute $\hat{z}(k+1)$ *according to (11.15)*
S4: Increase k *by one, receive* $y(k), u(k)$
S3: Compute $r(k), \varphi(k)$ *according to (11.16) and (11.18).*

11.3.3 Stability and Exponential Convergence

The major advantage of applying the above adaptive technique is that the convergency of the parameter estimation and the whole system stability are guaranteed. We now prove the stability of the adaptive residual generator proposed above and check the condition under which the adaptive residual generator is exponentially stable. For our purpose, we follow the same proof procedure known in the literature for the adaptive observer design, and apply the known results on the gradient algorithms, which are given in the following two lemmas.

Lemma 11.3 *Given*

$$y(k) = \varphi(k)\theta$$

and let

$$\hat{\theta}(k+1) = \frac{\mu}{\delta + \varphi(k)\varphi^T(k)}\varphi^T(k)e(k) + \hat{\theta}(k)$$
$$e(k) = y(k) - \varphi(k)\hat{\theta}(k), \delta \geq 0, 0 < \mu < 2.$$

It then follows

$$\lim_{k\to\infty} \frac{e(k)}{\sqrt{\delta + \varphi(k)\varphi^T(k)}} = 0. \tag{11.20}$$

Lemma 11.4 *The difference equation*

$$\tilde{\theta}(k+1) = \left(I - \frac{\mu\varphi^T(k)\varphi(k)}{\delta + \varphi(k)\varphi^T(k)}\right)\tilde{\theta}(k)$$

is globally exponentially stable if there exist positive constants β_1, β_2, *and integer* Π *such that for all* k

$$0 < \beta_1 I \leq \sum_{i=k}^{k+\Pi-1} \varphi^T(i)\varphi(i) \leq \beta_2 I < \infty.$$

To begin with the stability proof, we first define

$$\begin{aligned}\eta(k) &= \tilde{z}(k) - V(k)\tilde{\theta}(k)\\ \tilde{z}(k) &= z(k) - \hat{z}(k), \tilde{\theta}(k) = \theta - \hat{\theta}(k).\end{aligned}$$

Notice that $r(k)$ can be rewritten into

$$\begin{aligned}r(k) &= gy(k) - c_z\hat{z}(k) - d_z u(k) - \left(g - \hat{g}(k)\right) y(k) + \left(d_z - \hat{d}_z\right) u(k)\\ &= c_z\tilde{z}(k) - y^T(k)\tilde{g}^T(k) + u^T(k)\tilde{d}_z^T(k) = c_z\tilde{z}(k) - \left[-u^T(k)\ y^T(k)\right] \bar{I}\tilde{\theta}(k)\end{aligned}$$

and it holds

$$\begin{aligned}\tilde{z}(k+1) = \bar{A}_z\tilde{z}(k) + Q\left(u(k), y(k)\right)\tilde{\theta}(k) - V(k+1)\left(\hat{\theta}(k+1) - \hat{\theta}(k)\right)\\ c_z\tilde{z}(k) - \left[-u^T(k)\ y^T(k)\right]\bar{I}\tilde{\theta}(k) = c_z\eta(k) + \varphi(k)\tilde{\theta}(k).\end{aligned}$$

Moreover, it follows from (11.19) that

$$\tilde{\theta}(k+1) = -\gamma(k)\varphi^T(k)r(k) + \tilde{\theta}(k)$$

which lead to

$$\begin{aligned}\tilde{\theta}(k+1) &= \left(I - \gamma(k)\varphi^T(k)\varphi(k)\right)\tilde{\theta}(k) + \Theta(k)\eta(k)\\ \Theta(k) &= -\gamma(k)\varphi^T(k)c_z.\end{aligned}$$

Note that

$$\begin{aligned}V(k+1)\left(\hat{\theta}(k+1) - \hat{\theta}(k)\right) &= V(k+1)\left(\tilde{\theta}(k) - \tilde{\theta}(k+1)\right)\\ \eta(k+1) &= \tilde{z}(k+1) - V(k+1)\tilde{\theta}(k+1)\\ \Longrightarrow \eta(k+1) &= \bar{A}_z\tilde{z}(k) + Q\left(u(k), y(k)\right)\tilde{\theta}(k) - V(k+1)\tilde{\theta}(k) = \bar{A}_z\eta(k).\end{aligned}$$

We finally have the following compact form

$$\begin{bmatrix}\eta(k+1)\\ \tilde{\theta}(k+1)\end{bmatrix} = \begin{bmatrix}\bar{A}_z & 0\\ \Theta(k) & I - \gamma(k)\varphi^T(k)\varphi(k)\end{bmatrix}\begin{bmatrix}\eta(k)\\ \tilde{\theta}(k)\end{bmatrix}. \tag{11.21}$$

Based on (11.21) and Lemma 11.3–11.4, we are now able to prove the following theorems.

Theorem 11.1 *Given adaptive residual generator* (11.15)–(11.19)*, then*

$$\lim_{k\to\infty} r(k) = 0. \tag{11.22}$$

Proof It follows from (11.21) that $\eta(k) \to 0$ with exponential convergence. Hence, we only need to consider

$$\begin{aligned} r(k) &= \varphi(k)\tilde{\theta}(k) \\ \tilde{\theta}(k+1) &= \left(I - \gamma(k)\varphi^T(k)\varphi(k)\right)\tilde{\theta}(k) \end{aligned} \tag{11.23}$$

which, according to Lemma 11.3, leads to

$$\lim_{k\to\infty} \frac{r(k)}{\sqrt{\delta + \varphi(k)\varphi^T(k)}} = 0.$$

Considering that the auxiliary filter (11.17), (11.18) is exponentially stable and the inputs and outputs of the system are bounded, it turns out

$$\left|\varphi(k)\varphi^T(k)\right| < \infty, \lim_{k\to\infty} \left\|\gamma(k)\varphi^T(k)c_z\eta(k)\right\| = 0$$

which finally results in $\lim_{k\to\infty} r(k) = 0$.

It follows from Theorem 11.1 that

- the adaptive residual generator (11.15)–(11.19) is stable and
- the residual signal satisfies the basic requirement on residuals, that is, $\lim_{k\to\infty} r(k) = 0$ in the fault-free case.

On the other hand, Theorem 11.1 does not provide us with knowledge of the convergence speed. It is well-known that a reliable and early fault detection requires a quick convergence of $r(k)$. For this purpose, additional conditions are required, which are given in the following theorem.

Theorem 11.2 *Given adaptive residual generator* (11.15)–(11.19) *and assume that there exist positive constants* β_1, β_2, *and integer* Π *such that for all* k

$$0 < \beta_1 I \le \sum_{i=k}^{k+\Pi-1} \varphi^T(i)\varphi(i) \le \beta_2 I < \infty \tag{11.24}$$

then system (11.21) *is globally exponentially stable.*

Proof The proof is similar with the one for Theorem 11.1. It follows from Lemma 11.4 that the subsystem (11.23) is exponentially stable. Moreover, considering that $\eta(k)$ exponentially converges to zero and $\Theta(k)$ is bounded, it can be concluded that system (11.21) is exponentially stable.

Condition (11.24) is known as the existence condition for a persistent excitation which is needed for a successful parameter identification. In other words, the adaptive residual generator (11.15)–(11.19) is exponentially stable if the system under consideration is persistently excited. Note that in this case

$$\lim_{k\to\infty} \hat{\theta}(k) = \theta$$

with an exponential converging speed.

11.3.4 An Extension to the Adaptive State Observer

With a slight extension, we are also able to construct an adaptive state observer using Algorithm 11.4. To this end, we first find a vector κ_o, as given in Theorem 10.1 and Algorithm 10.3, and then using

$$\bar{y}(k) = \kappa_o y(k) \in \mathcal{R}$$

to construct an adaptive state observer as given in (11.15)–(11.19) with $s = n$. Concretely, the observer that is driven by the residual signal and the auxiliary filter will be realized as follows, while no change in the parameter estimator (11.19) is made,

Observer

$$\begin{aligned}\hat{z}(k+1) &= \bar{A}_z \hat{z}(k) + Q\left(u(k), \bar{y}(k)\right)\hat{\theta}(k) + V(k+1)\left(\hat{\theta}(k+1) - \hat{\theta}(k)\right) \in \mathcal{R}^n \\ r(k) &= \left[\, -u^T(k) \;\; \bar{y}^T(k) \,\right] \bar{I}\hat{\theta}(k) - c_z \hat{z}(k)\end{aligned}$$

Auxiliary filter

$$\begin{aligned}V(k+1) &= \bar{A}_z V(k) + Q(u(k), \bar{y}(k)) \\ \varphi(k) &= c_z V(k) - \left[\, -u^T(k) \;\; \bar{y}^T(k) \,\right] \bar{I}.\end{aligned}$$

Table 11.1 List of the simulated faults

Fault cases	Occur time	Fault description
Sensor fault in Tank 3	3,000 s	Offset 6 cm
Pump fault in Tank 2	3,000 s	Power reduction of 30 %
Leak in Tank 2	3,000 s	Magnitude of 15 %

11.4 Case Studies

11.4.1 Application of Adaptive SVD Based RPCA Scheme to Three-Tank System

In this section, we demonstrate the application of the adaptive SVD method introduced in Sect. 11.2 to PCA based fault detection. This method is also called recursive PCA (RPCA). We test this method on the Simulink model of the three-tank system in four case scenarios: (S1) slow normal plugging in the pipe connecting Tank 1 and Tank 3 (S2) a sensor fault (S3) a leak fault (S4) a pump fault.

Test Setup
The three-tank system is working in the closed-loop and steady mode around a fixed working point. The sampling time is 2 s. As a simulated change in the plant (case scenario S1), a slowly changing plugging in the pipe connecting Tank 1 and Tank 3 is generated as follows: at time instant 1,001 s, the plugging is added, which increases asymptotically according to

$$plugging = 5 \cdot (1 - e^{-\lambda t})$$

where $\lambda = 8 \cdot 10^{-5}$. The plugging causes change in the cross section area of the pipe connecting Tank 1 and Tank 3, which is 0.5 cm before the plugging is added. In the simulation, the cross section area of the pipe is reduced to 0.48 cm at most. In the case scenarios S2–S4, faults are simulated, which are summarized in Table 11.1.

In our test study, only liquid levels h_1, h_2 and h_3 are used for the process monitoring purpose.

The tests are designed as follows: (i) 300 samples are first collected to build the initial PCA model (ii) the process is then simulated for further 2,000 sampling instants for running the RPCA and the standard PCA. The number of significant PCs is determined using the cumulative percent variance such that the variance explained is approximately 99 % of the total variance. It should be mentioned that the threshold of *SPE* is set using the modified form given in (5.16). It follows the update in the *SPE* statistic.

Test Results
Figures 11.1 and 11.2 are the results of the test statistics of the standard PCA and RPCA. It is evident that the standard PCA leads to a (very) high FAR and cannot

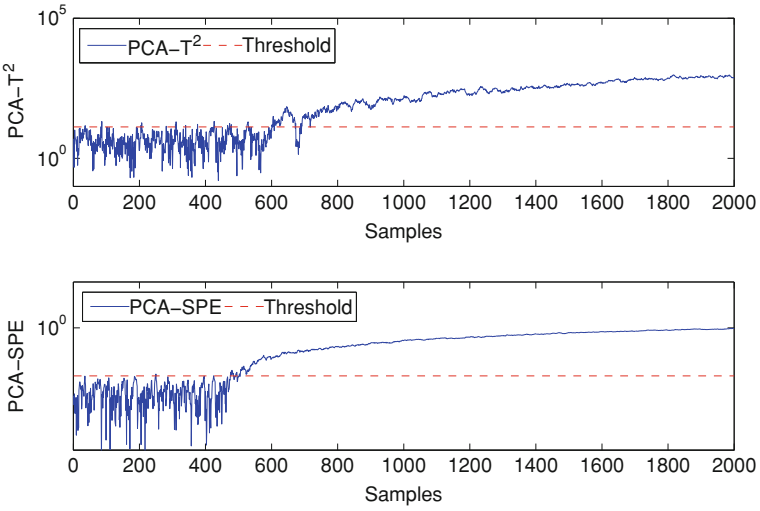


Fig. 11.1 Test statistics of PCA by operation point change

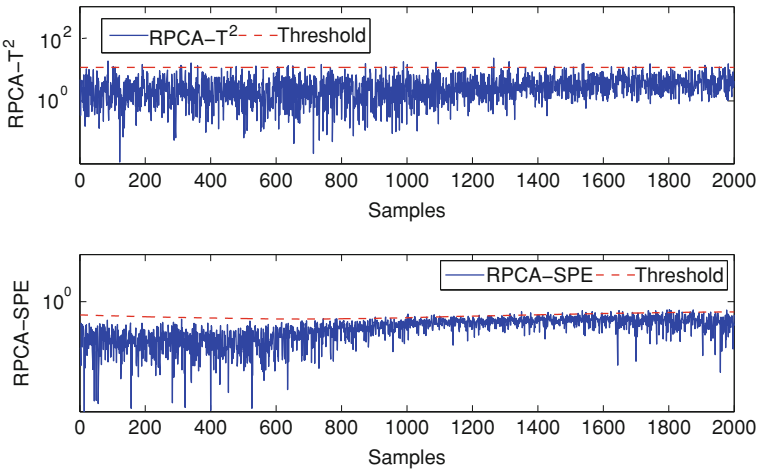


Fig. 11.2 Test statistics of RPCA by operation point change

be used for process monitoring. In against, the FAR of the RPCA is considerably moderate and can be adopted for the detection purpose. Note that in some figures in this subsection the *semilog-style* is adopted for the y-axis, in order to simplify the comparisons between the simulation results.

In Figs. 11.3, 11.4 and 11.5, the test statistics of the RPCA in the case scenarios S2–S4 are given. Successful fault detection is achieved.

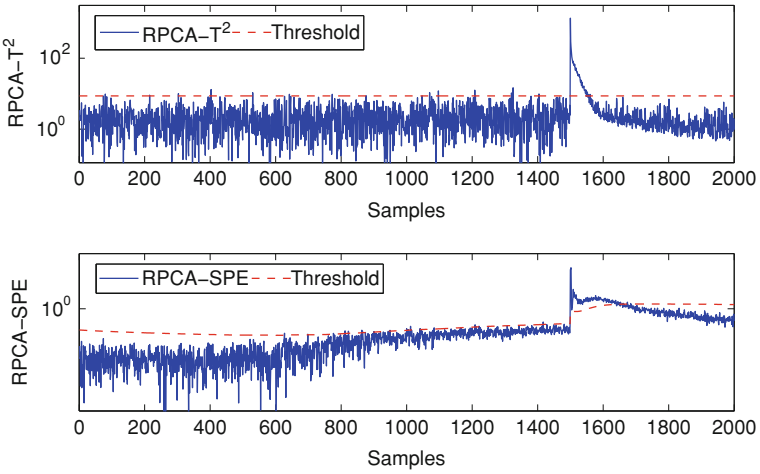


Fig. 11.3 Test statistics of RPCA by a sensor fault

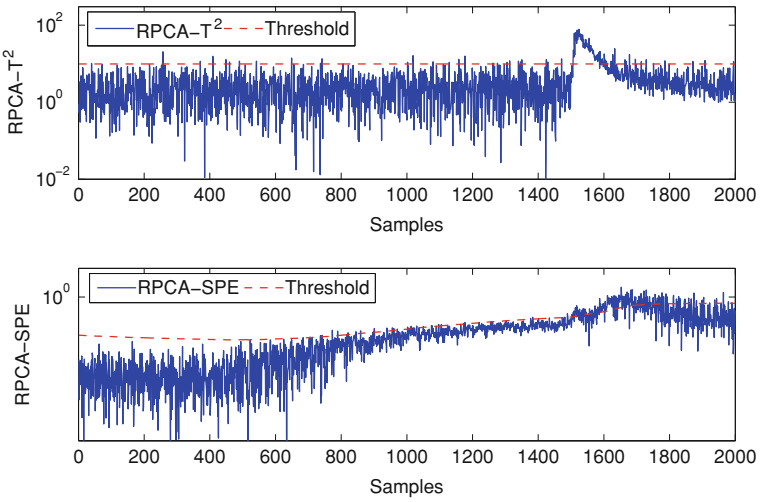


Fig. 11.4 Test statistics of RPCA by a leak fault

It should be pointed out that for the demonstration purpose, we have only used liquid levels as the process measurements. It is thus reasonable that the test statistics are more sensitive to the sensor fault. Although the actuator and process faults can be detected, the sensitivity to these faults is low. This problem can be solved when also the actuator signals are used for the detection purpose.

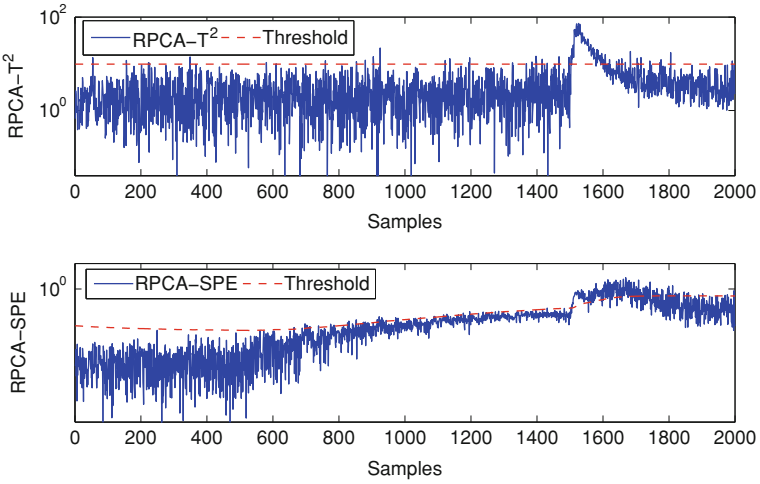


Fig. 11.5 Test statistics of RPCA by a pump fault

11.4.2 Application of the Adaptive Observer-Based Residual Generation Scheme to the Three-Tank System

Now, the adaptive observer-based residual generation scheme is illustrated by a simulation study on the three-tank system. Due to the nonlinearity in the process, linear models achieved by a linearization at different operation points would be different. Consequently, the performance of an LTI observer and residual generator would be generally poor. It will be demonstrated that the proposed adaptive observer and residual generation scheme deliver satisfactory results.

Consider the adaptive observer (11.15)–(11.19). The initial parameters are identified using the data-driven method proposed in Chap. 10. The parameter matrix L_0 is set to be

$$L_0 = \begin{bmatrix} 0.1 & 0.15 & 0.3 \end{bmatrix}^T$$

whose selection ensures that the eigenvalues of $\bar{A}_z$ lie in the unit circle.

Our simulation runs for 400 s. During the first 200 s, the set-points (representing the operating point) have been set to be

$$h_1 = 45\,\text{cm}, h_2 = 30\,\text{cm}$$

while in the next 200 s they have been set equal to

$$h_1 = 55\,\text{cm}, h_2 = 40\,\text{cm}.$$

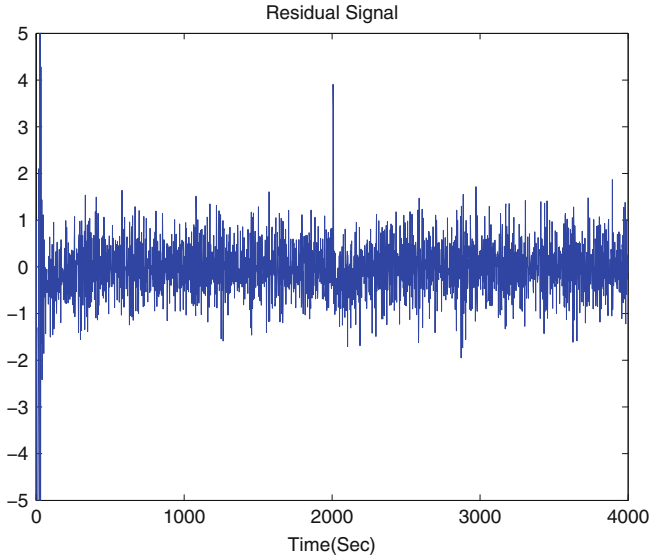


Fig. 11.6 Residual signal

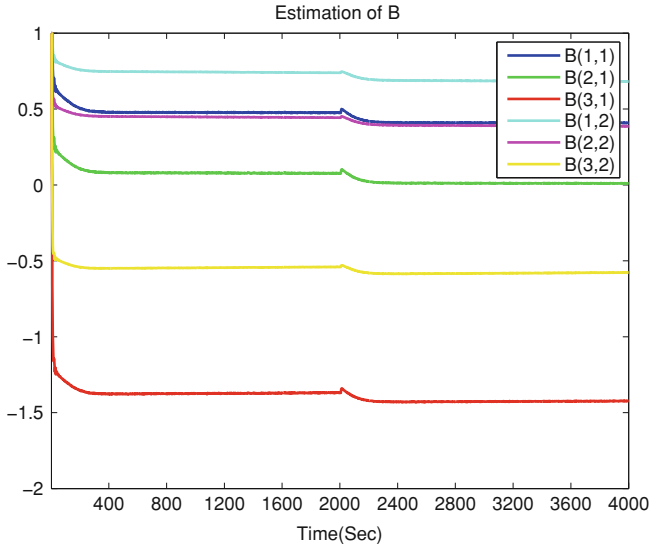


Fig. 11.7 The estimated B_z

Figure 11.6 shows a satisfactory residual generation, while Figs. 11.7, 11.8 and 11.9 give the updating results of system matrices B_z, L_z, g respectively.

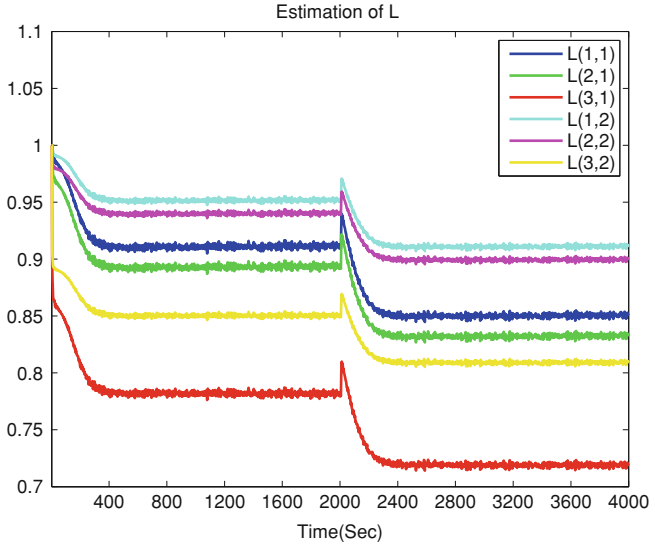


Fig. 11.8 The estimated L_z

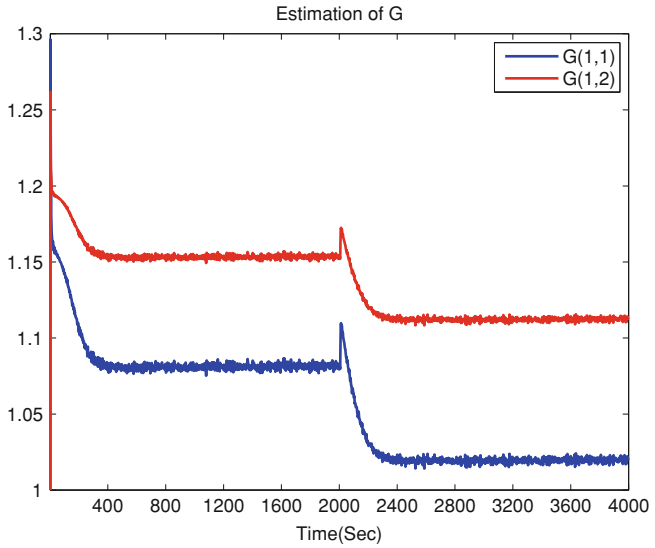


Fig. 11.9 The estimated g

11.5 Notes and References

The recursive and adaptive techniques have been well developed and, thanks to their efficient online computation ability, widely and successfully applied to the standard data-driven fault diagnosis methods. Since the early work by Helland et al. [1] and Qin [2], numerous recursive schemes have been reported. While the recursive updates of the mean, variance and the associated normalization of the new observation are trivial [3], the research focus is mainly on the recursive computation of SVD which is used both in the PCA and PLS. Li et al. [3] proposed to apply the rank-one modification technique for a recursive updating of the (normalized) covariance matrix and its SVD used in the PCA. In order to reduce the needed online computations, Elshenawy et al. [4] have proposed to apply the two adaptive/recursive SVD methods presented in the first and second sections of this chapter to the realization of recursive PCA (RPCA).

The DPM serves as a simple approach for recursively updating the singular values and associated singular vectors of interests and has been developed based on OI [5]. The DPM introduced in Sect. 11.1 was proposed in [6]. The details about Lemma 11.1, its proof and the update of μ given in (11.5) can be found in [5, 6].

The adaptive SVD method introduced in Sect. 11.2 is based on the first-order perturbation (FOP) theory [7]. The mathematical results presented in this section are mainly given by Willink [8]. For a detailed proof of Lemma 11.2 the reader is referred to [7, 8].

It is worth to mention that in [4] the standard PCA and different RPCA algorithms have been compared regarding to their online computation complexity. It is demonstrated that the DPM is the simplest algorithm if only a number of singular values and associated singular vectors are needed for the process monitoring and fault diagnosis purposes. In general, it is recommended to utilize the FOP method for the RPCA and RPLS. One important reason is that all singular values and the associated eigenvectors of the normalized covariance matrix are needed for an efficient and reliable FD.

Based on the similar idea behind the RPCA, Naik et al. [9] has proposed to apply the FOP and DPM algorithms for recursively updating parity vectors and diagnostic observer respectively. In this work, results of a benchmark study on the TEP have also been reported.

Recently, Wang et al. [10] pointed out numerous deficiencies in the existing RPCA algorithms due to ever-growing data set which leads to a reduction in the speed of adaptation. As a solution, they proposed the moving window PCA (MWPCA) scheme. He and Yang [11] also proposed an MWPCA scheme with an improved RPCA algorithm. In both studies, significant improvement of the adaptation performance is demonstrated. A combination of the MWPCA or MWPLS with the FOP-based RPCA or RPLS promises good process monitoring and fault diagnosis performance.

Model-based adaptive technique is an alternative and powerful tool to cope with the problems caused by changes in the process under monitoring. In the past

two decades, numerous adaptive observer based fault detection schemes have been reported, see for instance the recent work by [12, 13]. In Sect. 11.3, we have presented an adaptive observer scheme, based on the data-driven method developed in the previous chapters, for residual generation. The major difference between our results and the existing adaptive observer based fault detection schemes is that the SKR realization, instead of the standard system matrices in a state space representation/model, is online and recursively estimated in our adaptive scheme. To construct the adaptive residual generator, the standard adaptive technique for discrete-time systems [14], as well as the ideas of the adaptive observer design proposed by [12, 15] are applied.

Lemmas 11.3–11.4 are given in [14] (Theorem 6.3 and 6.4). Our proofs of Theorems 11.1–11.2 follow the same proof procedure proposed by Bastin and Gevers [12] for the adaptive observer design.

References

1. Helland DM, Bernstein HE, Borgen O, Martens H (1992) Recursive algorithm for partial least squares regression. Chemometr Intell Lab Syst 14:129–137
2. Qin SJ (1998) Recursive PLS algorithms for adaptive data modeling. Comput Chem Eng 22:503–514
3. Li W, Yue HH, Valle-Cervantes S, Qin SJ (2000) Recursive PCA for adaptive process monitoring. J Process Control 10:471–486
4. Elshenawy LM, Yin S, Naik AS, Ding SX (2010) Efficient recursive principal component analysis algorithms for proces monitoring. Ind Eng Chem Res 49:252–259
5. Golub GH, Loan CFV (1993) Matrix computations, 2nd edn. The John Hopkins University Press, Baltimore
6. Doukopoulos XG, Moustakides GV (2008) Fast and stable subspace tracking. IEEE Trans Sig Process 56:1452–1465
7. Stewart GW, Sun J-G (1990) Matrix perturbation theory. Academic Press, San Diego
8. Willink TJ (2008) Efficient adaptive SVD algorithm for MIMO applications. IEEE Trans Sig Process 56:615–622
9. Naik AS, Yin S, Ding SX, Zhang P (2010) Recursive identification algorithms to design fault detection systems. J Process Control 20:957–965
10. Wang X, Kruger U, Irwin GW (2005) Process monitoring approach using fast moving window PCA. Ind Eng Chem Res 44:5691–5702
11. He XB, Yang YP (2008) Variable MWPCA for adaptive process monitoring. Ind Eng Chem Res 47:419–427
12. Zhang Q (2002) Adaptive observer for multiple-input-multiple-output (MIMO) linear time-varying systems. IEEE Trans Autom Control 47:525–529
13. Caccavale F, Pierri F, Villani L (2008) Adaptive observer for fault aidgnosis in nonlinear discrete-time systems. J Dyn Syst Meas Control 130:021 005–1–021 005–9
14. Aström KJ, Wittenmark B (1995) Adaptive control. Addison-Wesley Publishing Company, Reading
15. Bastin G, Gevers M (1988) Stable adaptive observers for nonlinear time-varying systems. IEEE Trans Autom Control 33:650–658

Chapter 12
Iterative Optimization of Process Monitoring and Fault Detection Systems

In the previous chapter, we have introduced different adaptive techniques and discussed their applications to process monitoring and fault detection issues. The essential idea behind these adaptive methods is the real-time update of the parameters in the monitoring and detection system to match the possible changes in the process under monitoring. This idea is realized in the form of recursive algorithms. In this chapter, we address a different issue, although it seems, from the computational viewpoint, similar to the adaptive and recursive methods.

The motivation of our study is as follows. Suppose that by means of the data-driven design methods and the adaptive schemes a process monitoring and fault detection system is well constructed and in operation. Due to changes in the environment around the process, the performance of the process monitoring and fault detection system becomes sub-optimal or even poor. Our objective is to achieve a real-time optimization of the process monitoring and fault detection system during its operation.

Inspired by the reinforcement learning technique and its application to dealing with control problems, we are going to approach system optimization by means of repeated (iterative) interactions with the process environment and updating the process monitoring and fault detection algorithms based on new knowledge of the environment. This procedure is a real-time and iterative optimization.

12.1 Iterative Generalized Least Squares Estimation

In this section, we introduce the so-called iterative generalized least squares (IGLS) estimation scheme, which was proposed by Goldstein in 1986 and builds the fundament for the subsequent work.

Consider the measurement model

$$y = Ax + \varepsilon, \varepsilon \sim \mathcal{N}(0, \Sigma) \tag{12.1}$$

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_12, © Springer-Verlag London 2014

where $y \in \mathcal{R}^m$ represents the measurement vector, $x \in \mathcal{R}^n$ the internal process state vector. We assume that $A \in \mathcal{R}^{m\times n}$ is known and left invertible. In Sect. 3.5.1, it has been demonstrated that on the assumption of known Σ,

$$\hat{x} = \left(A^T \Sigma^{-1} A\right)^{-1} A^T \Sigma^{-1} y \tag{12.2}$$

delivers a unbiased and minimum variance estimate for x, which is also called generalized least squares (GLS) estimation. The covariance matrix of the GLS estimation error is

$$\mathcal{E}\left(x - \hat{x}\right)\left(x - \hat{x}\right)^T = \left(A^T \Sigma^{-1} A\right)^{-1}. \tag{12.3}$$

Applying the GLS estimation (12.2) to building a residual vector,

$$r = y - A\hat{x}$$

results in

$$\mathcal{E}\left(rr^T\right) = \Sigma - A\left(A^T \Sigma^{-1} A\right)^{-1} A^T \tag{12.4}$$

which leads to an optimal fault detection.

We now assume that Σ is unknown and study a simultaneous estimation of x and Σ using an iterative algorithm, the so-called IGLS algorithm proposed. To this end, rewrite (12.4) into

$$\Sigma = \mathcal{E}\left(y - A\hat{x}\right)\left(y - A\hat{x}\right)^T + A\left(A^T \Sigma^{-1} A\right)^{-1} A^T. \tag{12.5}$$

It follows from (12.5) that the iterative algorithm has been proposed

$$\Sigma_i = \mathcal{E}\left(y - A\hat{x}\right)\left(y - A\hat{x}\right)^T + A\left(A^T \Sigma_{i-1}^{-1} A\right)^{-1} A^T \tag{12.6}$$

that is, an update of the estimate for Σ at the ith iteration is achieved based on the value of the $(i-1)$-th iteration and the value of the cost function $\mathcal{E}\left(y - A\hat{x}\right)\left(y - A\hat{x}\right)^T$. It has been proven by Goldstein in 1989 that this iterative algorithm is equivalent to the so-called restricted maximum likelihood estimation.

Often, Σ has a certain structure and $vec\,(\Sigma)$ can be parametrized by

$$vec\,(\Sigma) = A_\Sigma \theta. \tag{12.7}$$

For instance, for

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

we have

$$vec\,(\Sigma) = \begin{bmatrix} \sigma_1 \\ 0 \\ \sigma_2 \\ 0 \end{bmatrix} = A_\Sigma \theta, A_\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \theta = \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix}.$$

In this case, the update of Σ_i can be realized by means of an LS estimation of θ based on the linear model

$$A_\Sigma \theta = vec\left(\left(y - A\hat{x}\right)\left(y - A\hat{x}\right)^T + A\left(A^T \Sigma_{i-1}^{-1} A\right)^{-1} A^T\right). \tag{12.8}$$

In the next sections, the basic ideas of IGLS will be adopted and applied to develop algorithms for iterative optimization of recursive LS (RLS) computation and Kalman filters.

12.2 Iterative RLS Estimation

12.2.1 The Basic Idea and Approach

Now, consider the measurement model

$$y(k) = A(k)x + \varepsilon(k), \mathcal{E}\left(\varepsilon(i)\varepsilon(j)^T\right) = \begin{cases} \Sigma, i = j \\ 0, i \neq j \end{cases}, \varepsilon(k) \sim \mathcal{N}(0, \Sigma) \tag{12.9}$$

with $y(k) \in \mathcal{R}^m$ representing the measurement vector, $x \in \mathcal{R}^n$ the internal process state vector. We assume that $A(k) \in \mathcal{R}^{m \times n}$ is known and measurement data $y(1), \ldots, y(N)$ are available for the estimation purpose. Let

$$\mathcal{Y} = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}, \mathcal{A} = \begin{bmatrix} A(1) \\ \vdots \\ A(N) \end{bmatrix}, E = \begin{bmatrix} \varepsilon(1) \\ \vdots \\ \varepsilon(N) \end{bmatrix}, \bar{\Sigma} = diag\,(\Sigma, \ldots, \Sigma).$$

The estimation problem can be solved using a standard LS

$$\hat{x} = \left(\mathcal{A}^T \bar{\Sigma}^{-1} \mathcal{A}\right)^{-1} \mathcal{A}^T \bar{\Sigma}^{-1} \mathcal{Y}. \tag{12.10}$$

It is obvious that solution (12.10) is computationally less attractive. Instead, the well-known RLS algorithm is an effective tool for dealing with the estimation of x. The standard RLS is summarized as follows:

- update of covariance matrix $P_k = \mathcal{E}\left(x - \hat{x}_k\right)\left(x - \hat{x}_k\right)^T$

$$P_k = \left(P_{k-1}^{-1} + A^T(k)\Sigma^{-1}A(k)\right)^{-1} \tag{12.11}$$

$$= P_{k-1} - P_{k-1}A^T(k)\left(\Sigma + A(k)P_{k-1}A^T(k)\right)^{-1}A(k)P_{k-1} \tag{12.12}$$

- update of the estimation

$$\hat{x}_k = \hat{x}_{k-1} - P_k A^T(k)\Sigma^{-1}\left(A(k)\hat{x}_{k-1} - y(k)\right). \tag{12.13}$$

It is worth to mention that $\hat{x}_k$ is equivalent to the LS estimation delivered by (12.10) using the data $y(1), \ldots, y(k)$.

Similar to (12.4), it is straightforward to prove that

$$\mathcal{E}\left(\left(y(k) - A(k)\hat{x}_k\right)\left(y(k) - A(k)\hat{x}_k\right)^T\right) = \Sigma - A(k)P_k A^T(k)$$

which leads to

$$\Sigma = \mathcal{E}\left(\left(y(k) - A(k)\hat{x}_k\right)\left(y(k) - A(k)\hat{x}_k\right)^T\right) + A(k)P_k(\Sigma)A^T(k) \tag{12.14}$$

where $P_k(\Sigma)$ is introduced to emphasize that

$$\mathcal{E}\left(\left(x - \hat{x}_k\right)\left(x - \hat{x}_k\right)^T\right) = P_k(\Sigma)$$

is a function of Σ. As a result, we introduce the update computation of Σ

$$\Sigma_i = \mathcal{E}\left(y(k) - A(k)\hat{x}_k\right)\left(y(k) - A(k)\hat{x}_k\right)^T + A(k)P_k(\Sigma_{i-1})A^T(k). \tag{12.15}$$

Σ_i denotes the ith update of the covariance matrix Σ. Note that both $\mathcal{E}\left(\left(y(k) - A(k)\hat{x}_k\right)\left(y(k) - A(k)\hat{x}_k\right)^T\right)$ and $A(k)P_k(\Sigma)A^T(k)$ are time-varying. Moreover, if Σ is of certain structure and can be parametrized by

$$vec\,(\Sigma) = A_\Sigma\theta \tag{12.16}$$

then the update of Σ_i will be realized by means of an LS of θ_i based on the linear relation

$$A_\Sigma\theta_i = vec\left(\Psi_{k,i-1}\right) \tag{12.17}$$

$$\Psi_{k,i-1} = \left(y(k) - A(k)\hat{x}_k\right)\left(y(k) - A(k)\hat{x}_k\right)^T + A(k)P_k(\Sigma_{i-1})A^T(k). \tag{12.18}$$

It is worth to point out that the recursion (12.15) or the model (12.17) are equivalent to (12.6) or (12.8) and can be considered as an extension to the recursive computation.

12.2.2 Algorithm, its Realization and Implementation

In the original form of the IGLS, the iterations of $\hat{x}_k$ and Σ_k run simultaneously, that is, updating $\hat{x}_k$ and Σ_k at each sampling instant. For our purpose of optimizing the RLS estimator by iterative interactions with the environment, we propose the following scheme to run the estimator:

- set initial values
- run the RLS estimator under a given Σ_i for some sampling instants so that the influence of the initial setting disappears
- update $\Sigma_i \rightarrow \Sigma_{i+1}$ using the measurement and estimation data collected in a sampling interval.

In Fig. 12.1, this scheme is sketched.

It is clear that a key step in this iterative scheme is the update of the covariance matrix of the noise. To this end, sufficient data, that is, $y(k) - A(k)\hat{x}_k$, should be collected, in order to gain sufficient knowledge of the environment around the process under consideration.

In summary, we have

Algorithm 12.1 *Iterative RLS*

S1: Initialize the parameters: $P_0, \hat{x}(0), \Sigma_0$
S2: For i=1 to N
  S2-1: For j=1 to L1
    run RLS
  End
  S2-2: For j=1 to L2
    run RLS and update Σ_i
  End
End

12.2.3 An Example

In this numerical example, we illustrate the application of the algorithm proposed in the last subsection. Consider the model (12.9) with

$$A(k) = \begin{bmatrix} 0.2 + 0.1sin(0.1k) & 0.5 + 0.1cos(0.5k) \\ 0.7 + 0.1cos(0.1k) & 0.4 + 0.1sin(0.4k) \\ 0.2 + 0.1sin(0.4k) & 0.1 + 0.1cos(0.6k) \end{bmatrix}$$

$$\Sigma = diag(\sigma_1, \sigma_2, \sigma_3).$$

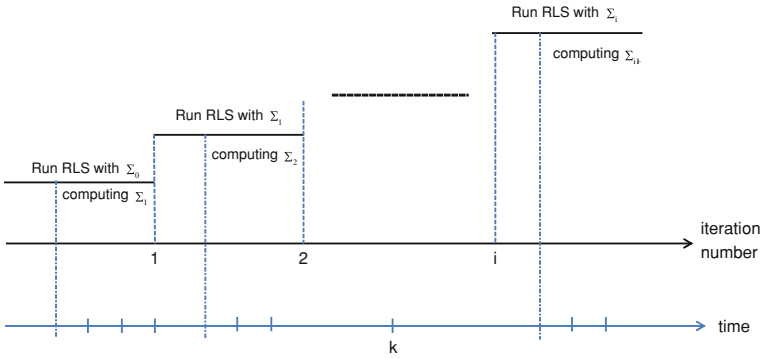


Fig. 12.1 Schematic description of the implementation of iterative RLS

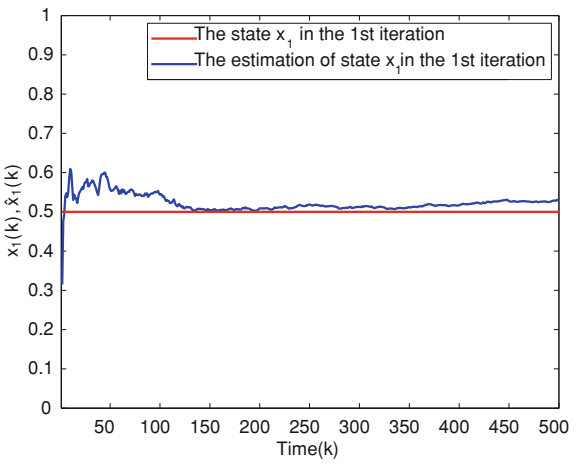


Fig. 12.2 Estimation of x_1 by the initial setting of Σ

For our test purpose, data $y(1), \ldots,$ are generated by setting

$$x = \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}, \Sigma = diag(0.03, 0.02, 0.01).$$

The update number of the covariance matrix Σ is $N = 20$. In Figs. 12.2 and 12.3, the estimations of x with the initial setting of Σ are shown, while Figs. 12.4 and 12.5 give the estimations of x after completing the Nth iteration. It is evident that the estimation performance is improved by updating the covariance matrix Σ.
The updates of $\sigma_1, \sigma_2, \sigma_3$ are shown in Figs. 12.6, 12.7, 12.8.

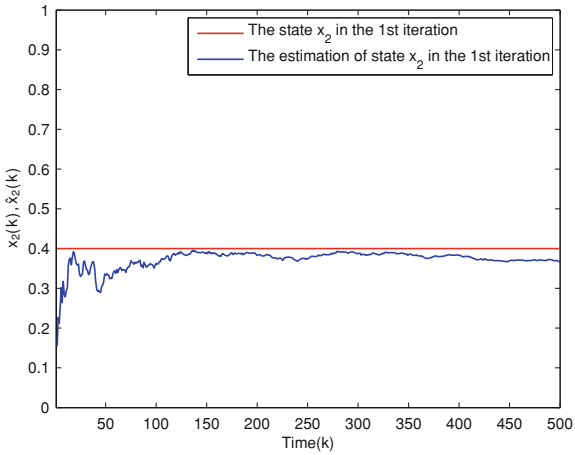


Fig. 12.3 Estimation of x_2 by the initial setting of Σ

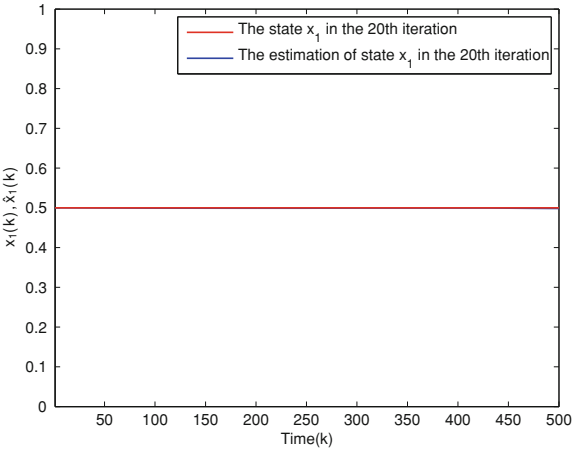


Fig. 12.4 Estimation of x_1 with the final update of Σ

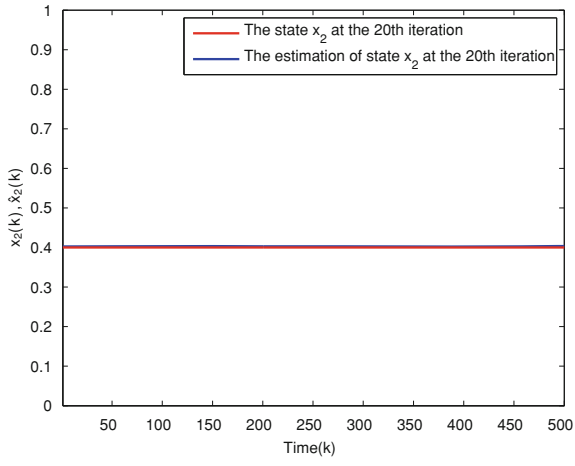


Fig. 12.5 Estimation of x_2 with the final update of Σ

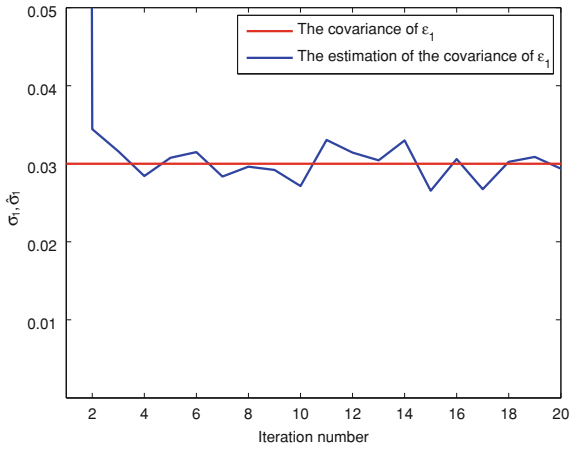


Fig. 12.6 Estimation of σ_1

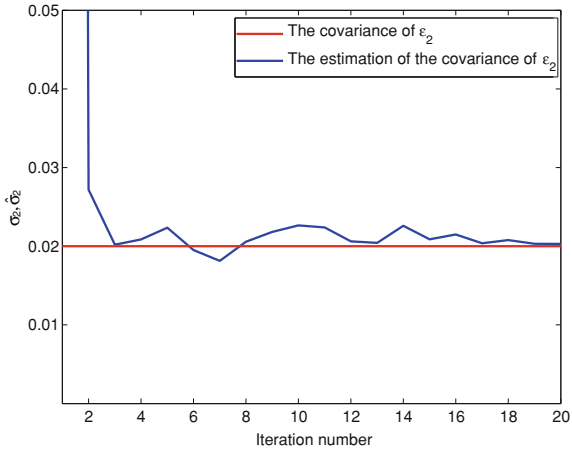


Fig. 12.7 Estimation of σ_2

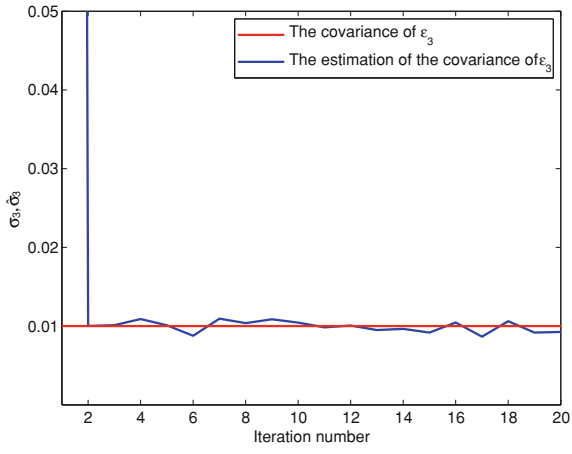


Fig. 12.8 Estimation of σ_3

12.3 Iterative Optimization of Kalman Filters

12.3.1 The Idea and Scheme

Consider a dynamic process described by

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{12.19}$$
$$y(k) = Cx(k) + Du(k) + v(k) \tag{12.20}$$

where $u \in \mathcal{R}^l$, $y \in \mathcal{R}^m$ and $x \in \mathcal{R}^n$, $w \in \mathcal{R}^n$ and $v \in \mathcal{R}^m$ denote noise sequences that are normally distributed, statistically independent of u and $x(0)$ and satisfy

$$\mathcal{E}\left(\begin{bmatrix} w(i) \\ v(i) \end{bmatrix}\begin{bmatrix} w(j) & v(j) \end{bmatrix}\right) = \begin{bmatrix} \Sigma_w \delta_{ij} & 0 \\ 0 & \Sigma_v \delta_{ij} \end{bmatrix}, \delta_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}.$$

A Kalman filter delivers an innovation sequence as residual vector. Thanks to its minimum variance property, the innovation sequence is optimal for the fault detection application.

It is well-known that a Kalman filter based state estimation can also be interpreted as an LS estimation problem. To this end, let

$$\begin{aligned}
&0 = x(0) - w_o,\ y(0) = Cx(0) + Du(0) + v(0),\ Bu(0) = x(1) - Ax(0) - w(0) \\
&y(1) = Cx(1) + Du(1) + v(1),\ Bu(1) = x(2) - Ax(1) - w(1), \ldots, \\
&y(i) = Cx(i) + Du(i) + v(i),\ Bu(i) = x(i+1) - Ax(i) - w(i), \ldots, \\
&y(k-1) = Cx(k-1) + Du(k-1) + v(k-1) \\
&Bu(k-1) = x(k) - Ax(k-1) - w(k-1)
\end{aligned}$$

where $w_o \sim \mathcal{N}(0, \Sigma_w)$. Our task is to estimate $x(0), \ldots, x(k)$ using the process input and output data $u(0), y(0), \ldots, u(k-1), y(k-1)$, and based on the model described by

$$\bar{y}_k = A_k \bar{x}_k + \bar{\varepsilon}_k$$

$$\bar{y}_k = \begin{bmatrix} 0 \\ y(0) - Du(0) \\ Bu(0) \\ \vdots \\ y(k-1) - Du(k-1) \\ Bu(k-1) \end{bmatrix}, \bar{x}_k = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(k) \end{bmatrix}$$

$$\bar{\varepsilon}_k = \begin{bmatrix} -w_o \\ v(0) \\ -w(0) \\ \vdots \\ v(k-1) \\ -w(k-1) \end{bmatrix} \sim \mathcal{N}\left(0, diag\left(\Sigma_w, \Sigma_v, \Sigma_w, \ldots, \Sigma_v, \Sigma_w\right)\right)$$

$$A_k = \begin{bmatrix} I & 0 & \cdots & 0 \\ C & 0 & \cdots & 0 \\ -A & I & 0 \cdots & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & C \; 0 \\ 0 & \cdots & -A & I \end{bmatrix}. \tag{12.21}$$

which can be solved by finding a RLS estimate for $\bar{x}_k$ with given $\bar{y}_k$.

On the assumption of known $\Sigma_w > 0$, $\Sigma_v > 0$, the Kalman filter algorithm is summarized as:

- update of covariance matrix of the innovation (residual vector)

$$\Sigma_{r,k} = \mathcal{E}\left(\left(y(k) - \hat{y}(k)\right)\left(y(k) - \hat{y}(k)\right)^T\right) = \Sigma_v + C P_k C^T \tag{12.22}$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

- update of the filter gain matrix

$$K_k = A P_k C^T \Sigma_{r,k}^{-1}$$

- update of covariance matrix P_k

$$P_{k+1} = A P_k A^T + \Sigma_w - K_k \Sigma_{r,k} K_k^T \tag{12.23}$$

- update of the state estimation

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K_k r(k) \tag{12.24}$$
$$r(k) = y(k) - C\hat{x}(k) - Du(k). \tag{12.25}$$

Next, assume that Σ_w, Σ_v are unknown. Along the lines of the iterative RLS algorithm presented in the previous section, we study iterative optimization of a Kalman filter. For this purpose, we first consider (12.22) and rewrite it into

$$\Sigma_v = \Sigma_{r,k} - C P_k C^T. \tag{12.26}$$

It follows from (12.23) that

$$\Sigma_v + C\Sigma_w C^T = \Sigma_{r,k+1} - \Psi_{k,0} \tag{12.27}$$
$$\Psi_{k,0} = C\left(AP_kA^T - K_k\Sigma_{r,k}K_k^T\right)C^T.$$

Repeating this step leads to

$$\Sigma_v + \sum_{i=0}^{j} CA^i\Sigma_w\left(CA^i\right)^T = \Sigma_{r,k+j+1} - \Psi_{k,j},\ j = 1, 2, \ldots \tag{12.28}$$
$$\Psi_{k,j} = C\left(A^{j+1}P_k\left(A^T\right)^{j+1} - \sum_{i=0}^{j} A^{j-i}K_{k+i}\Sigma_{r,k+i}K_{k+i}^T\left(A^{j-i}\right)^T\right)C^T.$$

Next, vectorize (12.26)–(12.28) using the relation

$$vec\,(T_1T_2T_3) = \left(T_3^T \otimes T_1\right)vec\,(T_2)$$

which yields

$$vec\,(\Sigma_v) = vec\left(\Sigma_{r,k} - CP_kC^T\right)$$
$$vec\left(\Sigma_v + \sum_{i=0}^{j} CA^i\Sigma_w\left(CA^i\right)^T\right) =$$
$$vec\,(\Sigma_v) + \sum_{i=0}^{j}\left(\left(CA^i\right)\otimes\left(CA^i\right)\right)vec\,(\Sigma_w) = vec\left(\Sigma_{r,k+j+1}\right) - vec\left(\Psi_{k,j}\right)$$

$$j = 0, 1, \ldots, \Theta \Longrightarrow \tag{12.29}$$

$$\begin{bmatrix} I & 0 \\ I & C\otimes C \\ \vdots & \vdots \\ I & \sum_{i=0}^{\Theta}\left(CA^i\right)\otimes\left(CA^i\right) \end{bmatrix}\begin{bmatrix} vec\,(\Sigma_v) \\ vec\,(\Sigma_w) \end{bmatrix} = \begin{bmatrix} vec\left(\Sigma_{r,k}\right) - vec(CP_kC^T) \\ vec\left(\Sigma_{r,k+1}\right) - vec\left(\Psi_{k,0}\right) \\ \vdots \\ vec\left(\Sigma_{r,k+\Theta+1}\right) - vec\left(\Psi_{k,\Theta}\right) \end{bmatrix}. \tag{12.30}$$

Recall that Σ_v, Σ_w are symmetric and have only, respectively, $m(m+1)/2$ and $n(n+1)/2$ independent elements. We denote them by θ_v, θ_w. As a result, there exists a matrix Π so that

$$\begin{bmatrix} vec\,(\Sigma_v) \\ vec\,(\Sigma_w) \end{bmatrix} = \Pi\begin{bmatrix} \theta_v \\ \theta_w \end{bmatrix}.$$

It yields

$$\begin{bmatrix} I & 0 \\ I & C \otimes C \\ \vdots & \vdots \\ I & \sum_{i=0}^{\Theta} \left(CA^i\right) \otimes \left(CA^i\right) \end{bmatrix} \Pi \begin{bmatrix} \theta_v \\ \theta_w \end{bmatrix} = \begin{bmatrix} vec\left(\Sigma_{r,k}\right) - vec(CP_kC^T) \\ vec\left(\Sigma_{r,k+1}\right) - vec\left(\Psi_{k,0}\right) \\ \vdots \\ vec\left(\Sigma_{r,k+\Theta+1}\right) - vec\left(\Psi_{k,\Theta}\right) \end{bmatrix}$$

which can be written as

$$A_{v,w} \begin{bmatrix} \theta_v \\ \theta_w \end{bmatrix} = y_{v,w} \tag{12.31}$$

$$A_{v,w} = \begin{bmatrix} I & 0 \\ I & C \otimes C \\ \vdots & \vdots \\ I & \sum_{i=0}^{\Theta} \left(CA^i\right) \otimes \left(CA^i\right) \end{bmatrix} \Pi,\ y_{v,w} = \begin{bmatrix} vec\left(\Sigma_{r,k}\right) - vec(CP_kC^T) \\ vec\left(\Sigma_{r,k+1}\right) - vec\left(\Psi_{k,0}\right) \\ \vdots \\ vec\left(\Sigma_{r,k+\Theta+1}\right) - vec\left(\Psi_{k,\Theta}\right) \end{bmatrix}.$$

For our purpose, Θ is so selected that

$$rank \begin{bmatrix} I & 0 \\ I & C \otimes C \\ \vdots & \vdots \\ I & \sum_{i=0}^{\Theta} \left(CA^i\right) \otimes \left(CA^i\right) \end{bmatrix} = \frac{m^2 + n^2 + n + m}{2}.$$

Note that (12.31) is a general form which also includes the case that Σ_v or/and Σ_w can be structurally parametrized. The iterative update of Σ_v and Σ_w follows from (12.31) and realized by an LS based on the form

$$A_{v,w} \begin{bmatrix} \theta_{v,i} \\ \theta_{w,i} \end{bmatrix} = y_{v,w}(k, \Sigma_{v,i-1}, \Sigma_{w,i-1}) \tag{12.32}$$

$$y_{v,w}(k, \Sigma_{v,i-1}, \Sigma_{w,i-1}) = \begin{bmatrix} \alpha(k) - vec(CP_kC^T) \\ \alpha(k+1) - vec\left(\Psi_{k,0}\right) \\ \vdots \\ \alpha(k+\Theta+1) - vec\left(\Psi_{k,\Theta}\right) \end{bmatrix}$$

$$\alpha(k+j) = vec\left(\left(y(k+j) - \hat{y}(k+j)\right)\left(y(k+1) - \hat{y}(k+1)^T\right)\right), j = 0, 1, \ldots, \Theta.$$

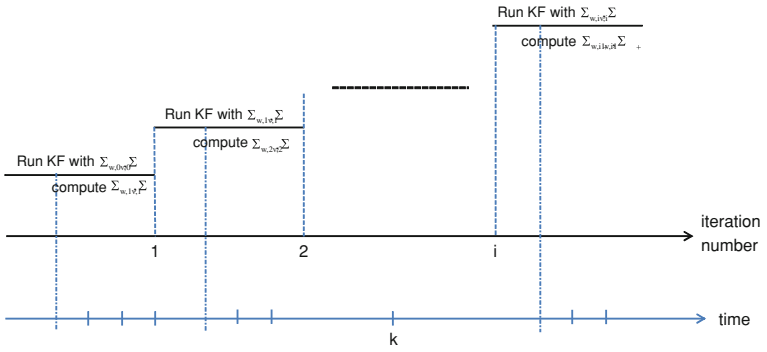


Fig. 12.9 Schematic description of the implementation of iterative Kalman filter (KF) scheme

12.3.2 Algorithm and Implementation

Similar to the iterative optimization of RLS, we propose the following scheme to optimize the Kalman filter by iterative interactions between the estimator and the environment:

- set initial values
- run the Kalman filter/estimator for given $\Sigma_{w,i}$, $\Sigma_{v,i}$ for some sampling instants so that the influence of the initial setting disappears
- update $\Sigma_{w,i} \rightarrow \Sigma_{w,i+1}$, $\Sigma_{v,i} \rightarrow \Sigma_{v,i+1}$ using the measurement and estimation data collected in a sampling interval.

The real-time implementation procedure is schematically sketched in Fig. 12.9 and the corresponding algorithm is summarized in Algorithm 12.2.

Algorithm 12.2 *Iterative Kalman filter (KF)*

S1: Initialize the parameters: P_0, $\hat{x}(0)$, $\Sigma_{w,0}$, $\Sigma_{v,0}$
S2: For i=1 to N
  S2-1: For j=1 to L1
    run FK algorithm
  End
  S2-2: For j=1 to L2
    run KF algorithm and update $\Sigma_{w,i}$, $\Sigma_{v,i}$
  End
End

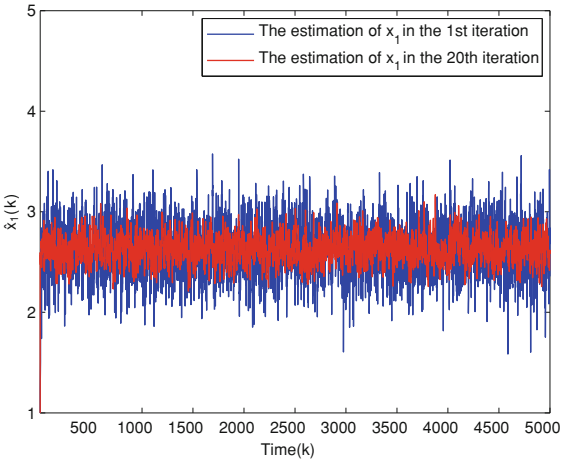


Fig. 12.10 Estimation of x_1 in comparison

12.3.3 An Example

We now illustrate the application of the above algorithm by a numerical example. Consider the model (12.19)–(12.20) with

$$A = \begin{bmatrix} 0.2 & -0.3 & 0.4 \\ 0.1 & 0.4 & -0.6 \\ 0.2 & 0.1 & 0.5 \end{bmatrix}, B = \begin{bmatrix} 0.5 \\ 0.6 \\ 0.4 \end{bmatrix}, C = \begin{bmatrix} 1 & 0.3 & 0.4 \\ 1 & 0.9 & 0.1 \end{bmatrix}.$$

For the purpose of data generation, we set

$$\Sigma_w = diag(\sigma_1, \sigma_2, \sigma_3) = diag(0.05, 0.04, 0.05)$$
$$\Sigma_v = diag(\sigma_4, \sigma_5) = diag(0.06, 0.06).$$

Moreover, for the sake of simplicity, the input is set to be $u(k) = 1$. In our simulation study, the case with unknown Σ_w and known Σ_v is considered. Figures 12.10, 12.11, 12.12 give a comparison of the state estimations at the beginning and end of the iterations. It is evident that improvement is achieved by the iterative optimization procedure. The iterative updating results for $\sigma_1, \sigma_2, \sigma_3$ are given in Figs. 12.13, 12.14, 12.15 respectively. Next, we demonstrate the application of the above algorithm to improving fault detection performance. To this end, we simulate a fault in the process modeled by fault vector $f(k)$ in

$$x(k+1) = Ax(k) + Bu(k) + f(k) + w(k),\ f(k) = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ 1.5 \end{bmatrix}, & k \geq 500 \\ 0, & k < 500 \end{cases}.$$

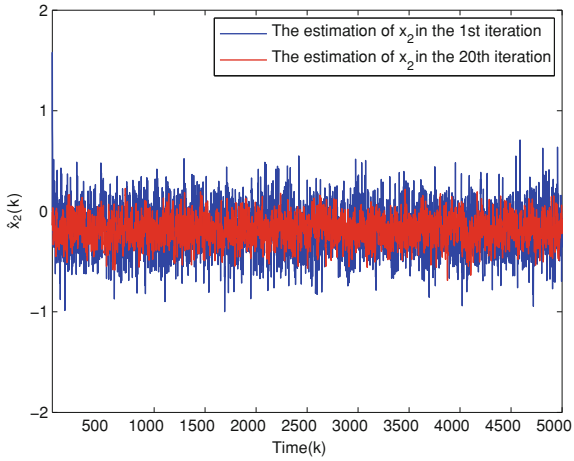


Fig. 12.11 Estimation of x_2 in comparison

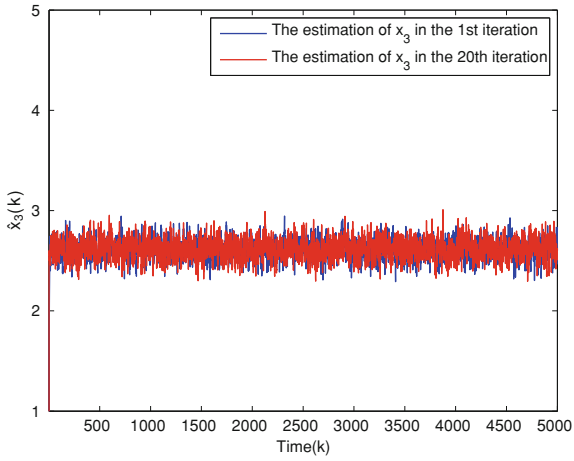


Fig. 12.12 Estimation of x_3 in comparison

For the detection purpose, T^2 test statistic of the residual vector has been adopted. Figure 12.16 shows the results of a comparison study between the fault detection performance at the 1st (T_1^2) and 20th (T_{20}^2) iterations. It can be seen that the fault detection performance is significantly improved by means of the iteration update.

12.4 Case Study

By a simulation study on the three-tank system, the application of the iterative optimization algorithm is illustrated and demonstrated. For our purpose, a linear model is adopted, which is achieved by a linearization at the operating point $h_1 = 45\,cm$, $h_2 = 30\,cm$, and described as follows:

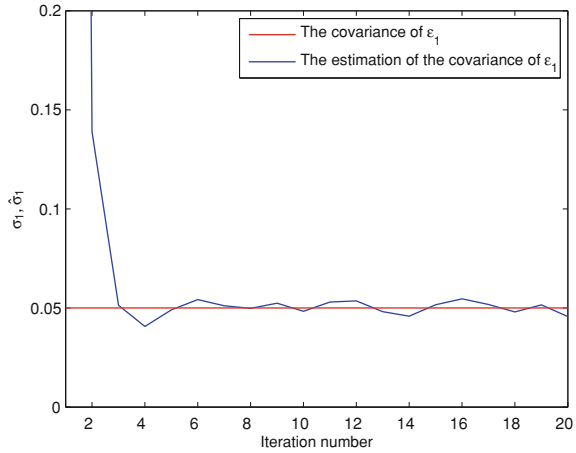


Fig. 12.13 Iterative updates of σ_1

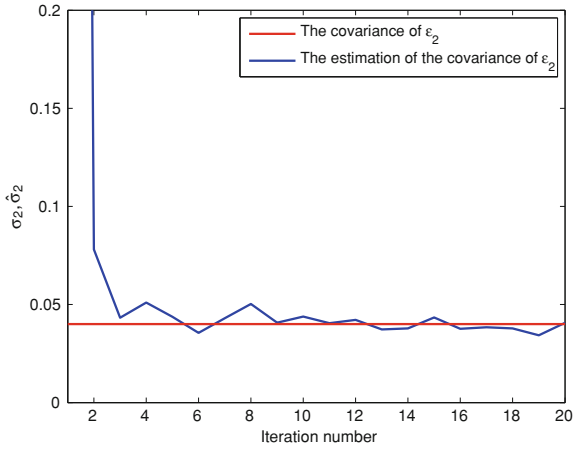


Fig. 12.14 Iterative updates of σ_2

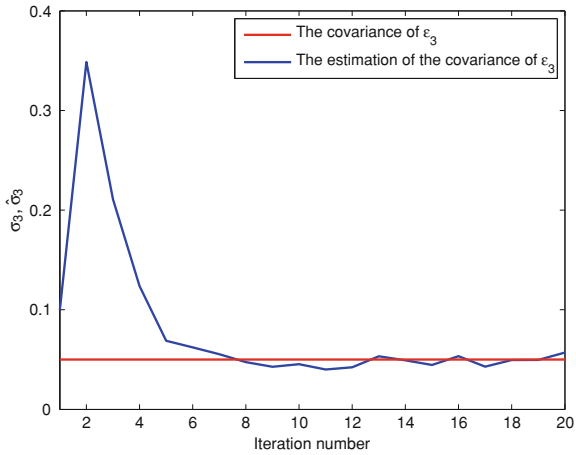


Fig. 12.15 Iterative updates of σ_3

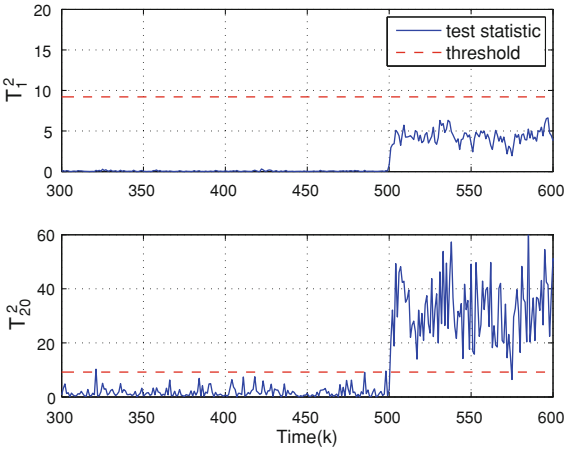


Fig. 12.16 T^2 test statistics at the 1st and 20th iterations

$$\dot{x} = Ax + Bu,\ y = Cx$$

$$A = \begin{bmatrix} -0.0085 & 0 & 0.0085 \\ 0 & -0.00195 & 0.0084 \\ 0.0085 & 0.0084 & -0.0169 \end{bmatrix}, B = \begin{bmatrix} 0.0065 & 0 \\ 0 & 0.0065 \\ 0 & 0 \end{bmatrix}.$$

We assume that h_1 and h_2 are measurement output variables. The discrete-time model, achieved by a discretization with a sampling time $t = 5s$, is given by

$$x(k+1) = A_d x(k) + B_d u(k),\ y(k) = Cx(k)$$

$$A_d = \begin{bmatrix} 0.9591 & 0.008279 & 0.0401 \\ 0.0008279 & 0.9079 & 0.03816 \\ 0.0401 & 0.03816 & 0.9206 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0.03179 & 9.129e-006 \\ 9.129e-006 & 0.03179 \\ 0.0006647 & 0.0006386 \end{bmatrix}, C = \begin{bmatrix} 1\ 0\ 0 \\ 0\ 1\ 0 \end{bmatrix}.$$

In the simulation study, zero-mean process and measurement noises with

$$\Sigma_w = diag\,(\sigma_1, \sigma_2, \sigma_3) = diag(0.0450, 0.054, 0.065)$$
$$\Sigma_v = diag\,(\sigma_2, \sigma_5) = diag(0.0360, 0.0219)$$

have been applied for generating the (simulated) process data.

12.4.1 Case 1: Σ_v is Unknown While Σ_w is Given

In this case, the two parameters of Σ_v are online estimated as Algorithm 12.2 runs. The iteration results are shown in Figs. 12.17 and 12.18. In order to demonstrate the

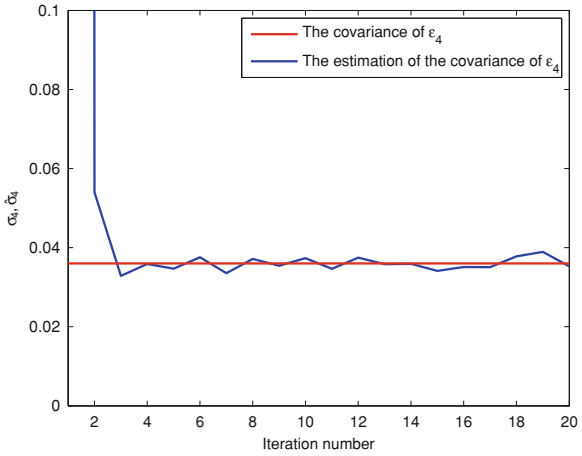
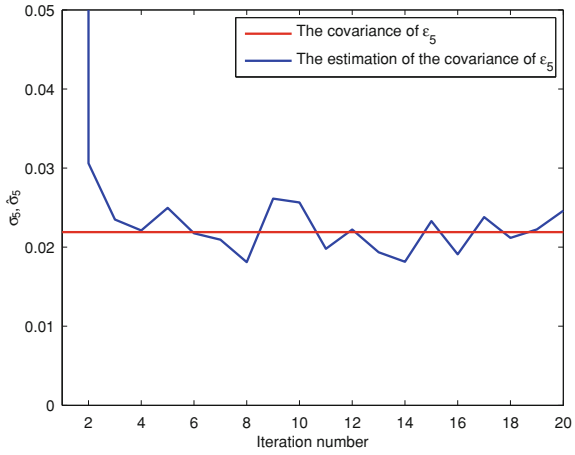


Fig. 12.17 Iterative updates of σ_4

Fig. 12.18 Iterative updates of σ_5



potential of the proposed method in improving fault detection performance, detection of a sensor fault has been tested. For this purpose, a 2 *cm* offset in tank 2 is simulated. It occurs at $k = 500s$. Figure 12.19 gives a comparison of T^2 test statistics of the residual signals at the 1st and 20th iterations. It is evident that the fault detectability is, thanks to the improved estimation of the covariance matrix, significantly improved after the iterations.

12.4.2 Case 2: Σ_w is Unknown While Σ_v is Given

Figures 12.20, 12.21, 12.22 give the iteration results with the updates of the three parameters of Σ_w as Algorithm 12.2 runs.

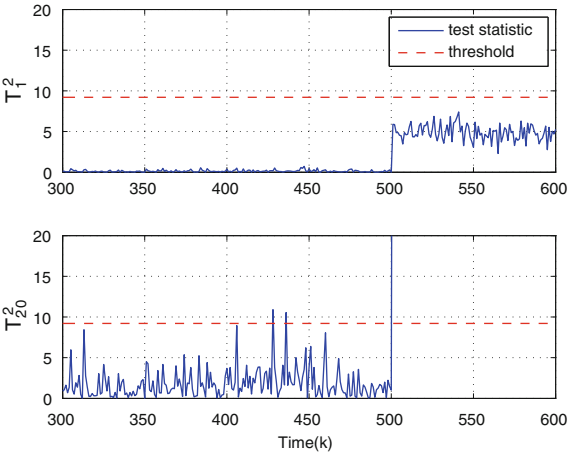


Fig. 12.19 T^2 test statistics at the 1st and 20th iterations

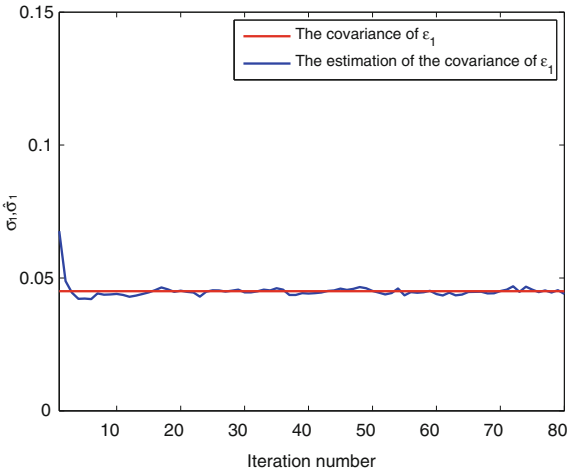


Fig. 12.20 Iterative updates of σ_1

12.5 Notes and References

This chapter deals with LS, RLS and Kalman filter algorithms, which are essential in the framework of process monitoring and diagnosis. When perfect information about the process and the environment around the process is available, the application of LS (to static processes) and Kalman filter (to dynamic processes) deliver the optimal monitoring and diagnosis performance. In the previous chapter, we have introduced adaptive techniques that result in updating the parameters in the monitoring and diagnosis systems to match changes in the process. Our focus in this chapter is on real-time updating of the RLS and Kalman filter algorithms as changes are under way in the environment around the process. The basic idea of our study consists in

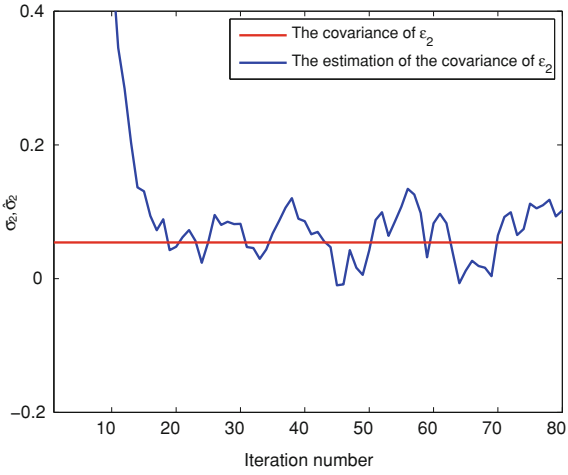


Fig. 12.21 Iterative updates of σ_2

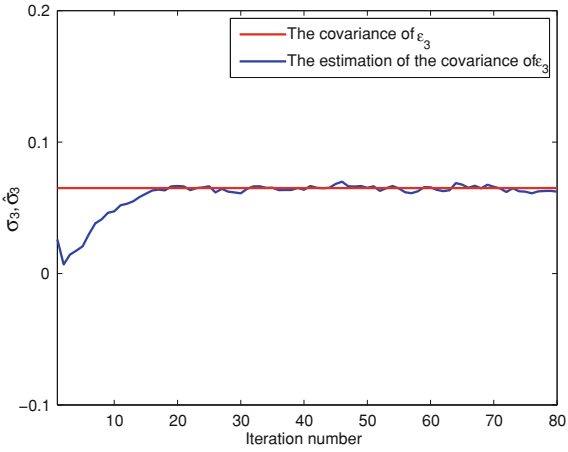


Fig. 12.22 Iterative updates of σ_3

learning the environment by iterative interactions between the system (estimator) and the environment (via measurements).

The LS (GLS), RLS and Kalman filter algorithms, (12.2), (12.11)–(12.13) and (12.22)–(12.24), are standard. For a more systematic and detailed description of these algorithms, the reader is, for instance, referred to [5].

The IGLS is fundamental for the study in this chapter. It was proposed by Goldstein [1–3].

Roughly speaking, Kalman filter and the LS algorithm are based on the same principle and applied for different process types. There were very interesting publications on their relationships in the 1970s [8] or more recently [4, 5].

Recently, application of reinforcement learning to dealing with optimal control issues has received considerable attention [6, 7]. It has been demonstrated that by

means of reinforcement learning methods real-time optimization of feedback control systems can be successfully achieved. The core of reinforcement learning methods is iterative interactions between the control system and environment and real-time evaluation of the system/control performance. Inspirited by these ideas, we have modified the IGLS algorithm and introduced an time interval for updating knowledge of the environment.

References

1. Goldstein H (1999) Multilevel statistical models. Institute of education, multilevel models project, London. http://www.arnoldpublishers.com/support/goldstein.htm
2. Goldstein H (1986) Multilevel mixed linear model analysis using iterative generalized least squares. Biometrika 73:43–56
3. Goldstein H (1989) Restricted unbiased iterative generalized least-squares estimation. Biometrika 76:622–623
4. Humpherys J, West J (2010) Kalman filtering with newton's method. IEEE Control Syst Mag 30(6):101–106
5. Kailath T, Sayed A, Hassibi B (1999) Linear estimation. Prentice Hall, New Jersey
6. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circuits Syst Mag 9(3):32–50
7. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control. IEEE Control Syst Mag 32(6):76–105
8. Sorenson HW (1970) Least-squares stimation: from gauss to kalman. IEEE Spectr 7:63–68

Part V
Data-driven Design and Lifetime Management of Fault-tolerant Control Systems

Chapter 13
Fault-Tolerant Control Architecture and Design Issues

In order to meet high requirements on system performance, more and more feedback control loops with embedded sensors, actuators, controllers as well as microprocessors are integrated into automatic control systems. In parallel to this development, fault-tolerant control technique is receiving considerably increasing attention, both in the research and application domains. Also, a new trend of integrating model-based FDI into automatic control systems can be observed. The research activities in these fields are strongly driven by the enhanced industrial demands for system reliability and availability, in particular when the systems are embedded in safety relevant plants like aeroplanes, vehicles or robots.

In this chapter, we present a framework for the design and implementation of fault-tolerant control systems and for the lifetime management of automatic control systems. The core of this framework is a fault-tolerant control architecture and parametrizations of the control, monitoring and detection systems/units integrated in this architecture. In this framework, both model-based and data-driven system design and implementation can be realized. For our purpose, we shall, in this chapter, deal with the model-based issues, while their data-driven realizations will be addressed in the next two chapters.

13.1 Preliminaries

We first introduce preliminary knowledge needed for the model-based controller configuration and design, which builds the basis for our fault-tolerant control architecture and the relevant study. We consider the nominal model (8.2)–(8.3), which is given below:

$$x(k+1) = Ax(k) + Bu(k), x(0) = x_0 \tag{13.1}$$

$$y(k) = Cx(k) + Du(k). \tag{13.2}$$

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_13, © Springer-Verlag London 2014

13.1.1 Image Representation and State Feedback Control

Recall that in Chap. 8 we have presented the concepts of LCF, RCF as well as SKR. where RCF is defined as follows. Let $G_{yu}(z) = C(zI - A)^{-1}B + D$. The pair $(M(z), N(z))$,

$$M(z) = I + F(zI - A_F)^{-1}B, N(z) = D + C_F(zI - A_F)^{-1}B$$
$$A_F = A + BF, C_F = C + DF$$

builds the RCF of $G_{yu}(z)$ with

$$G_{yu}(z) = N(z)M^{-1}(z)$$

where F is a matrix of appropriate dimension. It is well-known that the interpretation of RCF is state feedback control with

$$x(k+1) = (A + BF)x(k) + B\upsilon(k), y(k) = (C + DF)x(k) + D\upsilon(k)$$
$$u(k) = Fx(k) + \upsilon(k) \Longrightarrow u(z) = M(z)\upsilon(k), y(z) = N(z)\upsilon(k) \quad (13.3)$$

$\upsilon(k)$ being the reference vector. Alternatively, the nominal model (13.1)–(13.2) can be represented by

$$\text{for some } \upsilon, \begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = \begin{bmatrix} M(z) \\ N(z) \end{bmatrix} \upsilon(z). \quad (13.4)$$

As a dual form of the SKR introduced in Definition 8.3, (13.4) is called (stable) image representation of system (13.1)–(13.2).

Definition 13.1 Given system (13.1)–(13.2) then a stable linear system $\mathcal{I}$ is called stable image representation (SIR) of (13.1)–(13.2) if for any $u(z)$ and its response $y(z)$ a (reference) input $\upsilon(z)$ can be found such that

$$\begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = \mathcal{I}\upsilon(z). \quad (13.5)$$

Suppose that $\upsilon(z)$ is driven by a reference signal $y_{ref}(z)$ to be followed by the plant output $y(z)$. Let

$$\upsilon(z) = T(z)y_{ref}(z) \Longrightarrow N(z)\upsilon(z) = N(z)T(z)y_{ref}(z) \quad (13.6)$$

$T(z)$ is a pre-filter, also called feed-forward controller, which can be applied to achieve the desired tracking behavior. It yields

$$\begin{bmatrix} y(z) \\ u(z) \end{bmatrix} = \begin{bmatrix} N(z)T(z) \\ M(z)T(z) \end{bmatrix} y_{ref}(z).$$

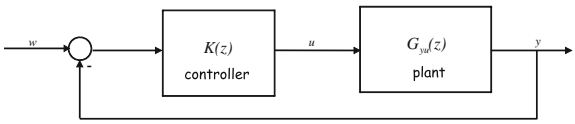


Fig. 13.1 Feedback control loop

Thus, by means of SIR we are able to describe the system tracking behavior, $N(z)T(z)y_{ref}(z)$, and the dynamics of the actuator, $M(z)T(z)y_{ref}(z)$, in a compact form.

13.1.2 Parametrization of Stabilizing Controllers

Consider the feedback control loop sketched in Fig. 13.1. The so-called Youla parameterization described by

$$K(z) = \left(\hat{X}(z) - Q_c(z)\hat{N}(z)\right)^{-1}\left(\hat{Y}(z) - Q_c(z)\hat{M}(z)\right) \tag{13.7}$$

parameterizes all stabilizing controllers, where $Q_c(z)$ is the stable parameter matrix, $\hat{M}(z)$, $\hat{N}(z)$ build the LCF of $G_{yu}(z)$, and $\hat{X}(z)$, $\hat{Y}(z)$ are stable and defined by

$$\hat{X}(z) = I - F(zI - A_L)^{-1}B_L, \hat{Y}(z) = F(zI - A_L)^{-1}L \tag{13.8}$$
$$A_L = A - LC, B_L = B - LD$$

with F, L being matrices of appropriate dimensions and ensuring that $A_F = A + BF$, A_L are Schur matrices.

For our study, the observer-based realization of the above Youla parameterization of all stabilizing controllers plays a central role. It is given by

$$u(z) = F\hat{x}(z) + Q(z)r_o(z), Q(z) = -Q_c(z) \tag{13.9}$$
$$r_o(z) = y(k) - \hat{y}(k) \tag{13.10}$$

where $\hat{x}(k)$, $\hat{y}(k)$ are estimates of the state and output variables, respectively, which are delivered by an observer

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L\left(y(k) - \hat{y}(k)\right), \hat{y}(k) = C\hat{x}(k) + Du(k). \tag{13.11}$$

It is worth to mention the following two facts:

- $u(z)$ satisfying (13.9) is in fact a functional observer for $Fx(k)$
- the selection of $Q(z)$, so far it is stable, has no influence on the stability of the closed-loop system. In other words, the stability of the overall system is determined by F and L.

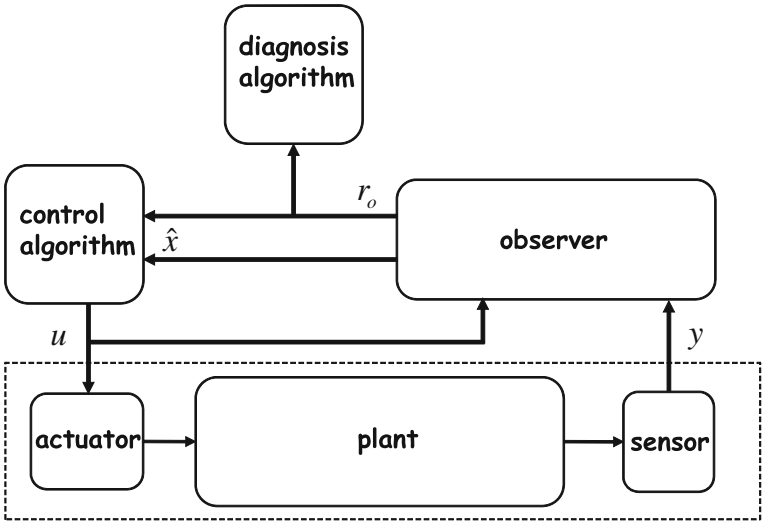


Fig. 13.2 Basic configuration of the fault-tolerant control architecture

13.2 Fault-Tolerant Control Architecture and Relevant Issues

13.2.1 An Observer-Based Fault-Tolerant Control Architecture

In Fig. 13.2, the basic configuration of the fault-tolerant control system architecture is sketched. It is composed of three functional units: (i) an observer that delivers a state estimation, $\hat{x}$, and the preliminary residual, r_o, (ii) control algorithm (law) (iii) fault detection and diagnosis algorithm. It should be emphasized that the above-mentioned three functional units are the basic components. For a successful fault-tolerant control, further functional units like system re-configuration and adaptation will be integrated into the architecture shown in Fig. 13.2.

Next, the realization of the fault-tolerant control system architecture in the model-based framework is addressed, which serves as the basis for our data-driven design study in the sequel. Consider the plant model (13.1)–(13.2). We propose the following realization scheme:

- observer

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) + Bu(k) + Lr_o(k) \\ r_o(k) &= y(k) - \hat{y}(k),\ \hat{y}(k) = C\hat{x}(k) + Du(k) \end{aligned} \tag{13.12}$$

- control law

$$u(z) = F\hat{x}(z) + Q(z)r_o(z) + \upsilon(z) \tag{13.13}$$

- diagnosis algorithm

$$r(z) = R(z)r_o(z) \tag{13.14}$$

where $Q(z), R(z)$ are stable transfer matrices and serve as the design parameters of the controller and detection system. Remember that for $\upsilon(z) = 0, u(z) = F\hat{x}(z) + Q(z)\left(y(z) - \hat{y}(z)\right)$ is the observer-based realization of the well-known Youla parameterization of all stabilizing controllers. For $Q(z) = 0$, we then have a standard observer-based state feedback control. Moreover, $R(z)$ is a post-filter, $r(z) = R(z)r_o(z)$ forms the parametrization of all LTI residual generators, as described in Chap. 8.

In order to gain a deeper insight into the above control configuration, we extend the plant model (13.1)–(13.2) to

$$x(k+1) = Ax(k) + Bu(k) + E_\alpha \alpha(k) \tag{13.15}$$
$$y(k) = Cx(k) + Du(k) + F_\alpha \alpha(k) \tag{13.16}$$

where $\alpha(k)$ represents a unknown input vector in the process and sensor models and can be interpreted as process and sensor faults or disturbances/noises. We have the following system dynamics, expressed in terms of the SIR and SKR,

$$\begin{bmatrix} y(z) \\ u(z) \\ r(z) \end{bmatrix} = \begin{bmatrix} T_1(z) & N(z) & N(z)Q(z) \\ 0 & M(z) & M(z)Q(z) + \hat{Y}(z) \\ 0 & 0 & R(z) \end{bmatrix} \begin{bmatrix} \alpha(z) \\ \upsilon(z) \\ r_o(z) \end{bmatrix} \tag{13.17}$$

$$r_o(z) = T_\alpha(z)\alpha(z), T_\alpha(z) = F_\alpha + C\left(zI - A_L\right)^{-1} E_{\alpha,L}, E_{\alpha,L} = E_\alpha - LF_\alpha \tag{13.18}$$

$$T_1(z) = F_\alpha + \begin{bmatrix} C + DF & -DF \end{bmatrix} \begin{bmatrix} zI - A_F & -BF \\ 0 & zI - A_L \end{bmatrix}^{-1} \begin{bmatrix} E_\alpha \\ E_{\alpha,L} \end{bmatrix} \tag{13.19}$$
$$\hat{Y}(z) = F\left(zI - A_F\right)^{-1} L.$$

In (13.17), the dynamics of $y(z)$ is known from the Youla parameterization, the residual dynamics of $r(z)$ is given by the parametrization of residual generators. Let

$$\upsilon(z) = T(z)y_{ref}(z) \tag{13.20}$$

$T(z)$ is feed-forward controller for achieving the desired tracking behavior. Then, we finally have

$$\begin{bmatrix} y(z) \\ u(z) \\ r(z) \end{bmatrix} = \begin{bmatrix} T_1(z) + N(z)Q(z)T_\alpha(z) & N(z)T(z) \\ \left(M(z)Q(z) + \hat{Y}(z)\right)T_\alpha(z) & M(z)T(z) \\ R(z)T_\alpha(z) & 0 \end{bmatrix} \begin{bmatrix} \alpha(z) \\ y_{ref}(z) \end{bmatrix}. \tag{13.21}$$

It is evident that five parameters are available in the above fault-tolerant control architecture for the design objectives, and they have different functionalities, as summarized below:

- F, L determine the stability and eigen-dynamics of the closed-loop
- $Q(z), R(z), T(z)$ have no influence on the system stability, and
- $Q(z)$ is used to enhance the system robustness and fault-tolerant performance
- $R(z)$ to optimize the fault detectability
- $T(z)$ to optimize the tracking behavior.

In the context of fault-tolerant control of the overall control loop, these five parameters have to be, due to their different functionalities, treated with different priorities. Recall that system stability is the minimum requirement on an automatic control system. This requires that a real-time adaptation of F, L to the possible changes in the system eigen-dynamics, possibly caused by faults, is needed to guarantee the overall system stability. For this reason, we call F, L H-PRIO (high-priority) parameter whose adaptation has the highest priority. Differently, $Q(z), R(z), T(z)$ are used to optimize control or FDI performance. In case that a temporary system performance degradation is tolerable, the real-time demand and the priority for the optimization of $Q(z), R(z), T(z)$ are relatively low. These three parameters are called L-PRIO (low-priority) parameter.

13.2.2 Design and Optimal Settings

In this section, we study the design and implementation of the fault-tolerant system (13.17)–(13.18). At first, we present the well-known LQG (Linear Quadratic Gaussian) and $\mathcal{H}_2$ optimal control schemes.

Theorem 13.1 (LQG optimal control) *Given system model*

$$x(k+1) = Ax(k) + Bu(k) + \eta(k), y(k) = Cx(k) + \nu(k) \tag{13.22}$$

where $\eta \sim \mathcal{N}(0, \Sigma_\eta), \nu \sim \mathcal{N}(0, \Sigma_\nu)$ *are white noise sequence and independent, and assume that the system is stabilizable and detectable, then the control law*

$$u(k) = F_{LQG}\hat{x}(k), F_{LQG} = -\left(B^T Y B + R\right)^{-1} B^T Y A \tag{13.23}$$

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L_{Kal} r_o(k), r_o(k) = y(k) - C\hat{x}(k) \tag{13.24}$$

$$L_{Kal} = APC^T \left(CPC^T + \Sigma_\nu\right)^{-1} \tag{13.25}$$

minimizes the cost function

$$J = \lim_{N\to\infty} \frac{1}{N}\mathcal{E}\left(\sum_{k=0}^{N}\left(x^T(k)Qx(k) + u^T(k)Ru(k)\right)\right), R > 0, Q \geq 0$$

where $P \geq 0, Y \geq 0$ *are, respectively, the solutions of the Riccati equations*

$$P = APA^T - APC^T\left(CPC^T + \Sigma_v\right)^{-1} CPA^T + \Sigma_\eta \tag{13.26}$$

$$Y = A^T YA - A^T YB\left(B^T YB + R\right)^{-1} B^T YA + Q. \tag{13.27}$$

Theorem 13.2 ($\mathcal{H}_2$ optimal control) *Given system model*

$$x(k+1) = Ax(k) + Bu(k) + E_d d(k), y(k) = Cx(k) + F_d d(k) \tag{13.28}$$

$$z(k) = C_z x(k) + D_z u(k) + F_z d(k) \tag{13.29}$$

where $d \in \mathcal{R}^{k_d}$ *is (unknown) disturbance vector,* $z \in \mathcal{R}^{k_z}$ *denotes the control output (objective), and assume that*

- (A, B) *is stabilizable and* (C, A) *is detectable*
- F_z *is full column rank and* $F_z^T F_z = I$, F_d *is full row rank and* $F_d F_d^T = I$
- *for all* $\theta \in [0, 2\pi]$

$$rank\begin{bmatrix} A - e^{j\theta} I & B \\ C_z & D_z \end{bmatrix} = n + l$$

- *for all* $\theta \in [0, 2\pi]$

$$rank\begin{bmatrix} A - e^{j\theta} I & E_d \\ C & F_d \end{bmatrix} = n + m$$

then the control law

$$u(k) = F_2\hat{x}(k) + L_o r_o(k) \tag{13.30}$$

$$F_2 = -\left(B^T YB + I\right)^{-1}\left(B^T YA + D_z^T C_z\right) \tag{13.31}$$

$$L_o = \left(F_2 PC^T + F_o F_d^T\right)\left(CPC^T + I\right)^{-1} \tag{13.32}$$

$$F_o = -\left(B^T YB + I\right)^{-1}\left(B^T YE_d + D_z^T F_z\right) \tag{13.33}$$

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L_2 r_o(k), r_o(k) = y(k) - C\hat{x}(k) \tag{13.34}$$

$$L_2 = \left(APC^T + E_d F_d^T\right)\left(CPC^T + I\right)^{-1} \tag{13.35}$$

minimizes the $\mathcal{H}_2$*-norm of the transfer matrix from the disturbance vector* d *to the control output* z*, where* $P \geq 0, Y \geq 0$ *are respectively the solutions of the Riccati equations*

$$P = APA^T + E_d E_d^T - L_2 \left(CPC^T + I \right) L_2^T \tag{13.36}$$

$$Y = A^T YA + C^T C - F_2^T \left(B^T YB + I \right) F_2. \tag{13.37}$$

LQG and $\mathcal{H}_2$ control schemes belong to the basic ingredients of optimal and robust control theory. The results given in the above two theorems and their proof can be found in standard textbooks of advanced control theory. Some of them are listed at the end of this chapter.

It is clear that

- both control schemes presented in Theorems 13.1–13.2 are observer-based state feedback control
- the separation principle holds, and
- from the Youla parametrization viewpoint, we have $Q(z) = 0$ in the LQG control, while for $\mathcal{H}_2$ scheme $Q(z) = L_o$.

It is of practical interest to note that both observers, the Kalman filter in the LQG control and the $\mathcal{H}_2$ optimal observer in the $\mathcal{H}_2$ configuration, deliver a residual vector which allows the maximum fault detectability. Without proof, we give the following results for the optimal design of the detection algorithms. The reader is referred to the literatures given in the chapter end.

Theorem 13.3 *Given system model* (13.22) *and the fault-tolerant architecture* (13.12)–(13.14) *with an LQG controller* (13.23) *and Kalman filter* (13.24)–(13.25) *(as the observer), then setting*

$$R(z) = \left(CPC^T + \Sigma_v \right)^{-1/2} \tag{13.38}$$

generates a residual vector satisfying

$$r(z) = R(z) r_o(z), \, r(k) \sim \mathcal{N}(0, I)$$

and thus T^2 *test statistic*

$$J_{T^2} = r^T(k) r(k) \sim \chi^2(m) \tag{13.39}$$

with the threshold

$$J_{T^2,th} = \chi_\alpha \tag{13.40}$$

can be applied for an optimal fault detection.

Remark 13.1 The above detection problem has been studied in Sect. 3.2, and for the threshold setting Algorithm 3.2 is available.

Theorem 13.4 *Given system model* (13.28) *and the fault-tolerant architecture* (13.12)–(13.14) *with an* $\mathcal{H}_2$ *controller* (13.30) *and* $\mathcal{H}_2$ *observer* (13.34)–(13.35)*, then setting*

$$R(z) = \left(CPC^T + I\right)^{-1/2} \tag{13.41}$$

generates a residual vector satisfying, for all $\theta \in [0, 2\pi]$,

$$H_{rd}(e^{j\theta})H_{rd}^T(e^{-j\theta}) = I \tag{13.42}$$

and thus for the residual evaluation function

$$J = \|r\|_2^2 \tag{13.43}$$

the threshold is set to be

$$J_{th} = \delta_d := \sup \|d\|_2^2 \tag{13.44}$$

where $H_{rd}(z)$ *denotes the transfer function matrix from the disturbance vector* d *to the residual vector* r*, and it is assumed that* d *is* l_2*-bounded with the boundedness equal to* δ_d.

Remark 13.2 The above-described fault detection scheme is the so-called unified solution that delivers an optimal fault detectability. The property (13.42) means that $H_{rd}(z)$ is a co-inner (an all-pass).

13.2.3 A Residual-Based Fault-Tolerant and Lifetime Management Structure

To meet high industrial demands, great engineering efforts are made to achieve optimal design and implementation of automatic control systems. On the other hand, the optimal settings have rarely a guarantee for an optimal operation during the lifetime of the automatic control system. There is a number of reasons for this. Below are three typical scenarios:

- changes in the operation conditions and in the environment around the process
- failure of some system components or faults in the process
- component or device replacement in a maintenance or repair action.

It is well-known that robust, adaptive and fault-tolerant control techniques are powerful tools to deal with the possible performance degradation in the first two scenarios. Indeed, development of advanced robust, adaptive and fault-tolerant control schemes build the mainstream in the research field of control theory. In comparison, only few research efforts have been made to integrate controller optimization into the maintenance plan. In industrial practice, automatic calibration, start and tuning of automatic control systems after a maintenance or repair action are of considerable interest.

It can be recognized that adaptive and fault-tolerant control schemes have one action in common: the parameters or even the configuration of the running automatic control system will be updated to fit the above-mentioned changes. The basic requirement on the control system during the adaptation or re-configuration is the stability guarantee. Next, we introduce some theoretic results for developing an alternative scheme for a lifetime management of automatic control systems.

Theorem 13.5 *Given a control loop with the plant model* (13.1)–(13.2) *and a controller* $u_o(k)$ *which stabilizes the control loop, then all controllers which stabilize the control loop can be parametrized by*

$$u(z) = u_o(z) + Q(z)r(z) \tag{13.45}$$

where $Q(z)$ *is the stable parametrization matrix and* $r(z)$ *is the residual vector as defined in* (13.11).

Proof It follows from the observer-based Youla parametrization (13.9) that $u_o(z)$ can be written into

$$u_o(z) = F_o\hat{x}(k) + Q_o(z)r(z)$$

with some state feedback gain matrix F_o that ensures the closed-loop stability, and some stable parameter matrix $Q_o(z)$. On the other hand, we can write the observer-based Youla parametrization (13.9) as

$$u(z) = F_o\hat{x}(k) + \bar{Q}(z)r(z)$$

with $\bar{Q}(z)$ as the (stable) parametrization matrix. Let $\bar{Q}(z) = Q(z) + Q_o(z)$, where $Q(z)$ denotes any (arbitrarily selectable) stable matrix. We have finally (13.45).

Although (13.45) is an alternative form of the well-known Youla parametrization, it is of considerable practical interests. It allows us

- to update the controller for some performance optimization objective without changing the existing controller ($u_o(k)$) and thus with a stability guarantee.
- to find an optimal controller by updating $Q(z)$ as far as an optimum one exists.

These two properties are useful for a practical application of the Youla parametrization and make the major difference to the most adaptive or fault-tolerant control schemes.

It is also interesting to notice that a residual signal can also be integrated into feed-forward controller, for instance, to compensate disturbances, without affecting the system stability, as described in the following corollary.

Corollary 13.1 *Given plant model* (13.1)–(13.2) *and the fault-tolerant architecture* (13.17)*, then an integration of the residual vector* $r(z)$ *into the feed-forward controller*

$$\upsilon(z) = T(z)y_{ref}(z) + V(z)r(z) \tag{13.46}$$

does not affect the system stability.

Proof Since

$$u(z) = F\hat{x}(k) + Q(z)r(z) + \upsilon(z) = F\hat{x}(k) + \bar{Q}(z)r(z) + T(z)y_{ref}(z)$$

and $\bar{Q}(z) = Q(z)+V(z)$ is stable, $F\hat{x}(k)+\bar{Q}(z)r(z)$ is a Youla stabilizing controller that ensures the system stability.

Analogous to Theorem 13.4, we have the following result for the update of residual signals.

Theorem 13.6 *Given the plant model* (13.1)–(13.2) *and a residual vector* $\bar{r}(k)$*, then all residual generators can be parametrized by*

$$r(z) = \bar{r}(z) + R(z)r(z) \tag{13.47}$$

where $R(z)$ *is the stable parametrization matrix and* $r(z)$ *is the residual vector as defined in* (13.11)*.*

This result follows immediately from the definition of a residual signal.

13.2.4 System Dynamics and Design Parameters

In this subsection, we briefly study relations between the design parameters and system dynamics in the fault-tolerant architecture (13.17). Our objective is to demonstrate that by tuning L-PRIO parameters required optimal system performance, for instance as defined in LQG or $\mathcal{H}_2$ schemes, can be approached.

Lemma 13.1 *Given any two realizations of SIR and SKR,*

$$\begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = \begin{bmatrix} M_1(z) \\ N_1(z) \end{bmatrix} \upsilon_1(z) = \begin{bmatrix} M_2(z) \\ N_2(z) \end{bmatrix} \upsilon_2(z)$$
$$r_1(z) = \hat{M}_1(z)y(z) - \hat{N}_1(z)u(z), r_2(z) = \hat{M}_2(z)y(z) - \hat{N}_2(z)u(z)$$
$$M_i(z) = \left(A_{F_i}, B, F_i, I\right), N_i(z) = \left(A_{F_i}, B, C + DF_i, D\right)$$
$$\hat{M}_i(z) = \left(A_{L_i}, -L_i, C, I\right), \hat{N}_i(z) = \left(A_{L_i}, B - L_iD, C, D\right)$$
$$A_{F_i} = A + BF_i, A_{L_i} = A - L_iC, i = 1, 2$$

it holds

$$\begin{bmatrix} M_2(z) \\ N_2(z) \end{bmatrix} = \begin{bmatrix} M_1(z) \\ N_1(z) \end{bmatrix} Q_{SIR}(z) \Longrightarrow \tag{13.48}$$

$$\upsilon_1(z) = Q_{SIR}(z)\upsilon_2(z),\ Q_{SIR}(z) = I + (F_2 - F_1)\left(zI - A_{F_2}\right)^{-1} B$$

$$\left[\hat{M}_2(z)\ \hat{N}_2(z)\right] = Q_{SKR}(z)\left[\hat{M}_1(z)\ \hat{N}_1(z)\right] \Longrightarrow \tag{13.49}$$

$$r_2(z) = Q_{SKR}(z)r_1(z),\ Q_{SKR}(z) = I - C\left(zI - A_{L_2}\right)^{-1}(L_2 - L_1).$$

The proof is straightforward computations and thus omitted here.

Equations (13.48)–(13.49) reveal that any change in the system dynamics caused by tuning the state feedback gain and observer gain can be equivalently expressed by a pre-filter added to $\upsilon(z)$ and a post-filter of $r(z)$. A further result of this lemma is that the residual performance can be achieved, independent of the observer gain matrix L, by tuning the post-filter $R(z)$.

The following theorem illustrates that once the system stability is ensured by a right state feedback gain F and observer gain L the tracking, detection and robustness performances can be separately optimized by selecting pre-filter $T(z)$, post-filter $R(z)$ and Youla parametrization matrix $Q(z)$.

Theorem 13.7 *Given plant model* (13.15)–(13.16)*,* L_1, L_2, F *which ensure the stability of* A_{L_1}, A_{L_2} *and* A_F*, and a stable transfer matrix* $Q_1(z)$*, then it holds*

$$\left(M(z)Q_1(z) + \hat{Y}_1(z)\right) r_1(z) = \left(M(z)Q_2(z) + \hat{Y}_2(z)\right) r_2(z) \tag{13.50}$$

$$Q_2(z) = \left(Q_1(z) - F\left(zI - A_{L_2}\right)^{-1}(L_2 - L_1)\right) Q_{SKR}^{-1}(z) \tag{13.51}$$

$$Q_{SKR}^{-1}(z) = I - C\left(zI - A_{L_1}\right)^{-1}(L_1 - L_2)$$

$$\hat{Y}_1(z) = F(zI - A_F)^{-1}L_1,\ \hat{Y}_2(z) = F(zI - A_F)^{-1}L_2$$

where $r_1(z), r_2(z)$ *are residual vectors as defined in Lemma 13.1.*

Proof It follows from Lemma 13.1 that $r_2(z) = Q_{SKR}(z)r_1(z)$. Rewrite $\hat{Y}_2(z)$ $Q_{SKR}(z)$ into

$$\hat{Y}_2(z)Q_{SKR}(z) = F(zI - A_F)^{-1}\left(L_1 + \left(I - L_2C\left(zI - A_{L_2}\right)^{-1}\right)(L_2 - L_1)\right).$$

Note that

$$I - L_2C\left(zI - A_{L_2}\right)^{-1} = (zI - A)\left(zI - A_{L_2}\right)^{-1}$$

$$F(zI - A_F)^{-1}(zI - A) = \left(I + F(zI - A_F)^{-1}B\right)F.$$

It turns out

$$\hat{Y}_2(z)Q_{SKR}(z) = \hat{Y}_1(z) + M(z)F\left(zI - A_{L_2}\right)^{-1}(L_2 - L_1).$$

Hence, setting

$$Q_2(z) = \left(Q_1(z) - F\left(zI - A_{L_2}\right)^{-1}(L_2 - L_1)\right) Q_{SKR}^{-1}(z)$$

gives

$$M(z)Q_2(z)Q_{SKR}(z) = M(z)Q_1(z) - M(z)F\left(zI - A_{L_2}\right)^{-1}(L_2 - L_1)$$

which leads to (13.50).

Recall that the control input can be expressed by see (13.17)

$$u(z) = M(z)\upsilon(z) + \left(M(z)Q(z) + \hat{Y}(z)\right) r_o(z).$$

As a result of Theorem 13.7, we claim that the change in the system control performance caused by a change of observer gain can be fully compensated by tuning the (control) parameter matrix $Q(z)$.

Theorem 13.8 *Given plant model* (13.15)–(13.16)*,* F_1, F_2, L *which ensue the stability of* A_{F_1}, A_{F_2} *and* A_L*, and a stable transfer matrix* $Q_1(z)$*, then it holds*

$$M_1(z)Q_1(z) + \hat{Y}_1(z) = M_2(z)Q_2(z) + \hat{Y}_2(z) \tag{13.52}$$

$$Q_2(z) = Q_{SIR}^{-1}(z)\left(Q_1(z) + (F_1 - F_2)(zI - A_{F_2})^{-1}L\right) \tag{13.53}$$

$$Q_{SIR}^{-1}(z) = I + (F_1 - F_2)\left(zI - A_{F_1}\right)^{-1}B$$

$$\hat{Y}_1(z) = F_1(zI - A_{F1})^{-1}L, \hat{Y}_2(z) = F_2(zI - A_{F_2})^{-1}L$$

where $M_1(z)$, $M_2(z)$ *are defined in Lemma 13.1.*

Proof From Lemma 13.1 we have $M_2(z) = M_1(z)Q_{SIR}(z)$. Note that

$$\begin{aligned} &M_1^{-1}(z)\left(\hat{Y}_1(z) - \hat{Y}_2(z)\right) \\ &= \left(I - F_1(zI - A)^{-1}B\right)\left(F_1\left(zI - A_{F_1}\right)^{-1} - F_2(zI - A_{F_2})^{-1}\right)L \\ &= (F_1 - F_2)\left(zI - A_{F_2}\right)^{-1}L. \end{aligned}$$

It turns out

$$\begin{aligned} M_2(z)Q_2(z) + \hat{Y}_2(z) &= M_1(z)\left(Q_1(z) + (F_1 - F_2)(zI - A_{F_2})^{-1}L\right) + \hat{Y}_2(z) \\ &= M_1(z)Q_1(z) + \hat{Y}_1(z). \end{aligned}$$

Thus, (13.52) is proven.

An immediate result of Lemma 13.1 and Theorem 13.8 is that the change in the system control performance caused by a change of state feedback gain F can be fully compensated by tuning

- pre-filter $T(z)$, that is $T(z) = Q_{SIR}^{-1}(z) \Longrightarrow M_2(z)T(z)\upsilon(z) = M_1(z)\upsilon(z)$
- and (control) parameter matrix $Q(z)$, $Q(z) = Q_2(z) \Longrightarrow M_2(z)Q_2(z) + \hat{Y}_2(z) = M_1(z)Q_1(z) + \hat{Y}_1(z)$.

As a summary of Theorems 13.7–13.8, we have then the following corollary.

Corollary 13.2 *Given plant model* (13.15)–(13.16)*,* L_1, L_2, F_1, F_2 *which ensure the stability of* $A_{L_1}, A_{L_2}, A_{F_1}$ *and* A_{F_2}*, and a stable transfer matrix* $Q_1(z)$*, then it holds*

$$\begin{aligned} u(z) &= M_1(z)\upsilon(z) + \left(M_1(z)Q_1(z) + \hat{Y}_1(z)\right) r_1(z) \\ &= M_2(z)T(z)\upsilon(z) + \left(M_2(z)Q_2(z) + \hat{Y}_2(z)\right) r_2(z) \end{aligned} \tag{13.54}$$

$$Q_2(z) = \left(\bar{Q}_1(z) - F_2\left(zI - A_{L_2}\right)^{-1}(L_2 - L_1)\right) Q_{SKR}^{-1}(z) \tag{13.55}$$

$$\bar{Q}_1(z) = Q_{SIR}^{-1}(z)\left(Q_1(z) + (F_1 - F_2)(zI - A_{F_2})^{-1}L_1\right) \tag{13.56}$$

$$T(z) = Q_{SIR}^{-1}(z) \tag{13.57}$$

where $r_1(z), r_2(z), M_1(z), M_2(z)$ *are as defined in Lemma 13.1 and*

$$\hat{Y}_1(z) = F_1(zI - A_{F1})^{-1}L_1, \hat{Y}_2(z) = F_2(zI - A_{F2})^{-1}L_2.$$

Proof It is clear that (13.57) follows directly from Lemma 13.1. By Theorem 13.7 it holds

$$\begin{aligned} \left(M_2(z)Q_2(z) + \hat{Y}_2(z)\right) r_2(z) &= \left(M_2(z)\bar{Q}_1(z) + \tilde{Y}_2(z)\right) r_1(z) \\ \tilde{Y}_2(z) &= F_2(zI - A_{F2})^{-1}L_1. \end{aligned}$$

Now, applying Theorem 13.8 to $M_2(z)\bar{Q}_1(z) + \tilde{Y}_2(z)$ leads to (13.54)–(13.55).

Corollary 13.2 tells us that setting F, L will have no influence on the control performances, as long as an optimization of pre-filter $T(z)$ and parametrization matrix $Q(z)$ is followed. Considering moreover that the residual performance can be, independent of L, achieved by tuning the post-filter $R(z)$, the fault-tolerant system (13.21) can be designed in the following procedure:

- determining L for the residual generation and state estimation purpose
- determining F for the system stabilization
- determining $T(z)$ for given $N(z)$ (i.e. for given F) aiming at achieving the desired tracking performance

- determining $R(z)$ for given L for optimizing fault detection performance
- determining $Q(z)$ for given F, L for optimizing fault-tolerance or/and robustness.

In particular, the above results are useful for the implementation of the system configurations given in Theorems 13.5–13.6 and Corollary 13.1 in dealing with lifetime management of automatic control systems.

In the next two chapters, we shall study these issues in the data-driven design context.

13.3 Notes and References

This chapter is dedicated the introduction of preliminary knowledge of model-based controller and observer design, which is needed for our study on the fault-tolerant system architecture and lifetime management of automatic control systems. The addressed issues and the presented results are essential for our data-driven design study in the next two chapters.

As a dual form of the SKR presented in Chap. 8, we have introduced the definition stable image representation which is similar to the one given by [1] for nonlinear systems. SIR is a useful tool for the design of feedback and feed-forward controllers.

Youla parametrization of stabilizing controllers is essential in robust control and fault-tolerant controller design. Both the original form and its observer-based form can be found in [2, 3].

For a detailed study of LQG scheme and the proof of Theorem 13.1, the reader is referred to, for instance, [4–6]. In [2], extensive description of the $\mathcal{H}_2$ control scheme and the proof of Theorem 13.2 can be found. Note that the presentation form adopted here is slightly different from the one given in [2].

In Theorems 13.3–13.4, known results for an integrated design of observer-based fault detection systems have been presented. The reader is referred to [7] for more details and the proofs.

The study on fault-tolerant control architecture with an observer-based residual generator in its core, as shown in Fig. 13.2, has been initiated by [8] and extensively investigated in [9].

It is worth to mention that the study in this chapter is related to the integrated design of control and diagnostic systems, which has been initiated by Nett et al. in 1988 [10] and extensively studied in the recent decade, see for instance [11–13]. The main idea of the integrated design scheme is to formulate the design of the controller and FDI unit unified as a standard optimization problem, e.g. an $\mathcal{H}_\infty$ optimization problem, and then to solve it using the available tools and methods. In this context, the results given in Theorems 13.3–13.4 can be interpreted as integrated design solutions for $\mathcal{H}_2$ optimization.

References

1. Van der Schaft A (2000) L2—gain and passivity techniques in nonlinear control. Springer, London
2. Zhou K, Doyle J, Glover K (1996) Robust and optimal control. Prentice-Hall, Upper Saddle River, New Jersey
3. Anderson BDO (1998) From youla-kucera to identification, adaptive and nonlinear control. Automatica 34:1485–1506
4. Kucera V (1991) Analysis and design of discrete linear control systems. Prentice Hall, London
5. Chen G, Chen G, Hsu S-H (1995) Linear stochastic control systems. CRC Press, Boca Raton, FL
6. Hassibi B, Sayed AH, Kailath T (1999) Indefinite-quadratic estimation and control: a unified approach to H2 and H-Inf theories. SIAM studies in applied and numerical mathematics. SIAM, Philadelphia
7. Ding SX (2013) Model-based fault diagnosis techniques—design schemes, algorithms and tools, 2nd edn. Springer-Verlag, London
8. Zhou K, Ren Z (2001) A new controller architecture for high performance, robust, and fault-tolerant control. IEEE Trans Autom Control 46:1613–1618
9. Ding SX, Yang G, Zhang P, Ding E, Jeinsch T, Weinhold N, Schulalbers M (2010) Feedback control structures, embedded residual signals and feedcak control schemes with an integrated residual access. IEEE Trans Control Syst Tech 18:352–367
10. Nett CN, Jacobson C, Miller AT (1988) An integrated approach to controls and diagnostics. In: Proceedings of American control conference, pp 824–835
11. Niemann H, Stoustrup J (1997) Integration of control and fault detection: nominal and robust design. In: Proceedings of the 3rd IFAC symposium on SAFEPROCESS, vol 1. pp 341–346
12. Marcos A, Balas GJ (2005) A robust integrated controller/disgnosis aircraft application. Int J Robust Nonlinear Control 15:531–551
13. Ding SX (2009) Integrated design of feedback controllers and fault detectors. Annu Rev Control 33:124–135

Chapter 14
Data-Driven Design of Observer-Based Control Systems

In the previous chapter, we have introduced the fault-tolerant architecture and the associated design parameters. In this chapter, we focus on the data-driven design of the state feedback gain matrix, an H-PRIO parameter of the fault-tolerant architecture, and its application to the observer-based control schemes.

14.1 Problem Formulation

Remember that the SIR plays a key role in establishing the fault-tolerant architecture. As a dual form of the data-driven realization of SKR presented in Chap. 9, we first give the definition of the data-driven realization of SIR. Consider a system modelled by

$$x(k+1) = Ax(k) + Bu(k), x(0) = x_0 \tag{14.1}$$

$$y(k) = Cx(k) + Du(k) \tag{14.2}$$

and its corresponding I/O data model

$$Y_{k,s} = \Gamma_s X_{k-s} + H_{u,s} U_{k,s} \in \mathcal{R}^{(s+1)m \times N} \tag{14.3}$$

$$\Gamma_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^s \end{bmatrix} \in \mathcal{R}^{(s+1)m \times n}, H_{u,s} = \begin{bmatrix} D & 0 & & \\ CB & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ CA^{s-1}B & \dots & CB & D \end{bmatrix}$$

where $Y_{k,s}$, $U_{k,s}$ are I/O data sets defined in Sect. 8.3.

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_14, © Springer-Verlag London 2014

Definition 14.1 Given system (14.1), (14.2), matrix $\mathcal{I}_{d,s_1,s_2}$ is called a data-driven realization of the SIR, if for some integers s_1, s_2 it holds

$$\forall u_{s_2}(k), x_0, \exists v_{s_1}(k) \text{ s.t. } \begin{bmatrix} u_{s_2}(k) \\ y_{s_2}(k) \end{bmatrix} = \mathcal{I}_{d,s_1,s_2} v_{s_1}(k) \tag{14.4}$$

where the adopted data format $w_s(k)$ has been introduced in Sect. 8.3.

With Definition 14.1, our first and key problem to be addressed in the sequel is formulated as: Given the I/O data model (14.3), find the conditions for determining $\mathcal{I}_{d,s_1,s_2}$. Moreover, algorithms for identifying $\mathcal{I}_{d,s_1,s_2}$ by means of available process I/O data sets will be developed.

Our further study is the data-driven realizations of the observer-based state feedback term, $F\hat{x}(k)$, and feed-forward controller (pre-filter), $v(z) = T(z)y_{ref}(z)$, as given in the fault-tolerant control architecture (13.17).

14.2 Data-Driven Realization Form of the Image Representation

In this section, we study data-driven representation of the SIR. To this end, the noise-free form of the I/O data model (14.3) is considered. Motivated by the state feedback interpretation of the SIR, let

$$u(k) = Fx(k) + v(k)$$

and substitute it into (14.3). It leads to, after a straightforward computation,

$$\begin{bmatrix} u_s(k) \\ y_s(k) \end{bmatrix} = \begin{bmatrix} H_{u,v,s} & \Gamma_{s,u} \\ H_{y,v,s} & \Gamma_{s,y} \end{bmatrix} \begin{bmatrix} v_s(k) \\ x(k) \end{bmatrix} \tag{14.5}$$

$$\Gamma_{s,u} = \begin{bmatrix} F \\ FA_F \\ \vdots \\ FA_F^s \end{bmatrix} \in \mathcal{R}^{(s+1)l \times n}, \ \Gamma_{s,y} = \begin{bmatrix} C_F \\ C_F A_F \\ \vdots \\ C_F A_F^s \end{bmatrix} \in \mathcal{R}^{(s+1)m \times n} \tag{14.6}$$

$$H_{u,v,s} = \begin{bmatrix} I & 0 & & \\ FB & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ FA_F^{s-1}B & \dots & FB & I \end{bmatrix} \in \mathcal{R}^{(s+1)l \times (s+1)l} \tag{14.7}$$

$$H_{y,v,s} = \begin{bmatrix} D & 0 & & \\ C_F B & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ C_F A_F^{s-1}B & \dots & C_F B & D \end{bmatrix} \in \mathcal{R}^{(s+1)m \times (s+1)l}. \tag{14.8}$$

Now, set F so that all eigenvalues of $A_F = A + BF$ are zero, which means that the state feedback controller is a deadbeat controller. It results in

$$x(k) = \left[A_F^{n-1} B \ \ldots \ B \right] \begin{bmatrix} \upsilon(k-n) \\ \vdots \\ \upsilon(k-1) \end{bmatrix} \tag{14.9}$$

Substituting $x(k)$ in (14.5) by (14.9) yields

$$\begin{bmatrix} u_s(k) \\ y_s(k) \end{bmatrix} = \begin{bmatrix} \Gamma_{s,u}\Sigma_{n-1,F} & H_{u,\upsilon,s} \\ \Gamma_{s,y}\Sigma_{n-1,F} & H_{y,\upsilon,s} \end{bmatrix} \upsilon_{s+n}(k-n) \tag{14.10}$$

$$\upsilon_{s+n}(k-n) = \begin{bmatrix} \upsilon(k-n) \\ \vdots \\ \upsilon(k+s) \end{bmatrix}, \Sigma_{n-1,F} = \left[A_F^{n-1} B \ \ldots \ B \right]$$

As a result, we have the following theorem.

Theorem 14.1 *Given system* (14.1), (14.2) *and* $s \geq n$, *then*

$$\mathcal{I}_{d,s+n,s} = \begin{bmatrix} \Gamma_{s,u}\Sigma_{n-1,F} & H_{u,\upsilon,s} \\ \Gamma_{s,y}\Sigma_{n-1,F} & H_{y,\upsilon,s} \end{bmatrix} \in \mathcal{R}^{(l+m)(s+1)\times(s+1+n)l} \tag{14.11}$$

gives a data-driven realization of the SIR, where F *is selected so that all eigenvalues of* A_F *are zero.*

It is worth to emphasize that $\mathcal{I}_{d,s+n,s}$ is achieved for a special feedback gain, the deadbeat controller. The idea behind it is that in this way $\mathcal{I}_{d,s+n,s}$ can be identified independent of the initial value of the state vector. Notice that, as a result, $u_s(k)$, $y_s(k)$ are driven by $\upsilon_{s+n}(k-n)$, i.e. by $\upsilon(j)$, $j \in [k-n, k+s]$.

Recall that in the model-based framework SIR is related to the RCF as follows

$$\text{for some } \upsilon, \begin{bmatrix} u(z) \\ y(z) \end{bmatrix} = \begin{bmatrix} M(z) \\ N(z) \end{bmatrix} \upsilon(z).$$

In this sense,

$$\left[\Gamma_{s,u}\Sigma_{n-1,F} \ \ H_{u,\upsilon,s} \right], \left[\Gamma_{s,y}\Sigma_{n-1,F} \ \ H_{y,\upsilon,s} \right]$$

can be interpreted as a data-driven realization of the RCF.

14.3 An Identification Scheme for the Image Representation

14.3.1 A Brief Review of the I/O Data Set Model and Relevant Issues

For our purpose, we first briefly review the I/O data set model,

$$Y_{k,s} = \Gamma_s X_k + H_{u,s} U_{k,s} + H_{w,s} W_{k,s} + V_{k,s} \in \mathcal{R}^{(s+1)m \times N} \tag{14.12}$$

which is introduced in Sect. 8.3, and some relevant results. Rewrite (14.12) into

$$\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \begin{bmatrix} I & 0 \\ H_{u,s} & \Gamma_s \end{bmatrix} \begin{bmatrix} U_{k,s} \\ X_k \end{bmatrix} + \begin{bmatrix} 0 \\ H_{w,s} W_{k,s} + V_{k,s} \end{bmatrix}. \tag{14.13}$$

Let

$$Z_{k,s} = \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix}.$$

For the identification purpose, process data are typically collected and recorded in the following two I/O data sets:

$$Z_f = \begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix}, Z_p = \begin{bmatrix} U_{k-s_p-1,s_p} \\ Y_{k-s_p-1,s_p} \end{bmatrix} \tag{14.14}$$

where s_p is a large integer. By a QR-decomposition of the process data as follows

$$\begin{bmatrix} Z_p \\ U_{k,s} \\ Y_{k,s} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{14.15}$$

we have

$$\begin{bmatrix} U_{k,s} \\ Y_{k,s} \end{bmatrix} = \begin{bmatrix} R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} + \begin{bmatrix} 0 \\ R_{33} Q_3 \end{bmatrix}. \tag{14.16}$$

It is known that

$$R_{33} Q_3 = H_{w,s} W_{k,s} + V_{k,s}. \tag{14.17}$$

14.3.2 The Identification Scheme

Recall that $\mathcal{I}_{d,s+n,s}$ depends on the deadbeat controller matrix F. Thus, our study consists of two steps: identification of F and identification of the sub-matrices in $\mathcal{I}_{d,s+n,s}$.

14.3.2.1 On the Deadbeat Feedback Gain Matrix

For our purpose of identifying F, finding relations between the I/O data model and F plays a central role. The following theorem is given for this purpose.

Theorem 14.2 *Given system model* (14.3)*, let* $s \geq n$,

$$\Pi_s = \left[B \; AB \; \ldots \; A^s B \right] \in \mathcal{R}^{n \times (s+1)l}$$

and w_s *solve*

$$\Pi_s w_s = 0, w_s = \begin{bmatrix} w_{s,0} \\ \vdots \\ w_{s,s} \end{bmatrix} \in \mathcal{R}^{(s+1)l}, w_{s,s} \neq 0 \tag{14.18}$$

then matrices

$$F_c = \left[w_{s,0} \; \ldots \; w_{s,s-1} \right] \in \mathcal{R}^{l \times s} \tag{14.19}$$

$$T_c = \left[B \; AB \; \ldots \; A^{s-1} B \right] \begin{bmatrix} w_{s,1} & w_{s,2} & \ldots & w_{s,s-1} & w_{s,s} \\ w_{s,2} & \ldots & w_{s,s-1} & w_{s,s} & 0 \\ \vdots & & & 0 & \vdots \\ w_{s,s-1} & w_{s,s} & 0 & \vdots & \vdots \\ w_{s,s} & 0 & \ldots & 0 & 0 \end{bmatrix} \in \mathcal{R}^{n \times s} \tag{14.20}$$

solve the following equations

$$T_c A_c = A T_c + B F_c, T_c b_c = B w_{s,s} \tag{14.21}$$

$$A_c = \begin{bmatrix} 0 & 1 & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \\ 0 & \ldots & 0 & 0 \end{bmatrix} \in \mathcal{R}^{s \times s}, b_c = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{14.22}$$

This theorem is the dual result of the Parity vector based solution of Luenberger equations, which has been introduced in Sect. 10.2.1, and whose proof can be found in the reference given at the chapter end. Thus, the proof is omitted.

It is evident that for $s = n$ and $l = 1$, $rank\,(T_c) = n$. It follows from (14.21) that

$$A_c = T_c^{-1} (A + BF) T_c, F = F_c T_c^{-1}$$

and $u(k) = F_c x_c(k) = F x(k)$ is a deadbeat controller, where $x(k) = T_c x_c(k)$ with $x_c(k)$ denoting the state vector after a regular state transformation by T_c.

14.3.2.2 Identification of $H_{u,s}$, F_c

The basic idea behind the algorithm proposed below is to identify the null-matrix (vector) of Π_n using process I/O data sets for w_n. To this end, we first identify $H_{u,s}$, which is also essential for the identification of $\mathcal{I}_{d,s+n,s}$. In the SIM-framework, existing algorithms are available for this purpose. Below, we propose an alternative solution, which is, from the computational viewpoint, more effective. Recall that

$$\begin{aligned} Y_{k,s} &= \Gamma_s X_k + H_{u,s} U_{k,s} + H_{w,s} W_{k,s} + V_{k,s} \\ &\approx \left[\Gamma_s L_p \;\; H_{u,s} \right] \begin{bmatrix} Z_p \\ U_{k,s} \end{bmatrix} + H_{w,s} W_{k,s} + V_{k,s} \end{aligned} \tag{14.23}$$

where

$$\begin{aligned} L_p &= \left[L_{p,u} \;\; L_{p,y} \right], L_{p,u} = \left[A_K^{s_p} B_K \;\; \ldots \;\; B_K \right] \\ L_{p,y} &= \left[A_K^{s_p} K \;\; \ldots \;\; K \right], A_K = A - KC, B_K = B - KD. \end{aligned}$$

K denotes the Kalman filter gain matrix. It follows from the QR-decomposition (14.15) that

$$\begin{bmatrix} Z_p \\ U_{k,s} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}.$$

Considering moreover

$$\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} Q_2^T = \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix}$$

it holds

$$\begin{aligned} Y_{k,s} Q_2^T &= \left[\Gamma_s L_p \;\; H_{u,s} \;\; I \right] \begin{bmatrix} Z_p \\ U_{k,s} \\ R_{33} Q_3 \end{bmatrix} Q_2^T = H_{u,s} R_{22} \\ \Longrightarrow H_{u,s} &= Y_{k,s} Q_2^T R_{22}^- \end{aligned} \tag{14.24}$$

where R_{22}^- is the pseudo-inverse of R_{22}. This allows us to identify $H_{u,s}$ using the following algorithm.

Algorithm 14.1 *Identification of $H_{u,s}$*

S1: Collect data, build Z_p, $U_{k,s}$, $Y_{k,s}$ and do QR-decomposition (14.15)
S2: Form $Y_{k,s} Q_2^T$ and compute R_{22}^-
S3: Compute $H_{u,s}$ according to (14.24).

Having identified $H_{u,s}$ with $s \geq 2n$, we have

$$\left[\Theta_0 \ \ldots \ \Theta_n\right] = \Gamma_{n-1}\Pi_n \tag{14.25}$$

$$\Theta_i = H_{u,s}\left((n+1)m+1 : (2n+1)m, (n-i)l+1 : (n-i+1)l\right) \tag{14.26}$$

$i = 0, \ldots, n$. Since $rank\,(\Gamma_{n-1}) = n$, (14.25) allows us to determine w_n using the relation

$$\Pi_n w_n = 0 \Longleftrightarrow \left[\Theta_0 \ \ldots \ \Theta_n\right] w_n = 0.$$

To this end, doing an SVD of

$$\Gamma_{n-1}\Pi_n = \left[U_1 \ U_2\right]\begin{bmatrix}\Sigma_1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}V_1^T\\ V_2^T\end{bmatrix}$$

it holds

$$w_n \text{ is a column of } V_2,\ \Pi_n = \Gamma_{n-1}^{\dagger} U_1 \Sigma_1 V_1^T. \tag{14.27}$$

In summary, we have the following algorithm.

Algorithm 14.2 *Identification of deadbeat control gain matrix*

S1: Run Algorithm 14.1
S2: Form $\Gamma_{n-1}\Pi_n$ according to (14.25) and (14.26)
S3: Determine w_n e.g., by an SVD and using (14.27)
S4: Set F_c according to (14.19).

14.3.2.3 Identification of $\mathcal{I}_{d,s+n,s}$

With the identified $H_{u,s}$, F_c, we are now in a position to determine $\mathcal{I}_{d,s+n,s}$. For our purpose, we first introduce the following identity whose proof is straightforward: *for* $i = 1, \ldots$

$$\left[C_F A_F^i B \ \ldots \ C_F B \ D\right] = \left[CA^i B \ \ldots \ CB \ D\right] H_{u,v,i+1} \tag{14.28}$$

$$H_{u,v,i+1} = \begin{bmatrix} I & 0 & \ldots & 0\\ FB & I & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ FA_F^i B & \ldots & FB & I\end{bmatrix}.$$

It leads to

$$H_{y,v,s} = \begin{bmatrix} D & 0 & & \\ C_F B & \ddots & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ C_F A_F^{s-1} B & \dots & C_F B & D \end{bmatrix} = H_{u,s} H_{u,v,s} \tag{14.29}$$

$$\Longrightarrow \begin{bmatrix} H_{u,v,s} \\ H_{y,v,s} \end{bmatrix} = \begin{bmatrix} I \\ H_{u,s} \end{bmatrix} H_{u,v,s} \tag{14.30}$$

$$\Gamma_{s,y} \Sigma_{n-1,F} = \begin{bmatrix} \Gamma_{n-1,y} \Sigma_{n-1,F} \\ 0 \end{bmatrix} = \begin{bmatrix} H_{y,v,s}\,(nm+1:2nm, 1:nl) \\ 0 \end{bmatrix}$$

$$H_{y,v,s}\,(nm+1:2nm, 1:nl) =$$

$$H_{u,s}\,(nm+1:2nm, 1:2nl)\, H_{u,v,2n-1}(:, 1:nl) \tag{14.31}$$

$$\Gamma_{s,u} \Sigma_{n-1,F} = \begin{bmatrix} \Gamma_{n-1,u} \Sigma_{n-1,F} \\ 0 \end{bmatrix} = \begin{bmatrix} H_{u,v,s}\,(nl+1:2nl, 1:nl) \\ 0 \end{bmatrix}. \tag{14.32}$$

Thus, if we are able to determine $H_{u,v,s}$, the problem of identifying

$$\mathcal{I}_{d,s+n,s} = \begin{bmatrix} \Gamma_{s,u} \Sigma_{n-1,F} & H_{u,v,s} \\ \Gamma_{s,y} \Sigma_{n-1,F} & H_{y,v,s} \end{bmatrix}$$

is solved. Considering the definition of $H_{u,v,s}$ given in (14.7) and $A_F^i = 0$, $i \geq n$, the remaining problem for our study is the determination of $F A_F^i B$, $i = 0, \dots, n-1$. To simplify our study, we only consider the case with $l = 1$. For the multiple input case, the dual form of the study in Sect. 10.2.2 can be directly applied. Remember that according to (14.19)–(14.22)

$$FB = F_c T_c^{-1} B = F_c b_c / w_{n,n} = w_{n,n-1} / w_{n,n} \tag{14.33}$$

$$F A_F^i B = F_c T_c^{-1} A_F^i T_c T_c^{-1} B = w_{n,n-1-i} / w_{n,n}, i = 1, \dots, n-1. \tag{14.34}$$

As a result, by (14.29)–(14.34) $\mathcal{I}_{d,s+n,s}$ can be identified. The needed computations are summarized in the following algorithm.

Algorithm 14.3 *Identification of* $\mathcal{I}_{d,s+n,s}$

S1: Run Algorithm 14.2 with outputs: F_c, $H_{u,s}$
S2: Compute $H_{y,v,s}$, $\Gamma_{s,y}\Sigma_{n-1,F}$, $H_{u,v,s}$, $\Gamma_{s,u}\Sigma_{n-1,F}$ *using (14.29)-(14.34)*
S3: Form $\mathcal{I}_{d,s+n,s}$ *according to (14.11) with* $s = n$.

At the end of this section, we would like to remark that the key computations for the data-driven realization $\mathcal{I}_{d,s+n,s}$ consist of running (14.24) on the basis of the QR decomposition of the data sets Z_p, Z_f. Neither Γ_s nor Π_s need to be identified, which will be helpful in the data-driven design of an observer-based feedback controller.

Moreover, in comparison with the standard SIM algorithms, computation of (14.24) is more efficient.

14.4 A Data-Driven Design Scheme of Observer-Based Control Systems

In this section, the data-driven kernel and image representations are applied to the design of observer-based control systems, which is a part of the fault-tolerant architecture introduced in Chap. 13. We assume that

- by means of the data-driven realization of the SKR a state observer is constructed, as described in Sects. 10.2.2 and 10.4
- the data-driven realization of the SIR $\mathcal{I}_{d,s+n,s}$ is available.

14.4.1 Data-Driven Design of Feed-Forward Controller

In this subsection, we study the design of $\upsilon(z)$. Recalling the interpretation of the SIR, it is clear that $N(z)$ or its data realization form

$$\mathcal{N}_{d,s} = \left[\Gamma_{s,y}\Sigma_{n-1,F} \;\; H_{y,\upsilon,s} \right]$$

describes the (closed-loop) transfer behavior from the (external) input $\upsilon(k)$ to the plant output $y(k)$. In the model-based framework, setting the feed-forward controller

$$\upsilon(z) = T(z)y_{ref}(z),\, T(z) = N^{-1}(z) \tag{14.35}$$

yields, in the disturbance-free case,

$$y(z) = y_{ref}(z).$$

It is clear that in practice only few (feed-forward) controllers satisfying (14.35) are real-time realizable in sense of causality and stability. Instead, the data-driven realization of the SIR offers the possibility for an alternative solution in the data-driven framework. To this end, set $s = \gamma$ and assume that

$$\mathcal{N}_{d,\gamma} = \left[\Gamma_{\gamma,y}\Sigma_{n-1,F} \;\; H_{y,\upsilon,\gamma} \right] \in \mathcal{R}^{m(\gamma+1)\times(\gamma+1+n)l} \tag{14.36}$$

is right invertible. Now, setting

$$\upsilon_{\gamma+n}(k-n) = \mathcal{N}_{d,\gamma}^{-}y_{ref,\gamma}(k),\, \mathcal{N}_{d,\gamma}\mathcal{N}_{d,\gamma}^{-} = I \tag{14.37}$$

we have, in the disturbance-free case and after the calibration phase,

$$y_\gamma(k) = y_{ref,\gamma}(k).$$

Note that $\gamma \geq 1$ represents the (feed-forward) control interval length and, if

$$\frac{nl}{\gamma + 1} \geq m - l$$

the column number of $\mathcal{N}_{d,\gamma}$ is not smaller than the row number, which means that the assumption on the right invertible is realistic. In case that $\mathcal{N}_{d,\gamma}$ is not right invertible, the pseudo-inverse of $\mathcal{N}_{d,\gamma}$ can be, as an alternative solution, applied for our purpose.

14.4.2 Observer-Based State Feedback Controller Design

We now study the observer-based feedback controller design. To explain the basic idea and design procedure, we first assume that $l = 1$, the state feedback gain F_c is identified by means of Algorithm 14.2 and the state observer,

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k) \in \mathcal{R}^n \tag{14.38}$$

$$r(k) = g y(k) - c_z z(k) - d_z u(k) \in \mathcal{R} \tag{14.39}$$

is constructed using the following algorithm, which is the result of the study in Chap. 10.

Algorithm 14.4 *Observer and residual generator design*

S1: Run Algorithm 9.3
S2: Select a row of $\mathcal{K}_{d,s}$ with $s = n$
S3: Form A_z, B_z, c_z, d_z, g, L_z according to (10.1)-(10.3)
S4: Construct (14.38)-(14.39).

It is evident that our major task of designing an observer-based state feedback controller is to find the (state) transformation between the state vectors $z(k)$ and $x_c(k)$, which will then allow us to apply the following feedback controller

$$u(k) = F_c \hat{x}_c(k)$$

with $\hat{x}_c(k)$ denoting an estimate of $x_c(k)$. Note that

$$z(k) = Tx(k), x(k) = T_c x_c(k) \Longrightarrow z(k) = TT_c x_c(k) \tag{14.40}$$

where T is the state transformation introduced in the observer design (see 10.2) and T_c is defined in (14.20), and both of them are of full rank. It turns out

$$TT_c = \mathcal{A}\Gamma_{n-1}\Pi_{n-1}\mathcal{W} \tag{14.41}$$

$$\mathcal{A} = \begin{bmatrix} \alpha_{n,1} & \alpha_{n,2} & \dots & \alpha_{n,n-1} & \alpha_{n,n} \\ \alpha_{n,2} & \dots & \dots & \alpha_{n,n} & 0 \\ \vdots & \dots & \dots & \vdots & \vdots \\ \alpha_{n,n} & 0 & \dots & \dots & 0 \end{bmatrix} \tag{14.42}$$

$$\mathcal{W} = \begin{bmatrix} w_{n,1} & w_{n,2} & \dots & w_{n,n-1} & w_{n,n} \\ w_{n,2} & \dots & \dots & w_{n,n} & 0 \\ \vdots & \dots & \dots & & \vdots \\ w_{n,n} & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{14.43}$$

Recall that

$$\Gamma_{n-1}\Pi_{n-1} = \begin{bmatrix} \Theta_0 & \dots & \Theta_{n-1} \end{bmatrix} \tag{14.44}$$

where $\Theta_i, i = 0, \dots, n-1$, are given in (14.26) and can be constructed using $H_{u,s}$ identified by Algorithm 14.1. As a result, the design of the observer-based state feedback controller can be realized fully using the data-driven realization of the SKR and SIR as follows

$$u(k) = F_c \left(TT_c\right)^{-1} z(k) = F_z z(k) \tag{14.45}$$

Next, we extend this design scheme to the multiple input case $l > 1$. We first give the dual result of Theorem 10.1.

Theorem 14.3 *Given a controllable system*

$$x(k+1) = Ax(k) + Bu(k) \in \mathcal{R}^n, u(k) \in \mathcal{R}^l$$

and assume that the system matrix A is non-derogatory. Let

$$\kappa_c = \begin{bmatrix} \kappa_{c,1} \\ \vdots \\ \kappa_{c,k_u} \end{bmatrix} \in \mathcal{R}^l$$

be a vector generated randomly from a uniform distribution, then it holds with probability one that

$$rank \begin{bmatrix} B\kappa_c & AB\kappa_c & \dots & A^{n-1}B\kappa_c \end{bmatrix} = n. \tag{14.46}$$

Since this theorem is the dual form of Theorem 10.1, the proof is omitted.

For selecting κ_c, we propose to apply the dual form of the algorithm given by Mercere and Bako, in which computations based on the I/O data sets are necessary. Notice that we can, alternatively, also apply the data delivered by the observer for

this purpose. Once κ_c is selected, (14.25) will be modified to

$$\left[\Theta_0\kappa_c \ \ldots \ \Theta_n\kappa_c\right] = \Gamma_{n-1}\Pi_n. \tag{14.47}$$

This modification yields

$$\Pi_n = \left[B\kappa_c \ AB\kappa_c \ \ldots \ A^nB\kappa_c\right] \in \mathcal{R}^{n\times(n+1)}. \tag{14.48}$$

As a result, the last step in Algorithm 14.2 for the computation of F_c is changed to

$$F_c = \kappa_c \left[w_{n,0} \ \ldots \ w_{n,n-1}\right] \in \mathcal{R}^{l\times n} \tag{14.49}$$

where $w_{n,i}, i = 0, \ldots, n-1$, are the resulted from the application of Algorithm 14.2 with $\Gamma_{n-1}\Pi_n$ as defined in (14.47). Correspondingly, the observer-based state feedback controller becomes

$$\begin{aligned} u(k) &= F_z z(k) = F_c \left(TT_c\right)^{-1} z(k) \\ TT_c &= \mathcal{A}\Gamma_{n-1}\Pi_{n-1}\mathcal{W}, \Gamma_{n-1}\Pi_{n-1} = \left[\Theta_0\kappa_c \ \ldots \ \Theta_{n-1}\kappa_c\right]. \end{aligned} \tag{14.50}$$

In summary, we have the following data-driven design algorithm.

Algorithm 14.5 *Data-driven design of feedback controller*

S1: Run Algorithms 14.4 and 14.2
S2: Form $\mathcal{A}$ and $\mathcal{W}$ defined in (14.41)-(14.43) and compute $F_c \left(TT_c\right)^{-1}$
S3: Construct controller (14.50).

14.5 Concluding Remarks

In this chapter and Chaps. 9, 10, we have presented a data-driven framework for the realizations of kernel and image representations and their application to the design of an observer-based fault detection and control system. This control system can be a part of a fault-tolerant architecture, but can also work independently. In Fig. 14.1, the major results and the main steps in the data-driven design procedure are summarized.

From the viewpoint of data set models and the applied identification technique, our work is similar to the well-established SIM technique. On the other hand, the objective of our study is to identify and construct the observer-based fault detection and control system directly from the process data sets. This has been realized with the aid of the kernel and image representations and their data-driven realization forms.

With the efforts made in this work, we intend to establish a framework, in which data-driven design of observer-based fault detection and control systems can be realized. The next chapter will be devoted to the data-driven optimization of Youla

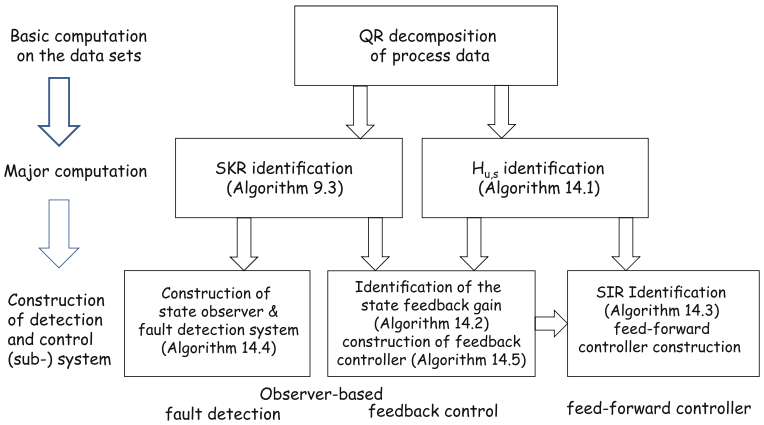


Fig. 14.1 Data-driven design scheme

parametrization matrix $Q(z)$ for the system robustness and optimization of post-filter $R(z)$ to improve the fault detection performance in a trade-off between the fault detectability.

14.6 Experimental Study on Laboratory CSTH System

In this section, experimental results are presented to demonstrate the application of Algorithms 14.1–14.5 to data-driven design of feedback controllers and further the potential of the fault-tolerant architecture introduced in the previous chapter in performing FTC.

14.6.1 System Setup and Process Measurements

The laboratory setup CSTH process described in Sect. 2.2 is adopted here for the case study. More specifically, we focus on the control of water level h_T as an application example. The system sampling time is 0.01 s. We have collected 5000 data for the feedback control gain identification, and additional 1000 ones for the validation of the constructed observer and residual generator.

14.6.2 Towards the Observer-Based Controller Design

Based on Algorithms 14.1, 14.2 the identification of $H_{u,s}$ is first carried out and later, the vector w_n is obtained as

$$w_n = \begin{bmatrix} -0.3626 \\ 0.3950 \\ -0.6040 \\ 0.5896 \end{bmatrix}.$$

It then leads to the construction of the deadbeat control gain matrix F_c according to (14.19)

$$F_c = \begin{bmatrix} -0.3626 & 0.3950 & -0.6040 \end{bmatrix}.$$

On the other hand, as a result of Algorithms 14.4–14.5, the state observer and residual generator (14.38), (14.39) are obtained by the identification of the parity vector with $s = n$ as follows

$$\begin{aligned} z(k+1) &= A_z z(k) + B_z u(k) + L_z y(k) \\ r(k) &= g y(k) - c_z z(k) - d_z u(k) \end{aligned}$$

where

$$A_z = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, B_z = \begin{bmatrix} -0.0030 \\ -0.0038 \\ -0.0003 \end{bmatrix}, L_z = \begin{bmatrix} 0.1322 \\ 0.6828 \\ -0.1016 \end{bmatrix}$$
$$g = 0.7112, c_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, d_z = 0.0030.$$

In addition, the transformation matrices satisfy

$$TT_c = \mathcal{A}\Gamma_{n-1}\Pi_{n-1}\mathcal{W}$$

where

$$\mathcal{A} = \begin{bmatrix} -0.6828 & 0.1016 & 0.7112 \\ 0.1016 & 0.7112 & 0 \\ 0.7112 & 0 & 0 \end{bmatrix}, \mathcal{W} = \begin{bmatrix} 0.3950 & -0.6040 & 0.5896 \\ -0.6040 & 0.5896 & 0 \\ 0.5896 & 0 & 0 \end{bmatrix}$$

and $\Gamma_{n-1}\Pi_{n-1}$ can be constructed by $\Theta_i, i = 0, \ldots, n-1$, given in (14.26). It follows that the observer-based state feedback controller (14.50)

$$\begin{aligned} u(k) &= F_z z(k) = F_c \left(TT_c\right)^{-1} z(k) \\ F_z &= \begin{bmatrix} -42.6634 & 353.3758 & -93.3105 \end{bmatrix}. \end{aligned}$$

14.6.3 Towards the Fault-Tolerant Control Scheme

As described in the previous chapter, once the gain matrices F, L are designed for system stabilization and state estimation, the follow-up issues, say achieving

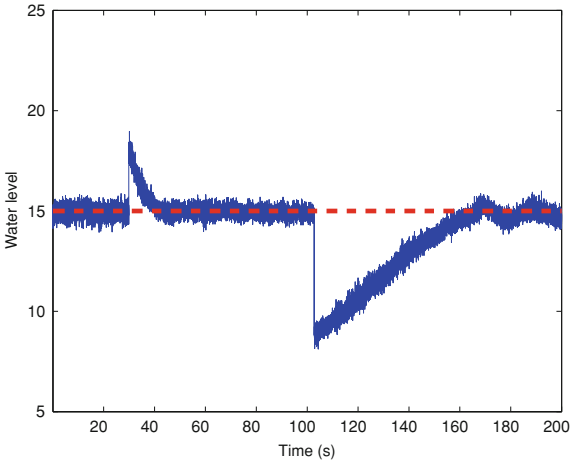


Fig. 14.2 Water level in the fault-tolerant case

the desired tracking performance and optimizing fault-tolerance/robustness, can be realized by determining $T(z)$ and $Q(z)$ respectively. With respect to this design strategy, some experimental results have been obtained. In Fig. 14.2, the tracking performance is demonstrated in a test with faults in the level sensor.

Since the controller adopted here is based on the state observer/residual generator (14.38), (14.39), the fault-tolerant feature can also be revealed by the variation of state estimates and residual signal as shown in Figs. 14.3, 14.4.

In order to show the superior of the proposed approach, a comparison study, as can be seen in Fig. 14.5, has been done between the default PID controller and the observer-based fault-tolerant controller. In this study, a sensor fault has been simulated and added in the level sensor at $k = 100$. To be compared is the tracking behavior, that is, the response of the water level to the fault. It is evident that the proposed scheme is more tolerant to the sensor fault than the default PID controller, which needed (considerably) more time to recover from the poor performance caused by the fault.

14.7 Notes and References

The study in this chapter has been originally motivated by the data-driven design of fault-tolerant architecture and its parameters. Although the major results can be, in combination with the data-driven design of observers and observer-based residual generation presented in Chap. 10, applied to a direct design of an observer-based control and fault detection system, the achievable system performance is limited. The main reason is that both the observer and state feedback controller are deadbeat

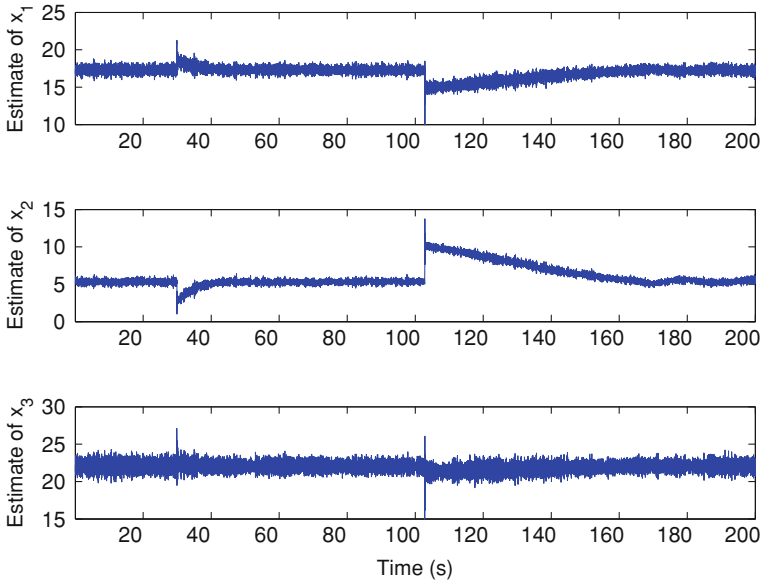
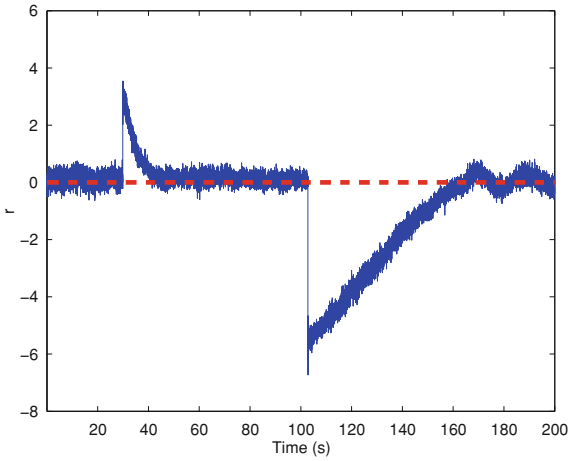


Fig. 14.3 State estimates in the fault-tolerant case

Fig. 14.4 Residual signal in the fault-tolerant case



dynamic systems, which may result in unsatisfactory performance in robustness and dynamic behavior. This is indeed the motivation for our investigation in the next chapter. On the other hand, it should be kept in mind that the state feedback gain and observer gain are the H-PRIO parameters of the fault-tolerant architecture. The results in this chapter also builds the basis for a real-time adaptation of the state feedback gain F to guarantee the system stability during the system operation period.

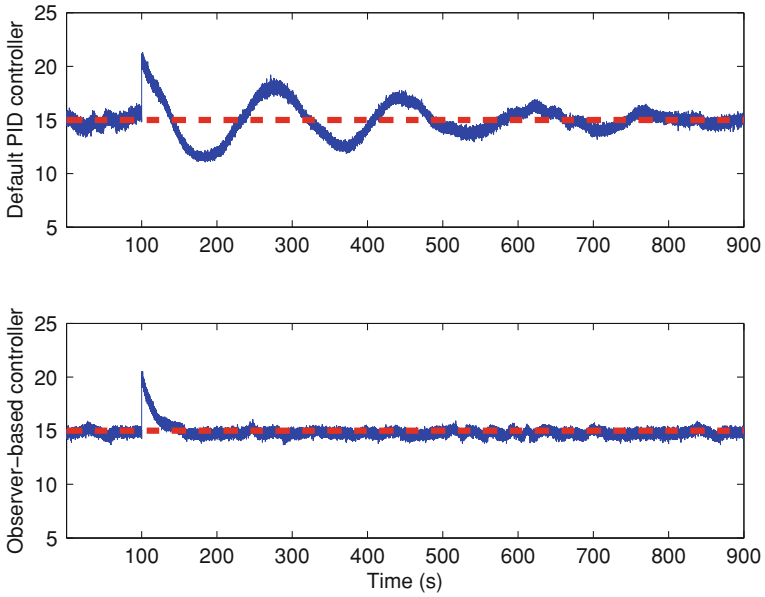


Fig. 14.5 Comparison between default PID controller and proposed scheme

Theorem 14.2 plays a key role in our study in this chapter. As mentioned, it is the dual result of the Parity vector based solution of Luenberger equations, which is described in details in [1] (Theorem 5.7).

Theorem 14.3 deals with the controllability and is the dual form of the study on the observability by Mercere and Bako [2]. In their work, Mercere and Bako also provide an algorithm for selecting a linear combination of the output variables by means of the process data. In its dual form, this algorithm can also be applied for selecting κ_c, a linear combination of the input variables.

References

1. Ding SX (2013) Model-based fault diagnosis techniques—design schemes, algorithms and tools, 2nd edn. Springer-Verlag, London
2. Mercere G, Bako L (2011) Parameterization and identification of multivariable state-space systems: a canonical approach. Automatica 47:1547–1555

Chapter 15
Realization of Lifetime Management of Automatic Control Systems

Roughly speaking, the objective of lifetime management of automatic control systems (LMACS) is to optimize the operation of automatic control systems in the broader sense of system reliability, availability, production performance as well as product quality. Fault-tolerant control is only one aspect of LMACS, which deals with the reliable online operation of control systems. In practice, further scenarios are common:

- replacement or repair of faulty system components in a maintenance action
- changes of operation conditions before and after a maintenance action
- changes of operation conditions due to, for instance, the change of the production line.

It is of significant practical interests to manage these procedures in such a manner that they run automatically or at least semi-automatically. In this chapter, we present a scheme for the LMACS realization on the basis of the fault-tolerant control architecture sketched in Fig. 13.2. This scheme consists of two parts:

- realization of an adaptive update of H-PRIO parameters and
- realization of iterative optimization of L-PRIO parameters.

of the automatic control loops under consideration.

15.1 Adaptive Update of H-PRIO Parameters

Recall that the observer and state feedback gain matrices L_z, F_z are H-PRIO parameters which should be updated online in case of changes in the process caused by uncertainties or faults or a maintenance action, in order to guarantee the closed-loop stability. In Sect. 11.3, Algorithm 11.4 has been proposed for designing the adaptive observer (11.15)–(11.19) and for updating L_z. In this section, we address the update issue of F_z using adaptive control technique (Fig. 15.1).

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4_15, © Springer-Verlag London 2014

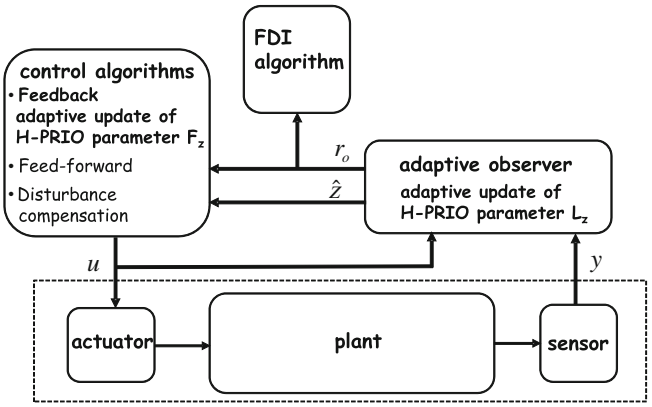


Fig. 15.1 Schematic description of adaptive updates of the H-PRIO parameters

15.1.1 Problem Formulation

Consider the process model that is controllable, observable and described by

$$x(k+1) = Ax(k) + Bu(k) \in \mathcal{R}^n, \, y(k) = Cx(k) \tag{15.1}$$

and the observer-based state feedback control law

$$u(k) = F_z \hat{z}\,(k)\,.$$

$\hat{z}\,(k) \in \mathcal{R}^n$ is the state estimate delivered by the adaptive observer (11.15)–(11.19), which is designed using Algorithm 11.4. In Theorem 14.2, it has been demonstrated that the feedback gain matrix F_z is in fact a system parameter matrix. Any change in the process may cause variations in F_z. As a result, an (online) estimate of F_z is needed, in order to guarantee the system stability. Concretely, we would like to solve the following problem: given the system model (15.1), the adaptive observer (11.15)–(11.19), find an adaptive control law so that F_z is estimated and

$$\lim_{k\to\infty} \big(y(k) - y_{ref}\,(k)\big) = 0 \tag{15.2}$$

with $y_{ref}\,(k)$ denoting the reference signal. For the sake of simplicity, we only consider SISO case, that is, $l = m = 1$.

15.1.2 Basic Idea

Consider the adaptive observer (11.15)–(11.19) with $s = n$. We assume that the conditions given in Theorem 11.2 are satisfied. It follows from Theorems 11.1, 11.2 and the associated discussion that for all $u(k)$, which ensure the needed persistent excitation,

$$\lim_{k\to\infty} \left(z(k) - \hat{z}(k)\right) = 0 \tag{15.3}$$

where

$$z(k) = Tx(k)$$

with the regular matrix T defined in (10.2). On the other hand, in Theorem 14.2 it has been proven that system model (15.1) can be written into

$$x_c(k+1) = A_c x_c(k) - B_c F_c x_c(k) + B_c u(k) \tag{15.4}$$

$$T_c x_c(k) = x(k),\ A_c = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 \end{bmatrix},\ B_c = T_c^{-1} B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ w_{n,n}^{-1} \end{bmatrix}$$

where F_c, T_c satisfy (14.19) and (14.20). It holds

$$x_c(k+1) = A_c x_c(k) - B_c F_z \hat{z}(k) + B_c F_z e_z(k) + B_c u(k) \tag{15.5}$$

$$y(k) = C_c x_c(k),\ e_z(k) = \hat{z}(k) - T T_c x_c(k) = \hat{z}(k) - Tx(k). \tag{15.6}$$

Recall the definition of $w_{n,n}$ given in Theorem 14.2, we are able, without loss of generality, to set it equal to 1. As a result, (15.4) can be equivalently expressed by

$$x_c(k+1) = A_c x_c(k) - b_c F_z z(k) + b_c F_z e_z(k) + b_c u(k),\ b_c = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{15.7}$$

Suppose that F_z is constant but unknown due to the possible changes in the process. Let

$$u(k) = \hat{F}_z(k)\hat{z}(k) \tag{15.8}$$

where $\hat{F}_z(k)$ denotes an estimate of F_z at time instant k. It holds

$$x_c(k+1) = A_c x_c(k) + b_c \Delta F_z \hat{z}(k) + b_c F_z e_z(k) \tag{15.9}$$

$$\Delta F_z = \hat{F}_z(k) - F_z. \tag{15.10}$$

Remember that

$$z(k+1) = A_z z(k) + B_z u(k) + L_z y(k)$$

and moreover

$$C\left(zI - A\right)^{-1} B = C_c \left(zI - A_c + B_c F_c\right)^{-1} b_c = C_z \left(zI - A_z - L_z C_z\right)^{-1} B_z.$$

Considering that F_c, L_z only lead to changes in the system characteristic polynomial, it turns out

$$C_c = B_z^T. \tag{15.11}$$

Since (15.3) holds, in the remaining part of this section we consider the system of the form

$$y(z) = G_c(z)\left(\beta(z) - F_z \hat{z}(z)\right), \beta(k) = \hat{F}_z(k)\hat{z}(k) \tag{15.12}$$

$$G_c(z) = C_c\left(zI - A_c\right)^{-1} b_c. \tag{15.13}$$

Note that (15.12) is equivalent to an adaptive output tracking problem with $G_c(z)$ as the reference model, whose poles are all located at the origin, and $y(k)$ as tracking error. Let

$$\omega(z) = G_c(z)\beta(z), \varsigma^T(z) = G_c(z)\hat{z}^T(z), \xi(k) = \hat{F}_z(k)\varsigma(k) - \omega(k) \tag{15.14}$$

$$\epsilon(k) = y(k) + \xi(k). \tag{15.15}$$

It is straightforward that

$$\epsilon(k) = \left(\hat{F}_z(k) - F_z\right)\varsigma(k) = \Delta F_z \varsigma(k). \tag{15.16}$$

Equation (15.16) is a linear model with a unknown parameter vector, which can be estimated using standard parameter estimation methods.

15.1.3 The Adaptive Scheme

We are now in a position to derive an estimation algorithm for F_z and so an adaptive control algorithm. To this end, the so-called normalized gradient method can be applied, which is given in the following lemma.

Lemma 15.1 *Given*

$$y(k) = \theta^T \phi(k) \in \mathcal{R}$$

where $\theta \in \mathcal{R}^n$ *is a constant but unknown parameter vector,* $y(k), \phi(k) \in \mathcal{R}^n$ *are known functions, and assume that*

$$\varphi(k) = \frac{\phi(k)}{m(k)}$$

is persistently exciting, then the recursion

$$\hat{\theta}(k+1) = \hat{\theta}(k) - \frac{\Gamma \phi(k) e(k)}{m^2(k)} \tag{15.17}$$

$$e(k) = \hat{\theta}^T(k)\phi(k) - y(k), m(k) = \sqrt{\kappa + \phi^T(k)\phi(k)} \tag{15.18}$$

where $0 < \Gamma < 2I, \kappa > 0$ *are design parameters, results in*

$$\lim_{k \to \infty} \left(\hat{\theta}(k) - \theta \right) = 0$$

exponentially.

This lemma is a known result in the adaptive control framework. For more details, the reader is referred to, for instance, the book by Tao, which is cited in the end of this chapter.

Applying Lemma 15.1 to (15.16) gives the following adaptive law.

Theorem 15.1 *Given the process model* (15.1)*, the adaptive observer* (11.15)–(11.19) *and assume* $\frac{\varsigma(k)}{m(k)}$ *is persistently exciting, then the adaptive law*

$$\hat{F}_z(k+1) = \hat{F}_z(k) - \epsilon(k) \frac{\varsigma^T(k)\Gamma}{m^2(k)} \tag{15.19}$$

results in

$$\lim_{k \to \infty} \left(\hat{F}_z(k) - F_z \right) = 0$$

exponentially, where $0 < \Gamma < 2I, \kappa > 0$ *are design parameters,* $\varsigma(k)$ *is defined in* (15.14) *and*

$$m(k) = \sqrt{\kappa + \varsigma^T(k)\varsigma(k)}. \tag{15.20}$$

Proof The proof of this theorem follows from Lemma 15.1 and the known results on adaptive output tracking control. Since (15.12) is equivalent to an adaptive output

tracking problem, it is known that

$$\lim_{k\to\infty} y(k) = 0. \tag{15.21}$$

On the other hand, note the fact that $\varsigma(k)$ is persistently exciting and the system is controllable and observable. Thus, (15.21) holds only if

$$\lim_{k\to\infty} \Delta F_z = \lim_{k\to\infty} \left(\hat{F}_z(k) - F_z\right) = 0.$$

15.1.4 Realization of the Adaptive Scheme

Note that in the above study, no reference signal is considered in the control law (15.8). For our application purpose, this control law is extended to

$$u(k) = \hat{F}_z(k)\hat{z}(k) + \upsilon(k) \tag{15.22}$$

with $\upsilon(k)$ denoting a reference signal or a feed forward controller. In this configuration, the overall adaptive control system consists of

- Auxiliary compensator

$$\vartheta(z) = G_z(z)\left[\hat{z}^T(z)\ u(z)\right], \varsigma^T(z) = G_z(z)\hat{z}^T(z) \tag{15.23}$$

$$\epsilon(k) = y(k) + \left[\hat{F}_z(k)\ \ -1\right]\vartheta^T(k) \tag{15.24}$$

- Estimator

$$\hat{F}_z(k+1) = \hat{F}_z(k) - \epsilon(k)\frac{\varsigma^T(k)\Gamma}{m^2(k)} \tag{15.25}$$

$$m(k) = \sqrt{\kappa + \varsigma^T(k)\varsigma(k)} \tag{15.26}$$

  where $0 < \Gamma < 2I$, $\kappa > 0$ are design parameters
- Adaptive control law

$$u(k) = \hat{F}_z(k)\hat{z}(k) + \upsilon(k)$$

  with $\hat{z}(k)$ delivered by the adaptive observer (11.15)–(11.19).

Note that $G_c(z)\upsilon(z)$ can be considered as a reference model and

$$y(z) - G_c(z)\upsilon(z) := e(z)$$

equivalently as tracking error. In this context, $\epsilon(k)$ can be rewritten into

$$\epsilon(k) = e(k) + \xi(k)$$

where $\xi(k)$ is defined in (15.14, 15.15). As a result, it holds again

$$\epsilon(k) = \Delta F_z \varsigma(k)$$

by which the convergence of the adaptive law can be guaranteed if the excitation condition is satisfied, as mentioned in Theorem 15.1. In summary, we have the following algorithm for an online update of the state feedback gain matrix.

Algorithm 15.1 *Adaptive state feedback control*

S1: Run the adaptive observer (11.15)-(11.19)
S2: Generate $\varsigma(k)$, $\epsilon(k)$ according to (15.23)-(15.24)
S3: Run (15.25).

An application of Algorithm 15.1 to a case study is given in Sect. 15.3.

15.2 Iterative Update of L-PRIO Parameters

This section addresses the update of the L-PRIO parameters. We focus on the following scenario: the control and observer gain matrices F_z, L_z are updated after changes in or/and around the process have been detected, which might be caused by faults in the system components or by some maintenance actions. By running Algorithms 15.1 and 11.4, the updates of F_z, L_z guarantee the stability of the overall control system. On the other hand, due to the changes, the control and detection performance of the system becomes poor. Our task is to optimize the parametrization matrices of the controller and observer, the so-called L-PRIO parameters, in an iterative manner.

15.2.1 Problem Formulation

We consider again systems of the form

$$x(k+1) = Ax(k) + Bu(k) + \eta(k),\ y(k) = Cx(k) + \nu(k) \tag{15.27}$$

where $\eta \sim \mathcal{N}(0, \Sigma_\eta)$, $\nu \sim \mathcal{N}(0, \Sigma_\nu)$ are white noise sequence and independent. We also suppose that the system is controllable and observable. For the control, process monitoring and diagnosis purposes, a Kalman filter

$$\hat{z}(k+1) = A_z\hat{z}(k) + B_z u(k) + L_z y(k) + K_z r_o(k) \tag{15.28}$$

$$r_o(k) = y(k) - C_z\hat{z}(k) \tag{15.29}$$

is constructed, where $\hat{z}(k)$ is an estimation of $Tx(k)$, L_z is identified (see Chap. 10) before the systems is in operation and updated (see Chap. 11) after changes in the system occurred. It is clear that

$$\hat{z}(k+1) = A_z\hat{z}(k) + B_z u(k) + L_z y(k), r_o(k) = y(k) - C_z\hat{z}(k)$$

is a deadbeat observer and residual generator. Hence, Kalman filter can be considered as a post-filter driven by the residual signal $r_o(k)$. In this context, K_z, the Kalman filter gain matrix, is an L-PRIO parameter which serves optimizing the system performance. By applying the results from our previous study, K_z is first identified and then updated in an iterative procedure, as shown in Chap. 12. Note that without loss of generality matrix G in (10.20) is assumed to be an identity matrix.

Our study in this section is focused on updating the L-PRIO parameter for the control purpose. The controller is given by

$$u(k) = F_z\hat{z}(k) + u_1(k), u_1(z) = Q(z)r_o(z) \tag{15.30}$$

where F_z, similar with L_z, is first identified before the systems is in operation and updated after changes in the system have been detected. $Q(z)$ is the (stable) parametrization matrix, as introduced in Chap. 13.

Suppose that the control performance is evaluated by the cost function

$$J = \lim_{N\to\infty}\frac{1}{N}\mathcal{E}\left(\sum_{k=0}^{N}\left(z^T(k)Q_z z(k) + u^T(k)R_z u(k)\right)\right), R_z > 0, Q_z \geq 0$$

and the fault detection performance by the covariance of the residual vector $r(k)$. It follows from Theorems 13.1 and 13.3 that the LQG controller with the embedded Kalman filter delivers the optimal solutions both for the control and fault detection purposes. Let us rewrite (15.28, 15.29) into

$$\hat{z}(k+1) = (A_z + L_z C_z + B_z F_z)\,\hat{z}(k) + B_z u_1(k) + (L_z + K_z)\,r(k).$$

It is straightforward that finding an optimal $Q(z)$ regarding to the cost function J is equivalent with finding F_{LQG}, that is,

$$u_1(z) = Q(z)r(z) = F_{LQG}\hat{z}(z)$$

with $\hat{z}(k)$ as the state estimate delivered by the Kalman filter and F_{LQG} the LQG feedback control gain matrix. Hence, we formulate our task as finding

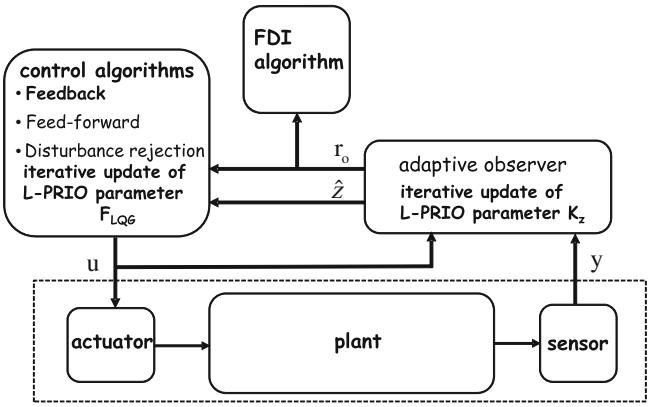


Fig. 15.2 Schematic description of iterative updates of the L-PRIO parameters

- an iterative Kalman filter to match the changes in the covariance matrices of the process and measurement noises using the iterative optimization method introduced in Chap. 12, and
- the LQG control gain matrix F_{LQG}, which will be addressed below.

Figure 15.2 gives a schematic description of our task.

15.2.2 Iterative Solution Algorithm

Consider the system model (15.27), which, following our discussion in Sect. 15.1, can be equivalent written as

$$z(k+1) = \left(\bar{A}_z + B_z F_{LQG}\right) z(k) - B_z \left(F_{LQG} + F_z\right) e_z(k) + \eta_z(k) \tag{15.31}$$

$$\bar{A}_z = A_z + L_z C_z + B_z F_z$$

$$y(k) = C_z z(k) + \nu(k) \tag{15.32}$$

where $z(k) = Tx(k)$, $\eta_z(k) = T\eta(k)$. Thanks to the separation principle, F_{LQG} can be determined using the result given by Theorem 13.1, that is,

$$F_{LQG} = -\left(B_z^T Y B_z + R_z\right)^{-1} B_z^T Y \bar{A}_z \tag{15.33}$$

$$Y = \bar{A}_z^T Y \bar{A}_z - \bar{A}_z^T Y B_z \left(B_z^T Y B_z + R_z\right)^{-1} B_z^T Y \bar{A}_z + Q_z. \tag{15.34}$$

Note that $\bar{A}_z$, B_z are the updated matrices delivered by the adaptive Kalman filter. On the other hand, it is clear that Riccati equation (15.34) should be solved online. To this end, the following well-established iterative algorithm can be used.

Algorithm 15.2 *Iterative LQG control gain computation*

S1: Initialization: set $F_{LQG,0}$
S2: Value update:

$$Y_{i+1} = \left(\bar{A}_z + B_z F_{LQG,i}\right)^T Y_i \left(\bar{A}_z + B_z F_{LQG,i}\right) + F_{LQG,i}^T R_z F_{LQG,i} + Q_z \quad (15.35)$$

S3: Policy update:

$$F_{LQG,i+1} = -\left(B_z^T Y_{i+1} B_z + R_z\right)^{-1} B_z^T Y_{i+1} \bar{A}_z. \quad (15.36)$$

(15.35, 15.36) build an iterative solution of Riccati equation (15.34), which converges to $Y > 0$, F_{LQG} given in (15.33). The reader is referred to the references given at the end of this chapter for details.

Remark 15.1 *Since the above optimization is embedded in a lifetime management process, the past setting of* F_{LQG} *can be applied as the initial value* $F_{LQG,0}$.

Remark 15.2 *Recall that our controller consists of two components,* $F_z\hat{z}(k)$ *and* $u_1(k)$, *as given in* (15.30). *Since the system stability is guaranteed by* $F_z\hat{z}(k)$ *and updating* F_{LQG} *means tuning* $u_1(k)$, *there is no need to update* $u_1(k)$ *at each iteration. Instead, it is reasonable to substitute* $F_{LQG,0}$ *by* $F_{LQG,N}$ *after the iteration is completed, where* $F_{LQG,N}$ *is the final iteration value.*

Our final solution consists of two algorithms which can run separately:

- run Algorithm 12.2 for an iterative Kalman filter, which delivers an optimal estimation of $z(k)$
- run Algorithm 15.2 for an iterative computation of F_{LQG}.

In Sect. 15.3, Algorithm 15.2 is applied to the case study and illustrated by simulations.

15.3 Implementation of the Lifetime Management Strategy

15.3.1 A General Description

The implementation of the LMACS strategy is event-driven. An event can be a successfully detected fault or a completed maintenance action. The LMACS strategy introduced in this chapter consists of two periods

- updating the H-PRIO parameters F_z, L_z and
- iterative optimization of the L-PRIO parameters K_z, F_{LQG}.

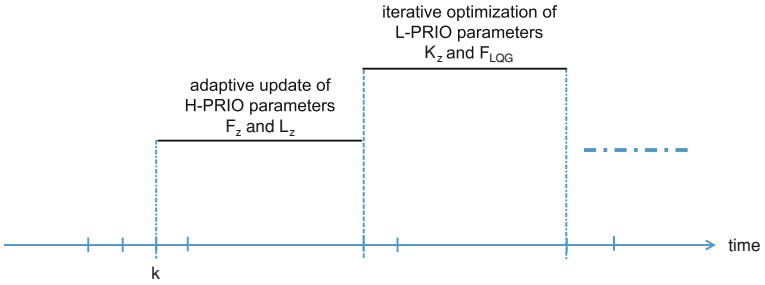


Fig. 15.3 Schematic description of the LMACS strategy: at time instant k an event is activated

In Fig. 15.3, this strategy is schematically sketched, where it is supposed that an event is activated at time instant k.

15.3.2 Case Study on Three-Tank System

15.3.2.1 System Setup

In our simulation study on the three-tank system, we only activate one pump (pump 1) as the process input variable and apply a linear combination of the three water level measurements as the process output variable. By setting the operation point as $h_1 = 20\,\text{cm}$, we have the (initial) linearized model achieved by a linearization at the operating point.

$$\dot{x}(t) = Ax(t) + Bu(t),\, y(t) = Cx(t) \tag{15.37}$$

where

$$A = \begin{bmatrix} -0.0120 & 0 & 0.0120 \\ 0 & -0.0319 & 0.0115 \\ 0.0120 & 0.0115 & -0.0235 \end{bmatrix},\, B = \begin{bmatrix} 0.0065 \\ 0 \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 1\ 2\ 4 \end{bmatrix}.$$

The corresponding discrete-time model, achieved by a discretization with a sampling time $t = 5\,\text{s}$, is given by

$$x(k+1) = A_d x(k) + B_d u(k),\, y(k) = Cx(k)$$
$$A_d = \begin{bmatrix} 0.9434 & 0.0001541 & 0.05498 \\ 0.0001541 & 0.854 & 0.05006 \\ 0.05498 & 0.05006 & 0.8923 \end{bmatrix},\, B_d = \begin{bmatrix} 0.03153 \\ 1.725e-005 \\ 0.0009189 \end{bmatrix}.$$

The structure of the observer-based state feedback controller is

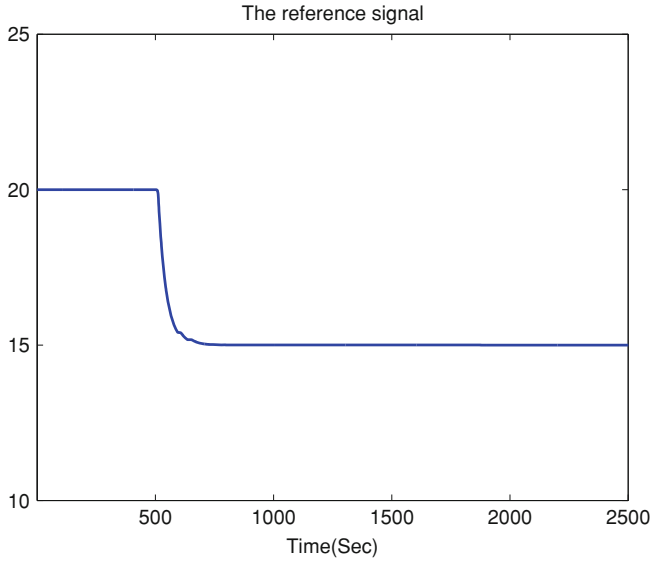


Fig. 15.4 Change of the reference signal $y_{ref}(k)$

$$u(z) = F_z\hat{z}(z) + Q(z)r_o(z) + Vy_{ref}(z)$$

where $y_{ref}(k)$ denotes the reference signal, $V = 4713$ is the pre-filter. $\hat{z}(k)$ is the estimate for $z(k) = Tx(k)$, delivered by a Kalman filter with r_o as residual signal. F_z is the deadbeat state feedback gain. $Q(z)r_o(z)$ is applied for the realization of an LQG controller, as described in the last section.

15.3.2.2 Adaptive Update of H-PRIO Parameters

By changing the operating point, variations in the process dynamics are simulated. This is realized as the reference signal moves to the operating point $y_{ref}(k) = 15\,\text{cm}$, which is shown in Fig. 15.4. The updating results of L_z,and F_z by running Algorithm 11.4 and Algorithm 15.1 are respectively given in Figs. 15.5 and 15.6. They, together with B_z, converge to

$$L_z = \begin{bmatrix} 0.7448 \\ -2.2152 \\ 3.0147 \end{bmatrix}, B_z = \begin{bmatrix} 0.0105 \\ -0.0207 \\ 0.0231 \end{bmatrix}$$
$$F_z = \begin{bmatrix} 0.0756 & 0.2625 & -0.0997 \end{bmatrix}.$$

Figure 15.6 demonstrates the simulation results of the water levels in the three tanks.

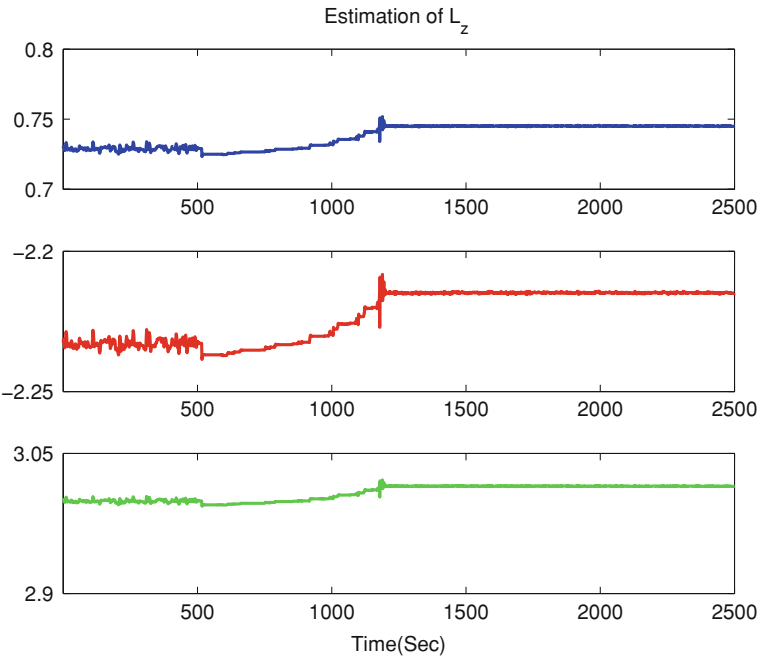


Fig. 15.5 Update of L_z

15.3.2.3 Iterative Update of L-PRIO Parameters

We now illustrate the application of Algorithm 12.2 and Algorithm 15.2.to iteratively updating the L-PRIO parameters K_z and F_{LQG}, respectively, starting with the final values of the L_z, F_z, B_z matrices achieved in the last updating period (Fig. 15.7).

In the simulation study on iterative Kalman filter, zero-mean measurement noise with the covariance $\sigma = 0.01$ is added. The application of Algorithm 12.2 results in updating the σ estimation, as shown in Fig.15.8. Correspondingly, the Kalman filter gain K_z is iteratively updated.

Concerning the quadratic cost function that evaluates the control performance, the state and input weighting matrices are selected as

$$Q_z = diag(10, 10, 10), R_z = 0.01.$$

Starting with Algorithm 15.2, we choose $F_{LQG,0}$ according to (15.33, 15.34) based on the initial $\bar{A}_z$ and B_z,

$$F_{LQG,0} = \begin{bmatrix} 12.1815 & -4.7145 & 0.0000 \end{bmatrix}$$

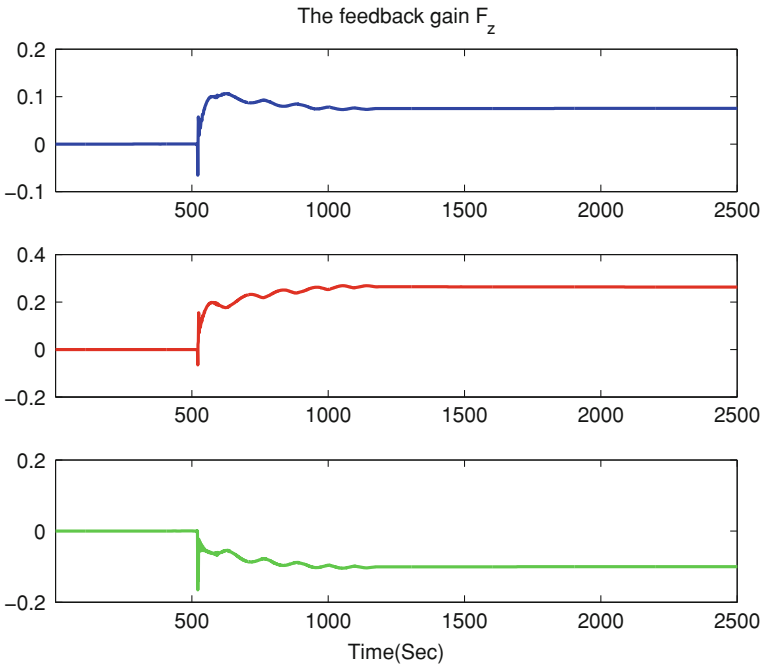


Fig. 15.6 Update of F_z

and the value of Y_0 simply as

$$Y_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

After 9 iterations, we have the following results

$$F_{LQG,9} = \begin{bmatrix} 12.0911 & -4.8000 & 0.0025 \end{bmatrix}, Y_9 = \begin{bmatrix} 17.3818 & 4.3295 & 0.0004 \\ 4.3295 & 18.2799 & -0.0001 \\ 0.0004 & -0.0001 & 10.0000 \end{bmatrix}$$

which are identical with the theoretical values. The convergence of the iterative update of F_{LQG} is illustrated in Fig. 15.9, which demonstrates the effectiveness of Algorithm 15.2.

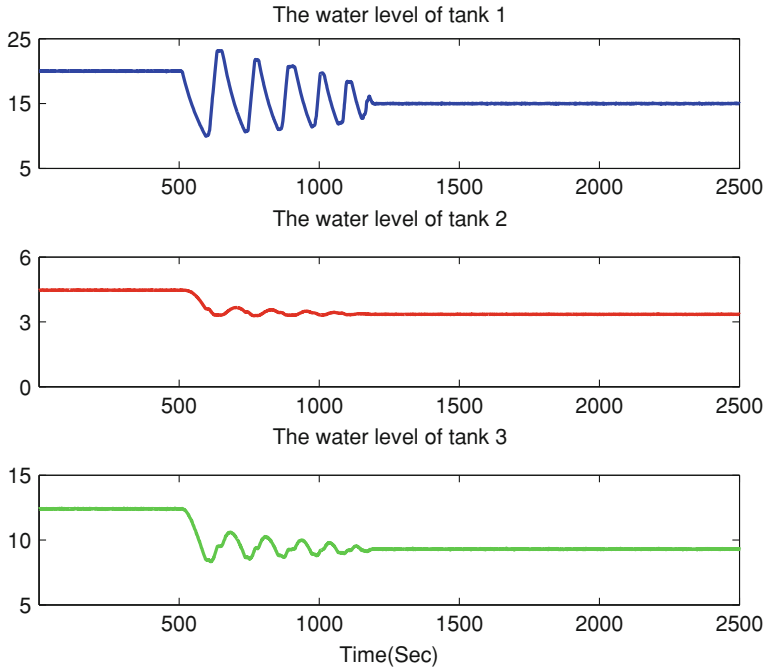


Fig. 15.7 Simulation results of the water levels in the tanks

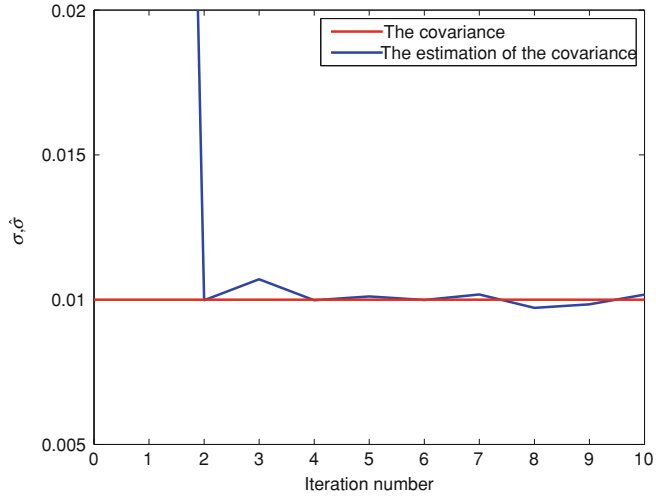


Fig. 15.8 Iterative update of σ

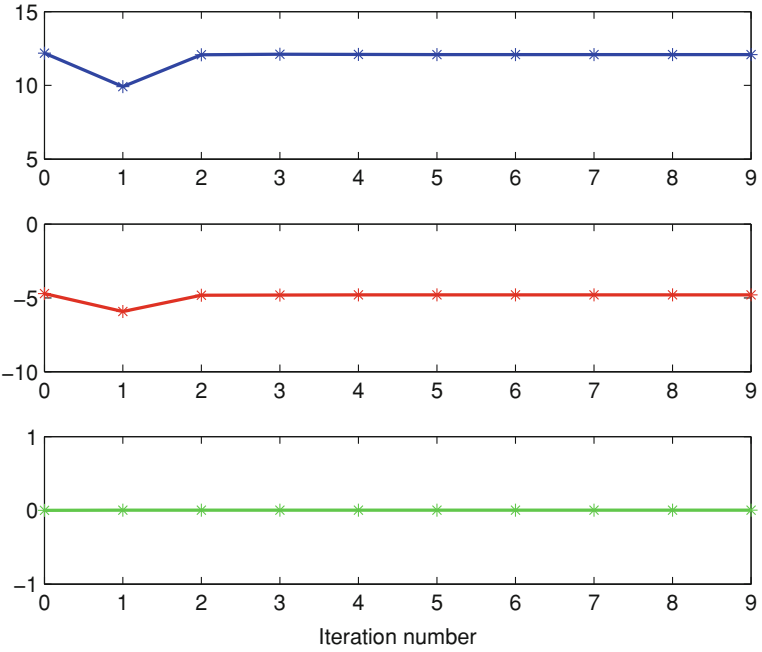


Fig. 15.9 Convergence of updating $F_{LQG}(1:3)$: top: $F_{LQG}(1)$, middle: $F_{LQG}(2)$, bottom: $F_{LQG}(3)$

15.4 Notes and References

[1] is an excellent book by Tao on adaptive control design and analysis. Lemma 15.1 is a summary of some results described in Chap. 3 in this book. A detailed description of the so-called adaptive output tracking control can be found in Chapter 4 in [1], where Lemma 4.3 and Theorem 4.3 are the main results adopted in our study.

The iterative solution of Riccati equation (15.34) given in (15.35, 15.36) is a known result, see, for instance, the book by Lancaster and Rodman [2]. This solution is also applied in the investigation on reinforcement learning and adaptive dynamic programming for feedback control [3, 4].

In this chapter, we have investigated and illustrated the implementation of the LMACS strategy. The basic idea behind the LMACS strategy is the functionalization and prioritization of control system parameters, which allow us to update and optimize the controller parameters

- by means of different control methods and
- in different time intervals according to the scheduler shown in Fig. 15.3.

With the update of the H-PRIO parameters, the system stability, the so-called fail-safe operation, is guaranteed. The performance optimization is then approached by an iterative update of the L-PRIO parameters. It should be mentioned that the LMACS

is a new strategy. Further efforts are necessary to improve it. For instance, algorithms for the (online) optimization of L-PRIO parameters like the parametrization matrices of the feed-forward controller and residual generator as well as the threshold setting are to be developed.

References

1. Tao G (2003) Adaptive control design and analysis. Wiley-Interscience, New Jersey
2. Lancaster P, Rodman L (1995) Algebraic Riccati equations. Oxford Univ. Press, London
3. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circuits Syst Mag 9:32–50
4. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control. IEEE Control Syst Mag 32:76–105

Index

S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*, Advances in Industrial Control,
DOI: 10.1007/978-1-4471-6410-4, © Springer-Verlag London 2014