Intelligent Systems, Control and Automation:
Science and Engineering

Suril Vijaykumar Shah
Subir Kumar Saha
Jayanta Kumar Dutt

Dynamics of Tree-Type Robotic Systems

Springer

Dynamics of Tree-Type Robotic Systems

International Series on
INTELLIGENT SYSTEMS, CONTROL AND AUTOMATION:
SCIENCE AND ENGINEERING

VOLUME 62

For further volumes:
http://www.springer.com/series/6259

Suril Vijaykumar Shah • Subir Kumar Saha
Jayanta Kumar Dutt

Dynamics of Tree-Type Robotic Systems

Springer

Dr. Suril Vijaykumar Shah
Postdoctoral Fellow
McGill University
Canada

Dr. Subir Kumar Saha
Department of Mechanical Engineering
IIT Delhi
New Delhi
India

Dr. Jayanta Kumar Dutt
Department of Mechanical Engineering
IIT Delhi
New Delhi
India

ISBN 978-94-007-5005-0 ISBN 978-94-007-5006-7 (eBook)
DOI 10.1007/978-94-007-5006-7
Springer Dordrecht Heidelberg New York London

Library of Congress Control Number: 2012954613

Printed on acid-free paper

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

Preface

Robots have evolved since birth of first industrial robot in 1961. Rapid industrialization, automation, high rate of production and maximizing human comfort in the modern times have necessitated various multifaceted applications of robots, which are required to perform complex tasks. As a result, multiple-chain tree-type robotic systems such as multi-fingered robotic arms, legged vehicles, humanoid robots, etc. have emerged. Successful as well as fast operation of such robots calls for systematic, efficient, and as far as possible generic computational framework to predict dynamic behavior. The framework should be useful in robot operation, trajectory planning, simulation and control. Driven by this motivation, an attempt has been made in this book to present a modular framework for dynamic modeling and analysis of tree-type robotic systems.

Robots having general tree-type architectures have been considered first. The concept of kinematic modules, where each module is essentially a serial-chain system, has been introduced. Macroscopically, module may be viewed similar to a link in the serial chain system. Next, module-level decoupled form of velocity transformation matrix, i.e., Decoupled Natural Orthogonal Complement matrices (DeNOC), is obtained by following the recursive relationships between any two adjoining modules. The constrained equations of motion are then obtained by using the module-level DeNOC matrices. Modular framework offers many advantages. For example, it makes the kinematic as well as dynamic representations compact, allows module-level expressions for various entities, and enables repeated use of module-level computations. A novel concept of Euler-Angle-Joints (EAJs) is also introduced to represent a spherical joint, which allows for unified representation of multiple-DOF joints that commonly appear in spatial systems.

Following the DeNOC-based approach, it is possible to obtain module-level analytical expression of the Generalized Inertia Matrix (GIM), which is used for module-level decomposition of the GIM. The decomposition allows one to invert the GIM analytically. The analytical inversion provides recursive forward dynamics algorithm, an insight into the associated dynamics, and help in predicting any inconsistency in the dynamic behavior of a robot.

Empowered with the modular framework, studies of dynamic behavior of fixed-base as well as floating-base robotic systems have been taken up. Initially, motion and force analyses of several fixed-base tree-type systems, e.g., a robotic gripper, biped, and hyper-degrees-of-freedom (hyper-DOF) system, are presented. This is followed by the approach where the legged robots are modeled as a floating-base with several branches (limbs or legs) emanating from the floating-base. Such approach provides more realistic modeling of the legged robots. In order to show the effectiveness of the modeling approach, legged robots are simulated under model-based control laws (e.g., computed-torque and feedforward). Computational complexities in terms of operation counts and CPU times are also reported to show that the methodology presented performs much better than many commonly used methodologies reported in literature. The efficacy of the approach increases with the complexity, primarily with total number of links and number of multiple-DOF joints in the system.

In summary, the book addresses dynamic modeling methodology and analyses of tree-type robotic systems. Such analyses are required to visualize the motion of a system without really building it. The book contains novelty in the form of treatment of the tree-type systems using concept of kinematic modules, unified representation of the multiple-degrees-of-freedom joints, efficient recursive dynamics algorithms, and detailed dynamic analyses of several legged robots.

This book is useful for teaching graduate-level (Master's and Ph.D.) courses specialized in Robot Dynamics and Legged robots. This book will also help researchers and practicing engineers in virtual testing, trajectory planning, designing and controlling complex robotic systems. In order to help the readers to quickly analyze their systems the ReDySim (Recursive Dynamic Simulator) developed based on the methodologies presented in this book is made freely available through the website http://www.redysim.co.nr/book.html.

Units and Notation

The international System of Units (SI) is used throughout this book. The boldface Latin/Greek letters in lower and upper cases denote vectors and matrices, respectively, whereas light face Latin/Greek letters in lower case with italic font denote scalars. In the case of any deviation in the above definitions, an entity is defined as soon as it appears in the text. Moreover, symbol '—' over an entity signifies that it is associated with a kinematic module of the tree-type systems under study.

Acknowledgments

We would like to thank all who have directly or indirectly helped in the preparation of this book. Special thanks are due to Indian Institute of Technology (IIT) Delhi where the first author did his Ph.D. We also thank people of Mechatronics Laboratory at IIT Delhi, and others with whom we had many discussions about life and education that may have influenced the presentation of this book indirectly. Special thanks are also due to our respective family members, Kruti and Arjav (with Suril Vijaykumar Shah), Bulu and Esha (with Subir Kumar Saha), and Mitali and Anabil (with Jayanta Kumar Dutt) for their patience and understanding while this book was under preparation. In addition, we express our sincere gratitude to Ms. Nathalie Jacobs of Springer Netherlands and anonymous reviewers for readily accepting the book for publication.

IIT Delhi

Suril Vijaykumar Shah
Subir Kumar Saha
Jayanta Kumar Dutt

Contents

Chapter 1
Introduction

Robots find applications in automobile, electronics, chemical and many other industries. The field of robotics has evolved since the development of industrial robots, which have mainly fixed base with serial-chain architecture. Since then many new applications like maintenance task of industrial plants, operations in dangerous and emergency environments, surveillance, maneuvering in unknown terrains, telesurgery, human care, grasping and manipulation of complex objects, multi-point force and tactile feeling, etc. have come into existence. As a result, multiple-chain tree-type robotic systems such as multi-fingered robotic arm, legged vehicle, humanoid robot, etc. have emerged. They are referred to as tree-type robotic systems in this book. Trajectory planning, dynamic analysis, obtaining stability criterion, and control are some of the basic issues with such systems. Research in these areas has growing interest and is active field of the robotics.

Dynamic analysis involves either force or motion analysis or both. Whereas force analysis attempts to find the driving and reactive forces for given input motion, the motion analysis obtains system's configuration under the input forces. Force analysis helps in design and control of robotic systems, whereas motion analysis allows one to study and test a design virtually without really building a real prototype and is called "virtual prototyping."

There has been significant increase in the use of computational dynamics for simulation, analysis, design, and model-based control of various robotic systems. Hence, an efficient framework is essential for the dynamic analysis of complex robotic systems. In this context, recursive dynamics algorithms play an important role. They are attractive due to simplicity and computational uniformity regardless of ever growing complex robotic systems. Recursive formulations in robotic applications have significantly helped in achieving real-time computations and model-based control laws.

In this book, an attempt has been made to present modular framework for kinematic and dynamic modeling of tree-type robotic systems, and recursive algorithms for predicting their dynamic behavior with or without control.

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_1,

1.1 Tree-Type Robotic Systems

The concept of robots has been around since medieval times in the form of human-like toys. Earlier perception of the robots was in the form of machine that performs human-like work or imitates animals. It wasn't until 1960s that the first modern day industrial robot was built. Since then the field of robotics has grown a lot and many new applications of robot have come into existence. An industrial robot is like a human arm and of many applications it finds maximum utility in pick and place operation. It has a fixed-base wherefrom bodies or links emanate one after another in a serial fashion, as shown in Fig. 1.1. The developments in the field of industrial robots have encouraged researchers to build robots which imitate mammals. As a result, many complex robotic systems have emerged. These robotic systems have multiple chains connected by single as well as multiple-degrees-of-freedom joints, and, in general, are referred to as tree-type robotic systems. Figure 1.2 shows several examples of tree-type robotic systems. Figure 1.2a is a multi-fingered gripper. Figure 1.2b illustrates an industrial robot, as in Fig. 1.1, with a secondary manipulator carrying the camera subsystem towards the end-effector, and hence forms a tree. A quadruped legged robot and a humanoid robot are shown in Fig. 1.2c, d, respectively.

Tree-type robotic systems have essentially multiple branches, like in a tree, with variable constraint topology. The base of the systems may be fixed or floating depending on the type of the robotic system. The robotic systems in Fig 1.2a, b may be considered to have a fixed-base whereas the systems in Fig. 1.2c or d have floating bases. Modeling and control of such robots are difficult due to their variable architecture. For example, the legged robot in Fig. 1.2c forms a closed-loop system when it is standing on the ground. However, when the robot lifts one or two of its legs to move forward or backward, the legs form open-chain architecture with respect to the trunk. The complexity involved with the tree-type robotic systems necessitates efficient computational framework for the dynamic analyses, which can be useful in simulation, design, trajectory planning, and control of those systems.

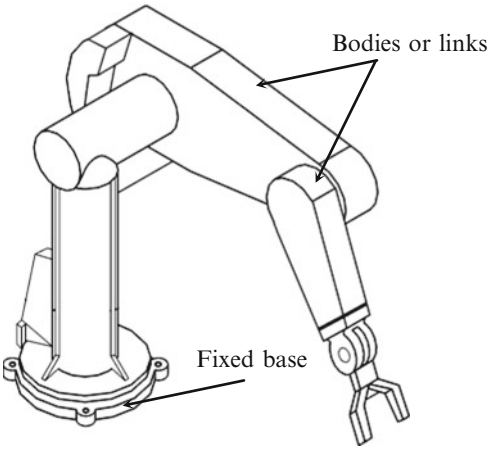


Fig. 1.1 An industrial robot

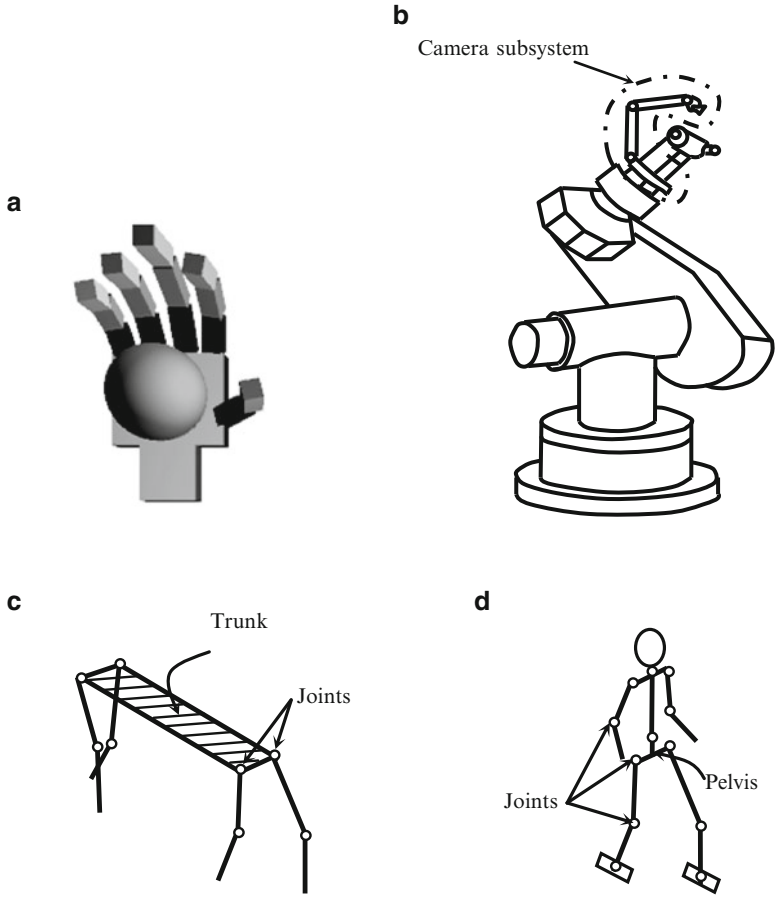


Fig. 1.2 Examples of tree-type robotic systems. (**a**) Multi-fingered gripper. (**b**) Industrial robot with camera subsystem. (**c**) A legged robot. (**d**) Humanoid

1.2 Dynamics

Computer-aided dynamic analysis of robotic systems has been a prime motive of the engineers since the evolution of high speed facilities using computers. In order to perform computer-aided analysis, the actual system is represented with its dynamic model, which has information in terms of link parameters, joint variables and constraints. Dynamic model of a robotic system is represented by a set of equations governed by physical laws of motions. For a system with few links, it is easier to obtain explicit expressions for the equations of motion. However, finding equations of motion is not an easy task for complex systems with many links. Sometimes even with four or five links, say, a 4-bar mechanism, it is difficult to find an explicit

expression for the system inertia in terms of the link length, mass and joint angle. Various techniques are available in the literature for automatic generation of the equations of motion. Development of the equations of motion is an essential step for dynamic analysis. Objective of dynamic analysis is mainly two fold.

1. First, to perform force analysis. This is referred to as inverse dynamics, in which driving forces of the system are computed for a given set of input joint motions. Knowledge of the driving forces help not only in control but also in actuator design of the robot. One may also obtain the reaction forces, which may be used for the design of associated linkages and joints from the strength point of view.
2. Second is motion analysis. This is called forward dynamics, where joint motions are computed under the application of external forces. This along with numerical integration helps in obtaining the configuration or state of the system at any instant of time. In other words, forward problem lets one to simulate the actual working of the system.

Whereas inverse dynamics requires only evaluation of algebraic expression of the equations of motion, forward dynamics requires the solution of the equations of motion. So, the forward dynamics problem may involve solutions of either Ordinary Differential Equations (ODE) or algebraic and differential equations together, i.e., Differential Algebraic Equations (DAE). Closed form solutions of the ODE or DAE for simple systems may be trivial, however, for a complex system they are not possible. Most common approach to solve them is to use numerical methods. It is worth noting here that no numerical method can give exact solution and, hence, it is prone to errors, either round-off or truncation or otherwise. Error propagation affects accuracy of the results leading to even numerical instability. The round-off error is caused mainly due to limitation of the computing system, whereas the truncation error depends on the algorithm used. Since the latter error is reduced with the reduced number of computational steps, so the computational count of an algorithm should be reduced as much as possible for numerical stability and efficiency. This reduction will certainly reduce the truncation errors as the computer needs to evaluate less number of functions. As a result, efficient computational framework to obtain the solution of the equations of motion for the dynamic analysis of tree-type robotic system is called for. To achieve efficiency, modular framework for the dynamic analysis of tree-type robotic systems consisting of multiple-degrees-of-freedom joints is presented in this book.

1.3 Important Features of the Book

The important features of this book are outlined below:

1. The concept of Euler-Angle-Joints (EAJs) for the representation of a spherical joint and its generalization for 1-, 2-, and 3-degrees-of-freedom joints.

2. Generalization of the link-to-link kinematic transformation of velocities, i.e., twist-propagation, to module-to-module twist-propagation using the concept of kinematic modules. As a consequence, definition of the Decoupled Natural Orthogonal Complement (DeNOC) matrices for a serial chain system is extended to a tree-type robotic system.
3. Dynamic modeling of a tree-type robotic system using the concept of the module-DeNOC matrices giving module-level expressions for the vectors and matrices appearing in the equations of motion.
4. Module-level analytical block $\mathbf{U}\,\mathbf{D}\,\mathbf{U}^T$ decomposition of the Generalized Inertia Matrix (GIM) for the tree-type robotic system using the Block Reverse Gaussian Elimination (BRGE), where $\mathbf{U}$ and $\mathbf{D}$ are the block upper triangular and block diagonal matrices, respectively. Subsequently analytical expressions of the inverse of the GIM are derived providing deeper insight into the dynamics.
5. Efficient recursive algorithms for the inverse and forward dynamics of the fixed- and floating-base tree-type robotic systems consisting of multiple-degrees-of-freedom joints introducing inter- and intra-modular recursions.
6. Dynamic analyses of several tree-type practical robotic systems, namely, (i) a robotic gripper, (ii) planar and spatial biped, (iii) a quadruped, and (iv) a hexapod, which would help researchers and practicing engineers to use these results for design, simulation and control of those robots.
7. Closed-loop control simulation of several robots, mainly, those listed in item 6 above, to study their controlled behaviors.

1.4 Book Organization

The book contains ten chapters which are organized as follows:

1 Introduction

The motivation and scope of the book are presented in this chapter. The important features and organization of the book are also highlighted.

2 Dynamics of Robotic Systems

Background and development in the field of dynamic modeling and tree-type robotic systems will be reviewed in this chapter.

3 Euler-Angle-Joints (EAJs)

A spherical joint in robotic literature is typically modeled as three-intersecting revolute joints. It is shown in this chapter that it is nothing but a variant of well-known Euler angle representation. Hence, the term Euler-Angle-Joints (EAJs) is introduced. The evolutions of different EAJs with examples will be shown in this chapter.

4 Kinematics of Tree-Type Robotic Systems

The concept of kinematic modules for the tree-type systems is introduced in this chapter. The recursive kinematic relationships in the form of twist-propagations are first obtained at intra-modular level (inside the module). Using the concept of Euler-Angle-Joints introduced in Chap. 3, it was possible to generalize 1-, 2-, and 3-degrees-of-freedom rotary joints, namely, revolute, universal and spherical joint, respectively. Next, inter-modular (between the module) kinematic constraints, i.e., module-level twist-propagations, are derived. This allows one to generalize the definition of the DeNOC matrices for a serial system to tree-type systems.

5 Dynamics of Tree-Type Robotic Systems

Dynamic modeling of the tree-type robotic system is presented in this chapter. The module-level expressions for the matrices and vectors appearing in the equations of motion are presented. Following the analytical expression of the Generalized Inertia Matrix (GIM), its module-level decomposition is obtained. The GIM is decomposed in the form of $\mathbf{U}\,\mathbf{D}\,\mathbf{U}^T$ using the concept of Block Reverse Gaussian Elimination (BRGE), where $\mathbf{U}$ and $\mathbf{D}$ are the block upper triangular and block diagonal matrices. This leads to physical interpretations of the terms associated with the dynamics. The decomposition helps in obtaining explicit analytical inversion of the GIM and the recursive forward dynamics.

6 Recursive Dynamics of Fixed-Base Systems

In this chapter, dynamic analyses of fixed-base robotic systems are presented. For this, recursive algorithms for inverse and forward dynamics are obtained. Dynamic analyses of robotic gripper, spatial biped and hyper-degrees-of-freedom system are presented in this chapter.

7 Recursive Dynamics of Floating-Base Systems

Modeling of a system with floating or mobile base is proposed in this chapter. Recursive algorithms for inverse and forward dynamics of floating-base robotic systems are derived here. Analyses for biped, quadruped and hexapod robotic systems are presented.

8 Closed-Loop Systems

In this chapter, dynamic analysis of a closed-loop system is presented using the algorithms presented for the tree-type robotic systems. It will be shown how the algorithm for a tree-type system can be suitably adopted when a closed-loop system is cut-open to form tree-type architecture.

9 Controlled Robots

Two model-based control schemes, namely, feedforward and computed-torque control, have been discussed with the help of proposed dynamic framework. Simulation of biped, quadruped and hexapod is presented by using them.

10 Recursive Dynamic Simulators (ReDySim)

Used of a MATLAB based solver, developed based on the algorithms presented in Chap. 6 and 7, will be illustrated in this chapter.

Appendices

A: Computational Complexity
B: Trajectory Generation of Legged Robot
C: Energy Balance
D: Foot-Ground Interaction

Chapter 2
Dynamics of Robotic Systems

The field of robotics has grown a lot in last three to four decades. In this chapter, background and developments in the field of dynamics of robotic systems are presented.

2.1 Robotic Systems

A robotic system can be divided into two categories based on their topology, i.e., open-chain or closed-chain. Robots with serial and tree-type architecture are open-chain systems, e.g., a PUMA industrial robot and multi-fingered robotic hands shown in Figs. 2.1a and 2.2, respectively. The cooperating industrial robot and robots with parallel architecture, e.g., Stewart platform and delta manipulator, are examples of closed-chain systems. Figure 2.3 shows a Stewart platform. On the contrary, legged robots as shown in Fig. 2.4 have time varying topology, i.e., a combination of open- and closed-chains. A legged robot may also be viewed as a robotic system with floating- or mobile-base with intermittent ground contacts. Thus, a robotic system has its base either fixed or floating. Industrial robots and parallel robots are examples of fixed-base systems while the space manipulators and legged robots are examples of floating-base systems. This book addresses dynamics of tree-type robotic systems with both fixed-base and floating-base. The analysis of closed-chain systems can be carried out by cutting appropriate joints to form a tree-type system, where the opened joints are substituted with appropriate constraint forces.

2.1.1 Serial Robots

Industrial robot (Fig. 2.1) is a classical example of a fixed-base serial-chain robotic system. Industrial robots have the capability to perform manipulation task similar

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_2,

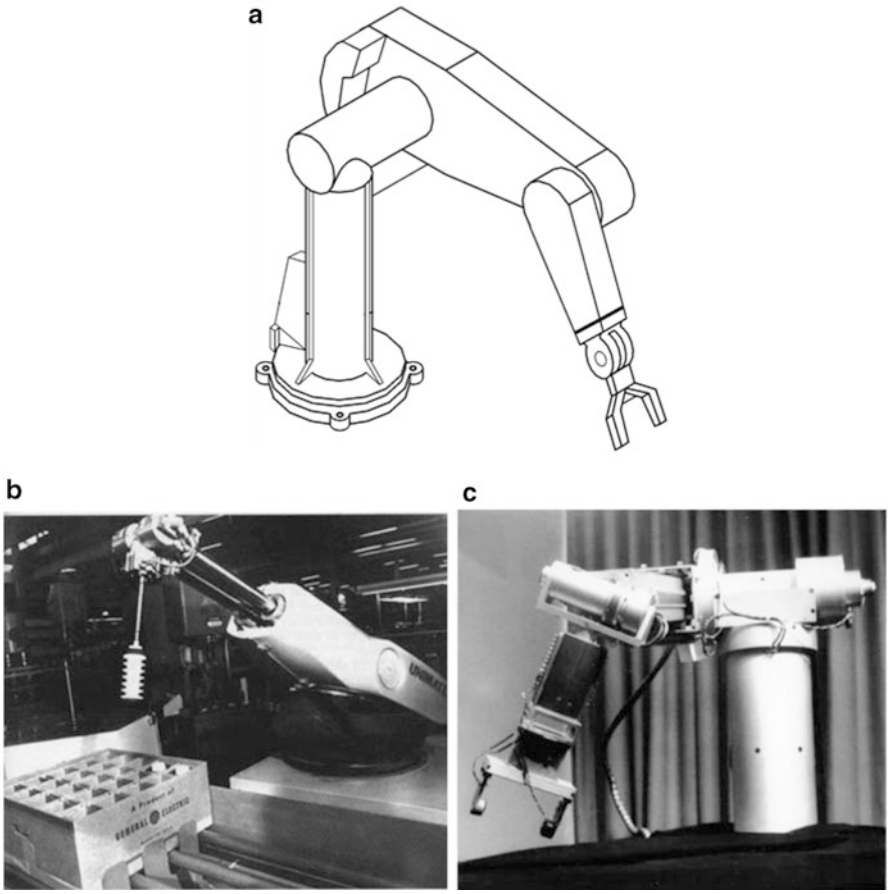


Fig. 2.1 Industrial robots. (**a**) Puma Robot. (**b**) UNIMATE robot (http://ijts). (**c**) Sanford arm (http://www.dipity)

to human arm. Development of such robots started after World War II. The first industrial robot, UNIMATE of Fig. 2.1b, was used in the assembly line in General Motors way back in 1961. The first electrically powered, computer-controlled robot, Stanford Arm (Fig. 2.1c), was developed in 1969. Research on industrial robots started in late 1970s and has gone a long way in the areas of kinematics, path planning, dynamics, simulation, control and design. Denavit and Hartenberg (1955), Hollerbach and Gideon (1983), Goldenberg et al. (1985), Khatib (1986), Ma and Angeles (1990), Coset et al. (2005) and others worked on different issues of kinematics and path planning of industrial robots. The issues of dynamics were addressed by Hollerbach (1980), Luh et al. (1980), Walker and Orin (1982), Khalil et al. (1986), Angeles et al. (1989), and Balafoutis and Patel (1991). Simulation

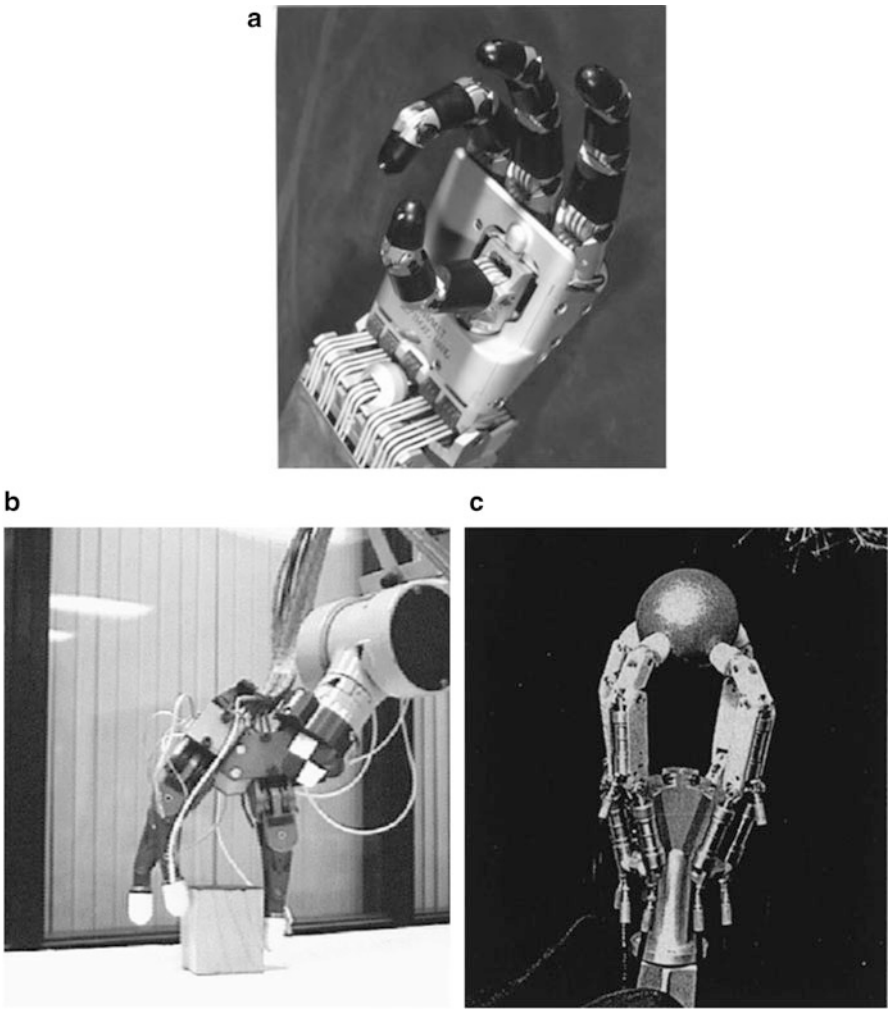


Fig. 2.2 Robotic hands. (**a**) Utah/MIT hand (Jacobsen et al. 1986). (**b**) Stanford/JPL hand (Mason and Salisbury 1985). (**c**) TUM hand (Pfeiffer 1996)

aspects were attempted by Featherstone (1987), Bae and Haug (1987), Rodriguez (1987), Lilly and Orin (1991), McMillan and Orin (1995), and Saha (1997). Issues on control of industrial robots were investigated in detail by Lewis et al. (2004) and Kelly et al. (2005), whereas aspects on design were reported by Asada (1984), Yoshikawa (1985), Kahtib and Burdick (1987), Craig (2006) and Saha et al. (2006). In this book, serial-chain robotic systems are assumed as special cases of a tree-type robotic system.

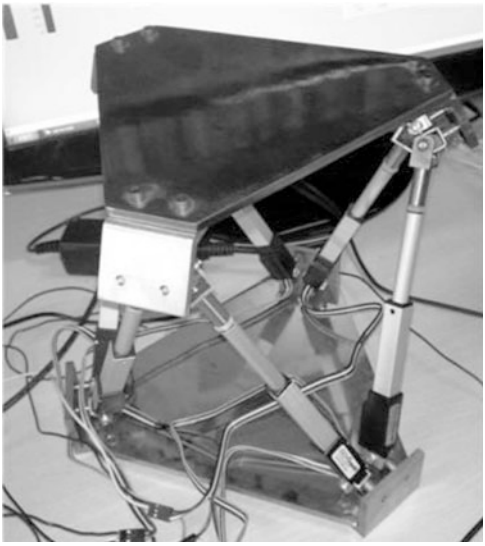


Fig. 2.3 Closed-chain Stewart-platform (Bhagat et al. 2011)

2.1.2 Tree-Type Robotic Hand

A robotic hand shown in Fig. 2.2 is a human-hand-like manipulating system. It has tree-type topology, where several links emanate from a common rigid link, i.e., the palm. The architecture of each finger is the same as that of a serial robot shown in Fig. 2.1. These robotic systems essentially perform the task of restraining an object and manipulating the same. In the task of restraining, fingers play the role of keeping the object rigidly attached to the palm. On the other hand, manipulation involves controlled motion of the grasped object with respect to the palm. This is also known as dexterous manipulation. Early groundwork on robotic hand was laid down by Salisbury and Craig (1982), whereas Mason and Salisbury (1985) and Jacobsen et al. (1986) designed early dexterous systems. As a result, several multi-fingered robotic hands such as the Stanford/JPL hand (Mason and Salisbury 1985), the Utah/MIT hand (Jacobsen et al. 1986), the TUM hand (Pfeiffer 1996) and others were developed. The Utah/MIT hand, Stanford/JPL hand, and TUM hand are shown in Fig. 2.2. Pons et al. (1999) and Bicchi (2000) presented various issues and comprehensive review of dexterous robotic hands. More recently, emphasis has been given on achieving dynamic re-grasping (Hasegawa et al. 2003; and Furukawa et al. 2006) with the help of a robotic hand.

2.1.3 Legged Robots

Desire to make devices that imitate mammals and to explore the unknown terrain has compelled the need for developing legged robots. In early twentieth century,

building walking machines was viewed as the task of designing kinematic linkages that would generate suitable stepping motion. However, later, it had become clear that a linkage providing fixed motion would not do the trick of walking or running. A walking machine, as shown in Fig. 2.4, would need control. The scientific study of legged locomotion began just over 50 year ago when Muybridge (1957) studied the motion of a horse. The first machine that balanced actively was based on automatically controlled inverted pendulum. Miura and Shimoyama (1984) built the first walking machine that balanced actively (Fig. 2.4a). According to Mark Raibert (1986) there are two main purposes for interest in legged locomotion, namely, (1) its potential for high mobility, and (2) the necessity to understand the motions of animals and humans. It is obvious that high mobility aspect is predominant as wheeled locomotion is primarily restricted to level surfaces. Raibert (1986) showed the advantage of legged robots in comparison to the wheeled systems, by illustrating that legs use isolated footholds for support, whereas wheels or tracks require a continuous path of support.

Note that during the flight phase of a legged robot, its degrees-of-freedom are more than the number of actuators, which makes them underactuated. Moreover, the dynamic model of the legged robot is highly nonlinear and it is difficult to model the interaction of the robot with its unknown environment. This makes control of a legged robot difficult. A controller plays major role in achieving stable periodic motion of the legged robot. The goal of the controller is to control velocity and posture of the robot while maintaining a cyclic gait. A statically balanced robot can be controlled in a kinematic way by neglecting the inertial forces. On the contrary, dynamically balanced robot can only be balanced through its motion, by taking into account the inertial forces and manipulating them in the right way. As a result, dynamics plays an important role in the control of legged robots. Research in the field of legged robots has made great stride in the last two decades. Various issues related to dynamics, trajectory planning and control have been addressed by researchers till date to obtain walking or running of the legged robots. As a result, various prototypes of the legged robots have been built. Raibert (1984), Ringrose (1997), Brown and Zeglin (1998), Ahmadi and Buehler (1999) and Hyon et al. (2003) have contributed in the development of monopod hopper. Development of bipedal robot and their issues were reported in the work by Miura and Shimoyama (1984), Vukobratovic et al. (1989), McGeer (1990), Shih et al. (1993), Hirai et al. (1998), Sakagami et al. (2002), Kuroki et al. (2003), Collins and Ruina (2005), and Kaneko et al. (2008). Various issues on quadruped robots have been investigated in the work by Buehler et al. (1998), Fukuoka et al. (2003), Marhefka et al. (2003), and Kimura et al. (2007), whereas development of hexapod robot was attempted by Espenschied et al. (1996), Nelson and Quinn (1998), Cham et al. (2002), and Saranli et al. (2004). Some of the legged robots are shown in Fig 2.4.

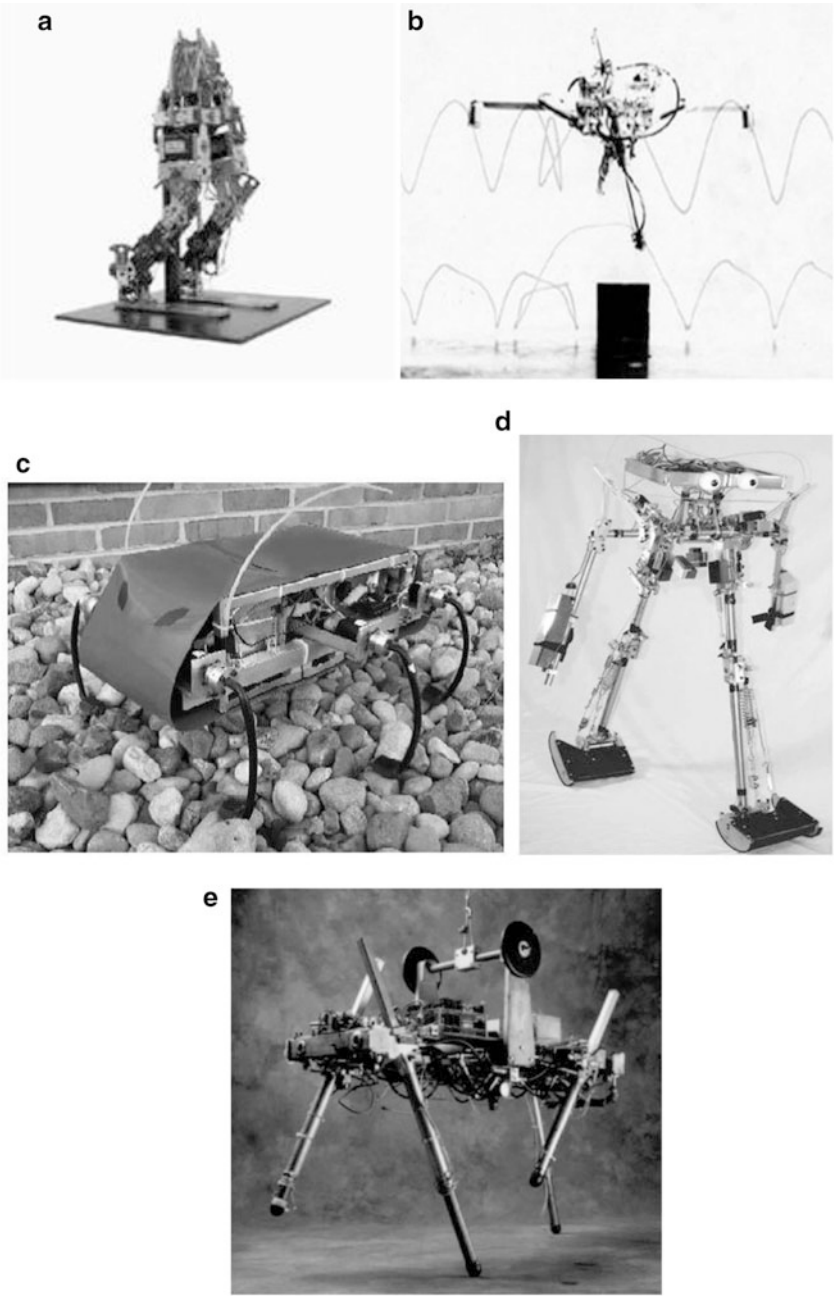


Fig. 2.4 Legged robots. (**a**) Biper4 (Miura and Shimoyama 1984). (**b**) Monopod (Raibert 1986). (**c**) Hexapod (Saranli et al. 2001). (**d**) Biped (Collins and Ruina 2005). (**e**) Quadruped legged robot (Raibert 1986)

2.2 Representations of Rotations

Rotation representation of a rigid link moving in a 3-dimensional Cartesian space is important. Selection of appropriate co-ordinates is vital, particularly, with multiple Degree-of-Freedom (DOF) joints, e.g., universal and spherical joints, connecting two neighboring links. Many schemes are available to represent a rotation in space as shown by Nikravesh (1988), Shuster (1993) and Shabana (2001). The representation of the rotation matrix with the help of nine direction cosines is one such scheme. However, it is not preferred by many as it uses dependent co-ordinates. Use of Euler angles (Shabana 2001) is the alternative choice. It has wide acceptability in the field of aerospace, bio-mechanics, and others due to its independent representation. The use of Euler parameters (Nikravesh 1988) is another popular choice, which uses four parameters though DOF of a rigid body rotation in three dimensional Cartesian space is three. Rodriguez parameters (Shabana 2001) consisting of essentially three independent Euler parameters are also used for rotation representations.

It may be noted that the Euler angles have three independent parameters but suffer from inherent numerical singularities, which, however, can be overcome by switching from one type of Euler angle set to another (Shuster and Oh 1981; Singla et al. 2004). On the other hand, Euler parameters have no singularity but they are not independent, and may suffer from constraint violations that may in turn lead to numerical instability. They are also difficult to visualize.

2.2.1 Denavit-Hartenberg Parameters

In robotic applications, the links are joined mainly with one-DOF joints, e.g., revolute and prismatic. Well-known Denavit-Hartenberg (DH) parameters (Denavit and Hartenberg 1955) are used to define the configuration of a link with respect to its previous one. Many modern robotic systems such as humanoid robot, legged robot, robotic hand, etc. contain multiple-DOF joints, say, universal, cylindrical or spherical joints. Such joints are generally modelled as several intersecting one-DOF joints, i.e., revolute or prismatic (Duffy 1978). Such representations suffer from unnecessary calculations associated with zero lengths and masses of the virtual links that need to be introduced to define the DH parameter of the system.

2.2.2 Euler-Angle-Joints

Recently Shah et al. (2009, 2012b) showed that there exists interesting correlation between Euler Angles and DH parameters. It was shown that the Euler Angles can

be described by the rotation about intersecting revolute joints represented by the DH parameters. Hence, it is formally introduced in this book and their benefits are also presented. The term Euler-Angle-Joints (EAJs) is coined to represent multi-DOF rotary joints. Moreover, it is shown that significant simplifications in the algorithms for dynamics can be obtained by not computing for the zero lengths and masses of the virtual links. This helped in obtaining recursive, fully O(n), forward dynamics algorithm (as will be explained in Sect. 6.2), which would have not been possible to construct by using original definition of the Euler angles (Shabana 2001).

2.3 Dynamic Modeling

Over the last two decades, applications of multibody dynamics have spread over the fields of robotics, automobile, aerospace, bio-mechanics, molecular modeling and many more. With continuous development in the field of robotics, and evolution of complex robotic systems, application of multibody dynamics has become more important. History of dynamics goes back to seventeenth century when Newton in 1686 presented the dynamics of a free particle and later Euler in 1776 introduced the concept of rigid body. This gave birth to Newton-Euler equations of motion. Lagrange in 1788 provided the systematic approach for mathematical formulation of the constrained rigid body systems. Since then multibody dynamics has grown a lot. Comprehensive discussion on dynamic formalisms can be found in the seminal text by Roberson and Schwertassek (1988), Schiehlen (1990), Stejskal and Valasek (1996), and Wittenburg (2008). Recent trends in dynamic formalisms can also be found in the work by Schiehlen (1997), Featherstone and Orin (2000), and Eberhard and Schiehlen (2006).

2.3.1 Equations of Motion

Derivation of the equations of motion is an important step in order to study the dynamics of any system. This is also referred to as dynamic modeling. There are several fundamental methods to formulate the equations of motion of a system under study. For example, Newton-Euler (NE) formulation, Euler-Lagrange principle, Gibbs-Appel approach, Kane's method, and D'Alembert's principle. All the above mentioned approaches, when applied to a robotic system, have their own advantages and disadvantages. Newton-Euler (NE) approach is one of the classical methods for dynamic formulation. This approach is based on the concept of "free-body." If it is constrained, associated forces of constraints are included in the free-body with those, which are externally applied. Mathematically, NE equations of motion lead to two vector equations, which in scalar form is equivalent to three translational equations of motion of the Centre-Of-Mass (COM), and three equations determining the rotational motion of the rigid body. The NE equations are related simply by

the constraint forces and hence for an open-chain system they can easily be solved recursively. For closed-loop systems, however, some of the NE equations need to be solved simultaneously in order to obtain constraint and driving forces. Hence, the use of the NE equations of motion for closed-chain systems is not as efficient as those for serial-chain systems.

Euler-Lagrange (EL) formulation is another classical approach widely used for dynamic modeling. The EL formulation uses the concept of generalized co-ordinates instead of Cartesian co-ordinates. It is based on minimization of a functional called "Lagrangian" which is nothing but the difference between kinetic energy and potential energy of the system at hand. For open-chain systems, where typically the number of generalized coordinates equals the DOF of the system, the constraint forces do not appear in the equations of motion. For the closed-chain systems, however, forces of constraints appear as Lagrange's multipliers. Kane's formulation, which is same as the Lagrange's form of D'Alembert's principle, has also been used by few researchers for the development of the equations of motion. It is found to be more beneficial than other formulations when used for systems with nonholonomic constraints.

2.3.2 Orthogonal Complements

It is pointed out here that NE equations of motion are still found popular in the literature to obtain recursive algorithms. However, it requires solution of the constraint forces and moments, which do not play any role in the motion of a system. Hence, in motion studies extra calculations are required. To avoid such extra calculations, there are formulations proposed in the literature where equations of motion in the EL form are obtained from the NE equations. Huston and Passerello (1974) were first to introduce computer oriented method to reduce the dimension of the unconstrained NE equations by eliminating the constraint forces. Later, Kim and Vanderploeg (1986) derived the equations of motion in terms of relative joint coordinates from Cartesian coordinates through the use of a velocity transformation matrix. A velocity transformation matrix relates linear and angular velocities of the links with joint velocities. It is worth noting that the vector of constraint forces and moments is orthogonal to the columns of the velocity transformation matrix. More precisely, the null space of the transpose of the velocity transformation matrix is orthogonal complement of the column space of the velocity transformation matrix. Hence, velocity transformation matrix is also referred to as orthogonal complement matrix. The concept of 'orthogonal complement' was first used by Hemami and Weimer (1981) for the modeling of nonholonomic systems. Orthogonal complement is not unique; in some approaches it was obtained numerically using singular value decomposition or Eigen value problem (Wehage and Haug 1982; Kamman and Huston 1984; Mani et al. 1985), which is computationally inefficient.

Alternatively, Angeles and Lee (1988) presented a methodology where they derived an orthogonal complement naturally from the velocity constraints. Hence,

the name Natural Orthogonal Complement (NOC) matrix was attached to their methodology. The NOC matrix, when combined with the uncoupled NE equations of motion, leads to minimal-order constrained dynamic equations of motion by eliminating the constraint forces. This facilitates the representation of the equations of motion in Kane's form that is suitable for recursive computation in inverse dynamics or in the EL form that is suitable for forward dynamics and integration. Later, Angeles and Ma (1988), Cyril (1988), Angeles et al. (1989), and Saha and Angeles (1991) showed the effectiveness of the use of the NOC matrix while applied to systems with holonomic and nonholonomic constraints. Subsequently, Saha (1997, 1999a, b) presented the decoupled form of the NOC matrix for the serial chain systems. The two resulting block matrices, namely, an upper block triangular and a block diagonal matrices, are referred to as the Decoupled NOC (DeNOC) matrices. In contrast to NOC, the DeNOC matrices allow one to recursively obtain the analytical expressions of the vectors and matrices appearing in the equations of motion. This in turn helps to analytically decompose the Generalized Inertia Matrix (GIM) of the system at hand, allowing one to obtain a recursive algorithm for forward dynamics. The DeNOC based formulation was later employed for the analysis of closed-chain systems (Saha and Schiehlen 2001; Khan et al. 2005; and Chaudhary and Saha 2007) and flexible manipulators (Mohan and Saha 2007). Blajer et al. (1994) also presented an orthogonal complement based formulation for constrained multibody systems.

2.3.3 Other Formulations

Several other dynamic formulations were also proposed in the literature. For example, Khatib (1987) presented the operational-space formulation, Park et al. (1995) presented robot dynamics using a Lie group formulation, while Stokes and Brockett (1996) derived the equations of motion of a kinematic chain using concepts associated with the special Euclidean group. McPhee (1996) showed how to use linear graph theory in multibody system dynamics. Cameron and Book (1997) described a technique based on Boltzmann-Hamel equations to derive dynamic equations of motion for serial-chain systems.

2.3.4 Open vs. Closed Chains

It is worth noting that the relative joint coordinates are independent in open-chain systems. As a result, a large number of unconstrained NE equations of motion can easily be converted into a reduced form of constrained equations using the velocity transformation matrix. On the other hand, the relative joint coordinates are not independent in the case of closed-chain system as they have to satisfy the loop closure constraints. In order to solve such a problem, methodology based on

cut-joints can be used where the closed kinematic loops are opened by cutting at appropriate joints and incorporating unknown Lagrange multipliers, λ's, at the cut joints as shown by Nikravesh and Gim (1993). Recently, Chaudhry and Saha (2009) proposed a methodology to obtain driving and constraint forces in a closed-chain system using a two-level recursion. They also used the cut-joint method but wrote the dynamic equations of motion using the DeNOC matrices for the system.

As far as the forward dynamics and simulation are concerned, one generally uses Differential Algebraic Equations (DAEs) for the closed-loop system whereas Ordinary Differential Equations (ODEs) are used for an open-chain system. Typically, three different approaches, namely, direct integration method, the constraint stabilization method (Baumgarte 1972; Yu and Chen 2000), and the generalized coordinate partitioning method (Wehage and Haug 1982; Yen and Petzold 1998), are used for the solution of the DAEs.

2.3.5 Dynamics of Legged Robots

In contrast to industrial robots, legged robots have time varying topology, as their feet-contacts vary during the motion cycle. Several methods of dynamic formulation for different legged robots were proposed by Freeman and Orin (1991), Shih et al. (1993), Perrin et al. (1997), McMillan and Orin (1998), Ouezdou et al. (1998), Berkemeier (1998), Hu et al. (2005), Vukobratovic et al. (2007) and others. These methods for formulating the dynamics of legged robots can be divided into two categories. In the first category contact is defined using a hard constrained model. Doing so increases the number of constraint forces in the system. The number is further increased with increase in the number of feet in a legged robot. This results into variable closed-open type of system for different combinations of feet-ground interactions. As a result, a robot has different sets of equations of motion for different configurations of movement, e.g., single support phase, double support phase, flight phase, etc. This is referred to as configuration-dependent approach. In the second approach, a legged robot is modeled as a floating- or mobile-base tree-type robotic system where foot-ground interactions are modeled as external forces and moments. This is referred to as the configuration-independent approach. The latter approach is generally preferred as it allows for a single set of dynamic equations of motion for the legged robots irrespective of their different configurations. Under this approach, as mentioned earlier, foot-ground interactions are modeled as contacts, where reactions on feet can be computed by using analytical method (Baraff 1994; Stewart and Trinkle 2000; Lloyd 2005), impulse based approach (Mirtich and Canny 1995), or penalty based method (Gerritsen et al. 1995; Nigg and Herzog 1999). In penalty based approach, the vertical reactions are approximated by using visco-elastic (spring-damper) model, whereas the horizontal reactions are approximated by using Coulomb or viscous friction. Variety of walking and running surfaces can be simulated by varying the parameters of the spring-damper and coefficient of friction. In this book, configuration independent approach is preferred even though both the approaches are showed for the legged robots.

2.4 Robot Dynamics

As discussed in Sect. 1.3, analysis of robotic systems involves problems of inverse and forward dynamics. Inverse dynamics problem attempts to find the joint torques and forces for a given set of joint motions, which are known for a given set of end-effector motion using inverse kinematic solution, as indicated in Fig. 2.5. Forward dynamics on the other hand attempts to find the joint motions from the knowledge of the external joint torques and forces. These joint motions are used to obtain the end-effector configurations using forward kinematics, as shown in Fig. 2.5. Inverse dynamics problem helps in control and design of a robot, while the forward dynamics enables simulation studies to obtain the configuration of the system at hand, and so helps in evaluation of a new design without actually building the physical robot.

2.4.1 Model-Based Control

With the advent of high speed processors, the dynamic model-based feedforward and computed-torque control schemes have shown better motion control performances of a robotic system than the conventional PID controller (Lewis et al. 2004; and Kelly et al. 2005). Figure 2.6 shows the application of inverse and forward dynamics algorithms to feedforward and computed-torque control schemes. In feedforward control scheme, as shown in Fig. 2.6a, the joint torques computed by using the dynamic model of a robot are fed forward to the controller, while linear servo feedbacks take care of any error in the trajectory to be followed. Selection of control gains is not very straightforward in the case of feedforward control and one may use the design methodology proposed by Kelly et al. (2005) to find the gains. On the other hand, computed-torque control showed in Fig. 2.6b works on the

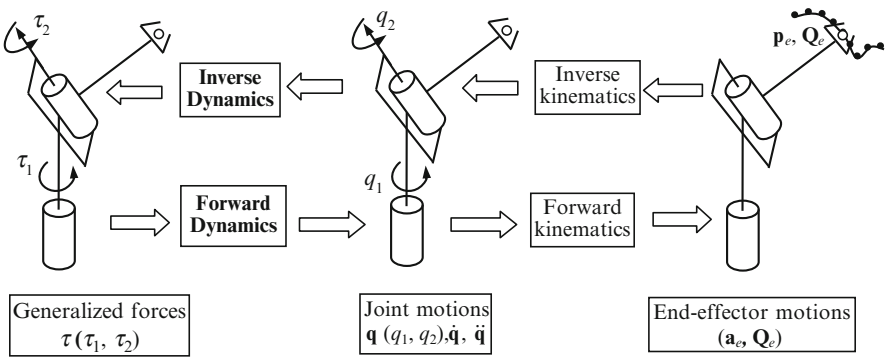


Fig. 2.5 Inverse and forward dynamics

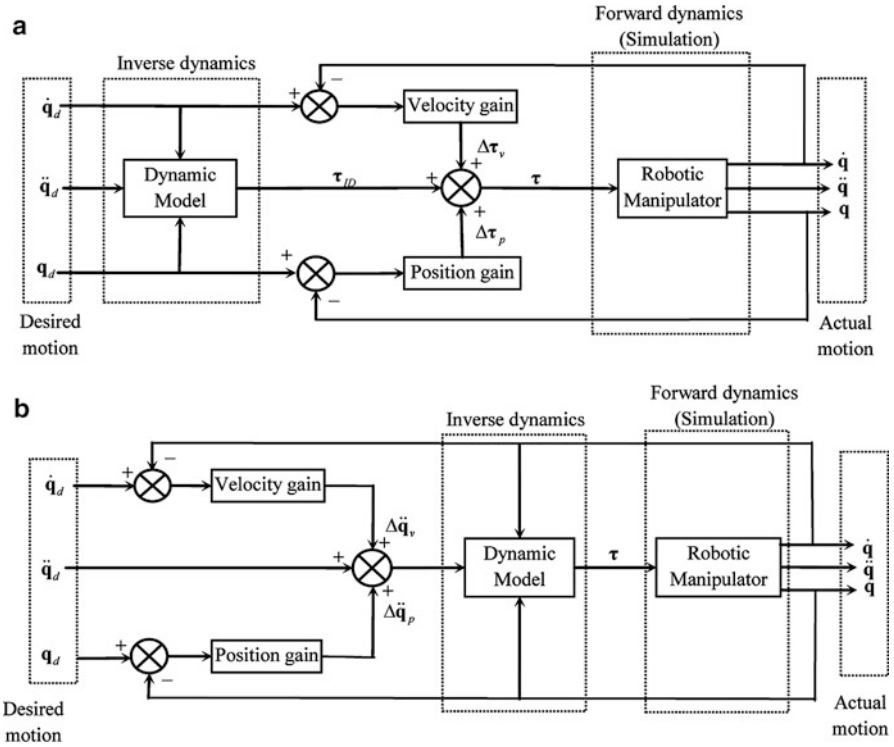


Fig. 2.6 Model-based control laws. (**a**) Feedforward control. (**b**) Computed torque control

principle of the feedback linearization (Craig 2006). In this control scheme, robot dynamic model is used along with linear servo feedback to calculate the driving torque. As the system reduces to a linear one controlled with a simple servo law, setting of control gains is much simpler. Estimating correct dynamic model and its real-time computation are the two major challenges in any model-based control scheme. Even if the dynamic model of the robot is not very accurate, computed-torque scheme eliminates some of the nonlinearities due to robot's inertia, and it is easy to remove the remaining nonlinearity using the associated PID controller.

It is evident from the above two schemes that the inverse dynamics is very essential for computation of control inputs. In the absence of a real robot, forward dynamics followed by numerical integration (i.e., simulation) will allow one to study the actual working of the robot. Accordingly, design modification in the robot can be made at design stage itself. Further, real-time simulation during actual working helps one in taking any preventive control measures. Hence, real-time computation of the robot's inverse and forward dynamics is very important. It is more crucial in the case of legged robots due to their complex dynamics and the presence of intermittent contact. Therefore, efficient computation of the robot dynamics should be of prime importance.

2.4.2 Recursive Algorithms

The formalisms of multibody dynamics, as described by Schiehlen (1990), can be divided into two, (a) symbolical formalism and (b) numerical formalism. In symbolical formalism, the equations of motion are obtained symbolically first, before using them for solving inverse or forward dynamics problems. However, the use of symbolical formalism is limited to smaller systems. In numerical formalism, the inverse and forward problems are solved at each time instant numerically. It may be noted that from numerical point of view recursive formulation is found to be the most efficient. With the advent of digital computers, various recursive formulations for dynamic analyses of robotic systems have emerged. Recursive formulations in dynamics are attractive due to simplicity and computational uniformity. This helps in achieving real-time computations and consequently helps in model-based control.

2.4.3 Inverse Dynamics

Study of dynamics in multibody systems is a classical problem. It was limited to analytical results for systems with small number of links. However, with emergence of high speed computers many methods have evolved to solve the dynamics problem of complex systems with many links. Uicker (1965) and then Kahn and Roth (1971) were first to develop methods based on the Euler-Lagrange equations to predict dynamic behavior of a system consisting of rigid bodies. Thereafter, several recursive algorithms for inverse dynamics were proposed in the literature. Recursive Newton-Euler Algorithm (RNEA) proposed by Luh et al. (1980) is one of the popular algorithms for inverse dynamics. Several other algorithms were also proposed by Walker and Orin (1982), Kane and Levinson (1983), Khalil et al. (1986), Featherstone (1987), Cyril (1988), Angeles et al. (1989), Balafoutis and Patel (1991), Li and Sankar (1992), and Saha (1999b). These algorithms were based on different methodologies but their main objective was to develop an efficient recursive algorithm. Typically, the computational complexity of a recursive inverse dynamics algorithm is of Order (n), where n is the number of joint variables in the system. Recently, Fang and Pollard (2003) presented an efficient algorithm to obtain partial derivatives of inverse dynamics problem with respect to design parameters.

2.4.4 Forward Dynamics

Forward dynamics problem require solution of nonlinear differential equations arising out of the equations of motion. An important step in forward dynamics

is to compute the joint accelerations, which require inversion of the Generalized Inertia Matrix (GIM), i.e., $\ddot{\mathbf{q}} = \mathbf{I}^{-1}\boldsymbol{\varphi}$, where $\ddot{\mathbf{q}}$ is the vector of joint accelerations, $\mathbf{I}$ is the GIM, and $\boldsymbol{\varphi}$ is the vector of generalized forces containing terms due to Coriolis, centrifugal, gravity, and external forces and moments. Forward dynamics algorithms can be divided based on the explicit or implicit inversion of the GIM. In explicit inversion of the GIM, it is inverted numerically using Gaussian elimination or Cholesky decomposition (Strang 1998). This method of inversion commonly leads to $O(n^3)$ algorithm. Walker and Orin (1982) proposed one such algorithm after introducing the concept of composite body. The algorithm is also referred to as Composite Rigid Body Algorithm (CRBA). Later, improved versions of the said algorithm were proposed by Featherstone (1987), Balafoutis and Patel (1991), Lilly and Orin (1991), Lilly (1993), McMillan and Orin (1995) and others. On the contrary, implicit inversion obtains joint accelerations analytically without actually inverting the GIM. Implicit inversion is complex but leads to $O(n)$ algorithms. In literature (Featherstone 2005) these explicit and implicit methods of inversion are also referred to as the "Inertia Matrix Method" and the "Propagation Method", respectively.

2.4.4.1 Implicit Inversion

Implicit inversion method is further divided into two categories. In the first category (type I) (Featherstone 1983; Bae and Haug 1987), one starts with the calculating unknowns, i.e., joint acceleration and joint constraint forces, locally at link-level and then transfers those to adjacent links till a link is obtained where dynamics can be solved locally. Adjacent links are then solved. In the second category (type-II) (Saha 1997), the GIM is factorized and then solved for the joint accelerations using backward and forward substitutions. Both the approaches are analogous. However, the latter looks at the global picture of the whole system while the former focuses at the local link level. Several such algorithms were also proposed in the literature for serial manipulators. Vereshchagin (1975) was first to propose a recursive algorithm for forward dynamics. Later, Armstrong (1979) presented a recursive algorithm for serial systems with 3-degrees-of-freedom spherical joints. Featherstone (1983) introduced recursive algorithm based on the concept of articulated-body inertia from the NE equations of motion of a free-body interacting with its neighboring bodies, whereas Bae and Haug (1987) used variational approach based on the principle of virtual work. In both the approaches, recursion was obtained from the recursive relationships of velocities and accelerations between the neighboring bodies. Later, Brandl et al. (1988), and McMillan and Orin (1995) presented efficient implementation of the Featherstone's articulated body algorithm. Rodriguez (1987) presented a complex algorithm based on Kalman filtering and smoothing theory. Soon after, they (Rodriguez et al. 1991, 1992) introduced the concept of spatial operator algebra for recursive forward dynamics. Rosenthal (1990) and Anderson (1991) also presented recursive formulations based on the use of Kane's equations of motion. Saha (1997) proposed a recursive formulation for the serial-chain systems

based on linear algebra theories and the DeNOC matrices. Several other algorithms were also proposed in the literature by Stejskal and Valasek (1996), Ascher et al. (1997), Critchley and Anderson (2003), Lee and Chirikjian (2005), and Mohan and Saha (2007).

Stelzle et al. (1995) compared different algorithms and showed that for $n > 7$, an O(n) forward dynamics algorithm is more efficient than an O(n^3) algorithm with proper selection of reference point and reference frame. Number of bodies in many robotic applications such as legged robot, humanoid, etc. is much more than seven. Hence, the use of O(n) recursive algorithm is certainly beneficial. Moreover, an O(n) algorithm is also reported to be numerically more stable than an O(n^3) algorithm due to smooth acceleration plots as shown by Ascher et al. (1997), and Mohan and Saha (2007). As a result numerical integrations are much faster in O(n) algorithms which were also reported by Ascher et al. (1997).

2.4.4.2 Other Forward Dynamics Algorithm

Other than explicit and implicit inversion of the GIM in ODE formulation, there is a third approach shown by Nikravesh (1988), Baraff (1996), Shabana (2001), and others, in which the equations of motion of individual bodies are clubbed together with kinematic constraint equations. This leads to the well-known DAE formulation which is more suitable to closed-chain systems. There are also forward dynamics algorithms that use parallel computations, e.g., those proposed by Bae et al. (1988), Fijany et al. (1995), Featherstone (1999), Anderson and Duan (2000), Yamane and Nakamura (2002), and Critchley and Anderson (2004).

2.5 Summary

Rapid development in the field of robotics has given birth to complex robotic systems. These robotic systems are very sophisticated and require intricate control. Hence, it is very important to predict their dynamic behavior at design stage itself. This necessitates efficient computational framework, which can be very useful in design, analysis, simulation, trajectory planning, and control of robotic systems.

These modern day robotic systems contain multiple-DOF joints, e.g., a spherical joint, in contrast to only one-DOF joints as in the case of industrial robots. One approach to model a spherical joint is to represent it by Euler angles. Modeling of a spherical joint can also be done by using three intersecting revolute joint axes that can be identified with DH parameters. It was revealed from the literature (Shah et al. 2012b) that the Euler angles can be described by using intersecting revolute joints, whose axes are identified with DH parameters. An attempt has been made here to correlate them by introducing a concept of Euler-Angle-Joints (EAJs). The concept was later adopted for a unified representation of multiple-DOF joints and to develop efficient recursive algorithms.

The Newton-Euler (NE) equations of motion are popular in dynamic modeling due to their simplicity, but, they do not give the minimal set as compared to Euler-Lagrange equations of motion. The velocity transformation matrix is used for systematic reduction of dimension of the unconstrained NE equations of motion. The DeNOC matrices are one such decoupled form of the velocity transformation matrix that gives the minimal set of equations of motion. The DeNOC-based methodology for serial systems showed several advantages, viz., (1) Analytical expressions for the elements of the vectors and matrices in the dynamic equations of motion (Saha 1999a); (2) Analytical decomposition of the GIM (Saha 1997); (3) Analytical inversion of the inertia matrix which gives deeper insight into the associated dynamics (Saha 1999b); (4) Uniform development of recursive inverse and forward dynamics algorithms; and (5) Extendibility to closed-chain systems (Saha and Schiehlen 2001). Hence, it is used in this book for the dynamic formulation of robotic systems including legged robots. In this book, the DeNOC based formulation for serial-chain systems proposed by Saha (1999b) is extended to dynamic modeling and analysis of more complex tree-type robotic systems consisting of multiple-DOF joints introducing the concept of kinematic modules (Shah et al. 2012a). Each kinematic module consists of a set of serially connected rigid links. In comparison to the work by Saha (1999b), where link-to-link velocity transformation was used to derive the DeNOC matrices, in this book, general module-to-module velocity transformation is used, instead, for the derivation of the DeNOC matrices of a tree-type system. This approach generalizes the concept of link-to-link transformation, and the work done by Saha (1999b) turns out to be a special case of the methodology presented here.

Chapter 3
Euler-Angle-Joints (EAJs)

As frequently noted in the literature on robotics (Sugihara et al. 2002; Kurazume et al. 2003; Vukobratovic et al. 2007; Kwon and Park 2009) and mechanisms (Duffy 1978; Chaudhary and Saha 2007), a higher Degrees-of-Freedom (DOF) joint, say, a universal, a cylindrical or a spherical joint, can be represented using a combination of several intersecting 1-DOF joints. For example, a universal joint also known as Hooke's joint is a combination of two revolute joints, the axes of which intersect at a point, whereas a cylindrical joint is a combination of a revolute joint and a prismatic joint. Similarly, the kinematic behavior of a spherical joint may be simulated by the combination of three revolute joints whose axes intersect at a point. The joint axes can be represented using the popular Denavit and Hartenberg (DH) parameters (Denavit and Hartenberg 1955). For the spherical joints, an alternative approach using the Euler angles can also be adopted, as there are three variables. For spatial rotations, one may also use other minimal set representations like Bryant (or Cardan) angles, Rodriguez parameters, etc. or non-minimal set representation like Euler parameters, quaternion, etc. The non-minimal sets are not considered here due the fact that the dynamic models obtained in this book are desired in minimal sets. The minimal sets, other than Euler/Bryant angles, are discarded here as they do not have direct correlation with the axis-wise rotations. It is worth mentioning that the fundamental difference between the Euler and Bryant angles lies in a fact that the former represents a sequence of rotations about the same axis separated with a rotation about a different axis, denoted as α–β–α, whereas the latter represents the sequence of rotations about three different axes, denoted as α–β–γ. They are also commonly referred to as symmetric and asymmetric sets of Euler angles in the literature. For convenience, the name Euler angles will be referred to both Euler and Bryant angles, hereafter.

Euler Angles are defined as rotations about three orthogonal axes, which may be identified using DH parameters. However, the only difficulty is that, under the DH parameter scheme the variable rotations always occur about Z axis, whereas the Euler Angles are defined by rotation about all three axes, i.e., X, Y and Z. In

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_3,

this chapter, a method will be presented to correlate these rotations so that one can take the advantage of the DH notations in defining the Euler angles to simulate the kinematic behavior of spherical joints. Such correlations are termed as Euler-Angle-Joints (EAJs) (Shah et al. 2012b). The EAJs have the specific advantage that they make a unified representation of multiple-DOF joints and also provide computational efficiency in formulating dynamics algorithms, as highlighted in Sects. 4.2.1 and 6.3, respectively. Such advantage is not available by following the Euler Angle representation.

3.1 Euler Angles

The definition of Euler angles is first revisited in this chapter before the Euler-Angle-Joints are introduced. According to Euler's rotation theorem (Shabana 2001), any three-dimensional spatial rotation can be described using three sequential angles of rotations about three independent axes. These angles of rotation are called Euler angles. Figure 3.1a–c show the sequence of rotations, (a) by an angle θ_1 about Z_M axis, (b) by an angle θ_2 about rotated Y_M axis, and (c) by an angle θ_3 about the current Z_M axis. The O_R-$X_R Y_R Z_R$ and O_M-$X_M Y_M Z_M$ denote the reference frame and moving frame, respectively. The three angles θ_1, θ_2, and θ_3 are called the ZYZ Euler angles. In a similar way, one can define 12 such sets of Euler/Bryant angles depending on the sequence of axes about which the rotations are carried out. They are ZYZ, ZXZ, ZXY, ZYX, YXY, YZY, YXZ, YZX, XYX, XZX, XZY, and XYZ. Out of these twelve Euler angle sets, the ZYZ scheme is widely used in the rotation representation of a robotic system, whereas the ZXZ scheme is more popular in formulating problems in multibody dynamics.

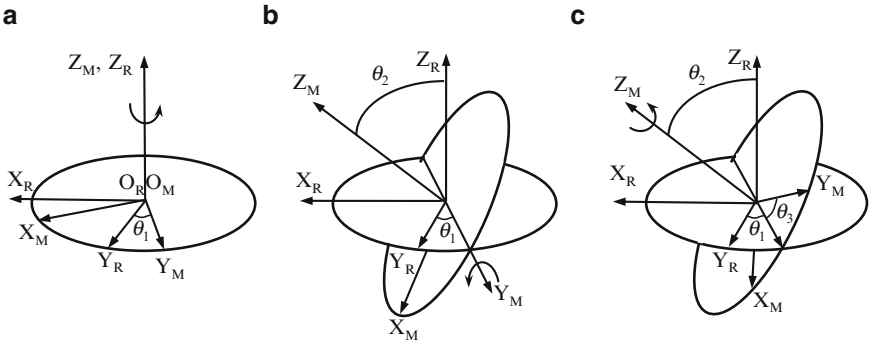


Fig. 3.1 ZYZ Euler angles

If ZYZ scheme of Euler angles is used, the elementary rotation matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$, and $\mathbf{Q}_3$ are given by

$$\mathbf{Q}_1 \equiv \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 \\ S\theta_1 & C\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{Q}_2 \equiv \begin{bmatrix} C\theta_2 & 0 & S\theta_2 \\ 0 & 1 & 0 \\ -S\theta_2 & 0 & C\theta_2 \end{bmatrix}, \text{and } \mathbf{Q}_3 \equiv \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 \\ S\theta_3 & C\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The overall rotation matrix between the frames O_R-$X_R Y_R Z_R$ and O_M-$X_M Y_M Z_M$ can then be obtained by multiplying the elementary rotation matrices, i.e., $\mathbf{Q} \equiv \mathbf{Q}_1\, \mathbf{Q}_2\, \mathbf{Q}_3$. The result is as follows:

$$\mathbf{Q}(\equiv \mathbf{Q}_{ZYZ}) = \begin{bmatrix} C\theta_1 C\theta_2 C\theta_3 - S\theta_1 S\theta_3 & -C\theta_1 C\theta_2 S\theta_3 - S\theta_1 C\theta_3 & C\theta_1 S\theta_2 \\ S\theta_1 C\theta_2 C\theta_3 + C\theta_1 S\theta_3 & -S\theta_1 C\theta_2 S\theta_3 + C\theta_1 C\theta_3 & S\theta_1 S\theta_2 \\ -S\theta_2 C\theta_3 & S\theta_2 S\theta_3 & C\theta_2 \end{bmatrix} \tag{3.2}$$

In a similar fashion, overall rotation matrices for all other Euler angle sets may be obtained. They are shown in Table 3.1.

3.2 Denavit-Hartenberg (DH) Parameters

The DH parameters were originally proposed by Denavit and Hartenberg in 1955, and widely used in robotics for the representation of the configuration of a link with respect to its previous one connected by a one degree-of-freedom (DOF) revolute or prismatic joint. Later, Khalil and Kleinfinger (1986) showed that the DH parameters form a powerful tool for serial robots but lead to ambiguity in the case of closed-loop and tree-type systems. In order to do away with the ambiguity they presented a scheme called Modified DH (MDH) parameters by retaining all the benefits of the original definition. The use of MDH parameters can be found in the work of Craig (2006).

In this section, the MDH notation, as proposed by Khalil and Kleinfinger (1986) is presented, and will be used throughout this book. For that, a co-ordinate frame is attached to each link. The frame O_k-$X_k Y_k Z_k$, denoted by F_k, is rigidly attached to the kth link, as shown in Fig. 3.2. The joint k couples links $(k-1)$ and k. The axis Z_k represents the kth joint axis. Moreover, the origin of F_k, namely, O_k is located at a point where the common normal to Z_k and Z_{k+1} intersects Z_k, whereas the common normal defines the axis X_k. Furthermore, the axis Y_k is such that axes X_k, Y_k, and Z_k form a right-handed triad, i.e., the unit vectors parallel to the axes X_k, Y_k, and Z_k denoted by $\mathbf{i}_k$, $\mathbf{j}_k$, and $\mathbf{k}_k$, satisfy $\mathbf{i}_k \times \mathbf{j}_k = \mathbf{k}_k$. The co-ordinate frame is referred to as

Table 3.1 Rotation matrices using Euler Angles

Euler Angles	Rotation matrix	Euler Angles	Rotation matrix
ZYZ	$\mathbf{Q}_{ZYZ} =$	ZXZ	$\mathbf{Q}_{ZXZ}$
	$\begin{bmatrix} C_1C_2C_3 - S_1S_3 & -C_1C_2S_3 - S_1C_3 & C_1S_2 \\ S_1C_2C_3 + C_1S_3 & -S_1C_2S_3 + C_1C_3 & S_1S_2 \\ -S_2C_3 & S_2S_3 & C_2 \end{bmatrix}$		$\begin{bmatrix} -S_1C_2S_3 + C_1C_3 & -S_1C_2C_3 - C_1S_3 & S_1S_2 \\ C_1C_2S_3 + S_1C_3 & C_1C_2C_3 - S_1S_3 & -C_1S_2 \\ S_2S_3 & S_2C_3 & C_2 \end{bmatrix}$
ZYX	$\mathbf{Q}_{ZYX} \equiv$	ZXY	$\mathbf{Q}_{ZXY} \equiv$
	$\begin{bmatrix} C_1C_2 & C_1S_2S_3 - S_1C_3 & C_1S_2C_3 + S_1S_3 \\ S_1C_2 & S_1S_2S_3 + C_1C_3 & S_1S_2C_3 - C_1S_3 \\ -S_2 & C_2S_3 & C_2C_3 \end{bmatrix}$		$\begin{bmatrix} -S_1S_2S_3 + C_1C_3 & -S_1C_2 & S_1S_2C_3 + C_1S_3 \\ C_1S_2S_3 + S_1C_3 & C_1C_2 & -C_1S_2C_3 + S_1S_3 \\ -C_2S_3 & S_2 & C_2C_3 \end{bmatrix}$
XYX	$\mathbf{Q}_{XYX} \equiv$	XZX	$\mathbf{Q}_{XZX} \equiv$
	$\begin{bmatrix} C_2 & S_2S_3 & S_2C_3 \\ S_1S_2 & -S_1C_2S_3 + C_1C_3 & -S_1C_2C_3 - C_1S_3 \\ -C_1S_2 & C_1C_2S_3 + S_1C_3 & C_1C_2C_3 - S_1S_3 \end{bmatrix}$		$\begin{bmatrix} C_2 & -S_2C_3 & S_2S_3 \\ C_1S_2 & C_1C_2C_3 - S_1S_3 & -C_1C_2S_3 - S_1C_3 \\ S_1S_2 & S_1C_2C_3 + C_1S_3 & -S_1C_2S_3 + C_1C_3 \end{bmatrix}$
XYZ	$\mathbf{Q}_{XYZ} \equiv$	XZY	$\mathbf{Q}_{XZY} \equiv$
	$\begin{bmatrix} C_2C_3 & -C_2S_3 & S_2 \\ S_1S_2C_3 + C_1S_3 & -S_1S_2S_3 + C_1C_3 & -S_1C_2 \\ -C_1S_2C_3 + S_1S_3 & C_1S_2S_3 + S_1C_3 & C_1C_2 \end{bmatrix}$		$\begin{bmatrix} C_2C_3 & -S_2 & C_2S_3 \\ C_1S_2C_3 + S_1S_3 & C_1C_2 & C_1S_2S_3 - S_1C_3 \\ S_1S_2C_3 - C_1S_3 & S_1C_2 & S_1S_2S_3 + C_1C_3 \end{bmatrix}$
YXY	$\mathbf{Q}_{YXY} \equiv$	YZY	$\mathbf{Q}_{YZY} \equiv$
	$\begin{bmatrix} -S_1C_2S_3 + C_1C_3 & S_1S_2 & S_1C_2C_3 + C_1S_3 \\ S_2S_3 & C_2 & -S_2C_3 \\ -C_1C_2S_3 - S_1C_3 & C_1S_2 & C_1C_2C_3 - S_1S_3 \end{bmatrix}$		$\begin{bmatrix} C_1C_2C_3 - S_1S_3 & -C_1S_2 & C_1C_2S_3 + S_1C_3 \\ S_2C_3 & C_2 & S_2S_3 \\ -S_1C_2C_3 - C_1S_3 & S_1S_2 & -S_1C_2S_3 + C_1C_3 \end{bmatrix}$
YXZ	$\mathbf{Q}_{YXZ} \equiv$	YZX	$\mathbf{Q}_{YZX} \equiv$
	$\begin{bmatrix} S_1S_2S_3 + C_1C_3 & S_1S_2C_3 - C_1S_3 & S_1C_2 \\ C_2S_3 & C_2C_3 & -S_2 \\ C_1S_2S_3 - S_1C_3 & C_1S_2C_3 + S_1S_3 & C_1C_2 \end{bmatrix}$		$\begin{bmatrix} C_1C_2 & -C_1S_2C_3 + S_1S_3 & C_1S_2S_3 + S_1C_3 \\ S_2 & C_2C_3 & -C_2S_3 \\ -S_1C_2 & S_1S_2C_3 + C_1S_3 & -S_1S_2S_3 + C_1C_3 \end{bmatrix}$

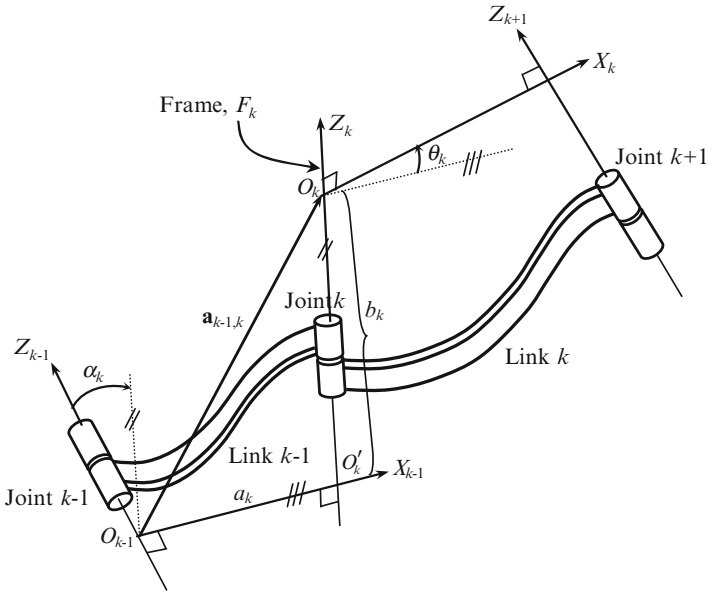


Fig. 3.2 Frame convention for modified Denavit and Hartenberg (DH) parameters

MDH frame or simply DH frame. Note that the frame attached to the fixed link, i.e., O_0-$X_0Y_0Z_0$, can be chosen arbitrarily and hence one can choose Z_0 coincident with Z_1. Once the link frames are established using the above scheme, the position and the orientation between any two frames, say, F_{k-1} and F_k, can be specified using four parameters known as MDH or DH parameters. As the system consists of 1-DOF joints only, out of the four DH parameters one is variable whereas the other three are constant. These parameters are now defined below:

Referring to Fig. 3.2,

- Twist angle (α_k) is the angle between Z_{k-1} and Z_k about X_{k-1}
- Link length (a_k) is the distance from Z_{k-1} to Z_k along X_{k-1}
- Joint offset (b_k) is the distance from X_{k-1} to X_k along Z_k
- Joint angle (θ_k) is the angle between X_{k-1} and X_k about Z_k

Depending on the type of joints, i.e., revolute or prismatic, θ_k or b_k is a variable quantity. Based on the definition of the above DH parameters, the homogeneous transformation matrix ($\mathbf{T}_k$) defining the orientation and position of the frame F_k with respect to frame F_{k-1} can be obtained as

$$\mathbf{T}_k \equiv \begin{bmatrix} \mathbf{Q}_k & [\mathbf{a}_{k-1,k}]_{k-1} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.3}$$

where the 3×3 orientation matrix $\mathbf{Q}_k$ is defined as

$$\mathbf{Q}_k \equiv \mathbf{Q}_{X(\alpha_k)}\mathbf{Q}_{Z(\theta_k)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha_k & -S\alpha_k \\ 0 & S\alpha_k & C\alpha_k \end{bmatrix} \begin{bmatrix} C\theta_k & -S\theta_k & 0 \\ S\theta_k & C\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} C\theta_k & -S\theta_k & 0 \\ S\theta_k C\alpha_k & C\theta_k C\alpha_k & -S\alpha_k \\ S\theta_k S\alpha_k & C\theta_k S\alpha_k & C\alpha_k \end{bmatrix} \tag{3.4}$$

In Eq. (3.3), the position vector $\mathbf{a}_{k-1,k}$, measured from the origin O_{k-1} of the frame F_{k-1} to the origin O_k of fame F_k (Fig. 3.2), in the frame F_{k-1} is given by

$$[\mathbf{a}_{k-1,k}]_{k-1} \equiv \begin{bmatrix} a_k \\ -b_k S\alpha_k \\ b_k C\alpha_k \end{bmatrix} \tag{3.5}$$

Note that, in Eqs. (3.4) and (3.5), $C(\bullet) \equiv \text{Cos}(\bullet)$ and $S(\bullet) \equiv \text{Sin}(\bullet)$.

3.3 Euler-Angle-Joints (EAJs)

As discussed in Sect. 3.2, DH parameters are widely used for representation of links connected by 1-DOF joints. Many modern day robotic systems, however, consist of multiple-DOF joints, e.g., universal and spherical. One approach to model such multiple-DOF joints is to represent them using intersecting 1-DOF revolute joints. Similarly, a spherical joint can be modeled using three intersecting 1-DOF revolute joints axes of which are identified with DH parameters. On the contrary, the Euler angles that are popular in representing any 3-dimensional spatial rotation can be employed to model a spherical joint as well.

Since the modeling of a spherical joint can be done using either Euler angles or three intersecting revolute joint defined with DH parameters, an attempt is made here to correlate them by introducing a concept of Euler-Angle-Joints (EAJs) (Shah et al. 2012b). EAJs are three intersecting revolute joints but their axes are defined using DH parameters such that they provide a rotation equivalent to the one described by a particular set of Euler angles. The concept is later adopted for a unified representation of revolute joint based rotations, i.e., revolute, universal and spherical, and used to develop an efficient fully O(n) recursive algorithm. A fully recursive algorithm is not possible if the representation of a spherical joint is done using the definition of Euler angles. This is highlighted in Sect. 6.1.2.

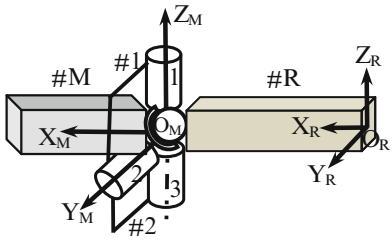


Fig. 3.3 A spherical joint represented by three intersecting revolute joints

3.3.1 DH Parameterization of Euler Angles

As mentioned earlier, the EAJs are intersecting revolute joints that give Euler angle rotations. Architecture of an EAJ is shown in Fig. 3.3, where a spherical joint connects a moving link #M to a reference link #R. The spherical joint is described by using three intersecting revolute joints, where joint 1 connects real link #R to an imaginary link #1, whereas joint 2 connects two imaginary links #1 and #2, and joint 3 connects imaginary link #2 with a real link #M. Two coordinate frames O_M-$X_MY_MZ_M$ and O_R-$X_RY_RZ_R$ are rigidly attached to links #M and #R, respectively. If these frames are denoted as F_M and F_R, the rotation matrix between these frames can be obtained by using any of the Euler angle sets defined in Sect. 3.1. If the ZYZ Euler angle set is used, one obtains the orientation matrix $\mathbf{Q}$ given in Eq. (3.2). Interestingly, the same rotation matrix is obtained by treating the spherical joint as a combination of three intersecting revolute pairs and by defining their axes using the DH parameters.

However, it is worth reminding that the definitions of DH parameters include (1) constant rotation about X axis representing twist angle α, and (2) variable rotation about Z axis representing joint angle θ. As a result, first step towards the development of EAJs, i.e., to correlate DH parameters with axes of Euler angle rotations, is to represent any Euler angle rotation with respect to Z or X axis only. Since the variable joint rotation in the DH parameter definitions is with respect to Z axis only, an Euler angle rotation about X or Y axis has to be expressed as a rotation about Z axis. This can be done by first orienting Z axis parallel to X or Y axis through a fixed set of rotations, typically, by 90°s, followed by the actual or variable rotation about the Z axis. The fixed rotations are then reversed. These will be clear from the following subsection where each elementary rotation about X, Y and Z axes is expressed as a rotation about Z axis only.

3.3.2 Elementary Rotations

In order to obtain Euler-Angle-Joints (EAJs) corresponding to a particular Euler angle set, it is necessary that every elementary rotation, say, about X and Y axes, is finally be considered as rotation about Z axis only. The concept is illustrated below:

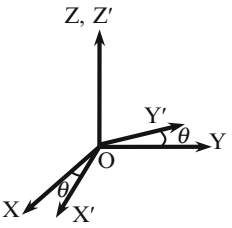


Fig. 3.4 Rotation about Z axis denoted as $Q_{Z(\theta)}$

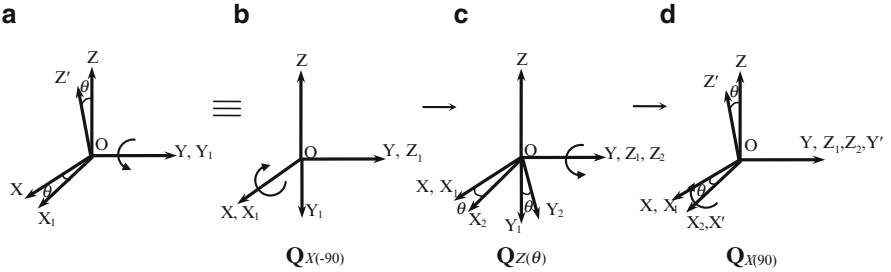


Fig. 3.5 Rotation about Y axis

3.3.2.1 Rotation About Z Axis

Figure 3.4 shows the elementary rotation about Z axis. In DH parameter definition, a rotation is defined about an axis that is identified as Z axis. The rotation matrix defining the rotation of frame O′-X′Y′Z′ with respect to frame O-XYZ is denoted as $Q_{Z(\theta)}$ or Q_{θ} for brevity.

3.3.2.2 Rotation About Y Axis

An elementary rotation about Y axis is shown in Fig. 3.5a. As per the DH nomenclature, a variable rotation has to be about Z axis only. Hence, Z axis of Fig. 3.5a has to be first brought parallel to Y axis before the desired rotation is applied. This can be done by rotation of the frame O-XYZ about X axis by −90° (clockwise rotation is negative), as shown in Fig. 3.5b. The new frame is indicated with O-$X_1Y_1Z_1$. The rotation is indicated with $Q_{X(-90)}$. The desired rotation by an angle θ is now given about Z_1 axis as shown in Fig. 3.5c, where the corresponding rotation is indicated by $Q_{Z(\theta)}$. The new frame is O-$X_2Y_2Z_2$. Finally, to take care of the initial rotation about X axis by −90°, an opposite rotation about X_2 axis is applied, which is indicated by $Q_{X(90)}$, as shown in Fig. 3.5d. The final frame is O′-X′Y′Z′.

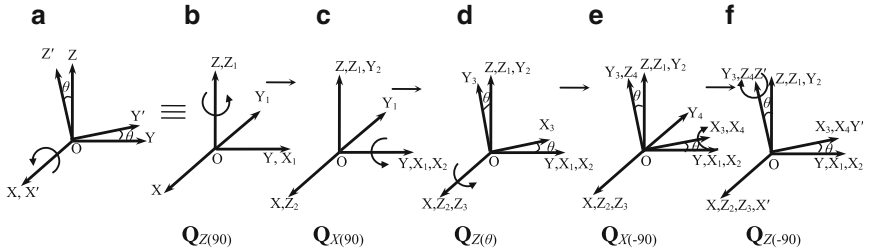


Fig. 3.6 Rotation about X axis

The resultant of three elementary rotations is the desired rotation about Y axis, which is given by

$$\mathbf{Q}_Y = \mathbf{Q}_{-X}\mathbf{Q}_\theta\mathbf{Q}_{+X} \tag{3.6}$$

where for brevity, $\mathbf{Q}_\theta \equiv \mathbf{Q}_{Z(\theta)}$, $\mathbf{Q}_{-X} \equiv \mathbf{Q}_{X(-90)}$, $\mathbf{Q}_{+X} \equiv \mathbf{Q}_{X(90)}$ are used.

3.3.2.3 Rotation About X Axis

Similar to the rotation about Y axis, rotation about X axis is equivalent to five elementary rotations shown in Fig. 3.6. They are

1. Rotation about Z axis by 90° to reach O-$X_1Y_1Z_1$ (Fig. 3.6b).
2. Rotation about X_1 axis by 90° to reach O-$X_2Y_2Z_2$ (Fig. 3.6c).
3. Rotation about Z_2 axis by θ to reach O-$X_3Y_3Z_3$ (Fig. 3.6d).
4. Rotation about X_3 axis by −90° to reach O-$X_4Y_4Z_4$ (Fig. 3.6e).
5. Rotation about Z_4 axis by −90° to reach the final orientation, O-X′Y′Z′ (Fig. 3.6f).

The above five rotations can be represented as

$$\mathbf{Q}_X = \mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_\theta\mathbf{Q}_{-X}\mathbf{Q}_{-Z} \tag{3.7}$$

where $\mathbf{Q}_{+Z} \equiv \mathbf{Q}_{Z(90)}$, $\mathbf{Q}_{+X} \equiv \mathbf{Q}_{X(90)}$, $\mathbf{Q}_\theta \equiv \mathbf{Q}_{Z(\theta)}$, $\mathbf{Q}_{-X} \equiv \mathbf{Q}_{X(-90)}$, and $\mathbf{Q}_{-Z} \equiv \mathbf{Q}_{Z(-90)}$.

3.3.3 Composite Rotations

Resultant of two elementary rotations, say, first about X followed by about Y, denoted as a composite rotation XY, can be obtained by using the elementary rotation representations derived in Sect. 3.3.2. They are shown below:

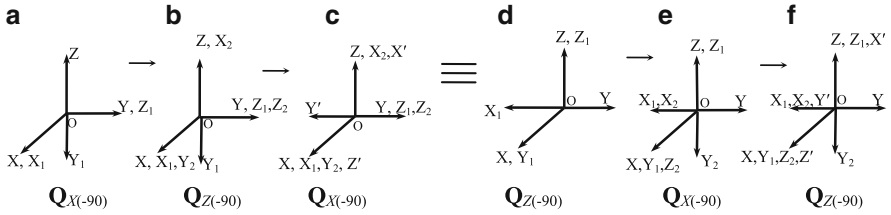


Fig. 3.7 Equivalent transformation

3.3.3.1 Rotation About X and Y Axes

Rotation matrix $\mathbf{Q}_{XY}$ due to rotations about X axis followed by about Y axis can be shown to be a combination of rotation matrices $\mathbf{Q}_X$ and $\mathbf{Q}_Y$ representing the elementary rotations about X and Y axes, obtained in Eqs. (3.7) and (3.6), respectively, i.e.,

$$\mathbf{Q}_{XY} = \mathbf{Q}_X \mathbf{Q}_Y = \overbrace{\mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{\theta_1}\underline{\mathbf{Q}_{-X}}\,\underline{\mathbf{Q}_{-Z}}}^{X}\overbrace{\underline{\mathbf{Q}_{-X}}\,\mathbf{Q}_{\theta_2}\,\mathbf{Q}_{+X}}^{Y} \tag{3.8}$$

where $\mathbf{Q}_{\theta_1}$ and $\mathbf{Q}_{\theta_2}$ represent rotation matrices due to the rotations about X and Y axes by angles θ_1 and θ_2, respectively. Note that in Eq. (3.8), $\mathbf{Q}_{-Z}$ represents the rotation matrix due to rotation about the Z axis by $-90°$. Since DH nomenclature requires all rotations about Z axis must be variable, such a term in the middle will not allow one to express the axes of rotations in terms of DH parameters. A close look into the underlined terms of Eq. (3.8), i.e., $\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}$, however reveals that it is equal to $\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}$, as evident from Fig. 3.7. The same can be proven using matrix representation as well.

For any three sequential rotations of 90° represented by the rotation matrix $\mathbf{Q}_{\alpha(\pm 90)}\,\mathbf{Q}_{\beta(\pm 90)}\,\mathbf{Q}_{\alpha(\pm 90)}$ is equal to $\mathbf{Q}_{\beta(\pm 90)}\,\mathbf{Q}_{\alpha(\pm 90)}\,\mathbf{Q}_{\beta(\pm 90)}$, where α and β represent rotation about X, Y or Z axis.

Hence, replacing $\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}$ by $\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}$, one obtains Eq. (3.8) as

$$\mathbf{Q}_{XY} = \mathbf{Q}_{+Z}\,\mathbf{Q}_{+X}\,\mathbf{Q}_{\theta_1}\,\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}\,\mathbf{Q}_{\theta_2}\,\mathbf{Q}_{+X} \tag{3.9}$$

In Eq. (3.9), even if $\mathbf{Q}_{-Z}$ appears in the middle, but it is next to $\mathbf{Q}_{\theta_1}$ which represents a variable rotation about Z axis. The two consecutive rotations about Z axis is equivalent to one rotation by $(\theta_1 - 90)°$. As a result, no difficulty is faced in defining the DH parameters.

It is worth mentioning here that the rotation matrix $\mathbf{Q}_{YX}$ due to the sequence of rotations about Y axis followed by X axis can be obtained by using the transpose rule of matrix multiplication, i.e.,

$$\mathbf{Q}_{YX} = \mathbf{Q}_{XY}^T = \mathbf{Q}_{-X}\,\mathbf{Q}_{\theta_1}\,\mathbf{Q}_{+Z}\,\mathbf{Q}_{+X}\,\mathbf{Q}_{+Z}\,\mathbf{Q}_{\theta_2}\,\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z} \tag{3.10}$$

where $\mathbf{Q}_{\theta_1}$ actually represents $\mathbf{Q}_{\theta_2}^T (= \mathbf{Q}_{-\theta_2})$ because the first joint rotation in composite rotation is denoted with θ_1. Similarly, $\mathbf{Q}_{\theta_2}$ of Eq. (3.10) can be interpreted. The composite rotation matrix $\mathbf{Q}_{YX}$ can also be verified independently using the expressions given in Eqs. (3.6) and (3.7), as done for $\mathbf{Q}_{XY}$ in Eq. (3.8).

3.3.3.2 Rotations About Y and Z Axes

Rotation about Y axis followed by a rotation about Z axis can be shown to be a combination of rotations about Y and Z axes obtained in the Sect. 3.3.2. This is given by

$$\mathbf{Q}_{YZ} = \mathbf{Q}_Y \mathbf{Q}_Z = \overbrace{\mathbf{Q}_{-X} \mathbf{Q}_{\theta_1} \mathbf{Q}_{+X}}^{Y} \overbrace{\mathbf{Q}_{\theta_2}}^{Z} \tag{3.11}$$

Similar to $\mathbf{Q}_{YX}$ in Eq. (3.10), $\mathbf{Q}_{ZY}$, can be shown to be equal to

$$\mathbf{Q}_{ZY} = \mathbf{Q}_{YZ}^T = \mathbf{Q}_{\theta_1} \mathbf{Q}_{-X} \mathbf{Q}_{\theta_2} \mathbf{Q}_{+X} \tag{3.12}$$

3.3.3.3 Rotations About Z and X Axes

As shown above for XY rotations, $\mathbf{Q}_{XZ}$ can be obtained as

$$\mathbf{Q}_{XZ} = \mathbf{Q}_X \mathbf{Q}_Z = \overbrace{\mathbf{Q}_{+Z} \mathbf{Q}_{+X} \mathbf{Q}_{\theta_1} \mathbf{Q}_{-X} \mathbf{Q}_{-Z}}^{X} \overbrace{\mathbf{Q}_{\theta_2}}^{Z} \tag{3.13}$$

Again using the transpose rule, one can show

$$\mathbf{Q}_{ZX} = \mathbf{Q}_{XZ}^T = \mathbf{Q}_{\theta_1} \mathbf{Q}_{+Z} \mathbf{Q}_{+X} \mathbf{Q}_{\theta_2} \mathbf{Q}_{-X} \mathbf{Q}_{-Z} \tag{3.14}$$

3.4 Euler Angles Using Euler-Angle-Joints (EAJs)

In this section, it will be shown how the Euler angle sets can be obtained by using the concept of Euler-Angle-Joints (EAJs) and the elementary rotations explained in Sects. 3.3.2 and 3.3.3.

3.4.1 ZYZ-EAJs

The rotation matrix for the ZYZ Euler angles set can be obtained as the combination of the rotation representation about Z and the composite rotation YZ, i.e.,

Table 3.2 DH parameters for ZYZ EAJs

	α_k	a_k	b_k	θ_k (JV)
1	0	a_1	0	θ_1
2	−90	0	0	θ_2
3	90	0	0	θ_3

JV joint variable

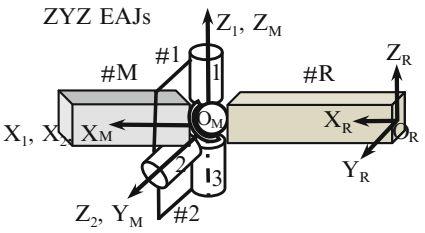


Fig. 3.8 Representation of DH frames for ZYZ EAJs

$$\mathbf{Q}_{ZYZ} = \mathbf{Q}_Z\,\mathbf{Q}_{YZ} = \overbrace{\mathbf{Q}_{\theta_1}}^{Z}\,\overbrace{\mathbf{Q}_{-X}\,\mathbf{Q}_{\theta_2}\,\mathbf{Q}_{+X}\,\mathbf{Q}_{\theta_3}}^{YZ} \tag{3.15}$$

In Eq. (3.15), $\mathbf{Q}_{\theta_k}$ for $k = 1, 2, 3$, represent the rotation matrices corresponding to angles θ_1, θ_2, and θ_3 about Z, Y and Z axes, respectively. Since, the variable rotation is always abut Z axis, θ_k, for $k = 1, 2, 3$, is interpreted as the variable DH parameter or the joint angle. Moreover, the rotation about X axis, i.e., $\mathbf{Q}_{\pm X}$, is interpreted as the rotation due to the twist angle α. Note that, according to the definition of the DH parameters given in Sect. 3.2, the transformation due to the twist angle precedes the one due to the joint angle. This is evident from Eq. (3.4). Therefore, the DH parameters for the ZYZ Euler angles can be extracted from Eq. (3.15) as

- First rotation matrix $\mathbf{Q}_{\theta_1}(\equiv \mathbf{1}\mathbf{Q}_{\theta_1}$, $\mathbf{1}$ being an identity matrix) corresponds to the rotation $\alpha_1 = 0$ about X axis, followed by a rotation of θ_1 about Z axis.
- The next rotation matrices $\mathbf{Q}_{-X}$ and $\mathbf{Q}_{\theta_2}$ correspond to the rotation of $\alpha_2 = -90°$ about X axis, followed by a rotation of θ_2 about Z axis.
- Finally, the rotation matrices $\mathbf{Q}_{+X}$ and $\mathbf{Q}_{\theta_3}$ correspond to a rotation of $\alpha_3 = 90°$ about X axis followed by a rotation of θ_3 about Z axis.

The DH parameters thus obtained for the ZYZ Euler-Angle-Joints (EAJs) are shown in Table 3.2.

Now, a spherical joint connecting the moving link #M with the reference link #R can be represented using the ZYZ EAJs as indicated in Fig. 3.8. Frames F_M, i.e., O_M-$X_M Y_M Z_M$, and F_R, i.e., O_R-$X_R Y_R Z_R$, are rigidly attached to the links #M and #R, respectively. The other frames are assigned in such a manner that they satisfy the DH parameters specified in Table 3.2. These are explained below:

- For $\alpha_1 = 0°$, axis Z_1 is parallel to Z_R and it represents the joint axis of revolute joint 1 connecting the imaginary link #1 to the reference link #R.

- For $\alpha_2 = -90°$, axis Z_2 is orthogonal to Z_1 and it represents the joint axis of the revolute joint 2 connecting the imaginary link #2 to the imaginary link #1. It is worth noting that the axis Z_2 is initially parallel to Y_R, as the second Euler angle rotation is about Y axis.
- For $\alpha_3 = 90°$, the third joint axis is orthogonal to Z_2 and parallel to Z_M. It connects the link #M to the imaginary link #2. The axis Z_M is initially parallel to the axis Z_R as final Euler angle rotation is about Z axis.

The resulting DH frames are shown in Fig. 3.8. With the simultaneous movement of the three revolute joints, the frame F_M attached to #M will then change its orientation with respect to the frame F_R attached to #R. Using the DH parameters in Table 3.2, the rotation matrices $\mathbf{Q}_k$ of Eq. (3.4), for $k = 1, 2, 3$, are obtained as

$$\mathbf{Q}_1 \equiv \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 \\ S\theta_1 & C\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{Q}_2 \equiv \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 \\ 0 & 0 & 1 \\ -S\theta_2 & -C\theta_2 & 0 \end{bmatrix}, \text{and } \mathbf{Q}_3 \equiv \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 \\ 0 & 0 & -1 \\ S\theta_3 & C\theta_3 & 0 \end{bmatrix} \tag{3.16}$$

where $\mathbf{Q}_1$ represents the orientation of Frame F_1 with respect to (w.r.t.) F_R. Similarly, $\mathbf{Q}_2$ and $\mathbf{Q}_3$ represent the orientations of F_2 w.r.t. F_1 and F_M w.r.t. F_2, respectively. Successive multiplications of $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$, i.e., $\mathbf{Q}_{ZYZ} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3$, will then provide the overall orientation of #M w.r.t. #R. Matrix $\mathbf{Q}_{ZYZ}$ is given by

$$\mathbf{Q}_{ZYZ} \equiv \begin{bmatrix} C\theta_1C\theta_2C\theta_3 - S\theta_1S\theta_3 & -C\theta_1C\theta_2S\theta_3 - S\theta_1C\theta_3 & C\theta_1S\theta_2 \\ S\theta_1C\theta_2C\theta_3 + C\theta_1S\theta_3 & -S\theta_1C\theta_2S\theta_3 + C\theta_1C\theta_3 & S\theta_1S\theta_2 \\ -S\theta_2C\theta_3 & S\theta_2S\theta_3 & C\theta_2 \end{bmatrix} \tag{3.17}$$

The matrix elements of $\mathbf{Q}$ given in Eq. (3.17) are nothing but those appearing in Eq. (3.2) that was obtained using the ZYZ Euler angle set. This proves that the three intersecting revolute joints shown in Fig. 3.8 are equivalent to the spherical joint whose rotations are denoted with ZYZ Euler angles. Hence, the revolute joints in Fig. 3.8 are termed as ZYZ Euler-Angle-Joints (EAJs). Note that the above derivations could also be done by combining ZY rotations given by Eq. (3.12), followed by the rotation about Z axis to be denoted as $\mathbf{Q}_{\theta_3}$, i.e.,

$$\mathbf{Q}_{ZYZ} = \mathbf{Q}_{ZY}\mathbf{Q}_Z = \overbrace{\mathbf{Q}_{\theta_1}\mathbf{Q}_{-X}\mathbf{Q}_{\theta_2}\mathbf{Q}_{+X}}^{ZY}\overbrace{\mathbf{Q}_{\theta_3}}^{Z} \tag{3.18}$$

where $\mathbf{Q}_{ZYZ}$ in Eq. (3.18) is nothing but the one obtained in Eq. (3.15) or Eq. (3.17). It may also be proved that $\mathbf{Q}_{\alpha\beta\gamma}$ is associative in nature provided the sequence of rotations is maintained, i.e.,

$$\mathbf{Q}_{\alpha\beta\gamma} = \mathbf{Q}_{\alpha\beta}\mathbf{Q}_{\gamma} = \mathbf{Q}_{\alpha}\mathbf{Q}_{\beta\gamma} \tag{3.19}$$

Table 3.3 DH parameter for ZXZ EAJs

	α_k	a_k	b_k	θ_k (JV)
1	0	a_1	0	$\theta_1 + 90$
2	90	0	0	θ_2
3	−90	0	0	$\theta_3 - 90$

JV joint variable

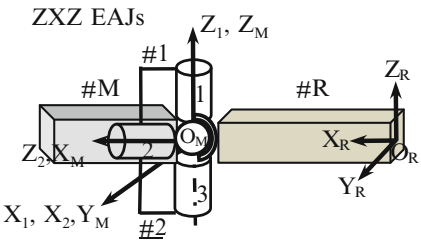


Fig. 3.9 Representation of DH frames for ZXZ EAJs

3.4.2 ZXZ-EAJs

The rotation matrix for the symmetric ZXZ Euler angles can be obtained as the combination of rotation about Z and a composite rotation XZ as

$$\mathbf{Q}_{ZXZ} = \mathbf{Q}_Z \mathbf{Q}_{XZ} = \overbrace{\mathbf{Q}_{\theta_1}}^{Z} \overbrace{\mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{\theta_2}\mathbf{Q}_{-X}\mathbf{Q}_{-Z}\mathbf{Q}_{\theta_3}}^{XZ} \tag{3.20}$$

Once again, in Eq. (3.20), $\mathbf{Q}_{\pm X}$ is the rotation matrix corresponding to the twist angle, and $\mathbf{Q}_{\theta_k}$ and $\mathbf{Q}_{\pm Z}$ correspond to the joint angles. Based on Eq. (3.20), the DH parameters are shown in Table 3.3.

Like the previous section, one can relate the rotation matrix by using ZXZ Euler angles as a combination of three intersecting revolute joints. The representations of DH frames, however, differ from what has been done for ZYZ EAJs. Assignment of the DH frames for the three intersecting revolute joints representing ZXZ EAJs is shown in Fig. 3.9.

The corresponding rotation matrices between F_1 and F_R, F_2 and F_1, and F_M and F_2 are given below:

$$\mathbf{Q}_1 \equiv \begin{bmatrix} -S\theta_1 & -C\theta_1 & 0 \\ C\theta_1 & -S\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{Q}_2 \equiv \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 \\ 0 & 0 & -1 \\ S\theta_2 & C\theta_2 & 0 \end{bmatrix}, \text{ and } \mathbf{Q}_3 \equiv \begin{bmatrix} S\theta_3 & C\theta_3 & 0 \\ 0 & 0 & 1 \\ C\theta_3 & -S\theta_3 & 0 \end{bmatrix} \tag{3.21}$$

The overall rotation matrix, $\mathbf{Q}_{ZXZ}$, between the frames F_M and F_R is then given by

$$\mathbf{Q}_{ZXZ} = \begin{bmatrix} -S\theta_1 C\theta_2 S\theta_3 + C\theta_1 C\theta_3 & -S\theta_1 C\theta_2 C\theta_3 - C\theta_1 S\theta_3 & S\theta_1 S\theta_2 \\ C\theta_1 C\theta_2 S\theta_3 + S\theta_1 C\theta_3 & C\theta_1 C\theta_2 C\theta_3 - S\theta_1 S\theta_3 & -C\theta_1 S\theta_2 \\ S\theta_2 S\theta_3 & S\theta_2 C\theta_3 & C\theta_2 \end{bmatrix} \tag{3.22}$$

Table 3.4 DH parameters for ZXY EAJs

	α_k	a_k	b_k	θ_k (JV)
1	0	a_1	0	$\theta_1 + 90$
2	90	0	0	$\theta_2 - 90$
3	-90	0	0	$\theta_3 - 90$
4	90	0	0	0 (Constant)

JV joint variable

Here too, it can be seen that the orientation matrix $\mathbf{Q}_{ZXZ}$ of Eq. (3.22) is same as that of the ZXZ Euler angles set given in Table 3.1.

3.4.3 ZXY-EAJs

In the previous subsections, ZYZ and ZXZ are referred to as symmetric Euler angle sets. In order to show how EAJs evolve for asymmetric Euler angle or Bryant (Cardan) angle set, the ZXY set is considered next in which the third rotation is not about Z. Similar to the previous subsections rotation matrix for the ZXY Euler angles can also be obtained as a combination of the rotation matrix $\mathbf{Q}_Z$ and the composite rotation matrix $\mathbf{Q}_{XY}$ as in Eq. (3.9). Hence

$$\mathbf{Q}_{ZXY} = \mathbf{Q}_Z \mathbf{Q}_{XY} = \overbrace{\mathbf{Q}_{\theta_1}}^{Z} \overbrace{\mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{\theta_2}\mathbf{Q}_{-Z}\mathbf{Q}_{-X}\mathbf{Q}_{-Z}\mathbf{Q}_{\theta_3}\mathbf{Q}_{+X}}^{XY} \tag{3.23}$$

The above equation forms the basis for the definition of the DH parameters for the ZXY EAJs, which are extracted from Eq. (3.23) as

- The first two terms $\mathbf{Q}_{\theta_1} (\equiv \mathbf{1}\mathbf{Q}_{\theta_1})$ and $\mathbf{Q}_{+Z}$ correspond to the twist angle $\alpha_1 = 0°$ and the net joint angle of $(\theta_1 + 90°)$.
- Next, three terms $\mathbf{Q}_{+X}$, $\mathbf{Q}_{\theta_2}$ and $\mathbf{Q}_{-Z}$ correspond to the twist angle $\alpha_2 = 90°$, and the net joint angle of $(\theta_2 - 90°)$, respectively.
- The terms $\mathbf{Q}_{-X}$, $\mathbf{Q}_{-Z}$ and $\mathbf{Q}_{\theta_3}$ correspond to twist angle $\alpha_3 = -90°$ and the net joint angle of $(\theta_3 - 90°)$, respectively.
- Finally, a constant rotation matrix $\mathbf{Q}_{+X}$ remains. This necessitates an additional set of DH parameters. It is worth mentioning here that the system under study has three joints and 3-DOF. As a result, three sets of DH parameters should be sufficient for defining its configuration. However while obtaining the DH parameters for ZXY Euler angle the term $\mathbf{Q}_{+X}$ is inevitable. It corresponds to a fourth set of all constant DH parameters, namely, $\alpha_4 = 90°$, $\theta_4 = 0°$. Presence of a constant rotation matrix $\mathbf{Q}_{+X}$ is the result of the final rotation of the Euler angles about Y axis. This is an important observation.

The DH parameters for ZXY Euler angles are now shown in Table 3.4. Based on the DH parameters obtained in Table 3.4, the ZXY EAJs are represented in Fig. 3.10, for which the DH frames are assigned using the following rules:

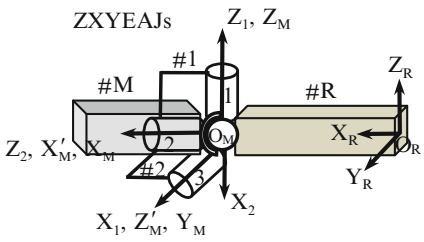


Fig. 3.10 Representation of DH frames for ZXY EAJs

- For $\alpha_1 = 0°$, axis Z_1 is parallel to Z_R and it represents the joint axis of revolute joint 1 connecting the imaginary link #1 to the reference link #R. Moreover, for the joint angle $(\theta_1 + 90°)$, X_1 is perpendicular to X_R initially.
- For $\alpha_2 = 90°$, axis Z_2 is perpendicular to Z_1 and it represents the joint axis of revolute joint 2 connecting the imaginary link #2 to the imaginary link #1. Once again, X_2 is perpendicular to X_1 for initial joint angle of $(\theta_2 - 90°)$. It is worth noting that axis Z_2 in its initial configuration is parallel to X_R as the second Euler angle rotation is about X axis.
- For $\alpha_3 = -90°$, the third joint axis is perpendicular to Z_2. It is important to note that the third frame is attached to the moving link #M, however Z_M cannot be chosen as the third joint axis, as $X_2 || Z_M$, which violates the definition of DH parameters. As a result, an intermediate frame O_M-$X'_M Y'_M Z'_M$ was assigned whose Z'_M axis is perpendicular to Z_2 and X_2 both. Axis Z'_M is parallel to Y_R in the initial configuration. Both the frames F'_M and F_M are attached to the moving link #M and have constant rotation between them.
- Finally, for $\alpha_4 = 90°$, i.e., corresponding to $\mathbf{Q}_{+X}$, the location of frame O_M-$X_M Y_M Z_M$ is obtained by 90° rotation of the frame O_M-$X'_M Y'_M Z'_M$ about X'_M.

Additional constant rotation matrix at the end, i.e., $\mathbf{Q}_{+X}$, may be interpreted as the rotation required to make the DH frame O_M-$X'_M Y'_M Z'_M$ parallel to the moving frame O_M-$X_M Y_M Z_M$ in order to obtain the rotation matrix same as ZXY Euler angles. The above DH parameters are then used to obtain rotation matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ and $\mathbf{Q}_{+X}$ representing the rotations between the frames F_R and F_1, F_1 and F_2, and F_2 and F'_M, and F'_M and F_M, respectively, as

$$\mathbf{Q}_1 \equiv \begin{bmatrix} -S\theta_1 & -C\theta_1 & 0 \\ C\theta_1 & -S\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{Q}_2 \equiv \begin{bmatrix} S\theta_2 & C\theta_2 & 0 \\ 0 & 0 & -1 \\ -C\theta_2 & S\theta_2 & 0 \end{bmatrix},$$

$$\mathbf{Q}_3 \equiv \begin{bmatrix} S\theta_3 & C\theta_3 & 0 \\ 0 & 0 & 1 \\ C\theta_3 & -S\theta_3 & 0 \end{bmatrix}, \text{and } \mathbf{Q}_{+X} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.24}$$

The overall rotation matrix between F_R and F_M is obtained from the above successive frame rotations as

$$\mathbf{Q}_{ZXY} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{+X} \equiv \begin{bmatrix} -S\theta_1S\theta_2S\theta_3 + C\theta_1C\theta_3 & -S\theta_1C\theta_2 & S\theta_1S\theta_2C\theta_3 + C\theta_1S\theta_3 \\ C\theta_1S\theta_2S\theta_3 + S\theta_1C\theta_3 & C\theta_1C\theta_2 & -C\theta_1S\theta_2C\theta_3 + S\theta_1S\theta_3 \\ -C\theta_2S\theta_3 & S\theta_2 & C\theta_2C\theta_3 \end{bmatrix} \tag{3.25}$$

The matrix expression of $\mathbf{Q}_{ZXY}$ in Eq. (3.25) is same as the one corresponding to ZXY Euler angles shown in Table 3.1. Interestingly, it may be concluded that one always encounters an additional constant rotation matrix $\mathbf{Q}_{+X}$ when DH parameterization of the Euler Angles is done with final rotation about Y, e.g., XZY, YXY, etc.

3.4.4 XYX-EAJs

The existence of constant rotation matrix for the ZXY EAJs, as shown in Eq. (3.25), leads to the exploration of how the EAJs evolve when the initial and/or final rotation is given about the X axis. Hence, the XYX Euler angles are considered next. Rotation matrix for the XYX Euler angles is written as a combination of rotation matrix $\mathbf{Q}_X$ and composite rotation matrix $\mathbf{Q}_{YX}$ as obtained in Eqs. (3.7) and (3.10), respectively, i.e.,

$$\mathbf{Q}_{XYX} = \mathbf{Q}_X\mathbf{Q}_{YX} = \overbrace{\mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{\theta_1}\underline{\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}}}^{X}\overbrace{\underline{\mathbf{Q}_{-X}}\mathbf{Q}_{\theta_2}\mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{+Z}\mathbf{Q}_{\theta_3}\mathbf{Q}_{-X}\mathbf{Q}_{-Z}}^{YX} \tag{3.26}$$

where the underlined terms $\underline{\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}}$ can be replaced by $\mathbf{Q}_{-Z}\,\mathbf{Q}_{-X}\,\mathbf{Q}_{-Z}$, using the property shown in Fig. 3.7. As a result, Eq. (3.26) is rewritten as

$$\mathbf{Q}_{XYX} = \mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{\theta_1}\mathbf{Q}_{-Z}\mathbf{Q}_{-X}\underline{\mathbf{Q}_{-Z}\,\mathbf{Q}_{\theta_2}\mathbf{Q}_{+Z}}\mathbf{Q}_{+X}\mathbf{Q}_{+Z}\mathbf{Q}_{\theta_3}\mathbf{Q}_{-X}\mathbf{Q}_{-Z} \tag{3.27}$$

Simplifying further, the terms $\underline{\mathbf{Q}_{-Z}\,\mathbf{Q}_{\theta_2}\mathbf{Q}_{+Z}}$ may be clubbed to result in the rotation matrix as $\mathbf{Q}_{\theta_2}$ for a net rotation of θ_2 about Z axis. Thus $\mathbf{Q}_{XYX}$ is given by

$$\mathbf{Q}_{XYX} = \mathbf{Q}_{+Z}\mathbf{Q}_{+X}\mathbf{Q}_{\theta_1}\mathbf{Q}_{-Z}\mathbf{Q}_{-X}\mathbf{Q}_{\theta_2}\mathbf{Q}_{+X}\mathbf{Q}_{+Z}\mathbf{Q}_{\theta_3}\mathbf{Q}_{-X}\mathbf{Q}_{-Z} \tag{3.28}$$

The DH parameters for the XYX EAJs are then extracted with the help of Eq. (3.28), i.e.,

- The first term $\mathbf{Q}_{+Z}(\equiv\mathbf{1}\mathbf{Q}_{+Z}$, $\mathbf{1}$ being an identity matrix) corresponds to the twist angle $\alpha_0 = 0°$ and the joint angle $\theta_0 = 90°$.

Table 3.5 DH parameters for XYX EAJs

	α_k	a_k	b_k	θ_k (JV)
0	0	0	0	90 (Constant)
1	90	0	b_1	$\theta_1 - 90$
2	−90	0	0	θ_2
3	90	0	0	$\theta_3 + 90$
4	−90	0	0	−90 (Constant)

JV joint variable

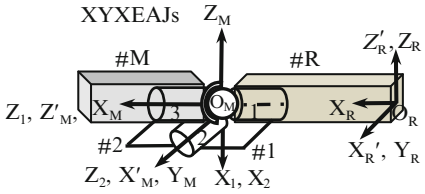


Fig. 3.11 Representation of DH frames for XYX EAJs

- The terms $\mathbf{Q}_{+X}$, $\mathbf{Q}_{\theta_1}$, and $\mathbf{Q}_{-Z}$ are corresponding to twist angle $\alpha_1 = 90°$ and net joint angle of $(\theta_1 - 90°)$, respectively.
- Next, the terms $\mathbf{Q}_{-X}$ and $\mathbf{Q}_{\theta_2}$ correspond to twist angle $\alpha_2 = -90°$ and joint angle of θ_2.
- Subsequently, the terms $\mathbf{Q}_{+X}$, $\mathbf{Q}_{+Z}$ and $\mathbf{Q}_{\theta_3}$ correspond to twist angle $\alpha_3 = 90°$ and net joint angle of $(\theta_3 + 90)$, respectively.
- Finally, the constant terms $\mathbf{Q}_{-X}$ and $\mathbf{Q}_{-Z}$ correspond to twist angle $\alpha_4 = -90°$ and joint angle of $\theta_4 = -90°$.

It is worth noting here that while constructing XYX EAJs, the constant orientation terms $\mathbf{Q}_{+Z}$ in the beginning and $\mathbf{Q}_{-X}\mathbf{Q}_{-Z}$ at the end are inevitable. The resulting DH parameters for the XYX EAJs are given in Table 3.5. This results into two additional constant set of DH parameters in the beginning and at the end as may be seen in Table 3.5.

Like any other Euler angle sets, shown earlier, $\mathbf{Q}_{\mathrm{XYX}}$ can also be obtained by XY and X rotations, respectively. The resulting XYX EAJs are shown in Fig. 3.11, where the DH frames are assigned using the following rules:

- For the constant terms $\alpha_0 = 0°$ and $\theta_0 = 90°$, i.e., $\mathbf{1}\mathbf{Q}_{+Z}$, the DH frame $\mathrm{O'_R\text{-}X'_R Y'_R Z'_R}$ is obtained from frame $\mathrm{O_R\text{-}X_R Y_R Z_R}$ by rotating $\mathrm{O_R\text{-}X_R Y_R Z_R}$ by 90° about $\mathrm{Z_R}$. Both the frames are rigidly attached to the reference link #R.
- For $\alpha_1 = 90°$, axis $\mathrm{Z_1}$ is perpendicular to $\mathrm{Z'_R}$ and it represents the joint axis of revolute joint 1 connecting the imaginary link #1 to the reference link #R. Moreover, for the joint angle $(\theta_1 - 90°)$, $\mathrm{X_1}$ is perpendicular to $\mathrm{X'_R}$ initially.
- For $\alpha_2 = -90°$, axis $\mathrm{Z_2}$ is perpendicular to $\mathrm{Z_1}$ and it represents the joint axis of revolute joint 2 connecting the imaginary link #2 to the imaginary link #1. It is worth noting that axis $\mathrm{Z_2}$ in initial configuration is parallel to $\mathrm{Y_R}$ as second Euler angle rotation is about Y axis.

- For $\alpha_3 = 90°$ and the net joint angle of $(\theta_3 + 90°)$, the third joint axis is perpendicular to Z_2. Third frame is attached to the moving link #M, however, Z_M is not the obvious choice for the third joint axis as $X_2 || Z_M$, which violates the definition of the DH notations. Hence, frame O_M-$X'_M Y'_M Z'_M$ is assigned such that $Z'_M \perp$ (Z_2 and X_2) and obviously axis Z'_M is parallel to X_R in the initial configuration.
- Finally, $\alpha_4 = -90°$ and $\theta_4 = -90°$, i.e., the result of constant rotation matrix $\mathbf{Q}_{-X}\mathbf{Q}_{-Z}$, give constant orientation of frame O_M-$X_M Y_M Z_M$, with respect to O_M-$X'_M Y'_M Z'_M$. Both the frames are attached to the moving link #M.

Using the DH parameters in Table 3.5, the rotation matrices $\mathbf{Q}_{+Z}$, $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$, and $\mathbf{Q}_{-XZ}$ representing rotations between frames F_R and F'_R, F'_R and F_1, F_1 and F_2, F_2 and F'_M, and F'_M and F_M, respectively, are obtained using Eq. (3.4) as

$$\mathbf{Q}_{+Z} \equiv \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{Q}_1 \equiv \begin{bmatrix} S\theta_1 & C\theta_1 & 0 \\ 0 & 0 & -1 \\ -C\theta_1 & S\theta_1 & 0 \end{bmatrix}, \mathbf{Q}_2 \equiv \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 \\ 0 & 0 & 1 \\ -S\theta_2 & -C\theta_2 & 0 \end{bmatrix},$$
$$\mathbf{Q}_3 \equiv \begin{bmatrix} -S\theta_3 & -C\theta_3 & 0 \\ 0 & 0 & -1 \\ C\theta_3 & -S\theta_3 & 0 \end{bmatrix}, \mathbf{Q}_{-XZ} \equiv \mathbf{Q}_{-X}\mathbf{Q}_{-Z} \equiv \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.29}$$

The overall rotation matrix between the frames F_R and F_M is obtained as

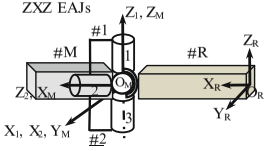
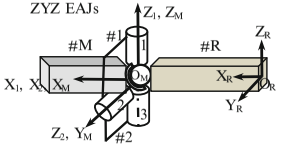
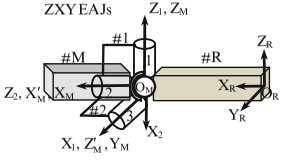
$$\mathbf{Q}_{XYX} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{-XZ}$$
$$\equiv \begin{bmatrix} C\theta_2 & S\theta_2 S\theta_3 & S\theta_2 C\theta_3 \\ S\theta_1 S\theta_2 & -S\theta_1 C\theta_2 S\theta_3 + C\theta_1 C\theta_3 & -S\theta_1 C\theta_2 C\theta_3 - C\theta_1 S\theta_3 \\ -C\theta_1 S\theta_2 & C\theta_1 C\theta_2 S\theta_3 + S\theta_1 C\theta_3 & C\theta_1 C\theta_2 C\theta_3 - S\theta_1 S\theta_3 \end{bmatrix} \tag{3.30}$$

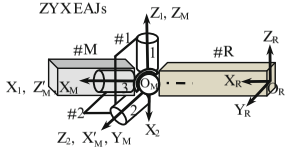
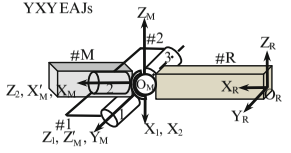
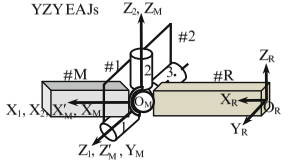
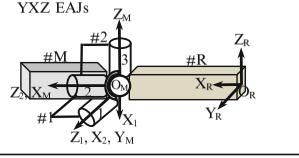
where $\mathbf{Q}_{XYX}$ represents the XYX Euler angles. Interestingly enough, EAJs for the Euler angle sets with first rotation about X, e.g., XYX, XYZ, XZY, require constant orientation of Q_{+Z} in the beginning, whereas the one with third rotation about X, e.g., XYX, ZYX, YZX, require constant orientation of Q_{-XZ} at the end.

3.4.5 Other-EAJs

Similarly, Euler-Angle-Joints (EAJs) (Shah et al. 2012b) for all 12 Euler angle sets were obtained. Table 3.6 shows the EAJs and their overall rotation matrices $\mathbf{Q}_{\alpha\beta\gamma}$,

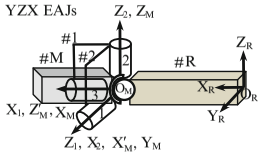
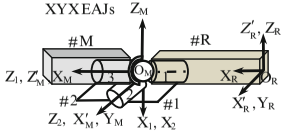
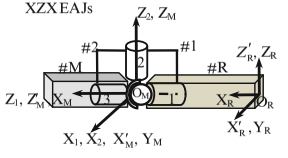
Table 3.6 Euler-Angle-Joints

	Euler Angle	DH parameters	α_k	θ_k	Equivalent EAJs	Rotation matrix using the DH parameters of the EAJs
1	ZXZ	1	0	$\theta_1 + 90$		$\mathbf{Q}_{ZXZ} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3$ $\equiv \begin{bmatrix} -S_1C_2S_3 + C_1C_3 & -S_1C_2C_3 - C_1S_3 & S_1S_2 \\ C_1C_2S_3 + S_1C_3 & C_1C_2C_3 - S_1S_3 & -C_1S_2 \\ S_2S_3 & S_2C_3 & C_2 \end{bmatrix}$
		2	90	θ_2		
		3	−90	$\theta_3 - 90$		
2	ZYZ	1	0	θ_1		$\mathbf{Q}_{ZYZ} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3$ $\equiv \begin{bmatrix} C_1C_2C_3 - S_1S_3 & -C_1C_2S_3 - S_1C_3 & C_1S_2 \\ S_1C_2C_3 + C_1S_3 & -S_1C_2S_3 + C_1C_3 & S_1S_2 \\ -S_2C_3 & S_2S_3 & C_2 \end{bmatrix}$
		2	−90	θ_2		
		3	90	θ_3		
3	ZXY	1	0	$\theta_1 + 90$		$\mathbf{Q}_{ZXY} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{+X}$ $\equiv \begin{bmatrix} -S_1S_2S_3 + C_1C_3 & -S_1C_2 & S_1S_2C_3 + C_1S_3 \\ C_1S_2S_3 + S_1C_3 & C_1C_2 & -C_1S_2C_3 + S_1S_3 \\ -C_2S_3 & S_2 & C_2C_3 \end{bmatrix}$
		2	90	$\theta_2 - 90$		
		3	−90	$\theta_3 - 90$		
		4	90	0		

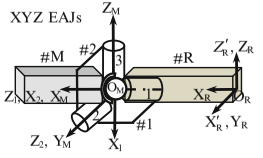
4	ZYX	1	0	θ_1		$\mathbf{Q}_{ZYX} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{-XZ}$ $\equiv \begin{bmatrix} C_1C_2 & C_1S_2S_3 - S_1C_3 & C_1S_2C_3 + S_1S_3 \\ S_1C_2 & S_1S_2S_3 + C_1C_3 & S_1S_2C_3 - C_1S_3 \\ -S_2 & C_2S_3 & C_2C_3 \end{bmatrix}$
		2	−90	$\theta_2 + 90$		
		3	90	$\theta_3 + 90$		
		4	−90	−90		
5	YXY	1	−90	$\theta_1 + 90$		$\mathbf{Q}_{YXY} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{+X}$ $\equiv \begin{bmatrix} -S_1C_2S_3 + C_1C_3 & S_1S_2 & S_1C_2C_3 + C_1S_3 \\ S_2S_3 & C_2 & -S_2C_3 \\ -C_1C_2S_3 - S_1C_3 & C_1S_2 & C_1C_2C_3 - S_1S_3 \end{bmatrix}$
		2	90	θ_2		
		3	−90	$\theta_3 - 90$		
		4	90	0		
6	YZY	1	−90	θ_1		$\mathbf{Q}_{YZY} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{+X}$ $\equiv \begin{bmatrix} C_1C_2C_3 - S_1S_3 & -C_1S_2 & C_1C_2S_3 + S_1C_3 \\ S_2C_3 & C_2 & S_2S_3 \\ -S_1C_2C_3 - C_1S_3 & S_1S_2 & -S_1C_2S_3 + C_1C_3 \end{bmatrix}$
		2	90	θ_2		
		3	−90	θ_3		
		4	90	0		
7	YXZ	1	−90	$\theta_1 + 90$		$\mathbf{Q}_{YXZ} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3$ $\begin{bmatrix} S_1S_2S_3 + C_1C_3 & S_1S_2C_3 - C_1S_3 & S_1C_2 \\ C_2S_3 & C_2C_3 & -S_2 \\ C_1S_2S_3 - S_1C_3 & C_1S_2C_3 + S_1S_3 & C_1C_2 \end{bmatrix}$
		2	90	$\theta_2 + 90$		
		3	−90	$\theta_3 - 90$		

(continued)

Table 3.6 (continued)

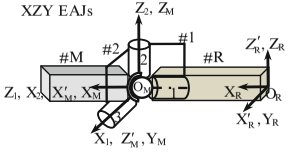
	Euler Angle	DH parameters			Equivalent EAJs	Rotation matrix using the DH parameters of the EAJs
			α_k	θ_k		
8	YZX	1	−90	θ_1		$\mathbf{Q}_{YZX} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{-XZ}$ $\equiv \begin{bmatrix} C_1C_2 & -C_1S_2C_3 + S_1S_3 & C_1S_2S_3 + S_1C_3 \\ S_2 & C_2C_3 & -C_2S_3 \\ -S_1C_2 & S_1S_2C_3 + C_1S_3 & -S_1S_2S_3 + C_1C_3 \end{bmatrix}$
		2	90	$\theta_2 + 90$		
		3	90	θ_3		
		4	−90	−90		
9	XYX	0	0	90		$\mathbf{Q}_{XYX} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{-XZ}$ $\equiv \begin{bmatrix} C_2 & S_2S_3 & S_2C_3 \\ S_1S_2 & -S_1C_2S_3 + C_1C_3 & -S_1C_2C_3 - C_1S_3 \\ -C_1S_2 & C_1C_2S_3 + S_1C_3 & C_1C_2C_3 - S_1S_3 \end{bmatrix}$
		1	90	$\theta_1 - 90$		
		2	−90	θ_2		
		3	90	$\theta_3 + 90$		
		4	−90	−90		
10	XZX	0	0	90		$\mathbf{Q}_{XZX} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{-XZ}$ $\equiv \begin{bmatrix} C_2 & -S_2C_3 & S_2S_3 \\ C_1S_2 & C_1C_2C_3 - S_1S_3 & -C_1C_2S_3 - S_1C_3 \\ S_1S_2 & S_1C_2C_3 + C_1S_3 & -S_1C_2S_3 - C_1C_3 \end{bmatrix}$
		1	90	θ_1		
		2	−90	θ_2		
		3	90	θ_3		
		4	−90	−90		

11	XYZ	0	0	90
		1	90	$\theta_1 - 90$
		2	-90	$\theta_2 - 90$
		3	90	θ_3



$$\mathbf{Q}_{XYZ} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3$$
$$\equiv \begin{bmatrix} C_2C_3 & -C_2S_3 & S_2 \\ S_1S_2C_3 + C_1S_3 & -S_1S_2S_3 + C_1C_3 & -S_1C_2 \\ -C_1S_2C_3 + S_1S_3 & C_1S_2S_3 + S_1C_3 & C_1C_2 \end{bmatrix}$$

12	XZY	0	0	90
		1	90	θ_1
		2	-90	$\theta_2 - 90$
		3	-90	θ_3
		4	90	0



$$\mathbf{Q}_{XZY} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_{+X}$$
$$\equiv \begin{bmatrix} C_2C_3 & -S_2 & C_2S_3 \\ C_1S_2C_3 + S_1S_3 & C_1C_2 & C_1S_2S_3 - S_1C_3 \\ S_1S_2C_3 - C_1S_3 & S_1C_2 & S_1S_2S_3 + C_1C_3 \end{bmatrix}$$

Table 3.7 Required constant matrix multiplication/Additional set of DH parameter for the EAJs (See last column of Table 3.6)

	Constant matrix multiplication/Additional set of DH parameter			
	Not required	Required at the beginning	Required at the end	Required at both the ends
EAJs with three rotations (Euler angles)			ZYX	
	ZYZ		YZX	XYX
	ZXZ	XYZ	ZXY	XZX
	YXZ		YXY	XZY
			YZY	

for $\alpha, \beta, \gamma = X, Y, Z$, corresponding to all the 12 Euler angles sets. The EAJs not only establish correlation between the DH parameters and Euler angles but also facilitates the systematic use of the intersecting revolute joints to describe the Euler Angle rotations even though the configurations of the links are defined using the DH parameters.

While developing EAJs from Euler angles it was seen that the definition of some of the EAJs required additional constant set of DH parameters other than the three regular sets corresponding to three physical revolute joints. As a result, a constant rotation matrix is required either in the beginning or at the end. Interestingly, the existence of constant rotation matrix depends on the axis about which the Euler angle is defined. This is summarized below:

1. EAJs having first rotation about Z or Y axis do not require any fixed rotation matrix in the beginning.
2. If the first rotation is about X axis, a fixed rotation matrix of $\mathbf{Q}_{+Z}$ in the beginning is required to achieve EAJs.
3. The EAJs having third rotation about Z axis do not require any fixed rotation at the end.
4. If the third rotation is about Y or X axis, it requires a fixed rotation matrix of $\mathbf{Q}_{+X}$ or $\mathbf{Q}_{-XZ}$ at the end.

Table 3.7 summarize of the requirement of constant matrix multiplication in the beginning or at the end or the both.

Table 3.7 provides an interesting conclusion. The symmetric EAJs ZYZ and ZXZ, and asymmetric EAJs YXZ are free from the requirement of multiplication of any constant rotation matrix. More specifically, only three sets of DH parameters are required to define these EAJs. Hence, one should use ZYZ and ZXZ EAJs if symmetric set is chosen for representing a three-dimensional rotation. On the other hand, YXZ EAJs is preferred if asymmetric set is chosen for the rotation representation of a 3-DOF joints.

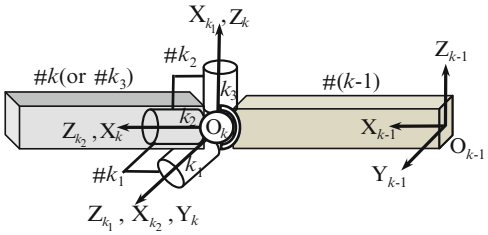


Fig. 3.12 Representation of a spherical joint using YXZ EAJs

3.5 Representation of a Spherical Joint Using EAJs

The spherical joint shown in Fig 3.3 connects the reference link #R with the moving link #M. In practice, a robotic system may have serial- or tree-type architecture with several multiple-DOF joints. This calls for a systematic numbering scheme for intersecting revolute joints and the associated imaginary or physical links. So, the use of Euler-Angle-Joints (EAJs) and the associated numbering scheme are presented here for the systematic representation of a spherical joint. For example, Fig. 3.12 shows a link, $\#(k-1)$, coupled to its neighboring link, $\#k$, by a spherical joint, k, which has three rotational DOF. In order to represent the spherical joint using YXZ EAJs, links $\#(k-1)$ and $\#k$ are considered to be connected by three orthogonally placed revolute joints, denoted as k_1, k_2, and k_3, connecting two virtual links ($\#k_1$ and $\#k_2$), each of zero length and mass. The link $\#k_3$ is the actual physical link $\#k$, which is attached to the third revolute joint, i.e., k_3. The corresponding frame assignment using the DH notation is shown in Fig. 3.12.

The YXZ scheme of EAJs is chosen and will be used throughout this book because (1) it does not require any fixed rotation at the beginning or at the end, and (2) unlike ZYZ scheme of EAJs, it does not lead to a singularity in zero-configuration. Zero-configuration is defined here as the one where all the joint angles are set to zeros.

3.6 Singularity in EAJs

Any three-parameter description of rotations including the Euler angles suffers from singularity. Singularity is encountered in EAJs when two joint axes become parallel. This phenomenon is also known as gimbal lock (Wittenburg 2008). All symmetric EAJs have zero-configurations as singular whereas all asymmetric EAJs are singular for $\theta_2 = 90$. Discussion on the singularity of EAJs is beyond of the scope of this book, hence, no further discussion on how to avoid them is provided in this chapter. One may, however, be referred to Shuster and Oh (1981) and Singla et al. (2004) for the singularity avoidance algorithm. In reality, most of the physical joints have restricted motion, and hence, areas of gimbal lock stay outside the domain of the

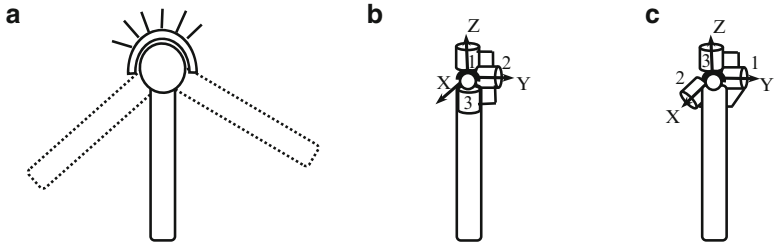


Fig. 3.13 EAJs and singularity. (**a**) A constrained spherical joint. (**b**) Use of ZYZ EAJs (Singular configuration). (**c**) Use of YXZ EAJs (Nonsingular configuration)

movement of joints. Typically, a spherical joint used in practice is constrained to move as shown in Fig. 3.13a. In such a situation, wise selection of EAJs helps in avoiding the singular configuration. For example, if one uses ZYZ EAJs as shown in Fig. 3.13b, singularity is encountered in the zero-configuration. This happens when axis of joint 1 coincides with that of joint 3. On the contrary, use of YXZ EAJs, as shown in Fig. 3.13c, has no singularity corresponding to the zero-configuration. Singularity occurs when joint 2 is rotated by 90° about X axis. However, such a configuration is never encountered due to the constrained movement of the spherical joint and hence singularity is avoided.

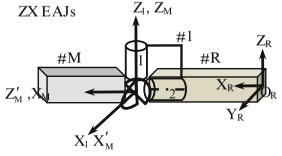
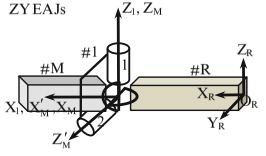
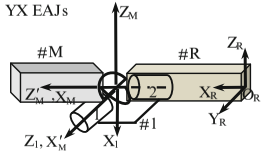
3.7 Multiple-DOF Joints

The concept of EAJs can conveniently be used for the representation of a universal joint by using the composite rotations as discussed in Sect. 3.3.3. Table 3.8 shows EAJs for 2-DOF rotations by universal joints. Such EAJs representation will allow one to treat the velocity transformation relation in a similar way to that of a revolute joint as given by Eq. (4.11). Thus, the concept of EAJs allows one to treat any rotational joint, be it 1-, 2- or 3-DOF, in a unified manner. This will be more clear in Chap. 4, where the kinematics is presented for a system with multiple-DOF joints. Besides, such relations lead to fully recursive O(n) dynamics algorithms, as highlighted in Sect. 6.1, and is not possible with the original definition of the Euler angles. Moreover, it is also shown in Chap. 6 that by careful treatment of zero lengths and masses, the most efficient algorithms can be obtained for a tree-type system consisting of multiple-DOF joints.

3.8 Summary

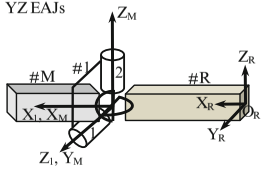
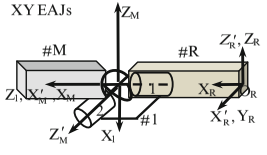
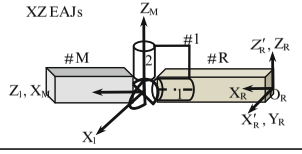
In this chapter, Euler angles and modified DH parameters were first revisited. The concept of the Euler-Angle-Joints (EAJs) was then introduced. Euler-angle-joints (EAJs) are essentially orthogonally intersecting revolute joints so connected by

Table 3.8 Euler-Angle-Joints for representation of a universal joint

	Euler Angle	DH parameters	α_k	θ_k	Equivalent EAJs	Rotation matrix using the DH parameters of the EAJs
1	ZX	1	0	$\theta_1 + 90$		$\mathbf{Q}_{ZX} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_{-XZ} \equiv \begin{bmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S_2 & C_2 \end{bmatrix}$
		2	90	θ_2		
		3	−90	−90		
2	ZY	1	0	θ_1		$\mathbf{Q}_{ZY} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_{+X} \equiv \begin{bmatrix} C_1C_2 & -S_1 & C_1S_2 \\ S_1C_2 & C_1 & S_1S_2 \\ -S_2 & 0 & C_2 \end{bmatrix}$
		2	−90	θ_2		
		3	90	0		
3	YX	1	−90	$\theta_1 + 90$		$\mathbf{Q}_{YX} \equiv \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_{-XZ} \equiv \begin{bmatrix} C_1 & S_1S_2 & S_1C_2 \\ 0 & C_2 & -S_2 \\ -S_1 & C_1S_2 & C_1C_2 \end{bmatrix}$
		2	90	$\theta_2 + 90$		
		3	−90	−90		

(continued)

Table 3.8 (continued)

	Euler Angle	DH parameters			Equivalent EAJs	Rotation matrix using the DH parameters of the EAJs
			α_k	θ_k		
4	YZ	1	-90	θ_1		$\mathbf{Q}_{YZ} \equiv \mathbf{Q}_1\mathbf{Q}_2 \equiv \begin{bmatrix} C_1C_2 & -C_1S_2 & S_1 \\ S_2 & C_2 & 0 \\ -S_1C_2 & S_1S_2 & C_1 \end{bmatrix}$
		2	90	θ_2		
5	XY	0	0	90		$\mathbf{Q}_{XY} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_{+X}$ $\equiv \begin{bmatrix} C_2 & 0 & S_2 \\ S_1S_2 & C_1 & -S_1C_2 \\ -C_1S_2 & S_1 & C_1C_2 \end{bmatrix}$
		1	90	$\theta_1 - 90$		
		2	-90	$\theta_2 - 90$		
		3	90	0		
6	XZ	0	0	90		$\mathbf{Q}_{XZ} \equiv \mathbf{Q}_{+Z}\mathbf{Q}_1\mathbf{Q}_2 \equiv \begin{bmatrix} C_2 & -S_2 & 0 \\ C_1S_2 & C_1C_2 & -S_1 \\ S_1S_2 & S_1C_2 & C_1 \end{bmatrix}$
		1	90	θ_1		
		2	-90	$\theta_2 - 90$		

imaginary links with zero lengths and masses that they represent a particular set of Euler angles. These EAJs are represented using the well-known DH parameters. Different evolutions of EAJs corresponding to different rotation sequences are also shown. The proposed EAJs are used in representing multiple-DOF joints of several robotic systems studied in this book.

Chapter 4
Kinematics of Tree-Type Robotic Systems

Kinematic modeling of a tree-type robotic system is presented in this chapter. In order to obtain kinematic constraints, a tree-type topology is first divided into a set of modules. The kinematic constraints are then obtained between these modules by introducing the concepts of module-twist, module-joint-rate, etc. This helps in obtaining the generic form of the Decoupled Natural Orthogonal Complement (DeNOC) matrices for a tree-type system with the help of module-to-module velocity transformations. Using the present derivation, link-to-link velocity transformation (Saha 1999b) turns out to be a special case of the module-to-module velocity transformation (Shah et al. 2012a) presented in this chapter.

4.1 Kinematic Modules

Conventionally, a tree-type system is considered to have a set of links or bodies connected by kinematic pairs as shown in Fig. 4.1a. However, here a more generic approach is introduced, where the tree-type architecture is considered to have a set of kinematic modules (Shah et al. 2012a). Each module is defined as a set of serially connected links. This is shown in Fig. 4.1b, where the kinematic modules are depicted by dotted boundaries. For the purpose of analysis, the tree-type system is first modularized before its kinematic constraints are obtained. In order to modularize a tree-type system, a link in the tree-type system is first identified as its base, for example, link #0 in Fig. 4.1b, which may be fixed or floating. This is referred to as module M_0. Once the base module is established, the system is then modularized outward such that each module

1. contains serially connected links only;
2. emerges from the last link of its parent module.

The first condition defines a module while the second one defines its connectivity with the adjoining modules. The resulting modules are indicated with M_1, M_2, etc.

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_4,

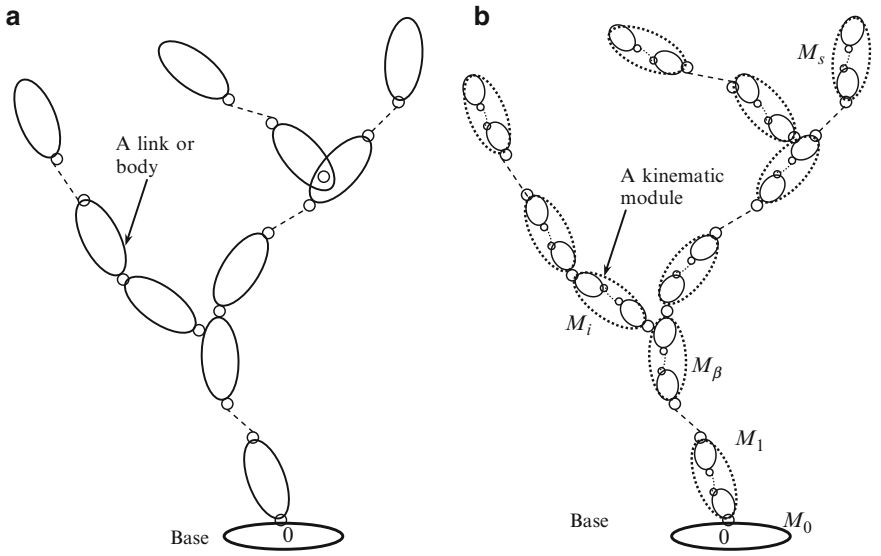


Fig. 4.1 Tree-type architectures (**a**) Conventional (**b**) Multi-modular

It is worth noting that any tree-type system can be modularized such that it follows the above two conditions. For a given tree-type system, however, there may be several module architectures which obey the above conditions. This is evident from Fig. 4.2a, b which represent two different module architectures of the same tree-type system shown in Fig. 4.1a. Moreover, if we consider each link in the tree-type system as one module then the resultant module architecture will follow the above two modularization conditions. As a result, Fig. 4.1a turns out to be a special case of the proposed architecture in Fig. 4.1b. Figure 4.2c shows another arbitrary way of modularization that satisfies only the first modularization condition, i.e., each module is a serial-chain, but does not satisfy the second one as modules M_2 and M_3 do not emanate from the last link of their parent module M_1. Such modularization is not advisable as one has to keep track of the links wherefrom the modules M_2 and M_3 emanate, which is essential to compute the velocities and accelerations of the links of those modules. The tracking index of parent bodies will cause additional burden on the computational resources, thereby, slowing down the analysis or simulation speed.

Referring to Fig. 4.1b, it is assumed that each module, other than the base, is a child module (M_i) that emerges from its parent module (M_β). Obviously the child module bears a higher module number than its parent, i.e., $i > \beta$. The links inside the ith module M_i are denoted as $\#1^i, \ldots, \#k^i, \ldots, \#\eta^i$, where the superscript i signifies the module number and η represents the total number of links in the ith module. Moreover, the joints are denoted as $1^i, \ldots, k^i, \ldots, \eta^i$, whereas the joint variables

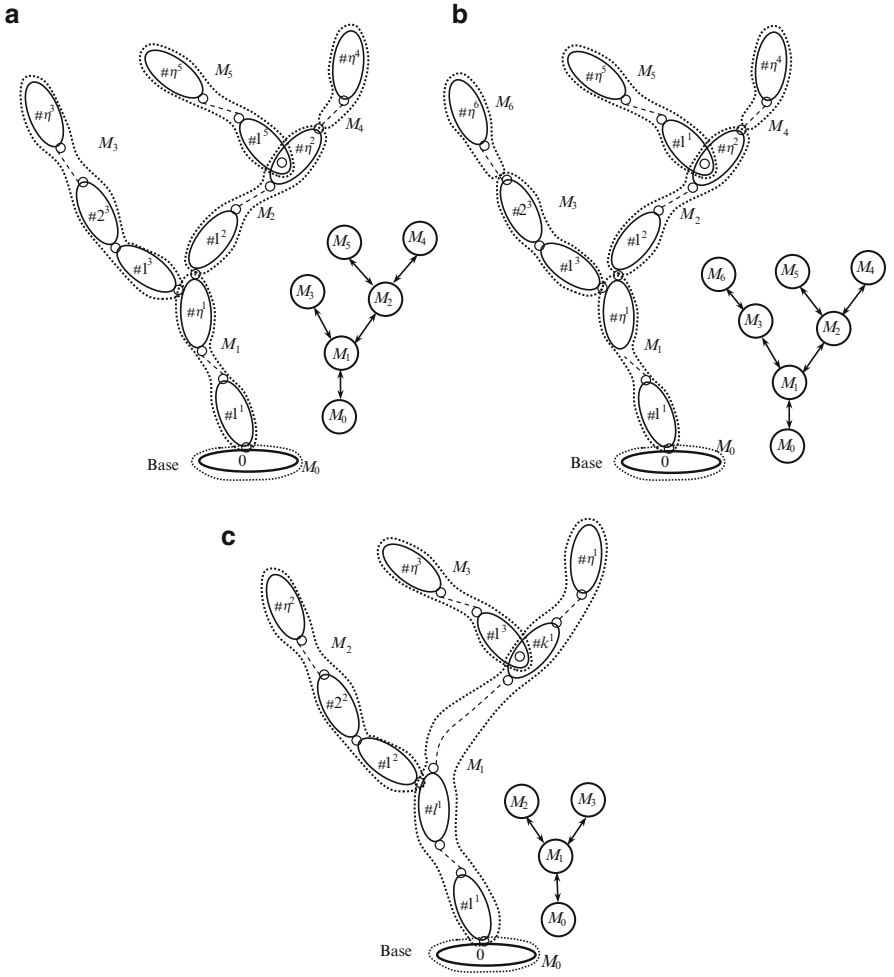


Fig. 4.2 Different module architectures for the tree-type system shown in Fig 4.1a

associated with joint k^i is denoted as ε_k. As a result, total number of joint variables in the ith module, denoted as n^i, is given by

$$n^i = \sum_{k=1}^{\eta^i} \varepsilon_k \tag{4.1}$$

Total number of modules excluding the base module M_0 is denoted as s, whereas total number of links in the s modules is denoted as η. Also, the total number of joint variables in s modules is n. Next, the kinematic constraints at the velocity level are derived amongst the links of a module.

4.2 Intra-modular Velocity Constraints

As mentioned in the previous section, module i, for $i = 1, \ldots, s$, in a tree-type system contains only serially connected links. Let us consider two serially connected links in the ith module, as shown in Fig. 4.3. The kth link, denoted as #k (or #k^i), is connected to the #$(k-1)$th link, i.e., #$(k-1)^i$, by a 1-DOF revolute or prismatic joint k (or k^i). The velocity constraints are then written in terms of the twist of these links. The 6-dimensional vector of twist associated with the angular velocity, $\boldsymbol{\omega}_k$, and linear velocity of the origin of link, $\dot{\mathbf{o}}_k$, of #k, is defined as $\mathbf{t}_k \equiv [\boldsymbol{\omega}_k^T \ \ \dot{\mathbf{o}}_k^T]^T$.

The origin of the kth link is defined using the Denavit-Hartenberg (DH) parameters described in Sect. 3.2, and shown in Fig. 4.3 by point O_k. This is chosen to simplify some of the calculations during dynamic analyses. Next, the twist of #k, $\mathbf{t}_k$, can be written in terms of the twist of #$(k-1)$, $\mathbf{t}_{k-1}$, as

$$\mathbf{t}_k = \mathbf{A}_{k,k-1}\mathbf{t}_{k-1} + \mathbf{p}_k\dot{\theta}_k \tag{4.2}$$

In Eq. (4.2), $\dot{\theta}_k$ is the time rate of change of angular or translational displacement of the kth joint depending on the type of joint, i.e., revolute or prismatic, respectively. The matrix $\mathbf{A}_{k,k-1}$ is the 6×6 twist-propagation matrix, and $\mathbf{p}_k$ is the 6-dimensional motion-propagation vector. They are given by

$$\begin{aligned}\mathbf{A}_{k,k-1} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{a}_{k,k-1}\times\mathbf{1} & \mathbf{1} \end{bmatrix}, \text{ and } \mathbf{p}_k &\equiv \begin{bmatrix} \mathbf{e}_k \\ \mathbf{0} \end{bmatrix} \text{ (for revolute)} \\ &\equiv \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_k \end{bmatrix} \text{ (for prismatic)}\end{aligned} \tag{4.3}$$

where $\mathbf{a}_{k,k-1}\times\mathbf{1}$ is the 3×3 cross-product tensor associated with the vector $\mathbf{a}_{k,k-1}(= -\mathbf{a}_{k-1,k} = [\,-a_k \ \ b_k S\alpha_k \ \ -b_k C\alpha_k\,]^T)$ which when operates on any 3-dimensional Cartesian vector, $\mathbf{x}$, results in a cross-product vector, $\mathbf{a}_{k,k-1}\times\mathbf{x}$. The tensor $\mathbf{a}_{k,k-1}\times\mathbf{1}$ has the following representation:

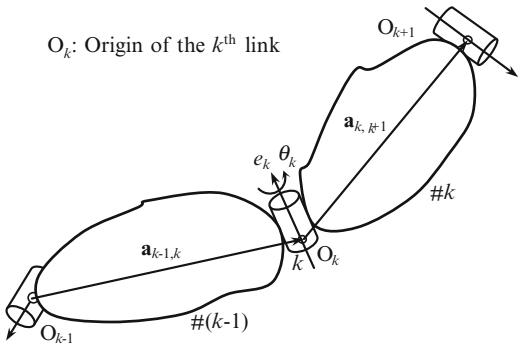


Fig. 4.3 The kth and $(k-1)$th links coupled by joint k

$$\mathbf{a}_{k,k-1} \times \mathbf{1} = \begin{bmatrix} 0 & b_k C\alpha_k & b_k S\alpha_k \\ -b_k C\alpha_k & 0 & a_k \\ -b_k S\alpha_k & -a_k & 0 \end{bmatrix} \tag{4.4}$$

In Eq. (4.3), $\mathbf{e}_k$ is the unit vector along the axis of rotation or translation of the kth joint. The notations '**O**', '**1**' and '**0**' in Eq. (4.3) represent null matrix, identity matrix and null vector of compatible dimensions, respectively. The matrix $\mathbf{A}_{k,k-1}$ and vector $\mathbf{p}_k$, in Eq. (4.3), have the following physical interpretations:

- Matrix $\mathbf{A}_{k,k-1}$ transforms the twist or velocities of the $\#(k-1)$ to the twist of $\#k$ as if they are rigidly attached and therefore referred to as "twist propagation matrix", which has the following properties

$$\mathbf{A}_{k,k-1}\mathbf{A}_{k-1,k-2} = \mathbf{A}_{k,k-2}; \quad \mathbf{A}_{k,k} = \mathbf{1}; \quad \text{and} \quad \mathbf{A}_{k,k-1}^{-1} = \mathbf{A}_{k-1,k} \tag{4.5}$$

- Vector $\mathbf{p}_k$ contributes to the additional motion due to joint k. Hence, it is named as "joint-motion propagation vector."

Next, the vectors of generalized twist and the generalized independent joint-rates for the η-coupled links of a serial module are defined as

$$\mathbf{t} \equiv \left[\mathbf{t}_1^T \cdots \mathbf{t}_k^T \cdots \mathbf{t}_\eta^T\right]^T, \quad \text{and} \quad \dot{\mathbf{q}} \equiv \left[\dot{\theta}_1 \cdots \dot{\theta}_k \cdots \dot{\theta}_\eta\right]^T \tag{4.6}$$

where $\mathbf{t}$ and $\dot{\mathbf{q}}$ are the 6η- and η-dimensional vectors, respectively. Substituting Eq. (4.2) into Eq. (4.6), for $k = 1, \ldots, \eta$, the expression for the generalized twist of a serial module, $\mathbf{t}$, is obtained as

$$\mathbf{t} = \mathbf{N}\dot{\mathbf{q}}, \text{ where } \mathbf{N} \equiv \mathbf{N}_l\mathbf{N}_d \tag{4.7}$$

In Eq. (4.7), $6\eta \times 6\eta$ matrix $\mathbf{N}_l$ and $6\eta \times \eta$ matrix $\mathbf{N}_d$ are given by

$$\mathbf{N}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{A}_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbf{A}_{k,k-1} & \ddots & \mathbf{O} \\ & & \ddots & & \\ \mathbf{A}_{\eta,1} & \cdots & & \mathbf{A}_{\eta,\eta-1} & \mathbf{1} \end{bmatrix}, \text{ and } \mathbf{N}_d \equiv \begin{bmatrix} \mathbf{p}_1 & & & \mathbf{0}'\text{s} \\ & \ddots & & \\ & & \mathbf{p}_k & \\ & & & \ddots \\ \mathbf{0}'\text{s} & & & \mathbf{p}_\eta \end{bmatrix} \tag{4.8}$$

The matrices $\mathbf{N}_l$ and $\mathbf{N}_d$ are referred here as the Decoupled Natural Orthogonal Complement (DeNOC) matrices of the serial-module of a tree-type system, which are nothing but those reported by Saha (1997) for a serial robot. Now, the velocity constraints for the module M_0, consisting of the fixed or floating-base #0 only, are derived. The twist of #0 can be expressed as

$$\begin{aligned} \mathbf{t}_0 &= \mathbf{P}_0\dot{\mathbf{q}}_0 && \text{if the base is floating} \\ &= \mathbf{0} && \text{if the base is fixed} \end{aligned} \tag{4.9}$$

where the 6×6 matrix $\mathbf{P}_0$ and the 6-dimensional vector $\dot{\mathbf{q}}_0$ have the following representations:

$$\mathbf{P}_0 \equiv \begin{bmatrix} \mathbf{L}_0 & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}, \text{ and } \dot{\mathbf{q}}_0 \equiv \begin{bmatrix} \dot{\boldsymbol{\theta}}_0 \\ \dot{\mathbf{o}}_0 \end{bmatrix} \tag{4.10}$$

In Eq. (4.10), $\dot{\boldsymbol{\theta}}_0$, $\mathbf{L}_0$ and $\dot{\mathbf{o}}_0$ are the time-rates of the independent rotation coordinates, the corresponding transformation matrix, and the linear velocity vector, respectively. Vector $\dot{\boldsymbol{\theta}}_0$ contains the time-rates of Euler angles if the Euler angles are used to define the rotation of the floating-base in the three-dimensional Cartesian space. The dimensions of $\mathbf{L}_0$, and $\dot{\boldsymbol{\theta}}_0$ change if the rotation is represented, say, by using Euler parameters. For planar rotations, the expression greatly simplifies, i.e., $\mathbf{L}_0$ and $\dot{\boldsymbol{\theta}}_0$ both become scalar quantities. More specifically, $\mathbf{L}_0$ is a scalar one, i.e., 1, whereas scalar $\dot{\boldsymbol{\theta}}_0$ is the rotation about the axis perpendicular to the plane of rotation.

4.2.1 Presence of Multiple-DOF Joints

The DeNOC matrices for a serial module obtained in Eq. (4.8) pertain to a system with 1-DOF joints only. However, a robotic system, e.g., a humanoid or quadruped, may contain multiple-Degrees-of-Freedom (multiple-DOF) joints, say a universal or a spherical one. One way of treating multiple-DOF joints is to treat them as a combination of several one-DOF joints connected by links of zero length and mass as shown by Duffy (1978), and Chaudhary and Saha (2007). Such consideration, however, increases unnecessary computations in the kinematic and dynamic algorithms, as zero link lengths and masses will not yield any non-zero results when operated on other variable. Therefore alternatively, Euler angles or Euler parameters are commonly used to represent the three-dimensional rotations due to the presence of, say, a spherical joint. Such representation, however, do not lead to unified representation of multiple-DOF joints. Moreover, one cannot avoid the inversion of 3×3 matrices in the recursive forward dynamics algorithm as pointed in Sect. 6.1.2. On the contrary, Euler-Angle-Joints (EAJs) introduced in Chap. 3 can be used, as this will avoid the above drawbacks present in the Euler angle representation. The EAJs can be used for unified representation of a generic multiple-DOF joint with the significant simplification in dynamics algorithms (Shah et al. 2009). The general expressions for the velocity constraints associated with the links connected by a multiple-DOF joint, Fig. 4.4, can then be given by

$$\begin{aligned} \text{for } j = 1 : \varepsilon_k \quad \mathbf{t}_{k_j} &= \mathbf{A}_{k,k-1}\mathbf{t}_{(k-1)_{\varepsilon_{(k-1)}}} + \mathbf{p}_{k_j}\dot{\theta}_{k_j} \quad \text{if } j = 1 \\ \mathbf{t}_{k_j} &= \mathbf{t}_{k_{j-1}} + \mathbf{p}_{k_j}\dot{\theta}_{k_j} \quad \text{if } j > 1 \end{aligned} \tag{4.11}$$

where ε_k corresponds to the joint variable associated with the kth joint, e.g., $\varepsilon_k = 1$ for 1-DOF joint, $\varepsilon_k = 2$ for 2-DOF and $\varepsilon_k = 3$ for 3-DOF joints. Next, the twists of associated links and corresponding joint rates are written as

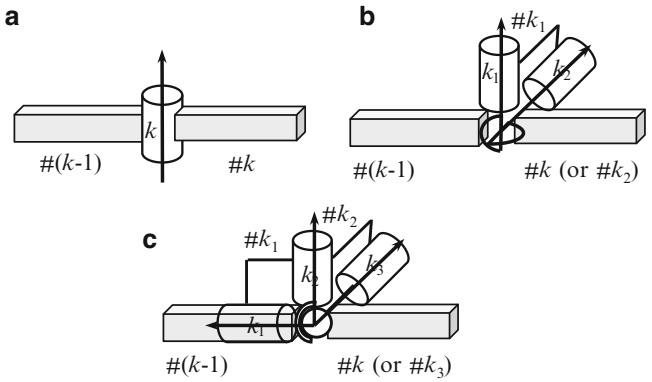


Fig. 4.4 Representation of multiple-DOF joints (**a**) A revolute joint ($\varepsilon_k = 1$) (**b**) A universal joint ($\varepsilon_k = 2$) (**c**) A spherical joint ($\varepsilon_k = 3$)

$$\mathbf{t}_k \equiv \left[\mathbf{t}_{k_1}^T \cdots \mathbf{t}_{k_{(\varepsilon_k)}}^T\right]^T, \text{ and } \dot{\boldsymbol{\theta}}_k \equiv \left[\dot{\theta}_{k_1} \cdots \dot{\theta}_{k_{(\varepsilon_k)}}\right]^T \tag{4.12}$$

Using Eqs. (4.11) and (4.12), $\mathbf{t}_k$ can be written in terms of $\mathbf{t}_{k-1}$ as

$$\mathbf{t}_k = \boldsymbol{\Lambda}_{k,k-1}\mathbf{t}_{k-1} + \mathbf{P}_k\dot{\boldsymbol{\theta}}_k \tag{4.13}$$

where the $6\varepsilon_k \times 6\varepsilon_{k-1}$ matrix $\boldsymbol{\Lambda}_{k,k-1}$ and $6\varepsilon_k \times \varepsilon_k$ matrix $\mathbf{P}_k$ are represented as

$$\boldsymbol{\Lambda}_{k,k-1} \equiv \begin{bmatrix} \mathbf{O} \cdots \mathbf{O} & \mathbf{A}_{k,k-1} \\ \vdots \quad \vdots & \vdots \\ \mathbf{O} \cdots \mathbf{O} & \mathbf{A}_{k,k-1} \end{bmatrix}, \text{ and } \mathbf{P}_k \equiv \begin{bmatrix} \mathbf{p}_{k_1} & \mathbf{0}'s \\ \vdots & \ddots \\ \mathbf{p}_{k_1} & \cdots \mathbf{p}_{k_{(\varepsilon_k)}} \end{bmatrix} \tag{4.14}$$

where '$\mathbf{O}$' and '$\mathbf{0}$' stand for the 6×6 null matrix and 6-dimensional null vector, respectively. Furthermore, $\mathbf{A}_{k,k-1}$ is the 6×6 twist-propagation matrix, and $\mathbf{p}_{k_j}$ is the 6-dimensional motion-propagation vector, as obtained in Eq. (4.3). It is worth mentioning that the unique form of $\boldsymbol{\Lambda}_{k,k-1}$ and $\mathbf{P}_k$ can only be attained if the origin is selected as the reference point. It is evident from Eq. (4.14) that the sizes of block matrices $\boldsymbol{\Lambda}_{k,k-1}$ and $\mathbf{P}_k$ change depending on the number of joint variables ε_k and ε_{k-1} associated with kth and $(k-1)$th joints, respectively. For example, if kth joint is universal, i.e., $\varepsilon_k = 2$, and $(k-1)$th joint is spherical, i.e., $\varepsilon_{(k-1)} = 3$, then the sizes of matrices $\boldsymbol{\Lambda}_{k,k-1}$ and $\mathbf{P}_k$ are 12×18 and 12×2, respectively. They are expressed as follows:

$$\boldsymbol{\Lambda}_{k,k-1} \equiv \begin{bmatrix} \mathbf{O} & \mathbf{O} & \mathbf{A}_{k,k-1} \\ \mathbf{O} & \mathbf{O} & \mathbf{A}_{k,k-1} \end{bmatrix} \text{ and } \mathbf{P}_k \equiv \begin{bmatrix} \mathbf{p}_{k_1} & \mathbf{0} \\ \mathbf{p}_{k_1} & \mathbf{p}_{k_2} \end{bmatrix} \tag{4.15}$$

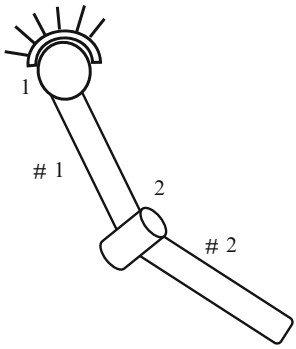


Fig. 4.5 A spatial double pendulum

Similarly if the kth joint is revolute, i.e., $\varepsilon_k = 1$, and $(k-1)$th joint is spherical, i.e., $\varepsilon_{k-1} = 3$, then the sizes of matrices $\mathbf{\Lambda}_{k,k-1}$ and $\mathbf{P}_k$ are 6×18 and 6×1, respectively. The matrices are given by

$$\mathbf{\Lambda}_{k,k-1} \equiv \begin{bmatrix} \mathbf{O} & \mathbf{O} & \mathbf{A}_{k,k-1} \end{bmatrix} \text{and } \mathbf{P}_k \equiv \mathbf{p}_{k_1} \tag{4.16}$$

In case a serial system has only 1-DOF joints then, $\mathbf{\Lambda}_{k,k-1} \equiv \mathbf{A}_{k,k-1}$ and $\mathbf{P}_k \equiv \mathbf{p}_k$. As a result Eq. (4.13) simplifies to Eq. (4.2). This shows that Eq. (4.2) is the special case of Eq. (4.13). Next, the general expression of the DeNOC matrices for a serial-chain system with multiple-DOF joints is obtained, similar to Eq. (4.8), as

$$\mathbf{N}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{\Lambda}_{2,1} & \mathbf{1} & \cdots & \mathbf{O} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{\Lambda}_{\eta,1} & \cdots & \mathbf{\Lambda}_{\eta,\eta-1} & \mathbf{1} \end{bmatrix}, \text{and } \mathbf{N}_d \equiv \begin{bmatrix} \mathbf{P}_1 & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \mathbf{P}_2 & \cdots & \mathbf{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{P}_\eta \end{bmatrix} \tag{4.17}$$

where $\mathbf{N}_l$ and $\mathbf{N}_d$ are the $6n \times 6n$ and $6n \times n$ matrices. In contrast to Saha (1997), Eq. 4.17 represent a general form of the DeNOC matrices of a serial-chain system with multiple-DOF joints.

4.2.2 An Illustration: A Spatial Double Pendulum

The concept of the DeNOC matrices for a serial-chain system with multiple-DOF joints is illustrated next using a spatial double pendulum with a spherical and a revolute joint, located at 1 and 2, respectively, as shown in Fig. 4.5. The DeNOC matrices for this system can be obtained using Eq. (4.17) as

$$\mathbf{N}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{\Lambda}_{2,1} & \mathbf{1} \end{bmatrix}, \text{ and } \mathbf{N}_d \equiv \begin{bmatrix} \mathbf{P}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{P}_2 \end{bmatrix} \tag{4.18}$$

where the 6×18 matrix $\mathbf{\Lambda}_{2,1}$, the 18×3 matrix $\mathbf{P}_1$, and the 6×1 vector $\mathbf{P}_2$ are obtained using Eq. (4.14) as

$$\mathbf{\Lambda}_{2,1} \equiv [\,\mathbf{O}\ \mathbf{O}\ \mathbf{A}_{2,1}\,],\ \mathbf{P}_1 \equiv \begin{bmatrix} \mathbf{p}_{1_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{p}_{1_1} & \mathbf{p}_{1_2} & \mathbf{0} \\ \mathbf{p}_{1_1} & \mathbf{p}_{1_2} & \mathbf{p}_{1_3} \end{bmatrix} \text{ and } \mathbf{P}_2 \equiv \mathbf{p}_2 \tag{4.19}$$

in which the 6×6 matrix $\mathbf{A}_{2,1}$ and the 6-dimensioinal vector $\mathbf{p}_{k_j}$ are defined in Eq. (4.3).

4.3 Inter-modular Velocity Constraints

Tree-type robotic systems have more than one kinematic module. Hence, the velocity constraints between the modules, i.e., at the inter-modular level, will be derived in this section. For this, recursive relationships between any two adjoining modules are established first. Figure 4.6 shows module M_i and its parent module M_β ('β' signifies a parent). For the module M_i, the $6n^i$-dimensional vector of module-twist, $\bar{\mathbf{t}}_i$, that has n^i link twists, and the n^i-dimensional vector of module-joint-rate, $\dot{\bar{\mathbf{q}}}_i$, having n^i joint rates are defined as

$$\bar{\mathbf{t}}_i \equiv \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_k \\ \vdots \\ \mathbf{t}_\eta \end{bmatrix}^i \text{ and } \dot{\bar{\mathbf{q}}}_i \equiv \begin{bmatrix} \dot{\boldsymbol{\theta}}_1 \\ \vdots \\ \dot{\boldsymbol{\theta}}_k \\ \vdots \\ \dot{\boldsymbol{\theta}}_\eta \end{bmatrix}^i \tag{4.20}$$

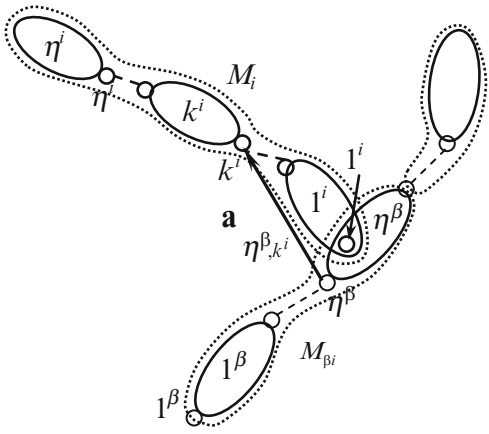


Fig. 4.6 Module M_i and its parent M_β

A bar ('−') over an entity in Eq. (4.20) signifies that the quantity is related to a module and the superscript, i, outside the bracket identifies the module. As a consequence, the generic notation $\mathbf{t}_k$ (or $\mathbf{t}_{k^i}$) in Eq. (4.20) is the 6-dimensional twist vector for the kth link in the ith module. Next, it is shown that the module-twist $\bar{\mathbf{t}}_i$, for M_i, can be written in terms of the module-twist $\bar{\mathbf{t}}_\beta$ (or) $\bar{\mathbf{t}}_{\beta_i}$ of its parent M_β as

$$\bar{\mathbf{t}}_i = \overline{\mathbf{A}}_{i,\beta}\bar{\mathbf{t}}_{\beta_i} + \overline{\mathbf{N}}_i\dot{\bar{\mathbf{q}}}_i \tag{4.21}$$

where $\overline{\mathbf{A}}_{i,\beta}$, and $\overline{\mathbf{N}}_i$ are the $6n^i \times 6n^\beta$ module-twist propagation and $6n^i \times n^i$ module-joint-motion propagation matrices, respectively, which are given by

$$\overline{\mathbf{A}}_{i,\beta} \equiv \begin{bmatrix} \mathbf{O} & \cdots & \mathbf{O} & \mathbf{A}_{1^i,\eta^\beta} \\ \vdots & & \vdots & \vdots \\ \mathbf{O} & & \mathbf{O} & \mathbf{A}_{k^i,\eta^\beta} \\ \vdots & & \vdots & \vdots \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{A}_{\eta^i,\eta^\beta} \end{bmatrix}, \text{ and } \overline{\mathbf{N}}_i \equiv \begin{bmatrix} \mathbf{p}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_{2,1}\mathbf{p}_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbf{A}_{k,k-1}\mathbf{p}_{k-1} & \ddots & \mathbf{0} \\ \mathbf{A}_{\eta,1}\mathbf{p}_1 & \cdots & & \ddots & \mathbf{A}_{\eta,\eta-1}\mathbf{p}_{\eta-1} & \mathbf{p}_\eta \end{bmatrix}^i \tag{4.22}$$

In Eq. (4.22), $\overline{\mathbf{A}}_{i,\beta}$ propagates the twist of the parent module (βth) to the child module (ith); the last column contains the twist propagation matrix from the ηth link of the βth module wherefrom the ith module emanates, to the different links of the ith module. Matrix $\mathbf{A}_{k^i,\eta^\beta}$ in the expression of $\overline{\mathbf{A}}_{i,\beta}$ denotes the 6×6 twist-propagation matrix from the twist of link $\#\eta^\beta$ in M_β to the twist of link $\#k^i$ in M_i. Its expression is obtained similar to $\mathbf{A}_{k,k-1}$ of Eq. (4.3) and is associated with the vector $\mathbf{a}_{k^i,\eta^\beta}$, as indicated in Fig. 4.6. Moreover for any three adjoining modules i, h and j, module-twist propagation matrix satisfies the following property

$$\overline{\mathbf{A}}_{j,h}\overline{\mathbf{A}}_{h,i} \equiv \overline{\mathbf{A}}_{j,i} \tag{4.23}$$

Equation (4.23) is, however, not true for non adjoining module. For example, referring to Fig. 4.2a, $\overline{\mathbf{A}}_{3,1}\overline{\mathbf{A}}_{1,0} \equiv \overline{\mathbf{A}}_{3,0}$ but $\overline{\mathbf{A}}_{3,2}\overline{\mathbf{A}}_{2,1} \neq \overline{\mathbf{A}}_{3,1}$ as modules M_2 and M_3 belong to different branches. It is worth noting that in Eq. (4.22), matrix $\mathbf{A}_{k,l}$ and vector $\mathbf{p}_k$ are corresponding to the system with only 1-DOF joints. If higher-DOF joints are present then $\mathbf{A}_{k,l}$ and $\mathbf{p}_k$ are to be replaced with $\mathbf{\Lambda}_{l,k}$ and $\mathbf{P}_k$ of Eq. (4.17). Moreover, for a fixed-base serial system with only 1-DOF joints, the expression in Eq. (4.21) is effectively the one given by Eq. (4.7), as the parent module is the fixed-base and $\bar{\mathbf{t}}_\beta = \mathbf{0}$. As a result, the module-joint-rate propagation matrix $\overline{\mathbf{N}}_i$ of Eq. (4.22) is the same as the NOC of a serial-chain system. Now, the module twist for module M_0 is expressed from Eq. (4.9) but written in the form of Eq. (4.21) as

$$\begin{aligned} \bar{\mathbf{t}}_0 &= \overline{\mathbf{N}}_0\dot{\bar{\mathbf{q}}}_0, && \text{if the base is floating} \\ &= \mathbf{0}, && \text{if the base is fixed} \end{aligned} \tag{4.24}$$

where $\overline{\mathbf{N}}_0 \equiv \mathbf{P}_0$ and $\dot{\overline{\mathbf{q}}}_0 \equiv \dot{\mathbf{q}}_0$. Considering all the links and joints, the generalized twist vector, $\mathbf{t}$, consisting of all module-twists, and the generalized joint-rate vector, $\dot{\mathbf{q}}$, consisting of all module-joint-rates are defined next as:

$$\mathbf{t} \equiv \begin{bmatrix} \bar{\mathbf{t}}_0 \\ \bar{\mathbf{t}}_1 \\ \vdots \\ \bar{\mathbf{t}}_i \\ \vdots \\ \bar{\mathbf{t}}_s \end{bmatrix} \text{ and } \dot{\mathbf{q}} \equiv \begin{bmatrix} \dot{\bar{\mathbf{q}}}_0 \\ \dot{\bar{\mathbf{q}}}_1 \\ \vdots \\ \dot{\bar{\mathbf{q}}}_i \\ \vdots \\ \dot{\bar{\mathbf{q}}}_s \end{bmatrix} \tag{4.25}$$

In Eq. (4.25), the vectors of module-twists $\bar{\mathbf{t}}_i$ and module-joint-rates $\dot{\bar{\mathbf{q}}}_i$ contain link twists and joint rates, respectively, as defined in Eq. (4.20). Hence, Eq. (4.25) treats modules of a tree-type system similar to the links in a serial system. This analogy helps one to extrapolate many concepts of the serial-chain systems directly to the tree-type systems. One such example is the formulation of Eq. (4.21) from Eq. (4.2). Substituting Eq. (4.21) into Eq. (4.25), the generalized twist $\mathbf{t}$ is expressed as

$$\mathbf{t} = \mathbf{A}\mathbf{t} + \mathbf{N}_d \dot{\mathbf{q}} \tag{4.26}$$

where $\mathbf{A}$ and $\mathbf{N}_d$ are the $6(n+\eta_0) \times 6(n+\eta_0)$ and $6n \times (n+n_0)$ matrices, where $n_0 = \eta_0 = 0$ for a fixed-base, and $\eta_0 = 1$ and $n_0 = 6$ for a floating-base. Matrices $\mathbf{A}$ and $\mathbf{N}_d$ are given by

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{O} & & & & \mathbf{O}'s \\ \overline{\mathbf{A}}_{1,\beta} & \ddots & \mathbf{O} & & \\ & \ddots & \overline{\mathbf{A}}_{i,\beta} & \ddots & \\ \mathbf{O}'s & & \ddots & \overline{\mathbf{A}}_{s,\beta} & \mathbf{O} \end{bmatrix}, \text{ and } \mathbf{N}_d \equiv \begin{bmatrix} \overline{\mathbf{N}}_0 & & & \mathbf{O}'\text{s} \\ & \ddots & & \\ & & \overline{\mathbf{N}}_i & \\ & & & \ddots \\ \mathbf{O}'\text{s} & & & \overline{\mathbf{N}}_s \end{bmatrix} \tag{4.27}$$

In Eq. (4.27), β in $\overline{\mathbf{A}}_{i,\beta}$ corresponds to the parent of i, for $i = 1, \ldots, \text{s}$. Moreover, $\mathbf{O}$'s are the null matrices of compatible dimensions. Rearranging Eq. (4.26), the $6(n+\eta_0)$-dimensional generalized twist is given by

$$\mathbf{t} = \mathbf{N}_l \mathbf{N}_d \dot{\mathbf{q}}, \text{ where } \mathbf{N}_l = (\mathbf{1} - \mathbf{A})^{-1} \tag{4.28}$$

The $6(n+\eta_0) \times 6(n+\eta_0)$ matrix $\mathbf{N}_l$ is as follows:

$$\mathbf{N}_l \equiv \begin{bmatrix} \overline{\mathbf{1}}_0 & & & & \\ \overline{\mathbf{A}}_{1,0} & \overline{\mathbf{1}}_1 & & \mathbf{O}'\text{s} & \\ \overline{\mathbf{A}}_{2,0} & \overline{\mathbf{A}}_{2,1} & \overline{\mathbf{1}}_2 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \overline{\mathbf{A}}_{s,0} & \overline{\mathbf{A}}_{s,1} & \ldots & \overline{\mathbf{A}}_{s,s-1} & \overline{\mathbf{1}}_s \end{bmatrix} (\overline{\mathbf{A}}_{j,i} \equiv \mathbf{O}, \text{if } M_j \notin \gamma_i) \tag{4.29}$$

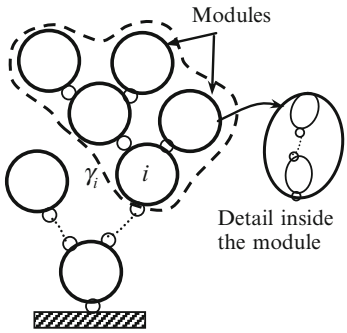


Fig. 4.7 Definition of γ_i

where $\overline{\mathbf{1}}_i$ is the $6n^i \times 6n^i$ identity matrix and γ_i stands for array of all the modules upstream from as well as including the module M_i, as shown within the dashed boundary of Fig. 4.7. Hence, if module M_j does not belong to γ_i then $\overline{\mathbf{A}}_{j,i}$ vanishes. This means that the twist of any link belonging to module M_j does not depend on any link of any module originating from M_i. This essentially takes care of branch-induced sparsity.

The matrices $\mathbf{N}_l$ and $\mathbf{N}_d$ of Eq. (4.28) are nothing but the DeNOC matrices for the tree-type system under study, written in terms of the module information rather than in terms of the link information. Hence, Eqs. (4.27) and (4.29) are the generic versions of Eq. (4.8). Under the special case, when the base is fixed, all the terms containing subscript '0', i.e., $\overline{\mathbf{A}}_{i,0}$, $\overline{\mathbf{1}}_0$, and $\overline{\mathbf{N}}_0$ (the elements in the 1st row and column), vanish. Moreover, if each module consists of one link with no branching, and is connected by 1-DOF joint only, then $\overline{\mathbf{N}}_i \equiv \mathbf{p}_i$, $\overline{\mathbf{A}}_{i,j} \equiv \mathbf{A}_{i,j}$ and $\mathbf{A}_{i,j}\mathbf{A}_{j,k} \equiv \mathbf{A}_{i,k}$. As a result, the expressions of $\mathbf{N}_l$ and $\mathbf{N}_d$ degenerate to Eq. (4.8), which shows that the work by Saha (1999b) is a special case of what is proposed here. This brings up an important fact that, for different module architectures, the NOC can be decoupled into different module-DeNOC matrices $\mathbf{N}_l$ and $\mathbf{N}_d$. This generalizes the DeNOC concept from a serial system of Saha (1997) to a tree-type system with general module architecture and multiple-DOF joints.

4.4 Examples

In order to illustrate the concept of the module-DeNOC matrices, and branch induced sparsity, several planar and spatial examples are given next.

4.4.1 A Robotic Gripper

Figure 4.8 shows a 4-link tree-type robotic gripper. The gripper is divided into four modules, as indicated in Fig. 4.8, using the scheme presented in Sect. 4.1.

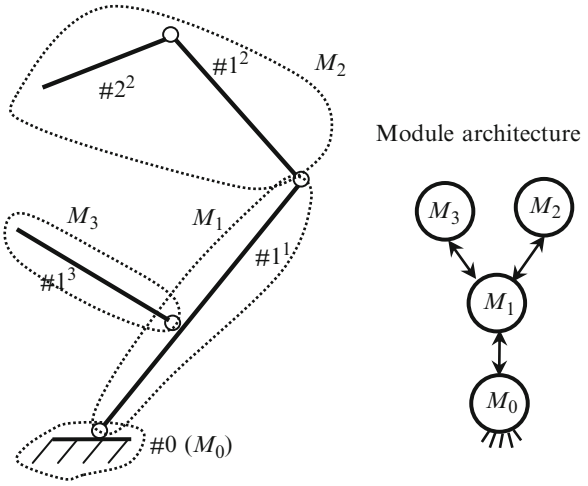


Fig. 4.8 A robotic gripper and its modularization

They are M_0, M_1, M_2, and M_3, where M_0 is the fixed-base. The expression for the module-level DeNOC matrices $\mathbf{N}_l$ and $\mathbf{N}_d$ are then obtained by using Eqs. (4.27) and (4.29) as

$$\mathbf{N}_l \equiv \begin{bmatrix} \overline{\mathbf{1}}_1 & & \mathbf{O}'\mathrm{s} \\ \overline{\mathbf{A}}_{2,1} & \overline{\mathbf{1}}_2 & \\ \overline{\mathbf{A}}_{3,1} & \mathbf{O} & \overline{\mathbf{1}}_3 \end{bmatrix}, \text{and } \mathbf{N}_d \equiv \begin{bmatrix} \overline{\mathbf{N}}_1 & & \mathbf{O}'\mathrm{s} \\ & \overline{\mathbf{N}}_2 & \\ \mathbf{O}'\mathrm{s} & & \overline{\mathbf{N}}_3 \end{bmatrix} \tag{4.30}$$

where $\mathbf{N}_l$ and $\mathbf{N}_d$ are the 24×24 and 24×4 matrices, respectively. It is worth noting that M_0 is fixed. Hence, the elements with subscripts '0', i.e., $\overline{\mathbf{A}}_{1,0}$, $\overline{\mathbf{A}}_{2,0}$, $\overline{\mathbf{A}}_{3,0}$, $\overline{\mathbf{1}}_0$, and $\overline{\mathbf{N}}_0$, in the expression of $\mathbf{N}_l$ and $\mathbf{N}_d$ vanish. It may also be noted that in Eq. (4.30) $\overline{\mathbf{A}}_{3,2} = \mathbf{O}$. This is due to the fact that module M_3 does not belong to the modules originating from the module M_2, i.e., $M_3 \notin \gamma_2$. Moreover, $\overline{\mathbf{1}}_1$, $\overline{\mathbf{1}}_2$, and $\overline{\mathbf{1}}_3$ are the 6×6, 12×12 and 6×6, identity matrices, respectively. The block elements $\overline{\mathbf{A}}_{2,1}$, $\overline{\mathbf{A}}_{3,1}$, $\overline{\mathbf{N}}_1$, $\overline{\mathbf{N}}_2$, and $\overline{\mathbf{N}}_3$ are obtained using Eq. (4.22) as

$$\overline{\mathbf{A}}_{21} \equiv \begin{bmatrix} \mathbf{A}_{1^2 1^1} \\ \mathbf{A}_{2^2 1^1} \end{bmatrix}, \overline{\mathbf{A}}_{31} \equiv \mathbf{A}_{1^3 1^1}, \overline{\mathbf{N}}_1 \equiv \mathbf{p}_{1^1}, \overline{\mathbf{N}}_2 \equiv \begin{bmatrix} \mathbf{p}_1 & 0 \\ \mathbf{A}_{21}\mathbf{p}_1 & \mathbf{p}_2 \end{bmatrix}^2, \text{and } \overline{\mathbf{N}}_3 \equiv \mathbf{p}_{1^3} \tag{4.31}$$

where matrix $\mathbf{A}_{k^i,\eta^\beta}$ denotes the 6×6 twist-propagation matrix from the twist of link $\#\eta^\beta$ in M_β to the twist of link $\#k^i$ in M_i. In Eq. (4.31), $\overline{\mathbf{A}}_{2,1}$, $\overline{\mathbf{A}}_{3,1}$ and $\overline{\mathbf{N}}_2$ are the 12×6, 6×6 and 12×2 matrices, respectively, and $\overline{\mathbf{N}}_1$ and $\overline{\mathbf{N}}_3$ are the 6-dimensional vectors. It is pointed out here that the size of block elements in $\mathbf{N}_l$ and $\mathbf{N}_d$ depends on the number of links and joint variables present in the modules. For example, $\overline{\mathbf{A}}_{2,1}$ corresponds to the module-twist propagation from module M_1 with one link to module M_2 with two links. Hence, its size is 12×6. Next, matrix $\overline{\mathbf{N}}_2$ corresponds to the module-joint-motion propagation for module M_2 containing two joint variables. Hence, its size is 12×2.

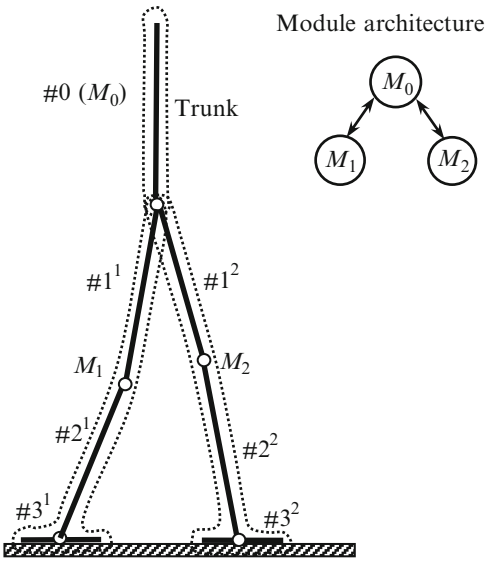


Fig. 4.9 A planar biped and its modularization

4.4.2 A Planar Biped

The concept of the DeNOC matrices is illustrated next for a 7-link floating-base biped. Torso of the biped is assumed to be the floating-base, M_0, and two legs form modules M_1 and M_2 as shown in Fig. 4.9. The DeNOC matrices are then obtained using Eqs. (4.27) and (4.29) as

$$\mathbf{N}_l \equiv \begin{bmatrix} \overline{\mathbf{1}}_0 & & \mathbf{O}'s \\ \overline{\mathbf{A}}_{1,0} & \overline{\mathbf{1}}_1 & \\ \overline{\mathbf{A}}_{2,0} & \mathbf{O} & \overline{\mathbf{1}}_2 \end{bmatrix}, \text{ and } \mathbf{N}_d \equiv \begin{bmatrix} \overline{\mathbf{N}}_0 & & \mathbf{O}'s \\ & \overline{\mathbf{N}}_1 & \\ \mathbf{O}'s & & \overline{\mathbf{N}}_2 \end{bmatrix} \tag{4.32}$$

where $\mathbf{N}_l$ and $\mathbf{N}_d$ are 42×42 and 42×12 matrices. Moreover, $\overline{\mathbf{1}}_0$, $\overline{\mathbf{1}}_1$, and $\overline{\mathbf{1}}_2$ are the 6×6, 18×18 and 18×18 identity matrices, respectively. The block elements $\overline{\mathbf{A}}_{1,0}$, $\overline{\mathbf{A}}_{2,0}$, $\overline{\mathbf{N}}_1$, $\overline{\mathbf{N}}_2$, and $\overline{\mathbf{N}}_0$ are obtained by using Eqs. (4.22) and (4.24) as

$$\overline{\mathbf{A}}_{i,0} = \begin{bmatrix} \mathbf{A}_{1,0} \\ \mathbf{A}_{2,0} \\ \mathbf{A}_{3,0} \end{bmatrix}^i, \overline{\mathbf{N}}_i = \begin{bmatrix} \mathbf{p}_1 & \mathbf{O} & \mathbf{O} \\ \mathbf{A}_{2,1}\mathbf{p}_1 & \mathbf{p}_2 & \mathbf{O} \\ \mathbf{A}_{3,1}\mathbf{p}_1 & \mathbf{A}_{3,2}\mathbf{p}_2 & \mathbf{p}_3 \end{bmatrix}^i \text{ for } i = 1, 2$$
$$\text{and } \overline{\mathbf{N}}_0 = \mathbf{P}_0 \tag{4.33}$$

where superscript i outside of the brackets indicates the module. Moreover, matrices $\overline{\mathbf{A}}_{1,0}$, $\overline{\mathbf{A}}_{2,0}$, $\overline{\mathbf{N}}_1$, $\overline{\mathbf{N}}_2$ and $\overline{\mathbf{N}}_0$ are of sizes 18×6, 18×6, 18×3, 18×3, and 6×6, respectively.

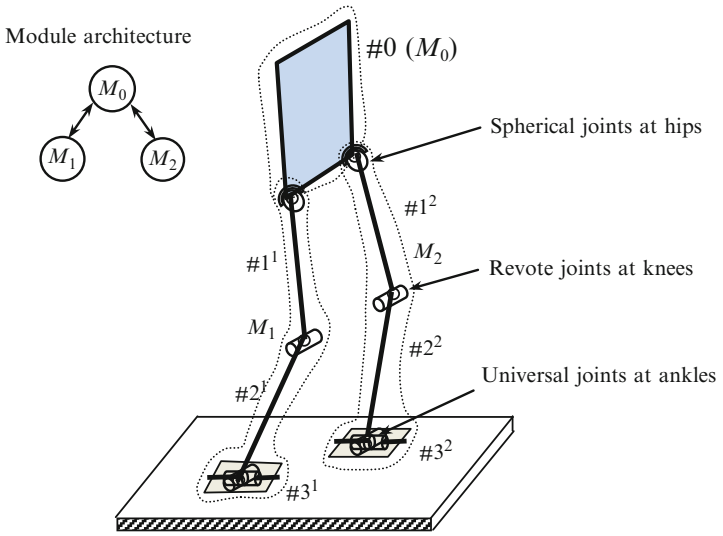


Fig. 4.10 A spatial biped and its modularization

4.4.3 A Spatial Biped

In order to study the application of the proposed concept for a spatial system with multiple-DOF joints, a 7-link spatial biped, as shown in Fig. 4.10, is considered. It has spherical joints at the hips, universal joints at the ankles and revolute joints at the knees. Again torso of the biped is assumed to be the floating-base, M_0, and legs form modules M_1 and M_2. The module-level DeNOC matrices are then obtained using Eqs. (4.27) and (4.29) as

$$\mathbf{N}_l \equiv \begin{bmatrix} \overline{\mathbf{1}}_0 & & \mathbf{O}'s \\ \overline{\mathbf{A}}_{1,0} & \overline{\mathbf{1}}_1 & \\ \overline{\mathbf{A}}_{2,0} & \mathbf{O} & \overline{\mathbf{1}}_2 \end{bmatrix}, \text{and } \mathbf{N}_d \equiv \begin{bmatrix} \overline{\mathbf{N}}_0 & & \mathbf{O}'s \\ & \overline{\mathbf{N}}_1 & \\ \mathbf{O}'s & & \overline{\mathbf{N}}_2 \end{bmatrix} \tag{4.34}$$

where $\mathbf{N}_l$ and $\mathbf{N}_d$ are 78×78 and 78×18 matrices. It is worth noting that the module-level expressions in Eq. (4.34) are exactly the same as Eq. (4.32) as they possess same number of modules and similar module architecture, however the size of their block elements differs. For example, $\overline{\mathbf{1}}_0$, $\overline{\mathbf{1}}_1$, and $\overline{\mathbf{1}}_2$ are the 6×6, 36×36 and 36×36 identity matrices. The block elements $\overline{\mathbf{A}}_{1,0}$, $\overline{\mathbf{A}}_{2,0}$, $\overline{\mathbf{N}}_1$, $\overline{\mathbf{N}}_2$, and $\overline{\mathbf{N}}_0$ are obtained by using Eqs. (4.22) and (4.24) as

$$\overline{\mathbf{A}}_{i,0} = \begin{bmatrix} \mathbf{\Lambda}_{1,0} \\ \mathbf{\Lambda}_{2,0} \\ \mathbf{\Lambda}_{3,0} \end{bmatrix}^i ; \overline{\mathbf{N}}_i = \begin{bmatrix} \mathbf{P}_1 & \mathbf{O} & \mathbf{O} \\ \mathbf{\Lambda}_{2,1}\mathbf{P}_1 & \mathbf{P}_2 & \mathbf{O} \\ \mathbf{\Lambda}_{3,1}\mathbf{P}_1 & \mathbf{\Lambda}_{3,2}\mathbf{P}_2 & \mathbf{P}_3 \end{bmatrix}^i ; \text{ for } i = 1, 2 \text{ and } \overline{\mathbf{N}}_0 = \mathbf{P}_0 \tag{4.35}$$

In contrast to the example of the planar biped in Sect. 4.4.2, $\mathbf{A}_{k,l}$ and $\mathbf{p}_k$ in the expression of $\overline{\mathbf{A}}_{i,0}$ and $\overline{\mathbf{N}}_i$ are replaced by $\mathbf{\Lambda}_{l,k}$ and $\mathbf{P}_k$ due to the presence of multiple-DOF joints. The expressions of matrices $\mathbf{\Lambda}_{l,k}$ and $\mathbf{P}_k$ in Eq. (4.35) depend on the number of joint variables associated with the joints, and are obtained by using Eq. (4.14). For example, the element $[\mathbf{\Lambda}_{1,0}]^i$ in the expression of $\overline{\mathbf{A}}_{i,0}$, and the elements $[\mathbf{\Lambda}_{2,1}]^i$ and $[\mathbf{P}_1]^i$ in the expression of $\overline{\mathbf{N}}_i$ are obtained by using Eq. (4.14) as

$$[\mathbf{\Lambda}_{1,0}]^i = \begin{bmatrix} \mathbf{A}_{1,0} \\ \mathbf{A}_{1,0} \\ \mathbf{A}_{1,0} \end{bmatrix}^i, [\mathbf{\Lambda}_{2,1}]^i = \begin{bmatrix} \mathbf{O} & \mathbf{O} & \mathbf{A}_{2,1} \end{bmatrix}^i, [\mathbf{P}_1]^i = \begin{bmatrix} \mathbf{p}_{1_1} & \mathbf{O} & \mathbf{O} \\ \mathbf{p}_{1_1} & \mathbf{p}_{1_2} & \mathbf{O} \\ \mathbf{p}_{1_1} & \mathbf{p}_{1_2} & \mathbf{p}_{1_3} \end{bmatrix}^i;$$
$$\text{for } i = 1, 2; \tag{4.36}$$

Other elements of $\overline{\mathbf{A}}_{i,0}$ and $\overline{\mathbf{N}}_i$ can similarly be obtained. Moreover, $\mathbf{A}_{l,k}$ and $\mathbf{p}_{k_j}$ in Eq. (4.36) can be obtained by using Eq. (4.3).

4.5 Summary

In this chapter, kinematics of a tree-type robotic system was presented with the help of the concept of kinematic modules. The kinematic constraints were obtained first at intra-modular level after introducing the efficient generic representation of multiple-DOF joints. Kinematic constraints were then obtained at inter-modular level. The modular representation enables one to extrapolate the concept of link-to-link velocity transformation to module-to-module velocity transformation. The modular approach lends its utility in writing module-level DeNOC matrices, which is a generic form of the DeNOC matrices developed for serial chain robotic system. In this way, several module-level concepts like module-twist propagation, module joint-motion propagation, etc. evolved and these concepts were found to be very useful in treating large tree-type robotic system with multiple-degrees-of-freedom joints.

Chapter 5
Dynamics of Tree-Type Robotic Systems

As reviewed in Chap. 2, Newton-Euler (NE) equations of motion are found to be popular in dynamic formulations. Several methods were also proposed by various researchers to obtain the Euler-Langrage's form of NE equations of motion. One of these methods is based on velocity transformation of the kinematic constraints, e.g., the Natural Orthogonal Complement (NOC) or the Decoupled NOC (DeNOC), as obtained in Chap. 4. The DeNOC matrices of Eq. (4.28) are used in this chapter to obtain the minimal order dynamic equations of motion that have several benefits.

5.1 Dynamic Formulation Using the DeNOC Matrices

The DeNOC-based methodology for dynamic modeling of a general multibody system, be it serial, tree-type or closed-loop, begins with the uncoupled Newton-Euler (NE) equations of motion of all the links constituting the system. The NE equations of motion are first formulated in a matrix-vector form for a serial kinematic module followed by a tree-type system consisting of the modules.

5.1.1 NE Equations of Motion for a Serial Module

Let us consider the *i*th module of Fig. 4.6. The *i*th module contains η^i serially connected links. The NE equations of motion for the *k*th link of the *i*th module can be written as (Greenwood 1988)

$$\begin{aligned} \mathbf{I}_k^c \dot{\boldsymbol{\omega}}_k + \boldsymbol{\omega}_k \times \mathbf{I}_k^c \boldsymbol{\omega}_k &= \mathbf{n}_k^c \\ m_k \ddot{\mathbf{c}}_k &= \mathbf{f}_k^c \end{aligned} \tag{5.1}$$

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_5,

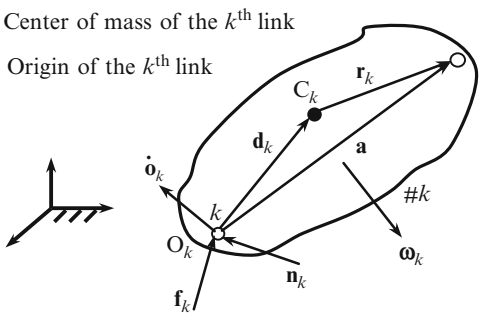


Fig. 5.1 Motion of link k^i in module M^i

where $\mathbf{n}_k^C$ and $\mathbf{f}_k^C$ are the resultant moment about and force applied at the Centre of Mass (COM), C_k, whereas, $\mathbf{I}_k^C$ is the inertia tensor about C_k, and m_k is the mass of kth link. Moreover, $\ddot{\mathbf{c}}_k$ is the linear acceleration of C_k and $\boldsymbol{\omega}_k$ is the angular velocity of the link. It is worth mentioning that the main objective of dynamic analysis is to calculate either joint torques or joint motions. Hence, if the origin of a link, which lies on the joint axis, is selected as a reference point, instead of the COM, efficient recursive inverse and forward dynamics algorithms can be obtained. This was also shown by Stelzle et al. (1995). In order to represent Eq. (5.1) with respect to the origin, O_k, of the kth link, Fig. 5.1, the entities $\ddot{\mathbf{c}}_k, \mathbf{I}_k^C, \mathbf{f}_k^C$ and $\mathbf{n}_k^C$, respectively, are represented in terms of the linear acceleration of origin O_k, i.e., $\ddot{\mathbf{o}}_k$, the inertia tensor about O_k, namely, $\mathbf{I}_k$, the resultant force applied at O_k, $\mathbf{f}_k$, and the resultant moment about O_k, $\mathbf{n}_k$, as

$$\begin{aligned}
\ddot{\mathbf{c}}_k &= \ddot{\mathbf{o}}_k - \mathbf{d}_k \times \dot{\boldsymbol{\omega}}_k - \boldsymbol{\omega}_k \times (\mathbf{d}_k \times \boldsymbol{\omega}_k) \\
\mathbf{I}_k^C &= \mathbf{I}_k + m_i(\mathbf{d}_k \times \mathbf{1})(\mathbf{d}_k \times \mathbf{1}) \\
\mathbf{f}_k^c &= \mathbf{f}_k \\
\mathbf{n}_k^c &= \mathbf{n}_k - \mathbf{d}_k \times \mathbf{f}_k
\end{aligned} \tag{5.2}$$

where $\mathbf{d}_k$ is the 3-dimensonal vector from the origin of the kth link to its COM, as shown in Fig. 5.1, whereas $\mathbf{d}_k \times \mathbf{1}$ is the 3×3 cross-product tensor associated with $\mathbf{d}_k \equiv [d_1 \;\; d_2 \;\; d_3]^T$. The product of $\mathbf{d}_k \times \mathbf{1}$ with any vector $\mathbf{x}$ gives the cross-product vector $\mathbf{d}_k \times \mathbf{x}$. Tensor $\mathbf{d}_k \times \mathbf{1}$ is defined as

$$\mathbf{d}_k \times \mathbf{1} = \begin{bmatrix} 0 & -d_3 & d_2 \\ d_3 & 0 & -d_1 \\ -d_2 & d_1 & 0 \end{bmatrix} \tag{5.3}$$

Substituting Eq. (5.2) into Eq. (5.1), the NE equations of motion for the kth link can be represented with respect to the origin O_k as

$$\begin{aligned} \mathbf{I}_k\dot{\boldsymbol{\omega}}_k + m_k\mathbf{d}_k \times \ddot{\mathbf{o}}_k + \boldsymbol{\omega}_k \times \mathbf{I}_k\boldsymbol{\omega}_k &= \mathbf{n}_k \\ m_k\ddot{\mathbf{o}}_k - m_k\mathbf{d}_k \times \dot{\boldsymbol{\omega}}_k - \boldsymbol{\omega}_k \times (m_k\mathbf{d}_k \times \boldsymbol{\omega}_k) &= \mathbf{f}_k \end{aligned} \tag{5.4}$$

The above equations of motion are then written in terms of the 6-dimensioinal twist $\mathbf{t}_k$, twist-rate $\dot{\mathbf{t}}_k$ and wrench $\mathbf{w}_k$ (Saha and Schiehlen 2001) as

$$\mathbf{M}_k\dot{\mathbf{t}}_k + \boldsymbol{\Omega}_k\mathbf{M}_k\mathbf{E}_k\mathbf{t}_k = \mathbf{w}_k, \text{where } \mathbf{w}_k \equiv \mathbf{w}_k^D + \mathbf{w}_k^C + \mathbf{w}_k^F \tag{5.5}$$

where $\mathbf{M}_k$, $\boldsymbol{\Omega}_k$ and $\mathbf{E}_k$ are 6×6 mass, angular velocity and coupling matrices, respectively, and $\mathbf{w}_k$ is the 6-dimensional vector of wrench for the kth link. The wrench vector $\mathbf{w}_k$ is composed of $\mathbf{w}_k^D$, the wrench due to driving moments and forces, $\mathbf{w}_k^C$, the wrench due to constrained moments and forces, and $\mathbf{w}_k^F$, the wrench due to external moments and forces other than driving. The matrices $\mathbf{M}_k$, $\boldsymbol{\Omega}_k$ and $\mathbf{E}_k$, and the vector $\mathbf{w}_k$ are obtained by comparing the Eqs. (5.4) and (5.5). They are given by

$$\begin{aligned} \mathbf{M}_k &\equiv \begin{bmatrix} \mathbf{I}_k & m_k\mathbf{d}_k \times \mathbf{1} \\ -m_k\mathbf{d}_k \times \mathbf{1} & m_k\mathbf{1} \end{bmatrix}, \boldsymbol{\Omega}_k \equiv \begin{bmatrix} \boldsymbol{\omega}_k\times\mathbf{1} & \mathbf{O} \\ \mathbf{O} & \boldsymbol{\omega}_k\times\mathbf{1} \end{bmatrix}, \\ \mathbf{E}_k &\equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}, \text{and } \mathbf{w}_k \equiv \begin{bmatrix} \mathbf{n}_k \\ \mathbf{f}_k \end{bmatrix} \end{aligned} \tag{5.6}$$

In Eq. (5.6), $\boldsymbol{\omega}_k \times \mathbf{1}$ is the 3×3 cross-product tensor associated with vector $\boldsymbol{\omega}_k$. For the ith module, i.e., M_i, comprising of η^i serially connected links the NE equations of motion, Eq. (5.5), can be combined in a compact form as

$$\overline{\mathbf{M}}_i\dot{\overline{\mathbf{t}}}_t + \overline{\boldsymbol{\Omega}}_i\overline{\mathbf{M}}_i\overline{\mathbf{E}}_i\overline{\mathbf{t}}_i = \overline{\mathbf{w}}_i \tag{5.7}$$

where the matrices $\overline{\mathbf{M}}_i$, $\overline{\boldsymbol{\Omega}}_i$, and $\overline{\mathbf{E}}_i$, are of sizes $6n^i\times6n^i$ each instead of 6×6 for a link in Eq. (5.6). Moreover, $\overline{\mathbf{w}}_i$ is the $6n^i$-dimensional vector of module-wrench associated with the module M_i. Matrices $\overline{\mathbf{M}}_i$, $\overline{\boldsymbol{\Omega}}_i$, $\overline{\mathbf{E}}_i$, and vector $\overline{\mathbf{w}}_i$ are defined as

$$\begin{aligned} \overline{\mathbf{M}}_i &\equiv \text{diag}\big[\,\mathbf{M}_1 \cdots \mathbf{M}_k \cdots \mathbf{M}_\eta\,\big]^i, \\ \overline{\boldsymbol{\Omega}}_i &\equiv \text{diag}\big[\,\boldsymbol{\Omega}_1 \cdots \boldsymbol{\Omega}_k \cdots \boldsymbol{\Omega}_\eta\,\big]^i, \text{ and } \overline{\mathbf{w}}_i \equiv \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_k \\ \vdots \\ \mathbf{w}_\eta \end{bmatrix}^i \\ \overline{\mathbf{E}}_i &\equiv \text{diag}\big[\,\mathbf{E}_1 \cdots \mathbf{E}_k \cdots \mathbf{E}_\eta\,\big]^i, \end{aligned} \tag{5.8}$$

where superscript i to the bracket '[]' of matrix and vector definitions corresponds to the ith module.

5.1.2 NE Equations of Motion for a Tree-Type System

The uncoupled NE equations of motion developed in Eq. (5.7) for a module are put together for $(s+\eta_0)$ modules as

$$\mathbf{M}\dot{\mathbf{t}} + \boldsymbol{\Omega}\mathbf{M}\mathbf{E}\mathbf{t} = \mathbf{w}, \text{ where } \mathbf{w} \equiv \mathbf{w}^D + \mathbf{w}^C + \mathbf{w}^F \tag{5.9}$$

In Eq. (5.9), $\mathbf{M}$, $\boldsymbol{\Omega}$, and $\mathbf{E}$ are the $6(n+\eta_0)\times 6(n+\eta_0)$ generalized matrices of mass, angular velocity, and coupling, respectively, and $\mathbf{w}$ is the $6(n+\eta_0)$-dimensional generalized vector of wrenches. They are defined as

$$\begin{aligned} \mathbf{M} &\equiv \text{diag}\left[\,\overline{\mathbf{M}}_0\ \overline{\mathbf{M}}_1\ \cdots \overline{\mathbf{M}}_i \cdots \overline{\mathbf{M}}_s\,\right], \boldsymbol{\Omega} \equiv \text{diag}\left[\,\overline{\boldsymbol{\Omega}}_0\ \overline{\boldsymbol{\Omega}}_1 \cdots \overline{\boldsymbol{\Omega}}_i \cdots \overline{\boldsymbol{\Omega}}_s\,\right], \\ \mathbf{E} &\equiv \text{diag}\left[\,\overline{\mathbf{E}}_0\ \overline{\mathbf{E}}_1\ \cdots \overline{\mathbf{E}}_i \cdots \overline{\mathbf{E}}_{\mathbf{s}}\,\right], \text{and } \mathbf{w} \equiv \left[\,\overline{\mathbf{w}}_0^T\ \overline{\mathbf{w}}_1^T\ \cdots \overline{\mathbf{w}}_i^T \cdots \overline{\mathbf{w}}_s^T\,\right], \end{aligned} \tag{5.10}$$

where matrices $\overline{\mathbf{M}}_i$, $\overline{\boldsymbol{\Omega}}_i$, $\overline{\mathbf{E}}_i$, and vector $\overline{\mathbf{w}}_i$ are defined in Eq. (5.8).

5.1.3 Minimal-Order Equations of Motion

Power due to constraint wrenches is equal to zero (Angeles and Ma 1988). As a result, the vector of constraint forces and moments is orthogonal to the columns of the velocity transformation matrix, i.e., the NOC or the resulting DeNOC matrices of Eq. (4.28), and one may show that the pre-multiplication of Eq. (5.9) by $\mathbf{N}_d^T\mathbf{N}_l^T$ yields the minimal set of equations of motion eliminating the constraint wrenches. More specifically, $\mathbf{N}_d^T\mathbf{N}_l^T\mathbf{w}^C = \mathbf{0}$. Thus Eq. (5.9) leads to

$$\mathbf{N}_d^T\mathbf{N}_l^T\left(\mathbf{M}\dot{\mathbf{t}} + \boldsymbol{\Omega}\mathbf{M}\mathbf{E}\mathbf{t}\right) = \mathbf{N}_d^T\mathbf{N}_l^T(\mathbf{w}^D + \mathbf{w}^F) \tag{5.11}$$

Now, substitution of $\mathbf{t} = \mathbf{N}_l\mathbf{N}_d\dot{\mathbf{q}}$, from Eq. (4.28), and its time derivative into Eq. (5.11) yields the following equations of motion:

$$\mathbf{I}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} = \boldsymbol{\tau} + \boldsymbol{\tau}^F \tag{5.12}$$

where the expressions of the $(n+n_0)\times(n+n_0)$ generalized inertia matrix (GIM) $\mathbf{I}$, $(n+n_0)\times(n+n_0)$ matrix of convective inertia (MCI) terms $\mathbf{C}$, and the $(n+n_0)$-dimensional vectors of the generalized driving forces $\boldsymbol{\tau}$ and the external forces $\boldsymbol{\tau}^F$ are given as

$$\mathbf{I} \equiv \mathbf{N}^T\mathbf{M}\mathbf{N}, \mathbf{C} \equiv \mathbf{N}^T(\mathbf{M}\dot{\mathbf{N}} + \boldsymbol{\Omega}\mathbf{M}\mathbf{E}\mathbf{N}), \tau \equiv \mathbf{N}^T\mathbf{w}^D, \text{and } \boldsymbol{\tau}^F \equiv \mathbf{N}^T\mathbf{w}^F \tag{5.13}$$

where $\mathbf{N} \equiv \mathbf{N}_l\mathbf{N}_d$. Equation (5.13) represents the minimal-order dynamic equations of motion for a tree-type robotic system with general kinematic module architecture. Note that the formulation can also suitably be used to analyze a closed-loop system.

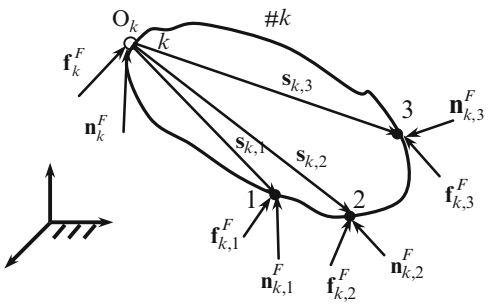


Fig. 5.2 External forces and moments on link #*k*

For that appropriate joints must be cut to form several open-loop serial or tree-type sub-systems. The cut joints are to be substituted by the constraint forces known as Lagrange multipliers. These constraint forces can then be treated as external forces to the resulting open-loop sub-systems, serial- or tree-type. Chapter 8 will illustrate the dynamic analyses of closed-loop systems using the proposed methodology.

Finally, it is to be noted that Eq. (5.12) has similar representation as that of a single-chain serial system (Saha 1999b), however, the elements of the matrices and vectors differ, and this will be evident in Sect. 5.2.

5.1.4 Wrench due to External Force, $\mathbf{w}^F$

The forces and moments due to gravitational acceleration or link-ground interactions are two typical examples of external forces that one needs to take care of, particularly, for the robotic systems analyzed in this book. Figure 5.2 shows the *k*th link subjected to external forces and moments at the points 1, 2 and 3. The vectors connecting the origin of the link O_k to the points of application of these forces and moments are represented by $\mathbf{s}_{k,1}$, $\mathbf{s}_{k,2}$, and $\mathbf{s}_{k,3}$, respectively. Resultant wrench acting at the origin of the link *k* is then defined as

$$\mathbf{w}_k^F \equiv \begin{bmatrix} \mathbf{n}_k^F \\ \mathbf{f}_k^F \end{bmatrix} \tag{5.14}$$

where $\mathbf{n}_k^F$ and $\mathbf{f}_k^F$ are the resultant moment about and the resultant force at O_k for the *k*th link. If there are n_f points of application of the external forces and moments, then wrench $\mathbf{w}_k^F$ is given by

$$\mathbf{w}_k^F = \sum_{j=1}^{n_f} \mathbf{S}_{k,j}\mathbf{w}_{k,j}^F, \quad \text{where} \quad \mathbf{S}_{k,j} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{s}_{k,j} \times \mathbf{1} \\ \mathbf{O} & \mathbf{1} \end{bmatrix} \text{and} \quad \mathbf{w}_{k,j}^F \equiv \begin{bmatrix} \mathbf{n}_{k,j}^F \\ \mathbf{f}_{k,j}^F \end{bmatrix} \tag{5.15}$$

In Eq. (5.15), $\mathbf{S}_{k,j}$ is the 6×6 matrix, and $\mathbf{s}_{k,j} \times \mathbf{1}$ is the cross-product tensor associated with vector $\mathbf{s}_{k,j}$, for $j = 1, 2, 3$, as shown in Fig. 5.2.

5.2 Generalized Inertia Matrix (GIM)

The elements of the GIM, $\mathbf{I}$, obtained in Eq. (5.13) will be derived in this section. These not only enable the factorization of the GIM but are also required for formulating the inverse dynamics algorithm reported in Chap. 7. For that, the GIM is written as

$$\mathbf{I} \equiv \mathbf{N}_d^T \tilde{\mathbf{M}} \mathbf{N}_d, \qquad \text{where} \quad \tilde{\mathbf{M}} \equiv \mathbf{N}_l^T \mathbf{M} \mathbf{N}_l \tag{5.16}$$

In Eq. (5.16), the matrix $\tilde{\mathbf{M}}$ is referred to as the generalized mass matrix of composite modules, and the matrices $\mathbf{N}_l$ and $\mathbf{N}_d$ are the module-DeNOC matrices. Substituting the expressions for $\mathbf{N}_l$ and the mass matrix $\mathbf{M}$ from Eqs. (4.29) and (5.10), respectively, into Eq. (5.16), the symmetric matrix $\tilde{\mathbf{M}}$ is obtained as

$$\tilde{\mathbf{M}} \equiv \begin{bmatrix} \tilde{\overline{\mathbf{M}}}_0 & & & & sym \\ \tilde{\overline{\mathbf{M}}}_1 \overline{\mathbf{A}}_{1,0} & \tilde{\overline{\mathbf{M}}}_1 & & & \\ \tilde{\overline{\mathbf{M}}}_2 \overline{\mathbf{A}}_{2,0} & \tilde{\overline{\mathbf{M}}}_2 \overline{\mathbf{A}}_{2,1} & \tilde{\overline{\mathbf{M}}}_2 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \tilde{\overline{\mathbf{M}}}_s \overline{\mathbf{A}}_{s,0} & \tilde{\overline{\mathbf{M}}}_s \overline{\mathbf{A}}_{s,1} & \cdots & \tilde{\overline{\mathbf{M}}}_s \overline{\mathbf{A}}_{s,s-1} & \tilde{\overline{\mathbf{M}}}_s \end{bmatrix} \quad (\tilde{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,i} \equiv \mathbf{O}, \text{if } M_j \notin \boldsymbol{\gamma}_i) \tag{5.17}$$

where $\overline{\mathbf{A}}_{i,j}$ is the $6n^i \times 6n^j$ module-twist-propagation matrix defined similar to $\overline{\mathbf{A}}_{i,\beta}$ in Eq. (4.22). Once again $\boldsymbol{\gamma}_i$ in Eq. (5.17) stands for array of all the modules originating from the module M_i including M_i, and $\tilde{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,i} \equiv \mathbf{O}$, when $M_j \notin \boldsymbol{\gamma}_i$. It is worth noting that in Eq. (5.17), the $6n^i \times 6n^i$ matrix $\tilde{\overline{\mathbf{M}}}_i$ represents the mass and inertia properties of the system comprising of rigidly connected links of modules which are upstream from the ith module as depicted in Fig. 5.3 inside the dotted boundary ($\boldsymbol{\gamma}_i$). Matrix $\tilde{\overline{\mathbf{M}}}_i$ is referred to as the mass matrix of composite-module. It is obtained as

$$\tilde{\overline{\mathbf{M}}}_i = \overline{\mathbf{M}}_i + \sum_{j \in \boldsymbol{\gamma}_i} \overline{\mathbf{A}}_{j,i}^T \overline{\mathbf{M}}_j \overline{\mathbf{A}}_{j,i} \tag{5.18}$$

In Eq. (5.18), $\tilde{\overline{\mathbf{M}}}_i$ is expressed in terms of the module mass matrix $\overline{\mathbf{M}}_j$ of all the modules upstream from module M_i, i.e., $\overline{\mathbf{M}}_j$ for all $j \in \boldsymbol{\gamma}_i$. Instead, $\tilde{\overline{\mathbf{M}}}_i$ may be expressed in terms of the already computed mass matrices of the composite-modules that are immediate children ($\boldsymbol{\xi}_i$) of the ith module as shown in Fig. 5.3. If these composite modules are denoted by $\tilde{\overline{\mathbf{M}}}_j$, then for all $j \in \boldsymbol{\xi}_i$

$$\tilde{\overline{\mathbf{M}}}_i = \overline{\mathbf{M}}_i + \sum_{j \in \boldsymbol{\xi}_i} \overline{\mathbf{A}}_{j,i}^T \tilde{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,i} \tag{5.19}$$

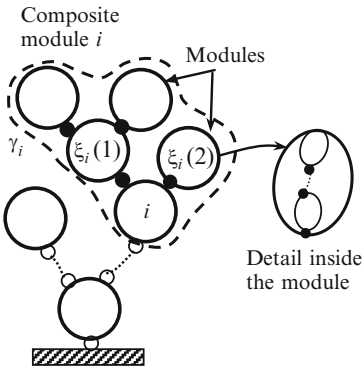


Fig. 5.3 The ith composite module

where $\boldsymbol{\xi}_i$ denotes the array of children of module M_i. If a module has no child, say, $\boldsymbol{\xi}_i = \{\}$, it is referred to as the leaf or the terminal module. For the terminal modules, the mass matrix of composite-module is simply equal to module mass matrix, i.e., $\tilde{\overline{\mathbf{M}}}_i \equiv \overline{\mathbf{M}}_i$.

Here the concept of composite-module is introduced, and this is the generalization over the similar concept of composite-body used by Featherstone (1983), Rodriguez et al. (1991) and Saha (1999b). It may be shown that their mass matrix of composite-body is a special case of Eq. (5.19), where each module has only one link, i.e., $\eta^i = 1$, for $i = 1, \ldots, s$. Finally, using the expression in Eq. (5.16), i.e., $\mathbf{I} \equiv \mathbf{N}_d^T \tilde{\mathbf{M}} \mathbf{N}_d$, the block elements of the GIM are given by

$$\mathbf{I} \equiv \begin{bmatrix} \bar{\mathbf{I}}_{0,0} & & & & sym \\ \bar{\mathbf{I}}_{1,0} & \bar{\mathbf{I}}_{1,1} & & & \\ \bar{\mathbf{I}}_{2,0} & \bar{\mathbf{I}}_{2,1} & \bar{\mathbf{I}}_{2,2} & & \\ \vdots & \vdots & \ddots & \ddots & \\ \bar{\mathbf{I}}_{s,0} & \bar{\mathbf{I}}_{s,1} & \ldots & \bar{\mathbf{I}}_{s,s-1} & \bar{\mathbf{I}}_{s,s} \end{bmatrix} \tag{5.20}$$

where the $n^i \times n^j$ block element $\bar{\mathbf{I}}_{i,j}$ is given, for $i = 1, \ldots, s$ and $j = 1, \ldots, i$ as

$$\begin{aligned} \bar{\mathbf{I}}_{i,j} \equiv \bar{\mathbf{I}}_{j,i} &= \overline{\mathbf{N}}_i^T \tilde{\overline{\mathbf{M}}}_i \overline{\mathbf{A}}_{i,j} \overline{\mathbf{N}}_j, \text{ if } M_i \in \boldsymbol{\gamma}_j (\text{For } i = j, \overline{\mathbf{A}}_{i,j} = \mathbf{1}) \\ &= \mathbf{O}, \text{otherwise} \end{aligned} \tag{5.21}$$

The analytical expression derived in Eq. (5.21) is the foundation for obtaining the analytical decomposition of the GIM, and thus Eq. (5.21) enables one to find analytical module-level inverse of the GIM and the recursive forward dynamics algorithm. Note that the word analytical is used as the expressions are obtained analytically, however it does not provide the explicit closed form expressions. It is just a compact representation.

5.3 Module-Level Decomposition of the GIM

Saha (1997) showed the $\mathbf{UDU}^{\mathrm{T}}$—**U** and **D** are upper triangular and diagonal matrices, respectively—decomposition of the generalized inertia matrix for a single-chain serial system. More recently, Featherstone (2005) proposed a similar $\mathbf{L}^{\mathrm{T}}\mathbf{DL}$ factorization, where **L** is lower triangular matrix for branched kinematic tree. However, the approach followed by the Featherstone was numerical in contrast to the analytical approach taken by Saha. In both the approaches the decomposition was carried out at the link-level. As compared to link-level approaches of Saha (1997) and Featherstone (2005), module-level decomposition is presented in this book for a tree-type system with general kinematic module architecture. This provides suitable recursive module-level expressions to obtain the implicit inverse of the GIM resulting in a recursive forward dynamics algorithm (Shah et al. 2012a). The proposed $\mathbf{UDU}^{T}$ decomposition is based on the Block Reverse Gaussian Elimination (BRGE) of the GIM that generalizes the concept of the Reverse Gaussian Elimination (RGE) of the GIM arrived from the link-level velocity transformation matrix, namely, the DeNOC matrices of a serial system proposed by Saha (1997). It is worth noting that here matrices **U** and **D** are block upper triangular and diagonal matrices, respectively. Gaussian Elimination (GE), as commonly used in linear algebra (Stewart 1973), begins with the annihilation of the first column of a matrix, whereas in Reverse Gaussian Elimination annihilation starts from the last column of the matrix. Hence the adjective 'Reverse' is added. It actually helps in obtaining various element of the decomposed matrix by establishing recursive relationships from the terminal link or module to the zeroth link or module, a process otherwise not possible with the conventional GE. The RGE or BRGE also preserves the sparsity pattern of the elements of the GIM into its factor **U**. It is worth noting that BRGE is performed on the block elements of the GIM, which are square matrices of variable sizes based on the module architectures.

The BRGE of the GIM given by Eq. (5.20) starts with the annihilation of the s-th block column and proceeds to the 1st block column by using elementary Block Upper Triangular Matrices (EBUTM), i.e.,

$$\mathbf{BI} = \mathbf{L}, \text{where } \mathbf{B} \equiv \overline{\mathbf{B}}_1 \cdots \overline{\mathbf{B}}_i \cdots \overline{\mathbf{B}}_s \tag{5.22}$$

where the $(n+n_0)\times(n+n_0)$ **B** and **L** are the block upper and lower triangular matrices. In Eq. (5.22), the $(n+n_0)\times(n+n_0)$ EBUTM $\overline{\mathbf{B}}_i$ is obtained as

$$\overline{\mathbf{B}}_i = \mathbf{1} - \overline{\mathbf{U}}_i \overline{\mathbf{V}}_i^T \tag{5.23}$$

In Eq. (5.23), the $(n+n_0)\times n^i$ matrices $\overline{\mathbf{U}}_i$ and $\overline{\mathbf{V}}_i$ have the following representation

$$\overline{\mathbf{U}}_i \equiv \left[\overline{\mathbf{U}}_{0,i}^T \ \cdots \ \overline{\mathbf{U}}_{i-1,i}^T \mathbf{O} \ \cdots \ \mathbf{O}\right]^T \quad \text{and} \quad \overline{\mathbf{V}}_i \equiv [\mathbf{O} \ \cdots \ \mathbf{O} \ \mathbf{1}_i \ \cdots \ \mathbf{O}]^T \tag{5.24}$$

where $\overline{\mathbf{U}}_{j,i}$ and $\mathbf{1}_i$ are the $n^j \times n^i$ and $n^i \times n^i$ matrices, respectively, and $\mathbf{O}$ is the null matrix of compatible dimension. Using Eqs. (5.23) and (5.24), the structure of $\overline{\mathbf{B}}_i$ is obtained as

$$\overline{\mathbf{B}}_i = \begin{bmatrix} \mathbf{1}_0 & \mathbf{O} & \cdots & -\overline{\mathbf{U}}_{0,i} & \cdots & \mathbf{O} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & \mathbf{1}_{i-1} & -\overline{\mathbf{U}}_{i-1,i} & \cdots & \mathbf{O} \\ & & & \mathbf{1}_i & \cdots & \mathbf{O} \\ & & & & \ddots & \vdots \\ & \mathbf{O}'s & & & & \mathbf{1}_s \end{bmatrix} \tag{5.25}$$

The GIM after following the BRGE may be represented as

$$\mathbf{I} = \mathbf{B}^{-1}\mathbf{L}, \text{ where } \quad \mathbf{B}^{-1} \equiv \overline{\mathbf{B}}_s^{-1} \cdots \overline{\mathbf{B}}_i^{-1} \cdots \overline{\mathbf{B}}_1^{-1} \tag{5.26}$$

In the above, matrix $\overline{\mathbf{B}}_i^{-1}$ is the inverse of EBUTM which can be obtained as

$$\overline{\mathbf{B}}_i^{-1} = (\mathbf{1} - \overline{\mathbf{U}}_i \overline{\mathbf{V}}_i^T)^{-1} = \mathbf{1} + \overline{\mathbf{U}}_i \overline{\mathbf{V}}_i^T \tag{5.27}$$

Next, the multiplication of the inverse of two neighboring EBUTMs are required which can be given by

$$\begin{aligned} \overline{\mathbf{B}}_i^{-1} \overline{\mathbf{B}}_{i-1}^{-1} &= (\mathbf{1} + \overline{\mathbf{U}}_i \overline{\mathbf{V}}_i^T)(\mathbf{1} + \overline{\mathbf{U}}_{i-1} \overline{\mathbf{V}}_{i-1}^T) \\ &= \mathbf{1} + \overline{\mathbf{U}}_i \overline{\mathbf{V}}_i^T + \overline{\mathbf{U}}_{i-1} \overline{\mathbf{V}}_{i-1}^T \end{aligned} \tag{5.28}$$

Similarly, the product $\overline{\mathbf{B}}_s^{-1} \ldots \overline{\mathbf{B}}_i^{-1} \ldots \overline{\mathbf{B}}_1^{-1}$ is obtained as

$$\mathbf{B}^{-1} = \mathbf{1} + \overline{\mathbf{U}}_s \overline{\mathbf{V}}_s^T + \ldots + \overline{\mathbf{U}}_1 \overline{\mathbf{V}}_1^T \tag{5.29}$$

Using Eqs. (5.24) and (5.29), the inverse of the EBUTM $\mathbf{B}^{-1}$, denoted by $\mathbf{U}$, is represented as

$$\mathbf{U} = \mathbf{B}^{-1} = \begin{bmatrix} \mathbf{1}_0 & \overline{\mathbf{U}}_{0,1} & \ldots & \overline{\mathbf{U}}_{0,i} & \ldots & \overline{\mathbf{U}}_{0,s} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & \mathbf{1}_{i-1} & \overline{\mathbf{U}}_{i-1,i} & \ldots & \overline{\mathbf{U}}_{i-1,s} \\ & & & \mathbf{1}_i & \ldots & \overline{\mathbf{U}}_{i,s} \\ & & & & \ddots & \vdots \\ & \mathbf{O}'s & & & & \mathbf{1}_s \end{bmatrix} \tag{5.30}$$

where the $(n+n_0)\times(n+n_0)$ matrix $\mathbf{U}$ in Eq. (5.30) is a upper block triangular matrix. Now, the GIM of Eq. (5.26) may also be expressed as $\mathbf{I} = \mathbf{UL}$. Since, $\mathbf{I}$ is a symmetric positive definite matrix, the lower block triangular matrix $\mathbf{L}$ can be written as

$$\mathbf{L} = \mathbf{D}\mathbf{U}^T \tag{5.31}$$

As a result the GIM is decomposed as

$$\mathbf{I} = \mathbf{U}\mathbf{D}\mathbf{U}^T \tag{5.32}$$

where the expression for the block upper triangular matrix $\mathbf{U}$ is obtained in (5.30) and the block diagonal matrix $\mathbf{D}$ is given by

$$\mathbf{D} = \begin{bmatrix} \hat{\overline{\mathbf{I}}}_0 & & \mathbf{O}'s \\ & \hat{\overline{\mathbf{I}}}_1 & \\ & & \ddots \\ \mathbf{O}'s & & \hat{\overline{\mathbf{I}}}_s \end{bmatrix} \tag{5.33}$$

where the $n^i \times n^i$ matrix $\hat{\overline{\mathbf{I}}}_i$ is essentially the block pivot of the BRGE. It is worth noting that the elements of the matrices resulting from the $\mathbf{UDU}^{\mathrm{T}}$ decomposition of the GIM of a serial system (Saha 1997) are scalars, whereas here the elements of the matrices $\overline{\mathbf{U}}_{i,j}$ and $\hat{\overline{\mathbf{I}}}_i$, in Eqs. (5.30) and (5.33), respectively, are matrices. They are given by

$$\begin{aligned} \overline{\mathbf{U}}_{i,j} &\equiv \overline{\mathbf{N}}_i^T \overline{\mathbf{A}}_{j,i}^T \overline{\mathbf{\Psi}}_j, \ \textit{if} \ j \in \boldsymbol{\gamma}_i \\ &\equiv \mathbf{O}, \text{otherwise} \\ \hat{\overline{\mathbf{I}}}_i &\equiv \overline{\mathbf{N}}_i^T \hat{\overline{\mathbf{\Psi}}}_i \end{aligned} \tag{5.34}$$

In Eq. (5.34), the $6n^i \times n^i$ matrices $\overline{\mathbf{\Psi}}_i$ and $\hat{\overline{\mathbf{\Psi}}}_i$ are as follows:

$$\overline{\mathbf{\Psi}}_i = \hat{\overline{\mathbf{\Psi}}}_i \hat{\overline{\mathbf{I}}}_i^{-1} \text{ and } \hat{\overline{\mathbf{\Psi}}}_i = \hat{\overline{\mathbf{M}}}_i \overline{\mathbf{N}}_i \tag{5.35}$$

The $6n^i \times 6n^i$ matrix $\hat{\overline{\mathbf{M}}}_i$ in Eq. (5.35) contains the mass and inertia properties of the articulated-module i, which is obtained as

$$\hat{\overline{\mathbf{M}}}_i = \overline{\mathbf{M}}_i + \sum_{j \in \xi_i} \overline{\mathbf{A}}_{j,i}^T \overline{\mathbf{\Phi}}_j \hat{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,i} \tag{5.36}$$

where if $\boldsymbol{\xi}_i = \{\}$, then $\hat{\overline{\mathbf{M}}}_i \equiv \overline{\mathbf{M}}_i$. The articulated-module corresponding to $\hat{\overline{\mathbf{M}}}_i$ is nothing but the composite-module of Fig. 5.3 in which fixed joints are replaced with actual joints. The $n^i \times n^i$ matrix $\hat{\overline{\mathbf{I}}}_i$ in Eq. (5.34), on the other hand, may be interpreted as the generalized inertia matrix of the articulated-module. The relationship between the matrices $\hat{\overline{\mathbf{M}}}_i$ and $\hat{\overline{\mathbf{I}}}_i$ can be viewed same as that between $\mathbf{M}$ and $\mathbf{I}$ of Eqs. (5.9) and (5.13), respectively. Moreover, the $6n^i \times 6n^i$ matrix $\overline{\boldsymbol{\Phi}}_j$ in Eq. (5.36) is obtained as

$$\overline{\boldsymbol{\Phi}}_j = \mathbf{1} - \overline{\boldsymbol{\Psi}}_j \overline{\mathbf{N}}_j^T \tag{5.37}$$

Matrix $\overline{\boldsymbol{\Phi}}_j$ is referred to as the matrix of articulated-module transformation. Similar matrix, however at link-level was also derived by Lilly and Orin (1991) and Saha (1997). The one introduced here is more generic as it is applicable to any general tree-type system in contrast to only a single-chain serial system. Comparing Eqs. (5.19) and (5.36) it may be seen that it is $\overline{\boldsymbol{\Phi}}_j$, that distinguishes mass matrix of the composite-module, $\tilde{\overline{\mathbf{M}}}_i$, from the mass matrix of the articulated module, $\hat{\overline{\mathbf{M}}}_i$. Again the concept of articulated-body inertia (Featherstone 1983; Rodriguez et al. 1991; Saha 1997) is generalized to tree-type system consisting of several kinematic modules. If each module of a tree-type system has only one link, the matrix of the articulated-module degenerates to the articulated-body inertia.

5.4 Inverse of the GIM

The block $\mathbf{UDU}^T$ decomposition of the GIM presented in the previous section facilitates the analytical inversion of the GIM expressed in the form of Eq. (5.20). The inverse is given by

$$\mathbf{I}^{-1} = \mathbf{U}^{-T}\mathbf{D}^{-1}\mathbf{U}^{-1} \tag{5.38}$$

In Eq. (5.38), $\mathbf{D}^{-1}$ is the block diagonal matrix, and, $\mathbf{U}^{-1}$ ($\equiv\mathbf{B}$ of Eq. 5.22) is block upper triangular matrix. From Eq. (5.22), $\mathbf{U}^{-1}$ or $\mathbf{B}$ is given by

$$\mathbf{U}^{-1} \equiv \mathbf{B} = \overline{\mathbf{B}}_1 \ldots \overline{\mathbf{B}}_i \ldots \overline{\mathbf{B}}_s \tag{5.39}$$

Using the definition of the EBUTM, the multiplication of the two neighboring EBUTM is obtained as

$$\begin{aligned}
\overline{\mathbf{B}}_{i-1}\overline{\mathbf{B}}_i &= (\mathbf{1} - \overline{\mathbf{U}}_{i-1}\overline{\mathbf{V}}_{i-1}^T)(\mathbf{1} - \overline{\mathbf{U}}_i\overline{\mathbf{V}}_i^T) \\
&= \mathbf{1} - \overline{\mathbf{U}}_{i-1}\overline{\mathbf{V}}_{i-1}^T - (\overline{\mathbf{U}}_i\overline{\mathbf{V}}_i^T - \overline{\mathbf{U}}_{i-1}\overline{\mathbf{V}}_{i-1}^T\overline{\mathbf{U}}_i\overline{\mathbf{V}}_i^T) \\
&= \mathbf{1} - \overline{\mathbf{U}}_{i-1}\overline{\mathbf{V}}_{i-1}^T - (\overline{\mathbf{U}}_i - \overline{\mathbf{U}}_{i-1} - \overline{\mathbf{U}}_{i-1,i})\overline{\mathbf{V}}_i^T
\end{aligned} \tag{5.40}$$

where $\overline{\mathbf{U}}_{i-1,i}(=\overline{\mathbf{V}}_{i-1}^T\overline{\mathbf{U}}_i)$ is the last nonzero block element of $\overline{\mathbf{U}}_i$ as obtained in Eq. (5.24). The product of EBUTM, i.e., $\mathbf{U}^{-1}=\overline{\mathbf{B}}_1\ldots\overline{\mathbf{B}}_i\ldots\overline{\mathbf{B}}_s$, is then calculated as

$$\mathbf{U}^{-1}=\overline{\mathbf{B}}_1\ldots\overline{\mathbf{B}}_i\ldots\overline{\mathbf{B}}_s=\mathbf{1}+\hat{\overline{\mathbf{U}}}_1\overline{\mathbf{V}}_1^T+\ldots\hat{\overline{\mathbf{U}}}_i\overline{\mathbf{V}}_i^T\ldots+\hat{\overline{\mathbf{U}}}_s\overline{\mathbf{V}}_s^T \tag{5.41}$$

where $\hat{\overline{\mathbf{U}}}_i$, for $i=1,\ldots,s$, is given by

$$\hat{\overline{\mathbf{U}}}_i=-(\overline{\mathbf{U}}_i-\hat{\overline{\mathbf{U}}}_{i-1}\overline{\mathbf{U}}_{i-1,i}-\cdots-\hat{\overline{\mathbf{U}}}_1\overline{\mathbf{U}}_{1,i});\hat{\overline{\mathbf{U}}}_1=\overline{\mathbf{U}}_1 \text{ and } \overline{\mathbf{U}}_{j,i}=\mathbf{O} \text{ if } i\notin\boldsymbol{\gamma}_j \tag{5.42}$$

Based on Eq. (5.41) the $(n+n_0)\times(n+n_0)$ block upper-triangular and diagonal matrices $\mathbf{U}^{-1}$ and $\mathbf{D}^{-1}$ have the following forms:

$$\mathbf{U}^{-1}\equiv\begin{bmatrix}\mathbf{1}_0 & \hat{\overline{\mathbf{U}}}_{0,1} & \ldots & \hat{\overline{\mathbf{U}}}_{1,s}\\ & \mathbf{1}_1 & & \vdots\\ & & \ddots & \hat{\overline{\mathbf{U}}}_{s-1,s}\\ \mathbf{O}'s & & & \mathbf{1}_s\end{bmatrix},\text{and } \mathbf{D}^{-1}\equiv\begin{bmatrix}\hat{\overline{\mathbf{I}}}_0^{-1} & & \mathbf{O}'s & \\ & \hat{\overline{\mathbf{I}}}_1^{-1} & & \\ & & \ddots & \\ & \mathbf{O}'s & & \hat{\overline{\mathbf{I}}}_s^{-1}\end{bmatrix} \tag{5.43}$$

where the block elements $\hat{\overline{\mathbf{U}}}_{i,j}$ and $\hat{\overline{\mathbf{I}}}_i^{-1}$ are obtained as

$$\begin{aligned}\hat{\overline{\mathbf{U}}}_{i,j}&\equiv-\overline{\mathbf{N}}_i^T\hat{\overline{\mathbf{A}}}_{j,i}{}^T\overline{\boldsymbol{\Psi}}_j,\text{if } j\in\boldsymbol{\gamma}_i\\&\equiv\mathbf{O},\text{otherwise}\\ \hat{\overline{\mathbf{I}}}_i^{-1}&\equiv(\overline{\mathbf{N}}_i^T\hat{\overline{\boldsymbol{\Psi}}}_i)^{-1}\end{aligned} \tag{5.44}$$

In Eq. (5.44), $\hat{\overline{\mathbf{A}}}_{j,i}$ may be interpreted as the articulated module-twist propagation matrix, and has following representation:

$$\hat{\overline{\mathbf{A}}}_{j,i}\equiv\hat{\overline{\mathbf{A}}}_{j,h}\overline{\boldsymbol{\Phi}}_h\hat{\overline{\mathbf{A}}}_{h,i} \tag{5.45}$$

where h is any intermediate serially connected module between modules j and i. Interestingly, it is shown in Eq. (4.23) that the module-twist propagation matrix $\overline{\mathbf{A}}_{j,i}$ is represented as $\overline{\mathbf{A}}_{j,h}\overline{\mathbf{A}}_{h,i}$. Hence, once again it is the matrix of articulated module transformation, $\overline{\boldsymbol{\Phi}}_h$, that separates matrix $\hat{\overline{\mathbf{A}}}_{j,i}$ from matrix $\overline{\mathbf{A}}_{j,i}$. Moreover for any two adjoining modules, i.e., module j and its parent β_j, the articulated module-twist propagation matrix $\hat{\overline{\mathbf{A}}}_{j,\beta}$ is simply equal to module-twist propagation matrix, i.e.,

$$\hat{\overline{\mathbf{A}}}_{j,\beta}=\overline{\mathbf{A}}_{j,\beta} \tag{5.46}$$

5.5 Examples

For the clearer understanding of the proposed concepts, the module-level expression of the GIM, its decomposition, and the inverse are shown for a robotic gripper and a biped.

5.5.1 A Robotic Gripper

The GIM of the tree-type robotic gripper with four modules, as shown in Fig. 4.7, is obtained by using Eqs. (5.20) and (5.21) as

$$\mathbf{I} \equiv \begin{bmatrix} \overline{\mathbf{N}}_1^T \tilde{\overline{\mathbf{M}}}_1 \overline{\mathbf{N}}_1 & & sym \\ \overline{\mathbf{N}}_2^T \tilde{\overline{\mathbf{M}}}_2 \overline{\mathbf{A}}_{2,1} \overline{\mathbf{N}}_1 & \overline{\mathbf{N}}_2^T \tilde{\overline{\mathbf{M}}}_2 \overline{\mathbf{N}}_2 & \\ \overline{\mathbf{N}}_3^T \tilde{\overline{\mathbf{M}}}_3 \overline{\mathbf{A}}_{3,1} \overline{\mathbf{N}}_1 & \mathbf{O} & \overline{\mathbf{N}}_3^T \tilde{\overline{\mathbf{M}}}_3 \overline{\mathbf{N}}_3 \end{bmatrix} \tag{5.47}$$

where $\mathbf{I}$ is the 4×4 matrix. In Eq. (5.47), the block elements $\overline{\mathbf{N}}_i$ and $\overline{\mathbf{A}}_{j,i}$ are obtained in Eq. (4.22). Moreover, the mass matrix of composite-module $\tilde{\overline{\mathbf{M}}}_i$, for $i = 3, 2, 1$, is obtained by using Eq. (5.19) as

$$\tilde{\overline{\mathbf{M}}}_3 = \overline{\mathbf{M}}_3, \; \tilde{\overline{\mathbf{M}}}_2 = \overline{\mathbf{M}}_2, \; \tilde{\overline{\mathbf{M}}}_1 = \overline{\mathbf{M}}_1 + \sum_{j=3,2} \overline{\mathbf{A}}_{j,1}^T \tilde{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,1} \tag{5.48}$$

Following the block reverse Gaussian elimination, the GIM of Eq. (5.47) is decomposed as $\mathbf{I} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where the expressions of the elements of matrices $\mathbf{U}$ and $\mathbf{D}$ are obtained using Eq. (5.34) as

$$\mathbf{U} \equiv \begin{bmatrix} \mathbf{1}_1 & \overline{\mathbf{N}}_1^T \overline{\mathbf{A}}_{2,1}^T \overline{\mathbf{\Psi}}_2 & \overline{\mathbf{N}}_1^T \overline{\mathbf{A}}_{3,1}^T \overline{\mathbf{\Psi}}_3 \\ & \mathbf{1}_2 & \mathbf{O} \\ \mathbf{O}'s & & \mathbf{1}_3 \end{bmatrix}, \text{ and } \mathbf{D} \equiv \begin{bmatrix} \overline{\mathbf{N}}_1^T \hat{\overline{\mathbf{\Psi}}}_1 & & \mathbf{O}'s \\ & \overline{\mathbf{N}}_2^T \hat{\overline{\mathbf{\Psi}}}_2 & \\ \mathbf{O}'s & & \overline{\mathbf{N}}_3^T \hat{\overline{\mathbf{\Psi}}}_3 \end{bmatrix} \tag{5.49}$$

where $\mathbf{U}$ and $\mathbf{D}$ are the 4×4 matrices. The block elements $\overline{\mathbf{\Psi}}_i$ and $\hat{\overline{\mathbf{\Psi}}}_i$, for $i = 1, 2, 3$, can be obtained using Eq. (5.35) as

$$\text{for } i = 1, 2, 3, \; \hat{\overline{\mathbf{\Psi}}}_i = \hat{\overline{\mathbf{M}}}_i \overline{\mathbf{N}}_i \text{ and } \overline{\mathbf{\Psi}}_i = \hat{\overline{\mathbf{\Psi}}}_i \hat{\overline{\mathbf{I}}}_i^{-1}, \text{ where } \hat{\overline{\mathbf{I}}}_i \equiv \overline{\mathbf{N}}_i^T \hat{\overline{\mathbf{\Psi}}}_i \tag{5.50}$$

In Eq. (5.50), the mass matrix of articulated-module $\hat{\overline{\mathbf{M}}}_i$, for $i = 3, 2, 1$, are then evaluated using Eq. (5.36), i.e.,

$$\hat{\overline{\mathbf{M}}}_3 = \overline{\mathbf{M}}_3, \; \hat{\overline{\mathbf{M}}}_2 = \overline{\mathbf{M}}_2, \; \hat{\overline{\mathbf{M}}}_1 = \overline{\mathbf{M}}_1 + \sum_{j=3,2} \overline{\mathbf{A}}_{j,1}^T \overline{\mathbf{\Phi}}_j \hat{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,1} \tag{5.51}$$

The $6n^i \times 6n^i$ matrix $\overline{\mathbf{\Phi}}_j$ in Eq. (5.51) is obtained below using Eq. (5.37):

$$\overline{\mathbf{\Phi}}_j = \mathbf{1} - \overline{\mathbf{\Psi}}_j \overline{\mathbf{N}}_j^T, \text{for } j = 2, 3 \tag{5.52}$$

The inverse of the decomposed GIM, i.e., $\mathbf{U}^{-1}$ and $\mathbf{D}^{-1}$ are obtained next from Eqs. (5.43) and (5.44) as

$$\begin{aligned} \mathbf{U}^{-1} &\equiv \begin{bmatrix} \mathbf{1}_1 & -\overline{\mathbf{N}}_1^T \hat{\overline{\mathbf{A}}}_{2,1}^T \overline{\mathbf{\Psi}}_2 & -\overline{\mathbf{N}}_1^T \hat{\overline{\mathbf{A}}}_{3,1}^T \overline{\mathbf{\Psi}}_3 \\ & \mathbf{1}_2 & \mathbf{O} \\ \mathbf{O}'s & & \mathbf{1}_3 \end{bmatrix}, \text{ and} \\ \mathbf{D}^{-1} &\equiv \begin{bmatrix} (\overline{\mathbf{N}}_1^T \hat{\overline{\mathbf{\Psi}}}_1)^{-1} & & \mathbf{O}'s \\ & (\overline{\mathbf{N}}_2^T \hat{\overline{\mathbf{\Psi}}}_2)^{-1} & \\ \mathbf{O}'s & & (\overline{\mathbf{N}}_3^T \hat{\overline{\mathbf{\Psi}}}_3)^{-1} \end{bmatrix} \end{aligned} \tag{5.53}$$

where $\hat{\overline{\mathbf{A}}}_{2,1}^T$ and $\hat{\overline{\mathbf{A}}}_{3,1}^T$ are available from Eq. (5.45), namely,

$$\hat{\overline{\mathbf{A}}}_{2,1} \equiv \overline{\mathbf{A}}_{2,1}, \text{and } \hat{\overline{\mathbf{A}}}_{3,1} \equiv \overline{\mathbf{A}}_{3,1} \tag{5.54}$$

In Eq. (5.54), the expressions for $\hat{\overline{\mathbf{A}}}_{2,1}$ and $\hat{\overline{\mathbf{A}}}_{3,1}$ are simply corresponding to module twist-propagation matrices, i.e., $\overline{\mathbf{A}}_{2,1}$ and $\overline{\mathbf{A}}_{3,1}$, respectively, as module M_1 is the parent of modules M_2 and M_3.

5.5.2 *A Biped*

The gripper mechanism presented in the previous subsection has its base fixed. In this subsection, the proposed concept is illustrated for a floating-base biped as shown in Fig. 4.8. The GIM of the biped with three modules is obtained using Eqs. (5.20) and (5.21) as

$$\mathbf{I} \equiv \begin{bmatrix} \overline{\mathbf{N}}_0^T \tilde{\overline{\mathbf{M}}}_0 \overline{\mathbf{N}}_0 & & sym \\ \overline{\mathbf{N}}_1^T \tilde{\overline{\mathbf{M}}}_1 \overline{\mathbf{A}}_{1,0} \overline{\mathbf{N}}_0 & \overline{\mathbf{N}}_1^T \tilde{\overline{\mathbf{M}}}_1 \overline{\mathbf{N}}_1 & \\ \overline{\mathbf{N}}_2^T \tilde{\overline{\mathbf{M}}}_2 \overline{\mathbf{A}}_{2,0} \overline{\mathbf{N}}_0 & \mathbf{O} & \overline{\mathbf{N}}_2^T \tilde{\overline{\mathbf{M}}}_2 \overline{\mathbf{N}}_2 \end{bmatrix} \tag{5.55}$$

where $\mathbf{I}$ is the 12×12 matrix. After decomposition of the GIM in Eq. (5.55) the 12×12 matrices $\mathbf{U}$ and $\mathbf{D}$ are obtained by using Eq. (5.34) as

$$\mathbf{U} \equiv \begin{bmatrix} \mathbf{1}_0 & \overline{\mathbf{N}}_0^T \overline{\mathbf{A}}_{1,0}^T \overline{\mathbf{\Psi}}_1 & \overline{\mathbf{N}}_0^T \overline{\mathbf{A}}_{2,0}^T \overline{\mathbf{\Psi}}_2 \\ & \mathbf{1}_1 & \mathbf{O} \\ \mathbf{O}'s & & \mathbf{1}_2 \end{bmatrix}, \text{and } \mathbf{D} \equiv \begin{bmatrix} \overline{\mathbf{N}}_0^T \hat{\overline{\mathbf{\Psi}}}_0 & & \mathbf{O}'s \\ & \overline{\mathbf{N}}_1^T \hat{\overline{\mathbf{\Psi}}}_1 & \\ \mathbf{O}'s & & \overline{\mathbf{N}}_2^T \hat{\overline{\mathbf{\Psi}}}_2 \end{bmatrix} \tag{5.56}$$

In Eq. (5.56), the block elements $\overline{\mathbf{\Psi}}_i$ and $\hat{\overline{\mathbf{\Psi}}}_i$, for $i = 0$, 1, and 2, are obtained by using Eq. (5.35) as

$$\text{for } i = 0, 1, 2, \hat{\overline{\mathbf{\Psi}}}_i = \hat{\overline{\mathbf{M}}}_i \overline{\mathbf{N}}_i \text{ and } \overline{\mathbf{\Psi}}_i = \hat{\overline{\mathbf{\Psi}}}_i \hat{\overline{\mathbf{I}}}_i^{-1} \text{ where } \hat{\overline{\mathbf{I}}}_i \equiv \overline{\mathbf{N}}_i^T \hat{\overline{\mathbf{\Psi}}}_i \tag{5.57}$$

The mass matrix of articulated-module $\hat{\overline{\mathbf{M}}}_i$, for $i = 2$, 1, 0, is then evaluated using Eq. (5.36) as

$$\hat{\overline{\mathbf{M}}}_2 = \overline{\mathbf{M}}_2, \hat{\overline{\mathbf{M}}}_1 = \overline{\mathbf{M}}_1, \hat{\overline{\mathbf{M}}}_0 = \overline{\mathbf{M}}_0 + \sum_{j=2,1} \overline{\mathbf{A}}_{j,0}^T \overline{\mathbf{\Phi}}_j \hat{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,0} \tag{5.58}$$

The $6n^i \times 6n^i$ matrix $\overline{\mathbf{\Phi}}_j$ in Eq. (5.58) is obtained below using Eq. (5.37):

$$\overline{\mathbf{\Phi}}_j = \mathbf{1} - \overline{\mathbf{\Psi}}_j \overline{\mathbf{N}}_j^T, \text{ for } j = 1, 2 \tag{5.59}$$

Accordingly, the inverses of the matrices $\mathbf{U}$ and $\mathbf{D}$ are obtained by using Eqs. (5.43) and (5.44) as

$$\begin{aligned} \mathbf{U}^{-1} &\equiv \begin{bmatrix} \mathbf{1}_0 & -\overline{\mathbf{N}}_0^T \hat{\overline{\mathbf{A}}}_{1,0}^T \overline{\mathbf{\Psi}}_1 & -\overline{\mathbf{N}}_0^T \hat{\overline{\mathbf{A}}}_{2,0}^T \overline{\mathbf{\Psi}}_2 \\ & \mathbf{1}_1 & \mathbf{O} \\ \mathbf{O}'s & & \mathbf{1}_2 \end{bmatrix}, \text{ and} \\ \mathbf{D}^{-1} &\equiv \begin{bmatrix} (\overline{\mathbf{N}}_0^T \hat{\overline{\mathbf{\Psi}}}_0)^{-1} & & \mathbf{O}'s \\ & (\overline{\mathbf{N}}_1^T \hat{\overline{\mathbf{\Psi}}}_1)^{-1} & \\ \mathbf{O}'s & & (\overline{\mathbf{N}}_2^T \hat{\overline{\mathbf{\Psi}}}_2)^{-1} \end{bmatrix} \end{aligned} \tag{5.60}$$

where $\mathbf{U}^{-1}$ and $\mathbf{D}^{-1}$ are 12×12 block upper triangular and diagonal matrices. Moreover, $\hat{\overline{\mathbf{A}}}_{2,1}^T$ and $\hat{\overline{\mathbf{A}}}_{3,1}^T$ are available from Eq. (5.45), namely,

$$\hat{\overline{\mathbf{A}}}_{1,0} \equiv \overline{\mathbf{A}}_{1,0}, \text{ and } \hat{\overline{\mathbf{A}}}_{2,0} \equiv \overline{\mathbf{A}}_{2,0} \tag{5.61}$$

5.6 Advantages of Modular Framework

The concept of kinematic modules consisting of serially connected links in tree-type systems brings several advantages. They are:

- Generalization of the body-to-body velocity transformation relationships to module-to-module velocity transformation relationships where the former is a special case of the latter.
- Compact representation of the system's kinematic and dynamic models.
- Provision of module-level analytical expressions for the matrices and vectors appearing in the equations of the motion. It is worth noting that the module-level expressions of the vectors and matrices appearing in the equations of motion facilitate the physical interpretation of many associated terms, decomposition of the GIM, and analytical inversion of the GIM.
- Macroscopic purview of several kinematic and dynamic properties like matrix of articulated-module, articulated-module transformation, etc.
- Uniform development of inverse and forward dynamics algorithms with inter- and intra-modular recursions, which will be presented in the following two chapters.
- Ease of adopting already developed serial-chain algorithms (Saha 1997, 1999b) for each individual serial module of a tree-type system at hand.
- Possibility of repeating the computations of a module to another module if the tree-type system has same number of links in each module.
- Ease of investigating any inconsistency in the results of a tree-type system by examining the results of one module at a time.
- Hybrid parallel-recursive dynamics algorithms can be obtained, where for a given module architecture, modules are solved parallelly whereas the links inside modules are solved recursively.

5.7 Summary

In this chapter, dynamic modeling of a tree-type robotic system is proposed based on kinematic modularization obtained in Chap. 4. The dynamic equations of motion are obtained using the concept of module-DeNOC matrices. The module-level expressions are obtained for the Generalized Inertia Matrix (GIM) which helped in introducing the concept of mass matrix of composite-module. The analytical expression of the GIM is then utilized to decompose it in the form of $\mathbf{UDU}^T$. The decomposition enables one to obtain many physical interpretation and analytical inversion of the GIM. The results of the analytical inversion will be used in Chaps. 6 and 7 to obtain recursive forward dynamics algorithms for several fixed- and floating-base robotic systems with tree-type structure and multiple-DOF joints.

Chapter 6
Recursive Dynamics for Fixed-Base Robotic Systems

In this chapter, dynamic analyses of fixed-base robotic systems are presented using the dynamic modeling presented in Chap. 5. For this, recursive inverse and forward dynamics algorithms are developed. The algorithms take care of the multiple-DOF joints in an efficient manner, as explained in Sect. 4.2.1; in contrast to treating them as a combination of several 1-DOF joints by taking into account the total number of links equal to number of 1-DOF joints or joint variables. In the presence of many multiple-DOF joints in a robotic system the latter approach is relatively inefficient due to the burden of unnecessary computations with zeros. The improvement in the computational efficiency in the presence of multiple-DOF joints are addressed in this chapter. Dynamic analyses, namely, the inverse and forward dynamics, of several systems are performed in this chapter.

6.1 Recursive Dynamics

As reviewed in Chap. 2, recursive algorithms are popular due to their efficiency, computational uniformity, and numerical stability. These essentially help in real-time computations, realistic visualization and model-based control of a robotic system. Recursive inverse dynamics algorithm helps in force analysis and control, whereas the recursive forward dynamics algorithm enables one to perform motion analysis. It is worth noting that the recursive algorithms available in the literature, e.g., Featherstone (1987) and Rodriguez et al. (1992), exploit only body-level recursions; however, here the recursion is obtained first amongst the kinematically serial modules followed by the body-level recursions inside a module (Shah 2011). In fact the latter approach encapsulates the former. As result, the problem is solved in a much elegant manner where a system has several modules, which in turn have several bodies or links in them. Hence, the inter-modular recursion with intra-modular recursion inside each module is a logical consequence of the system

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_6,

architecture. All of the above steps are achieved by defining suitable module-level Decoupled Natural Orthogonal Complement (DeNOC) matrices for a tree type system.

6.1.1 Inverse Dynamics

The problem of inverse dynamics attempts to find the joint torques and forces for a given set of joint motions of a robot under study. It is formulated with the help of Eq. (5.11) as

$$\boldsymbol{\tau} = \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^*, \text{where } \mathbf{w}^* = \mathbf{M}\dot{\mathbf{t}} + \boldsymbol{\Omega}\mathbf{M}\mathbf{E}\mathbf{t} \tag{6.1}$$

where $\mathbf{w}^*$ represents inertia wrench and $\boldsymbol{\tau}$ is the vector of joint torques and forces to be applied at the joint actuators to achieve the given motions. The dimension of $\boldsymbol{\tau}$ is 'n', that is equal to the joint variables of the robot at hand. The matrices $\mathbf{N}_l$ and $\mathbf{N}_d$ are the module-DeNOC matrices, derived in Eq. (4.28), whereas $\mathbf{M}$, $\boldsymbol{\Omega}$, and $\mathbf{E}$ are defined in Sect. 5.1.2. Equation (6.1) can then be evaluated recursively in two steps, similar to the one for a serial system (Saha 1999b), however, at the module level.

Step1: Forward recursion: Computation of $\mathbf{t}, \dot{\mathbf{t}}$ and $\mathbf{w}^*$

In this step, the generalized twist $\mathbf{t}$, generalized twist-rate $\dot{\mathbf{t}}$, and inertia wrench $\mathbf{w}^*$ are obtained recursively using two-level recursions, namely, the inter- and intra-modular levels. They are shown below:

A. Inter-modular computations: For $i = 1, \ldots, s$

$$\begin{aligned}
\bar{\mathbf{t}}_i &= \overline{\mathbf{A}}_{i,\beta}\bar{\mathbf{t}}_{\beta_i} + \overline{\mathbf{N}}_i\dot{\bar{\mathbf{q}}}_i \\
\dot{\bar{\mathbf{t}}}_i &= \dot{\overline{\mathbf{A}}}_{i,\beta}\bar{\mathbf{t}}_{\beta_i} + \overline{\mathbf{A}}_{i,\beta}\dot{\bar{\mathbf{t}}}_{\beta_i} + \dot{\overline{\mathbf{N}}}_i\dot{\bar{\mathbf{q}}}_i + \overline{\mathbf{N}}_i\ddot{\bar{\mathbf{q}}}_i \\
\overline{\mathbf{w}}_i^* &= \overline{\mathbf{M}}_i\dot{\bar{\mathbf{t}}}_i + \overline{\boldsymbol{\Omega}}_i\overline{\mathbf{M}}_i\overline{\mathbf{E}}_i\bar{\mathbf{t}}_i
\end{aligned} \tag{6.2}$$

where $\overline{\mathbf{w}}_i^*$ represents inertia wrench of the ith module, i.e., the value of the left-hand side of the NE equations given by Eq. (5.7). Moreover, $\overline{\mathbf{A}}_{i,\beta}$ and $\overline{\mathbf{N}}_i$ are module-twist propagation and module-joint-motion propagation matrices obtained in Eq. (4.22), whereas $\overline{\mathbf{M}}_i, \overline{\boldsymbol{\Omega}}_i$, and $\overline{\mathbf{E}}_i$ are defined in Eq. (5.8). To find out the above expressions one actually requires computations at the intra-modular level, which are shown next.

B. Intra-modular computations: For $k = 1^i, \ldots, \eta^i$

$$\begin{aligned}
&\text{For } j = 1 : \varepsilon_k \\
&\quad \mathbf{t}_{k_j} = \mathbf{A}_{k,\beta}\mathbf{t}_{\beta_{(k_j)}} + \mathbf{p}_{k_j}\dot{\theta}_{k_j}, \; j = 1 \\
&\qquad\; = \mathbf{t}_{\beta_{(k_j)}} + \mathbf{p}_{k_j}\dot{\theta}_{k_j}, \qquad j > 1
\end{aligned}$$

$$\begin{aligned}
\dot{\mathbf{t}}_{k_j} &= \mathbf{A}_{k,\beta}\dot{\mathbf{t}}_{\beta(k_j)} + \dot{\mathbf{A}}_{k,\beta}\mathbf{t}_{\beta(k_j)} + \boldsymbol{\Omega}_{k_j}\mathbf{p}_{k_j}\dot{\theta}_{k_j} + \mathbf{p}_{k_j}\ddot{\theta}_{k_j}, j = 1 \\
&= \dot{\mathbf{t}}_{\beta(k_j)} + \boldsymbol{\Omega}_{k_j}\mathbf{p}_{k_j}\dot{\theta}_{k_j} + \mathbf{p}_{k_j}\ddot{\theta}_{k_j}, \qquad j > 1 \\
\mathbf{w}^*_{k_j} &= 0, \qquad j < \varepsilon_k \\
&= \mathbf{M}_{k_j}\dot{\mathbf{t}}_{k_j} + \boldsymbol{\Omega}_{k_j}\mathbf{M}_{k_j}\mathbf{E}_{k_j}\mathbf{t}_{k_j}, \qquad j = \varepsilon_k
\end{aligned} \tag{6.3}$$

where $\mathbf{A}_{k,\beta}$ and $\mathbf{p}_{k_j}$ are defined in Eq. (4.3), whereas $\mathbf{M}_{k_j}$, $\boldsymbol{\Omega}_{k_j}$, and $\mathbf{E}_{k_j}$ are obtained in Eq. (5.6). Moreover, $\beta_k \equiv (k\text{-}1)$ for $k > 1$ and ε_k is the DOF of the kth joint. Hence, if the kth link is connected to its parent link by a higher-DOF joint k, i.e., $\varepsilon_k > 1$, then there are $\varepsilon_k - 1$ virtual links which have no dimensions and masses. As a result, many computations associated with these links are carefully avoided as evident from Eq. (6.3).

Step 2: Backward recursion: Computation of $\boldsymbol{\tau}$

In this step, joint torques and forces are obtained by using Eq. (6.1) as

$$\boldsymbol{\tau} = \mathbf{N}_d^T\tilde{\mathbf{w}}, \text{ where } \tilde{\mathbf{w}} \equiv \mathbf{N}_l^T\mathbf{w}^* \tag{6.4}$$

As $\mathbf{N}_l^T$ is a block upper-triangular matrix, $\tilde{\mathbf{w}}$ is obtained using backward substitutions, while the evaluation of $\boldsymbol{\tau}$ is straightforward as $\mathbf{N}_d^T$ is a block diagonal matrix. Hence, having the results of $\mathbf{w}^*$ from the above forward recursion step, this step computes the joint torques/forces using another set of inter- and intra-modular level computations that start from the last module and last body, respectively. They are shown below.

A. Inter-modular computations: For $i = s, \ldots, 1$

$$\begin{aligned}
\tilde{\overline{\mathbf{w}}}_i &= \overline{\mathbf{w}}^*_i + \sum_{j\in\boldsymbol{\xi}_i}\overline{\mathbf{A}}^T_{j,i}\tilde{\overline{\mathbf{w}}}_j \\
\overline{\boldsymbol{\tau}}_i &= \overline{\mathbf{N}}_i^T\tilde{\overline{\mathbf{w}}}_i
\end{aligned} \tag{6.5}$$

where $\tilde{\overline{\mathbf{w}}}_i \equiv \overline{\mathbf{w}}^*_i$ if $\boldsymbol{\xi}_i = \{\}$, and the intra-modular computations required to complete the above calculations are shown below:

B. Intra-modular computations: For $k = \eta^i, \ldots, 1$

$$\begin{aligned}
\text{For } j &= \varepsilon_k \ldots 1 \\
\tilde{\mathbf{w}}_{k_j} &= \mathbf{w}^*_{k_j} + \sum_{l\in\boldsymbol{\xi}_{(k_j)}}\mathbf{A}^T_{l,k}\tilde{\mathbf{w}}_l, j = \varepsilon_k, k = \eta^i \\
&= \mathbf{w}^*_{k_j} + \mathbf{A}^T_{k+1,k}\tilde{\mathbf{w}}_{(k+1)_1}, j = \varepsilon_k, k < \eta^i \\
&= \tilde{\mathbf{w}}_{k_{j+1}}, j < \varepsilon_k \\
\tau_{k_j} &= \mathbf{p}^T_{k_j}\tilde{\mathbf{w}}_{k_j}
\end{aligned} \tag{6.6}$$

where $\tilde{\mathbf{w}}_{k_j} \equiv \mathbf{w}^*_{k_j}$ if $\boldsymbol{\xi}_{(k_j)} = \{\}$. Once again, similar to Eq. (6.3), many computations in Eq. (6.6) associated with the links of multiple-DOF joints that do not have any length and mass are not carried out, thus, reducing the overall computational complexity. It may be noted that the computer implantation of the above steps requires assignment of an integer index for each of the link in a tree-type system. This is obtained as follows: An integer index for the link k^i of the ith module is

$$= k + \sum_{h=1}^{i-1} \eta^h \tag{6.7}$$

where η^h represents total number of links in the hth module. Accordingly, the computer implementation of the proposed inverse dynamics algorithm is given in Table 6.1. The computational complexities of the different steps in Table 6.1 and comparison of the computational complexity with those available in the literature are shown in Sect. 6.3.

6.1.2 *Forward Dynamics*

Forward dynamics problem attempts to find the joint motions from the knowledge of the external joint torques and forces. This enables simulation studies which provide configuration of the system at hand. Unlike the recursive inverse dynamics algorithm presented above, a recursive forward dynamics algorithm is rather complex. Inverse dynamics calculations require only matrix-vector operations which comprise of algebraic multiplications/divisions and additions/subtractions. In the case of forward dynamics, one needs to solve for the joint accelerations from a set of equations of motion, Eq. (5.12). A straightforward solution based on established Gaussian Elimination or Cholesky decomposition techniques (Strang 1998) require order (n^3) calculations. So for large n, typically $n > 7$ (Stelzle et al. 1995), computations increase so drastically that the overall computational efficiency is largely sacrificed. Moreover, such algorithms are known to have problems with numerical stability (Ascher et al. 1997; Mohan and Saha 2007). Hence, recursive order (n) forward dynamics algorithms are preferred. One such algorithm is presented here for the tree-type systems consisting of several kinematic modules and multiple-DOF joints. It is done by analytically decomposing the GIM, resulting out of the constrained equations of motion of a system, as derived in Sect. 5.3. The decomposed GIM allows one to calculate the joint accelerations recursively, as highlighted below:

- Based on the $\mathbf{UDU}^T$ decomposition of the GIM obtained in Eq. (5.32), the constrained equations of motion, Eq. (5.12), are rewritten as follows:

Table 6.1 Recursive O(n) inverse dynamics algorithm for fixed-base robotic systems

Step 1 (forward recursion): $\mathbf{t}_j, \dot{\mathbf{t}}_j$, and $\mathbf{w}_j^*$	Step 2 (backward recursion): τ_j
$j = 0$	$j = DOF$
For $i = 1 : s$ (Inter-modular)	For $i = s : 1$ (Inter-modular)
For $k = 1 : n^i$ (Intra-modular)	For $k = n^i : 1$ (Intra-modular)
$r = 1$ (Joint level)	For $r = \varepsilon_k : 2$ (Joint level)
$j = j + 1$	$\tau_j = \mathbf{p}_j^T \tilde{\mathbf{w}}_j$
if $(\beta_j \neq 0)$	$\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_j$
$\mathbf{t}_j = \mathbf{A}_{k,\beta}\mathbf{t}_{\beta_j} + \mathbf{p}_j\dot{\theta}_j$	$j = j - 1$
$\dot{\mathbf{t}}_j = \mathbf{A}_{k,\beta}\dot{\mathbf{t}}_{\beta_j} + \dot{\mathbf{A}}_{k,\beta}\mathbf{t}_{\beta_j} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	end
else	$r = 1$ (Joint level)
$\mathbf{t}_j = \mathbf{p}_j\dot{\theta}_j$	$\tau_j = \mathbf{p}_j^T \tilde{\mathbf{w}}_j$
$\dot{\mathbf{t}}_j = \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j + \boldsymbol{\rho}$	if $(\beta_j \neq 0)$
end	$\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\tilde{\mathbf{w}}_j$
$\tilde{\mathbf{w}}_j = \mathbf{0}$	end
For $r = 2 : \varepsilon_k$	$j = j - 1$
$j = j + 1$	end
$\mathbf{t}_j = \mathbf{t}_{j-1} + \mathbf{p}_j\dot{\theta}_j$	end
$\dot{\mathbf{t}}_j = \dot{\mathbf{t}}_{j-1} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	
$\tilde{\mathbf{w}}_j = \mathbf{0}$	
end	
$\mathbf{w}_k^* = \mathbf{M}_k\dot{\mathbf{t}}_j + \boldsymbol{\Omega}_j\mathbf{M}_k\mathbf{E}_k\mathbf{t}_j$	
$\tilde{\mathbf{w}}_j = \mathbf{w}_k^*$	
end	
end	

Note: The 6-dimensional vector $\boldsymbol{\rho} \equiv [\mathbf{0}^T \ \mathbf{g}^T]^T$, where '$\mathbf{g}$' is the 3-dimensional vector due to gravitational acceleration, is added to the links emanating from the base to take into account the effect of gravity to other links of the modules.

$$\mathbf{U}\mathbf{D}\mathbf{U}^T\ddot{\mathbf{q}} = \boldsymbol{\varphi}, \text{ where } \boldsymbol{\varphi} \equiv \boldsymbol{\tau}^F - \mathbf{h} \text{ and } \mathbf{h} \equiv \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}^F \tag{6.8}$$

The matrices $\mathbf{U}$ and $\mathbf{D}$ were obtained in Eqs. (5.30, 5.32, 5.33, and 5.34). Moreover, vector $\mathbf{h}$ is obtained recursively using any recursive inverse dynamics

algorithm (Walker and Orin 1982), say, the one presented in Sect. 6.1.1, where $\ddot{\mathbf{q}} = \mathbf{0}$ is substituted in step 1.

- The joint accelerations are then solved using the following three sets of linear algebraic equations:

$$\begin{aligned} &\text{(i) } \mathbf{U}\hat{\boldsymbol{\varphi}} = \boldsymbol{\varphi}, \text{where } \hat{\boldsymbol{\varphi}} \equiv \mathbf{D}\mathbf{U}^T\ddot{\mathbf{q}} \\ &\text{(ii) } \mathbf{D}\tilde{\boldsymbol{\varphi}} = \hat{\boldsymbol{\varphi}}, \text{where } \tilde{\boldsymbol{\varphi}} \equiv \mathbf{U}^T\ddot{\mathbf{q}} \\ &\text{(iii) } \mathbf{U}^T\ddot{\mathbf{q}} = \tilde{\boldsymbol{\varphi}} \end{aligned} \tag{6.9}$$

Using Eq. (6.9), the recursive algorithm, comprising of backward and forward recursions, for a general tree-type system is given next.

Step 1: Backward recursion: Computation of $\hat{\boldsymbol{\varphi}}$ and $\tilde{\boldsymbol{\varphi}}$

As $\mathbf{U}$ is a block upper-triangular matrix, $\hat{\boldsymbol{\varphi}}$ is obtained using backward substitutions, while the solution for $\tilde{\boldsymbol{\varphi}}$ is straightforward as $\mathbf{D}$ is a block diagonal matrix. The vectors $\hat{\boldsymbol{\varphi}}$ and $\tilde{\boldsymbol{\varphi}}$ are obtained using the inter- and intra-modular level computations given below:

A. Inter-modular computations: For $i = s, \ldots, 1$

$$\begin{aligned} &\text{(i) } \hat{\overline{\boldsymbol{\varphi}}}_i = \overline{\boldsymbol{\varphi}}_i - \overline{\mathbf{N}}_i^T \tilde{\overline{\boldsymbol{\eta}}}_i, \text{ where, } \tilde{\overline{\boldsymbol{\eta}}}_i = \sum_{j \in \boldsymbol{\xi}_i} \overline{\mathbf{A}}_{j,i}^T \overline{\boldsymbol{\eta}}_j, \\ &\qquad\qquad \text{and } \overline{\boldsymbol{\eta}}_j = \overline{\boldsymbol{\Psi}}_j \hat{\overline{\boldsymbol{\varphi}}}_j + \tilde{\overline{\boldsymbol{\eta}}}_j \\ &\text{(ii) } \tilde{\overline{\boldsymbol{\varphi}}}_i = \hat{\overline{\mathbf{I}}}_i^{-1} \hat{\overline{\boldsymbol{\varphi}}}_i \end{aligned} \tag{6.10}$$

where $\tilde{\overline{\boldsymbol{\eta}}}$ and $\overline{\boldsymbol{\eta}}_i$ are the $6n^i$-dimensional vectors, and $\tilde{\overline{\boldsymbol{\eta}}}_i = \mathbf{0}$, if $\boldsymbol{\xi}_i = \{\}$. Moreover $\hat{\overline{\mathbf{I}}}_i$ and $\overline{\boldsymbol{\Psi}}_i$ are obtained in Eqs. (5.34-35). Intra-modular level computations needed to complete the above steps are shown below:

B. Intra-modular computations: For $k = \eta^i, \ldots, 1$

For $j = \varepsilon_k \ldots 1$

$$\begin{aligned} &\text{i) } \hat{\varphi}_{k_j} = \varphi_{k_j} - \mathbf{p}_{k_j}^T \tilde{\boldsymbol{\eta}}_{k_j}, \text{ where } \tilde{\boldsymbol{\eta}}_{k_j} = \sum_{l \in \boldsymbol{\xi}_{(k_j)}} \mathbf{A}_{l,k}^T \boldsymbol{\eta}_l, j = \varepsilon_k, k = \eta^i \\ &\qquad\qquad = \mathbf{A}_{k+1,k}^T \boldsymbol{\eta}_{(k+1)_1}, j = \varepsilon_k, k < \eta^i \\ &\qquad\qquad = \boldsymbol{\eta}_{k_{j+1}}, j < \varepsilon_k \\ &\qquad \text{and } \boldsymbol{\eta}_l = \boldsymbol{\psi}_l \hat{\varphi}_l + \tilde{\boldsymbol{\eta}}_l \\ &\text{ii) } \tilde{\varphi}_{k_j} = \hat{\varphi}_{k_j} / \hat{m}_{k_j} \end{aligned} \tag{6.11}$$

where $\tilde{\boldsymbol{\eta}}_k$ and $\boldsymbol{\eta}_k$ are the 6-dimensional vectors. It is worth noting that if the original definition of the Euler angles (Shabana 2001) is used to define the motion of the 3-DOF spherical joint instead of EAJs as introduced in Chap. 3, $\mathbf{p}_k$, of Eq. (4.3) would have the size of 6×3 instead of 6×1 for EAJs. Hence, $\hat{m}_{k_j} = \mathbf{p}_{k_j}^T \hat{\mathbf{M}}_{k_j} \mathbf{p}_{k_j}$ of Eq. (6.11), which has been found as a scalar quantity using EAJs, would have been a 3×3 matrix using Euler Angles. As a result, it would require some extra computations to invert the 3×3 matrix in the range of $O(3^3)$, which is certainly disadvantageous computationally.

Step 2: Forward recursion: Computation of $\ddot{\mathbf{q}}$ from $\mathbf{U}^T\ddot{\mathbf{q}} = \tilde{\boldsymbol{\varphi}}$

As $\mathbf{U}^T$ is a block lower triangular matrix, the joint accelerations ($\ddot{\mathbf{q}}$) are solved next by using forward substitutions. These also require inter- and intra-modular level computations that are given below:

A. Inter-modular computations: For $i = 1, \ldots, s$

$$\ddot{\overline{\mathbf{q}}}_i = \tilde{\overline{\boldsymbol{\varphi}}}_i - \overline{\boldsymbol{\Psi}}_i^T \tilde{\overline{\boldsymbol{\mu}}}_i, \text{ where } \tilde{\overline{\boldsymbol{\mu}}}_i = \overline{\mathbf{A}}_{i,\beta} \overline{\boldsymbol{\mu}}_{\beta_i},$$
$$\text{and } \overline{\boldsymbol{\mu}}_{\beta_i} = \overline{\mathbf{N}}_{\beta_i} \ddot{\overline{\mathbf{q}}}_{\beta_i} + \tilde{\overline{\boldsymbol{\mu}}}_{\beta_i} \tag{6.12}$$

where $\overline{\boldsymbol{\mu}}_i$ and $\tilde{\overline{\boldsymbol{\mu}}}_i$ are the $6n^i$-dimensional vectors, and $\tilde{\overline{\boldsymbol{\mu}}}_i = \mathbf{0}$, if ith module emanates from the module M_0. Intra-modular level computations required for the above step are given next.

B. Intra-modular computations: For $k = 1, \ldots, \eta^i$

For $j = 1 : \varepsilon_k$

$$\ddot{\theta}_{k_j} = \tilde{\varphi}_{k_j} - \boldsymbol{\psi}_{k_j}^T \tilde{\boldsymbol{\mu}}_{k_j}, \text{ where } \tilde{\boldsymbol{\mu}}_{k_j} = \mathbf{A}_{k,\beta}\, \boldsymbol{\mu}_{\beta_{(k_j)}}, j = 1$$
$$= \boldsymbol{\mu}_{k_{j-1}}, j > 1$$
$$\text{and } \boldsymbol{\mu}_{\beta_{(k_j)}} = \mathbf{p}_{\beta_{(k_j)}} \ddot{\theta}_{\beta_{(k_j)}} + \tilde{\boldsymbol{\mu}}_{\beta_{(k_j)}} \tag{6.13}$$

In Eq. (6.13), $\boldsymbol{\mu}_k$ and $\tilde{\boldsymbol{\mu}}_k$ are the 6-dimensional vectors and $\beta_k = (k\text{-}1)$ for $k > 1$. Similar to the inverse dynamics, in forward dynamics also several computations associated with the dimension-less and mass-less virtual links in the multiple-DOF joints are not carried out. This was possible due to the introduction of the concept of Euler-Angle-Joints (EAJs) in Chap. 3. It may be noted that the derivation of the intra-modular steps from the inter-modular steps are not straightforward as in the case of inverse dynamics. Here, one essentially requires the analytical $\mathbf{U}_i \mathbf{D}_i \mathbf{U}_i^T$ decomposition of the matrix $\hat{\overline{\mathbf{I}}}_i$ obtained in Eq. (5.34). Next, $\ddot{\overline{\mathbf{q}}}_i$ of Eq. (6.12) in terms of the block expressions $\tilde{\overline{\boldsymbol{\varphi}}}_i$, $\overline{\boldsymbol{\Psi}}_i$ and $\tilde{\overline{\boldsymbol{\mu}}}_i$, are written to find $\ddot{\theta}_k$ of Eq. (6.13), for $k = 1, \ldots, \eta^i$. The actual computer implementation of the forward dynamics algorithm is shown in Table 6.2. The computational counts for various steps in Table 6.2 and

Table 6.2 Recursive O(n) forward dynamics algorithm for fixed-base robotic systems

Initialization	Step 1: $\hat{\varphi}_j$ and $\tilde{\varphi}_j$	Step 2: $\ddot{\theta}_j$
$j = 0$	$j = DOF$	$j = 0$
For $i = 1 : s$	For $i = s : 1$ (Inter-modular)	For $i = 1 : s$ (Inter-modular)
(Inter-modular)	For $k = n^i : 1$ (Intra-modular)	For $k = 1 : n^i$ (Intra-modular)
For $k = 1 : n^i$	For $r = \varepsilon_k : 2$ (Joint level)	$r = 1$ (Joint level)
(Intra-modular)	call function_1	$j = j + 1$
For $r = 1 : \varepsilon_k$	call function_2	if $(\beta_j \neq 0)$
(Joint level)	$\hat{\mathbf{M}}_{j-1} = \hat{\mathbf{M}}_{j,j}$	$\tilde{\boldsymbol{\mu}}_j = \mathbf{A}_{k,\beta}\boldsymbol{\mu}_{\beta_j}$
$j = j + 1$	$\tilde{\boldsymbol{\eta}}_{j-1} = \boldsymbol{\eta}_j$	call function_3
$\hat{\mathbf{M}}_j = \mathbf{O}$	$j = j - 1$	else
$\tilde{\boldsymbol{\eta}}_j = \mathbf{0}$	end	$\ddot{\theta}_j = \tilde{\varphi}_j$
$\tilde{\boldsymbol{\mu}}_j = \mathbf{0}$	$r = 1$	$\boldsymbol{\mu}_j = \mathbf{p}_j\ddot{\theta}_j$
end	call function_1	end
$\hat{\mathbf{M}}_j = \mathbf{M}_k$	if $(\beta_j \neq 0)$	For $r = 2 : \varepsilon_k$
end	call function_2	$j = j + 1$
end	$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\hat{\mathbf{M}}_{j,j}\mathbf{A}_{k,\beta}$	$\tilde{\boldsymbol{\mu}}_j = \boldsymbol{\mu}_{j-1}$
	$\tilde{\boldsymbol{\eta}}_{\beta_j} = \tilde{\boldsymbol{\eta}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\boldsymbol{\eta}_j$	call function_3
	end	end
	$j = j - 1$	end
	end	
	end	
	− − − − − − − − − − − − − − −	− − − − − − − − − − − − − − −
	function_1	**function_3**
	$\hat{\boldsymbol{\psi}}_j = \hat{\mathbf{M}}_j\mathbf{p}_j$	$\ddot{\theta}_j = \tilde{\varphi}_j - \boldsymbol{\psi}_j^T\tilde{\boldsymbol{\mu}}_j$
	$\hat{m}_j = \mathbf{p}_j^T\hat{\boldsymbol{\psi}}_j$	$\boldsymbol{\mu}_j = \mathbf{p}_j\ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_j$
	$\boldsymbol{\psi}_j = \hat{\boldsymbol{\psi}}_j/\hat{m}_j$	
	$\hat{\varphi}_j = \varphi_j - \mathbf{p}_j^T\tilde{\boldsymbol{\eta}}_j$	
	$\tilde{\varphi}_j = \hat{\varphi}_j/\hat{m}_j$	
	function_2	
	$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j\boldsymbol{\psi}_j^T$	
	$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j\hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_j$	

comparison of forward dynamics algorithm with those available in the literature are given in Sect. 6.3.

6.2 Applications

In this section, the developed recursive algorithms are applied to analyze several robotic systems and to present the efficiency of the algorithms. Numerical results are obtained using the Recursive Dynamics Simulator (ReDySim), a MATLAB-based software developed as a part of the research reported in Shah (2011), developed during this research work. Detailed discussion on use of ReDySim is provided in Chap. 10.

6.2.1 Robotic Gripper

A tree-type robotic gripper, as shown in Fig. 6.1, can hold objects to be manipulated by a robotic manipulator. It has four kinematic modules, namely, M_0, M_1, M_2, and M_3. Module M_0 is the fixed-base, whereas module M_1 has one link, module M_2 has two links, and module M_3 has one link. The expressions of the DeNOC matrices, the GIM, and its factors **U** and **D**, were already obtained in Eqs. (4.30), (5.47) and (5.49), respectively. Next, the steps of inverse and forward dynamics algorithms, given in Sects. 6.1.1 and 6.1.2, are shown in Tables 6.3 and 6.4, respectively.

Numerical results for inverse dynamics, i.e., to find the joint torques for a given set of input motions, are obtained using the inverse dynamics module of ReDySim. The detailed steps are shown in Table 6.3. The motion for each joint for this purpose was computed for the time function of the trajectory given below:

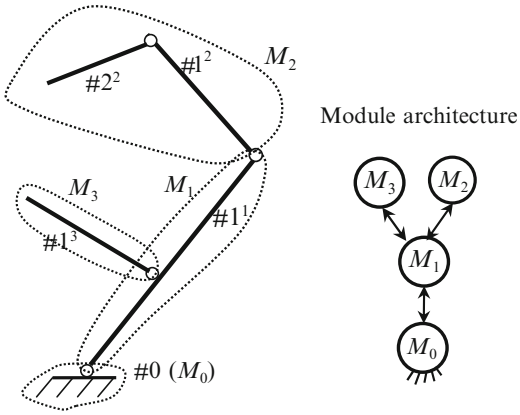


Fig. 6.1 A tree-type robotic gripper and it modularization

Table 6.3 Steps of inverse dynamics for robotic gripper

Module M_i	Inter-modular	Link k^i	Intra-modular
Step 1. Forward recursion [Based on Eqs. (6.2) and (6.3)]			
$i=1$	$\bar{\mathbf{t}}_1 = \overline{\mathbf{N}}_1\dot{\bar{\mathbf{q}}}_1; \dot{\bar{\mathbf{t}}}_1 = \dot{\overline{\mathbf{N}}}_1\dot{\bar{\mathbf{q}}}_1 + \overline{\mathbf{N}}_1\ddot{\bar{\mathbf{q}}}_1 + \bar{\boldsymbol{\rho}}$ $\overline{\mathbf{w}}_1^* \equiv \overline{\mathbf{M}}_1\dot{\bar{\mathbf{t}}}_1 + \overline{\boldsymbol{\Omega}}_1\overline{\mathbf{M}}_1\overline{\mathbf{E}}_1\bar{\mathbf{t}}_1$	$k=1^1$	$\mathbf{t}_{1^1} = \mathbf{p}_{1^1}\dot{\theta}_{1^1}$; $\dot{\mathbf{t}}_{1^1} = \boldsymbol{\Omega}_{1^1}\mathbf{p}_{1^1}\dot{\theta}_{1^1} + \mathbf{p}_{1^1}\ddot{\theta}_{1^1} + \boldsymbol{\rho}$ $\mathbf{w}_{1^1}^* \equiv \mathbf{M}_{1^1}\dot{\mathbf{t}}_{1^1} + \boldsymbol{\Omega}_{1^1}\mathbf{M}_{1^1}\mathbf{E}_{1^1}\mathbf{t}_{1^1}$
$i=2$	$\bar{\mathbf{t}}_2 = \overline{\mathbf{A}}_{2,1}\mathbf{t}_1 + \overline{\mathbf{N}}_2\dot{\bar{\mathbf{q}}}_2$ $\dot{\bar{\mathbf{t}}}_2 = \dot{\overline{\mathbf{A}}}_{2,1}\bar{\mathbf{t}}_1 + \overline{\mathbf{A}}_{2,1}\dot{\bar{\mathbf{t}}}_1 + \dot{\overline{\mathbf{N}}}_2\dot{\bar{\mathbf{q}}}_2 + \overline{\mathbf{N}}_2\ddot{\bar{\mathbf{q}}}_2$ $\overline{\mathbf{w}}_2^* \equiv \overline{\mathbf{M}}_2\dot{\bar{\mathbf{t}}}_2 + \overline{\boldsymbol{\Omega}}_2\overline{\mathbf{M}}_2\overline{\mathbf{E}}_2\bar{\mathbf{t}}_2$	$k=1^2$	$\mathbf{t}_{1^2} = \mathbf{A}_{1^2,1^1}\mathbf{t}_{1^1} + \mathbf{p}_{1^2}\dot{\theta}_{1^2}$; $\dot{\mathbf{t}}_{1^2} = \mathbf{A}_{1^2,1^1}\dot{\mathbf{t}}_{1^1} + \dot{\mathbf{A}}_{1^2,1^1}\mathbf{t}_{1^1} + \boldsymbol{\Omega}_{1^2}\mathbf{p}_{1^2}\dot{\theta}_{1^2} + \mathbf{p}_{1^2}\ddot{\theta}_{1^2}$ $\mathbf{w}_{1^2}^* \equiv \mathbf{M}_{1^2}\dot{\mathbf{t}}_{1^2} + \boldsymbol{\Omega}_{1^2}\mathbf{M}_{1^2}\mathbf{E}_{1^2}\mathbf{t}_{1^2}$
		$k=2^2$	$\mathbf{t}_{2^2} = \mathbf{A}_{2^2,1^2}\mathbf{t}_{1^2} + \mathbf{p}_{2^2}\dot{\theta}_{2^2}$; $\dot{\mathbf{t}}_{2^2} = \mathbf{A}_{2^2,1^2}\dot{\mathbf{t}}_{1^2} + \dot{\mathbf{A}}_{2^2,1^2}\mathbf{t}_{1^2} + \boldsymbol{\Omega}_{2^2}\mathbf{p}_{2^2}\dot{\theta}_{2^2} + \mathbf{p}_{2^2}\ddot{\theta}_{2^2}$ $\mathbf{w}_{2^2}^* \equiv \mathbf{M}_{2^2}\dot{\mathbf{t}}_{2^2} + \boldsymbol{\Omega}_{2^2}\mathbf{M}_{2^2}\mathbf{E}_{2^2}\mathbf{t}_{2^2}$
$i=3$	$\bar{\mathbf{t}}_3 = \overline{\mathbf{A}}_{3,1}\mathbf{t}_1 + \overline{\mathbf{N}}_3\dot{\bar{\mathbf{q}}}_3$ $\dot{\bar{\mathbf{t}}}_3 = \dot{\overline{\mathbf{A}}}_{3,1}\bar{\mathbf{t}}_1 + \overline{\mathbf{A}}_{3,1}\dot{\bar{\mathbf{t}}}_1 + \dot{\overline{\mathbf{N}}}_3\dot{\bar{\mathbf{q}}}_3 + \overline{\mathbf{N}}_3\ddot{\bar{\mathbf{q}}}_3$ $\overline{\mathbf{w}}_3^* \equiv \overline{\mathbf{M}}_3\dot{\bar{\mathbf{t}}}_3 + \overline{\boldsymbol{\Omega}}_3\overline{\mathbf{M}}_3\overline{\mathbf{E}}_3\bar{\mathbf{t}}_3$	$k=1^3$	$\mathbf{t}_{1^3} = \mathbf{A}_{1^3,1^1}\mathbf{t}_{1^1} + \mathbf{p}_{1^3}\dot{\theta}_{1^3}$ $\dot{\mathbf{t}}_{1^3} = \mathbf{A}_{1^3,1^1}\dot{\mathbf{t}}_{1^1} + \dot{\mathbf{A}}_{1^3,1^1}\mathbf{t}_{1^1} + \boldsymbol{\Omega}_{1^3}\mathbf{p}_{1^3}\dot{\theta}_{1^3} + \mathbf{p}_{1^3}\ddot{\theta}_{1^3}$ $\mathbf{w}_{1^3}^* \equiv \mathbf{M}_{1^3}\dot{\mathbf{t}}_{1^3} + \boldsymbol{\Omega}_{1^3}\mathbf{M}_{1^3}\mathbf{E}_{1^3}\mathbf{t}_{1^3}$
Step 2. Backward recursion [Based on Eqs. (6.5) and (6.6)]			
$i=3$	$\tilde{\bar{\mathbf{w}}}_3 = \overline{\mathbf{w}}_3^*; \bar{\tau}_3 = \overline{\mathbf{N}}_3^T\tilde{\bar{\mathbf{w}}}_3$	$k=1^3$	$\tilde{\mathbf{w}}_{1^3} = \mathbf{w}_{1^3}^*; \tau_{1^3} = \mathbf{p}_{1^3}^T\tilde{\mathbf{w}}_{1^3}$
$i=2$	$\tilde{\bar{\mathbf{w}}}_2 = \overline{\mathbf{w}}_2^*; \bar{\tau}_2 = \overline{\mathbf{N}}_2^T\tilde{\bar{\mathbf{w}}}_2$	$k=2^2$	$\tilde{\mathbf{w}}_{2^2} = \mathbf{w}_{2^2}^*; \tau_{2^2} = \mathbf{p}_{2^2}^T\tilde{\mathbf{w}}_{2^2}$
		$k=1^2$	$\tilde{\mathbf{w}}_{1^2} = \mathbf{w}_{1^2}^* + \mathbf{A}_{2^2,1^2}^T\tilde{\mathbf{w}}_{2^2}; \tau_{1^2} = \mathbf{p}_{1^2}^T\tilde{\mathbf{w}}_{1^2}$
$i=1$	$\tilde{\bar{\mathbf{w}}}_1 = \overline{\mathbf{w}}_1^* + \overline{\mathbf{A}}_{2,1}^T\tilde{\bar{\mathbf{w}}}_2 + \overline{\mathbf{A}}_{3,1}^T\tilde{\bar{\mathbf{w}}}_3; \bar{\tau}_1 = \overline{\mathbf{N}}_1^T\tilde{\bar{\mathbf{w}}}_1^*$	$k=1^1$	$\tilde{\mathbf{w}}_{1^1} = \mathbf{w}_{1^1}^* + \mathbf{A}_{1^2,1^1}^T\tilde{\mathbf{w}}_{1^2} + \mathbf{A}_{1^3,1^1}^T\tilde{\mathbf{w}}_{1^3}; \tau_{1^1} = \mathbf{p}_{1^1}^T\tilde{\mathbf{w}}_{1^1}$

Note that, in step 1 of Table 6.3, i.e., for $k=1^1$, the 6-dimensional vector $\boldsymbol{\rho} \equiv [\mathbf{0}^T\ \mathbf{g}^T]^T$, where '$\mathbf{g}$' is the 3-dimensional vector due to gravitational acceleration, is added to the acceleration $\dot{\mathbf{t}}_{1^1}$ of the first link of module M_1, to take into account the effect of gravity to other links of the modules

Table 6.4 Steps of forward dynamics for robotic gripper

Module M_i	Inter-modular	Link k^i	Intra-modular
Step 1. Backward Recursion: Computation of $\hat{\overline{\boldsymbol{\varphi}}}$ and $\hat{\tilde{\boldsymbol{\varphi}}}$ [Based on Eqs. (6.10) and (6.11)]			
$i=3$	$\hat{\overline{\boldsymbol{\varphi}}}_3 = \overline{\boldsymbol{\varphi}}_3; \hat{\tilde{\boldsymbol{\varphi}}}_3 = \hat{\tilde{\mathbf{I}}}_3^{-1}\hat{\overline{\boldsymbol{\varphi}}}_3$	$k=1^3$	$\hat{\varphi}_{1^3} = \varphi_{1^3}; \tilde{\varphi}_{1^3} = \hat{\varphi}_{1^3}/\hat{m}_{1^3}$
$i=2$	$\hat{\overline{\boldsymbol{\varphi}}}_2 = \overline{\boldsymbol{\varphi}}_2; \hat{\tilde{\boldsymbol{\varphi}}}_2 = \hat{\tilde{\mathbf{I}}}_2^{-1}\hat{\overline{\boldsymbol{\varphi}}}_2$	$k=2^2$	$\hat{\varphi}_{2^2} = \varphi_{2^2}; \tilde{\varphi}_{2^2} = \hat{\varphi}_{2^2}/\hat{m}_{2^2}$
		$k=1^2$	$\hat{\varphi}_{1^2} = \varphi_{1^2} - \mathbf{p}_{1^2}^T\tilde{\boldsymbol{\eta}}_{1^2}; \tilde{\boldsymbol{\eta}}_{1^2} = \mathbf{A}_{2^2,1^2}^T\boldsymbol{\eta}_{2^2};$ $\boldsymbol{\eta}_{2^2} = \boldsymbol{\psi}_{2^2}\hat{\varphi}_{2^2}$ $\tilde{\varphi}_{1^2} = \hat{\varphi}_{1^2}/\hat{m}_{1^2}$
$i=1$	$\hat{\overline{\boldsymbol{\varphi}}}_1 = \overline{\boldsymbol{\varphi}}_1 - \overline{\mathbf{N}}_1^T\tilde{\overline{\boldsymbol{\eta}}}_1;$ $\tilde{\overline{\boldsymbol{\eta}}}_1 = \overline{\mathbf{A}}_{2,1}^T\overline{\boldsymbol{\eta}}_2 + \overline{\mathbf{A}}_{3,1}^T\overline{\boldsymbol{\eta}}_3;$ $\overline{\boldsymbol{\eta}}_2 = \overline{\boldsymbol{\Psi}}_2\hat{\tilde{\boldsymbol{\varphi}}}_2\overline{\boldsymbol{\eta}}_3 = \overline{\boldsymbol{\Psi}}_3\hat{\tilde{\boldsymbol{\varphi}}}_3$ $\tilde{\overline{\boldsymbol{\varphi}}}_1 = \hat{\tilde{\mathbf{I}}}_1^{-1}\hat{\overline{\boldsymbol{\varphi}}}_1$	$k=1^1$	$\hat{\varphi}_{1^1} = \varphi_{1^1} - \mathbf{p}_{1^1}^T\tilde{\boldsymbol{\eta}}_{1^1};$ $\tilde{\boldsymbol{\eta}}_{1^1} = \mathbf{A}_{1^2,1^1}^T\boldsymbol{\eta}_{1^2} = \mathbf{A}_{1^3,1^1}^T\boldsymbol{\eta}_{1^3};$ $\boldsymbol{\eta}_{1^2} = \boldsymbol{\psi}_{1^2}\hat{\varphi}_{1^2} + \tilde{\boldsymbol{\eta}}_{1^2};$ $\boldsymbol{\eta}_{1^3} = \boldsymbol{\psi}_{1^3}\hat{\varphi}_{1^3};$ $\tilde{\varphi}_{1^1} = \hat{\varphi}_{1^1}/\hat{m}_{1^1}$
Step 2. Forward recursion: Computation of $\ddot{\mathbf{q}}$ [Based on Eqs. (6.12) and (6.13)]			
$i=1$	$\ddot{\overline{\mathbf{q}}}_1 = \tilde{\overline{\boldsymbol{\varphi}}}_1$	$k=1^1$	$\ddot{\theta}_{1^1} = \tilde{\varphi}_{1^1}$
$i=2$	$\ddot{\tilde{\mathbf{q}}}_2 = \tilde{\overline{\boldsymbol{\varphi}}}_2 - \overline{\boldsymbol{\Psi}}_2^T\tilde{\overline{\boldsymbol{\mu}}}_2;$ $\tilde{\overline{\boldsymbol{\mu}}}_2 = \overline{\mathbf{A}}_{2,1}\overline{\boldsymbol{\mu}}_1; \overline{\boldsymbol{\mu}}_1 = \overline{\mathbf{N}}_1\ddot{\overline{\mathbf{q}}}_1$	$k=1^2$	$\ddot{\theta}_{1^2} = \tilde{\varphi}_{1^2} - \boldsymbol{\psi}_{1^2}^T\tilde{\boldsymbol{\mu}}_{1^2};$ $\tilde{\boldsymbol{\mu}}_{1^2} = \mathbf{A}_{1^2,1^1}\boldsymbol{\mu}_{1^1}; \boldsymbol{\mu}_{1^1} = \mathbf{p}_{1^1}\ddot{\theta}_{1^1}$
		$k=2^2$	$\ddot{\theta}_{2^2} = \tilde{\varphi}_{2^2} - \boldsymbol{\psi}_{2^2}^T\tilde{\boldsymbol{\mu}}_{2^2};$ $\tilde{\boldsymbol{\mu}}_{2^2} = \mathbf{A}_{2^2,1^2}\boldsymbol{\mu}_{1^2};$ $\boldsymbol{\mu}_{1^2} = \mathbf{p}_{1^2}\ddot{\theta}_{1^2} + \tilde{\boldsymbol{\mu}}_{1^2}$
$i=3$	$\ddot{\tilde{\mathbf{q}}}_3 = \tilde{\overline{\boldsymbol{\varphi}}}_3 - \overline{\boldsymbol{\Psi}}_3^T\tilde{\overline{\boldsymbol{\mu}}}_3;$ $\tilde{\overline{\boldsymbol{\mu}}}_3 = \overline{\mathbf{A}}_{3,1}\overline{\boldsymbol{\mu}}_1; \overline{\boldsymbol{\mu}}_1 = \overline{\mathbf{N}}_1\ddot{\overline{\mathbf{q}}}_1$	$k=1^3$	$\ddot{\theta}_{1^3} = \tilde{\varphi}_{1^3} - \boldsymbol{\psi}_{1^3}^T\tilde{\boldsymbol{\mu}}_{1^3};$ $\tilde{\boldsymbol{\mu}}_{1^3} = \mathbf{A}_{1^3,1^1}\boldsymbol{\mu}_{1^1}; \boldsymbol{\mu}_{1^1} = \mathbf{p}_{1^1}\ddot{\theta}_{1^1}$

Table 6.5 Joint motions for robotic gripper

Joints	1^1	1^2	2^2	1^3
$\theta\ (0)$	0	0	0	90°
$\theta\ (T)$	60°	80°	80°	120°

$$\theta(t) = \theta(0) + \frac{\theta(T) - \theta(0)}{T}\left[t - \frac{T}{2\pi}\sin\left(\frac{2\pi}{T}t\right)\right] \tag{6.14}$$

where $\theta(0)$ and $\theta(T)$ denote the initial and final joint angles, respectively, as given in Table 6.5. The joint trajectory in Eq. (6.14) is so chosen that the initial and final joint rates and accelerations for all the joints are zeros. These ensure smooth motion of the joints. The length and mass of the links were taken as $l_{1^1} = 0.1$m, $l_{1^2} = l_{2^2} = l_{1^3} = 0.05$m, $m_{1^1} = 0.4$Kg, and $m_{1^2} = m_{2^2} = m_{1^3} = 0.2$Kg. The joint torques for the robotic gripper are then plotted in Fig. 6.2.

In order to validate the results a CAD model of the gripper mechanism was developed in ADAMS (2004) software and used for the computation of the joint torques. The results are also shown in Fig. 6.2, which show close match between the values obtained using the proposed inverse dynamics algorithm of ReDySim. Hence, the numerical results are validated. As far as CPU time is concerned, ReDySim took only 0.025 s on Intel T2300@1.66 GHz computing system. The

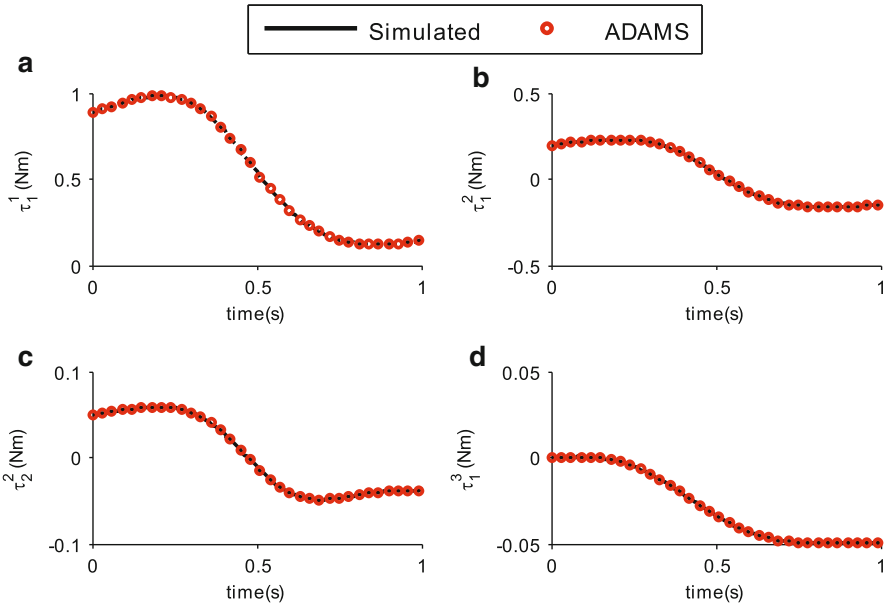


Fig. 6.2 Joint torques for robotic gripper. (**a**) Torques at joints 1^1. (**b**) Torques at joints 1^2. (**c**) Torques at joints 2^2. (**d**) Torques at joints 1^3

ADMAS software, however, took 1.95 s, which is longer as expected for general purpose software whose efficiency cannot really be compared with the customized program like ReDySim. The comparison of the efficiency with other existing algorithms is provided later in Sect. 6.3.

For simulation studies of the robotic gripper, forward dynamics module of the ReDySim based on the recursive algorithm in Table 6.2 is used. Numerical results for the acceleration were obtained for the free-fall of the gripper, i.e., it was left to move under gravity without any external torques applied at the joints. The accelerations were then numerically integrated twice using the Ordinary Differential Equation (ODE) solver 'ode45' (default in ReDySim) of MATLAB. The 'ode45' solver is based on the explicit Runge-Kutta formula given in Dormand and Prince (1980). The initial joint angles and rates are taken as $\theta_{1^1} = -60°$, $\theta_{2^1} = \theta_{2^2} = \theta_{1^3} = 0$, and $\dot{\theta}_{1^1} = \dot{\theta}_{1^2} = \dot{\theta}_{2^2} = \dot{\theta}_{1^3} = 0$, respectively. Figure 6.3 shows a comparison of the simulated joint angles with the same obtained in ADAMS (by using RKF45 solver) over the time duration of 1 s with the step size of 0.001 s. The ReDySim took 0.29 s in contrast to 39 s required by ADAMS. In order to show the convergence of the results, a variation of the simulated joint angle θ_{1^1} using the proposed algorithm and that of using the ADAMS software is plotted in Fig. 6.4. The resulting plot is a 45° line, and thus shows a close match between the results.

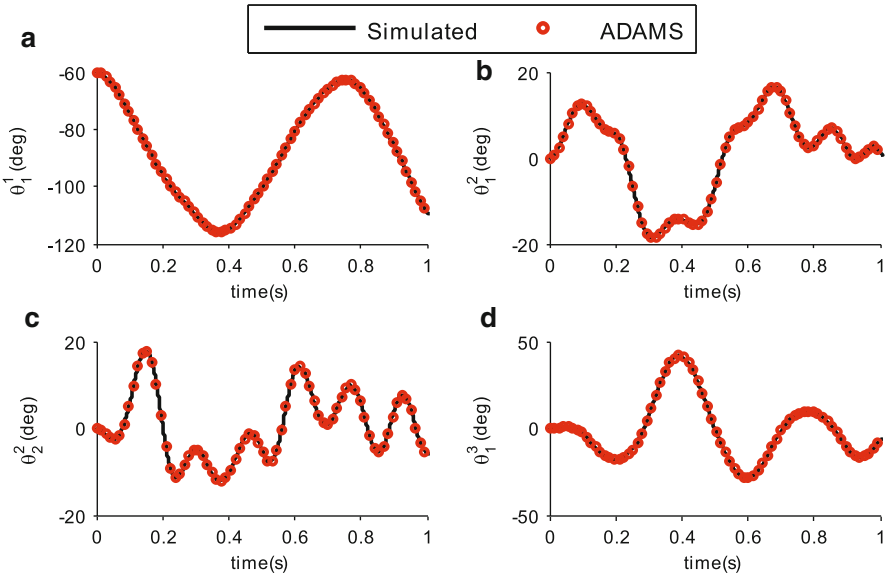


Fig. 6.3 Simulated joint angles for robotic gripper. (**a**) Joint 1^1. (**b**) Joint 2^2. (**c**) Joints 1^2. (**d**) Joint 1^3

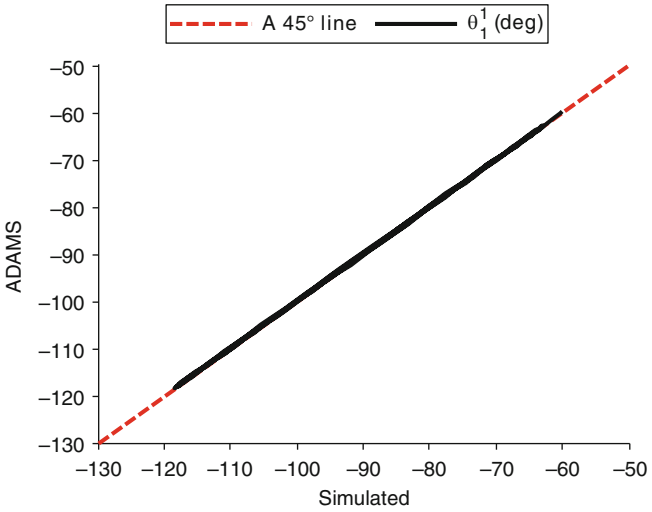


Fig. 6.4 Convergence of the simulated joint angle θ_{11}

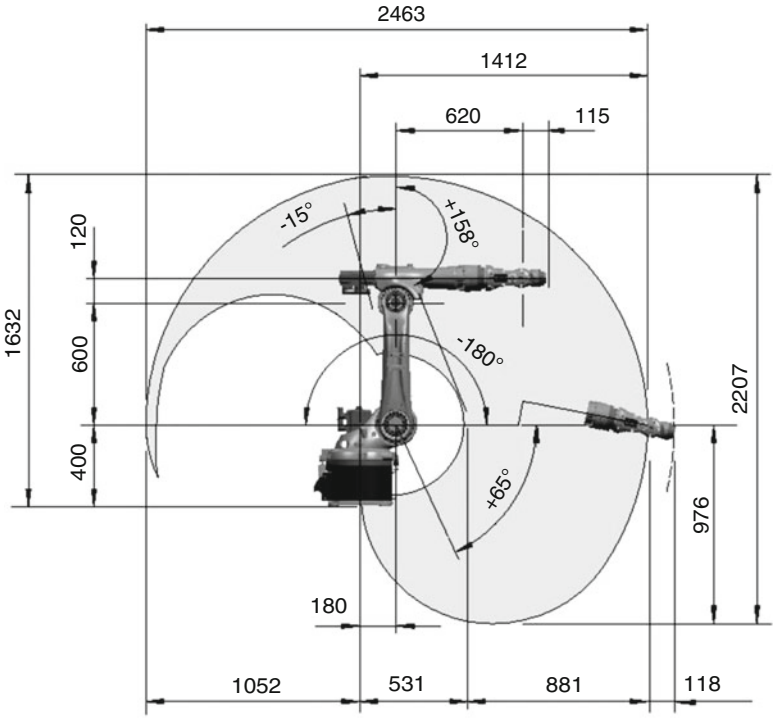


Fig. 6.5 KUKA KR5 robot (All the dimensions are in mm)

6.2.2 An Industrial Manipulator: KUKA KR5 Arc

Industrial robot is a serial manipulator and can be assumed as a special case of tree-type robot, where only one branch emanates from the fixed base. Hence, link index can be assumed to be the same as module index for the sake of simplicity. The kinematic architecture of the KUKA KR5 Arc industrial robot is used with user defined mass and inertia properties in order to study its dynamic behavior. The kinematic architecture of the KUKA robot is shown in Fig. 6.5. The DH parameters based on Fig. 6.5 are shown in Table 6.6. The assumed mass and inertia properties of the KUKA KR5 are also shown in Table 6.6.

First, inverse dynamics of the KUKA robot is performed for initial and final angles of all the six joints as 0° and 60°, respectively. Once again cycloidal trajectory of Eq. 6.14 was used as the input joint motion. The joint torques are shown in Fig. 6.6. Note that the torque required at joint 1 is zero in the beginning and at the end, as joint 1 is not affected by gravitational acceleration. Moreover, torque requirement at joint 2 is maximum, which is very obvious as this joint is required to overcome the effect of gravity on links 2, 3, 4, 5, and 6. Next forward dynamics of the KUKA robot was performed for the free-fall under gravity. For this both the initial joint angles and velocities are assumed to be zeros. Figure 6.7 show the joint

Table 6.6 DH parameter and inertia properties of the KUKA KR5

i	a_i (m)	α_i (deg)	b_i (m)	θ_i (deg)	m_i (kg)	$I_{i,xx}$	$I_{i,yy}$ (kg-m^2)	$I_{i,zz}$
1	0	0	0.400	θ_1	16.038	0.1769	0.2665	0.2622
2	0.180	90	0	θ_2	7.988	0.2716	0.2768	0.0218
3	0.600	0	0	θ_3	12.973	0.3889	0.3765	0.1041
4	0.120	90	0.620	θ_4	2.051	0.0047	0.0101	0.0121
5	0	−90	0	θ_5	0.811	0.0007	0.0017	0.0018
6	0	90	0	θ_6	0.008	0.000003	0.000001	0.000001

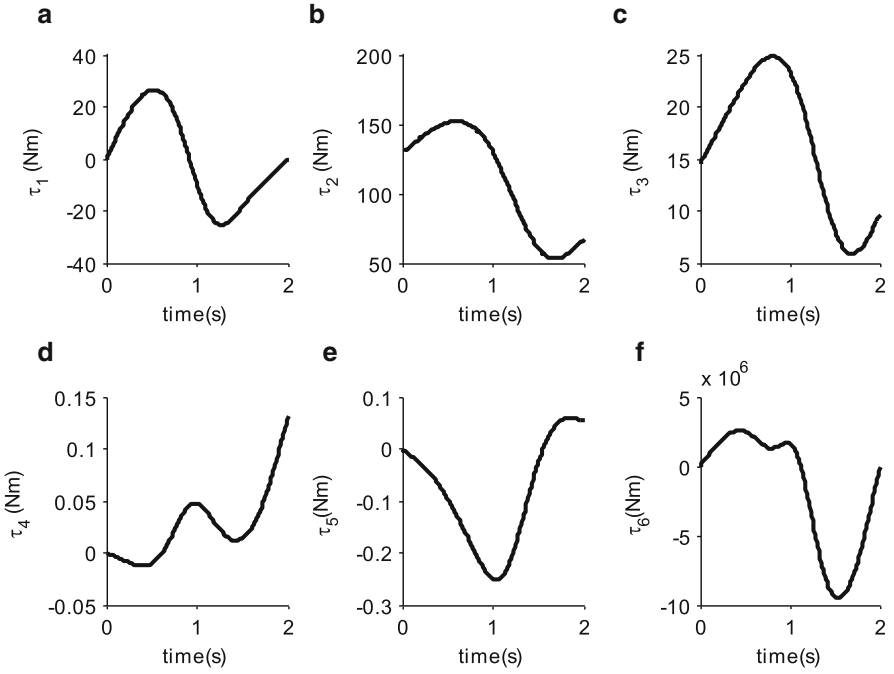


Fig. 6.6 Torque requirement at joints of the KUKA KR5. (**a**) τ_1. (**b**) τ_2. (**c**) τ_3. (**d**) τ_4. (**e**) τ_5. (**f**) τ_6

angles for the time period of 1 s. Total energy is also plotted in Fig. 6.8, which remains unchanged. This validates the simulation results.

6.2.3 A Biped

In order to study the effectiveness of the proposed algorithms for tree-type systems consisting of multiple-DOF joints, a spatial biped, as shown in Fig. 6.5a, is considered next. It has spherical joints at the hips, revolute joints at the knees,

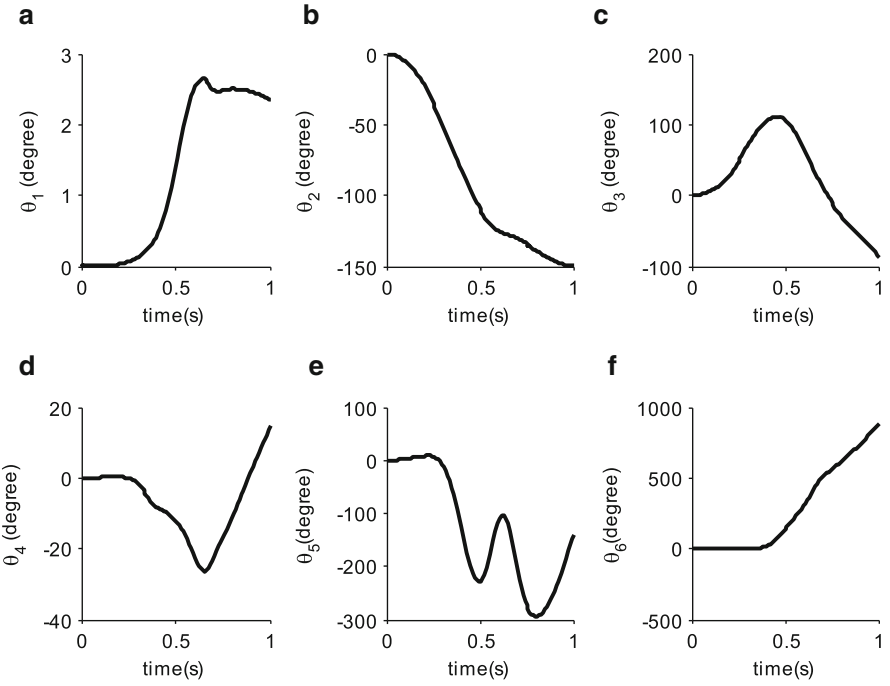


Fig. 6.7 Simulated joint angles of the KUKA KR5. (**a**) θ_1. (**b**) θ_2. (**c**) θ_3. (**d**) θ_4. (**e**) θ_5. (**f**) θ_6

and universal joints at the ankles. Biped motion was analyzed for a single support phase where one of the feet (the supporting foot during walking) is assumed to have no relative motion with respect to the ground. This is indicated in the figure with hatched lines. Hence, the biped dynamics can be solved as an open tree-type system (Shih et al. 1993). However, the following aspects distinguish the proposed modeling and simulation of the biped under study:

1. Treatment of multiple-DOF joints using the concept of Euler-Angle-Joints (EAJs) as proposed in Chap. 3.
2. Clever avoidance of the unnecessary operations with zeros associated with the dimensions and the masses of the intermediate links, as pointed out in Sect. 6.1.
3. The use of modularization of the biped. The biped is divided into three modules, M_0, M_1 and M_2, as shown in Fig. 6.9b. Such modularization helps one to treat module M_1 and M_2 in a similar manner as each module consists of three links, one spherical joint, one revolute joint, and one universal joint. Hence computationally one can use the same set of computer instructions. This result into a more elegant approach.

The model parameters for the biped are given in Table 6.7. In order to perform dynamic analysis, the desired trajectories of the swing foot and the Center-of-Mass

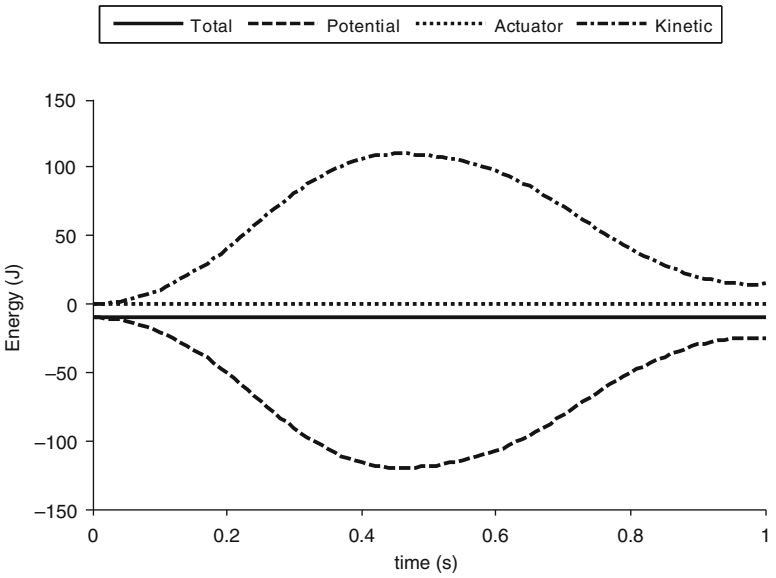


Fig. 6.8 Energy balance for the KUKA KR5

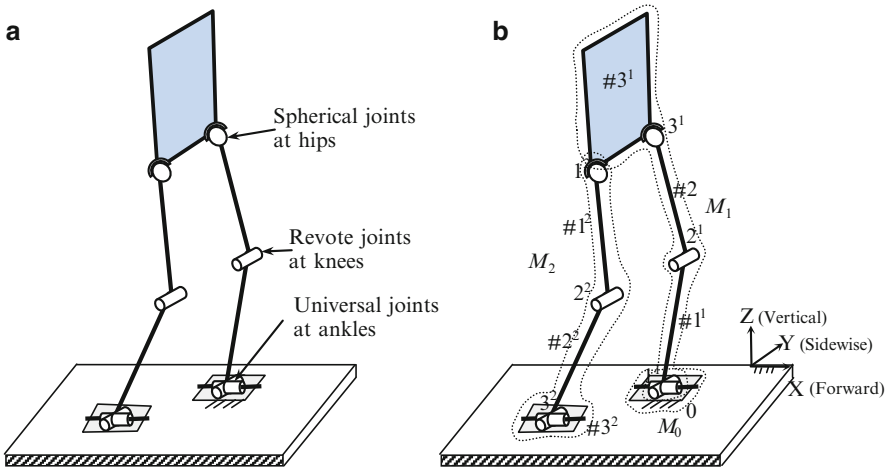


Fig. 6.9 A 7-link biped and its module architecture

(COM) of the trunk are synthesized first. The trunk's COM trajectory is designed based on the Inverted Pendulum Model (Kajita and Tani 1991; Park and Kim 1998), which assumes that the mass of the biped is concentrated at the trunk, the feet remain horizontal throughout the robot's motion and trunk moves horizontally with constant height from the ground. The swing foot trajectory is defined as cosine function. The trajectories of the trunk's COM and the swing foot are functions

Table 6.7 Model parameters for the biped

Link	Length (m)	Mass (Kg)
1^1,3^2	0.5	1
1^1, 2^2	0.6	1
1^2, 2^2	0.15	0.2
3^1	0.1×0.1(width)	5

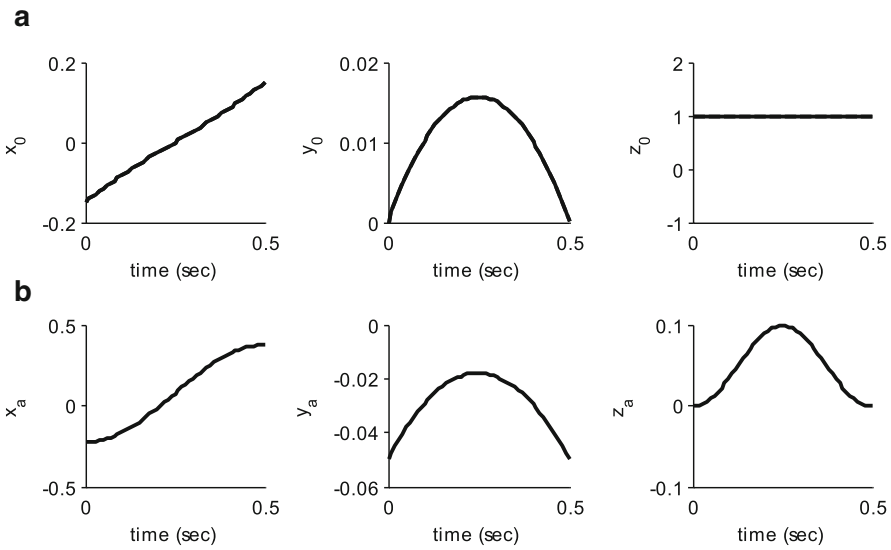


Fig. 6.10 Designed trajectories of the trunk's COM and ankle of the spatial biped. (**a**) Trunk's COM. (**b**) Ankle of the swing foot

of cycle time (T), co-ordinates of the COM $x_0(0)$, $y_0(0)$ and $z_0(0)$ at T = 0, stride length (l_s), and maximum foot height (h_f), as derived in Appendix B. For the spatial biped under study, T, $x_0(0)$, $y_0(0)$, $z_0(0)$, l_s, and h_f are assumed to be 0.5 s, −0.15, 0.08, 0.92, 0.3, and 0.1 m, respectively. The resulting trajectories of the COM of the trunk and the ankle of the swing foot are shown in Fig. 6.10. Based on the motion of the trunk and ankles, the desired joint level trajectories, i.e., $\boldsymbol{\theta}_{1^1} = \left[\theta_{1_1^1} \;\; \theta_{1_2^1}\right]^T$, θ_{2^1}, $\boldsymbol{\theta}_{3^1} = \left[\theta_{3_1^1} \;\; \theta_{3_2^1} \;\; \theta_{3_3^1}\right]^T$, $\boldsymbol{\theta}_{1^2} = \left[\theta_{1_1^2} \;\; \theta_{1_2^2} \;\; \theta_{1_3^2}\right]^T$, θ_{2^2}, and $\boldsymbol{\theta}_{3^2} = \left[\theta_{3_1^2} \;\; \theta_{3_2^2}\right]^T$, were calculated using the inverse kinematics relationships, which are also provided in Appendix B. The joint trajectories thus obtained are shown in Fig. 6.11.

Next, the inverse dynamics module of ReDySim was used to perform the force analyses of the biped. The detailed steps (like the robotic gripper of Sect. 6.2.1, as given in Table 6.3) are avoided here for brevity. Figure 6.12 shows the joint torques for a single support phase of the biped. It is worth noting that a complete walking

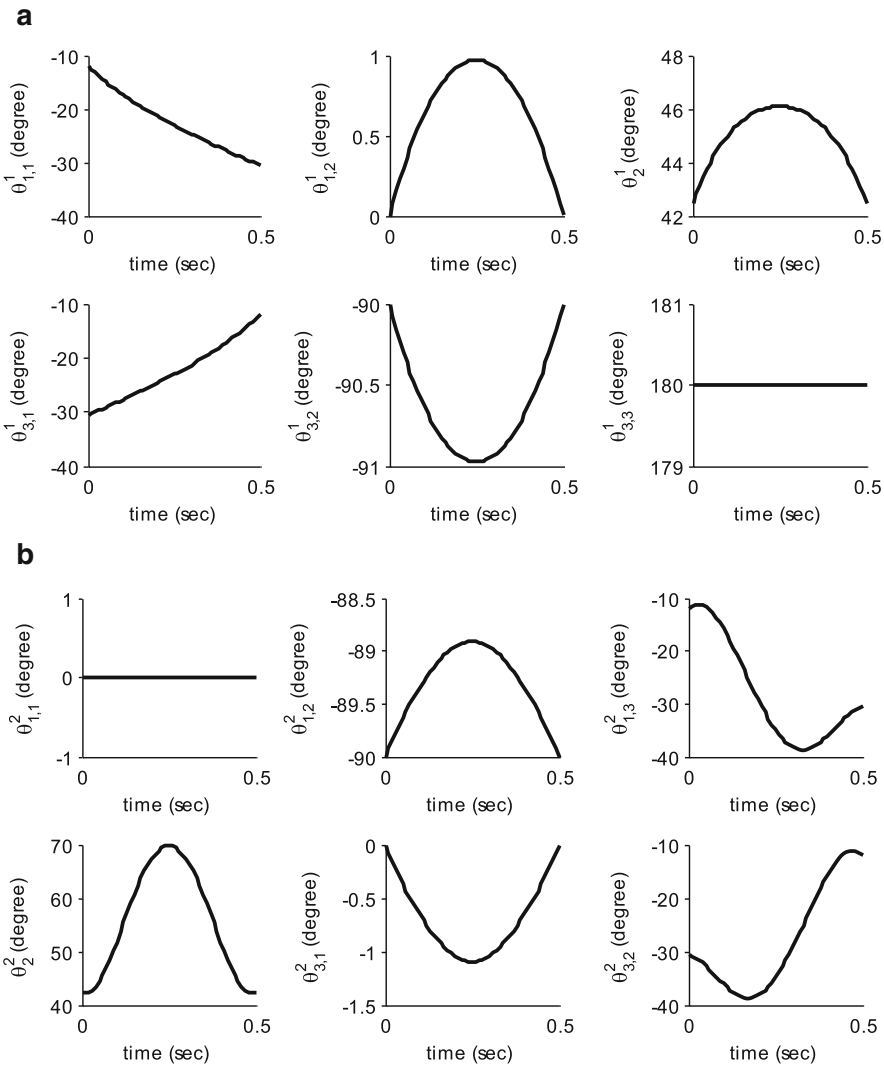


Fig. 6.11 Joint trajectories of the spatial biped obtained from trunk and ankle trajectories. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

sequence of a biped consists of single support phase as well as a momentary double support phase for change in legs forming a closed-loop system. The changeover of the legs demands the changing the equations of motion. Though the double support phase may be analyzed by following the approach given in Chap. 8, yet, the same is not an efficient one. More elegant way of analysis is reported in Chap. 7.

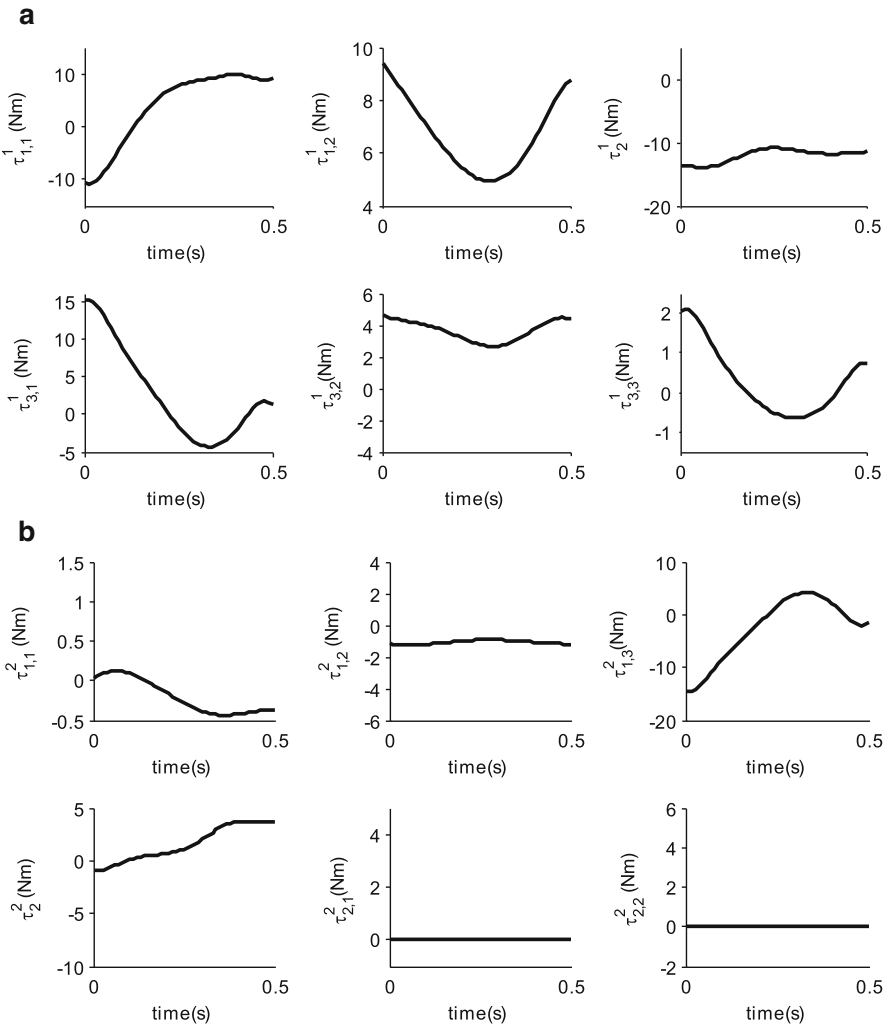


Fig. 6.12 Torque requirement at joints of the biped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

Forced simulation is performed next, where the motion of the biped is studied under the application of joint torques calculated above in Fig. 6.12. The joint motions were calculated using the forward dynamics module of ReDySim. The plots for the simulated joint angles are shown in Fig. 6.13 along with the desired one. It can be seen that the simulated joint angles match with the desired joint angles up to 0.1 s, i.e., until 0.1 s movement of the biped. After this, system behaves unexpectedly as evident from the divergent plots of the simulated angles in Fig. 6.13.

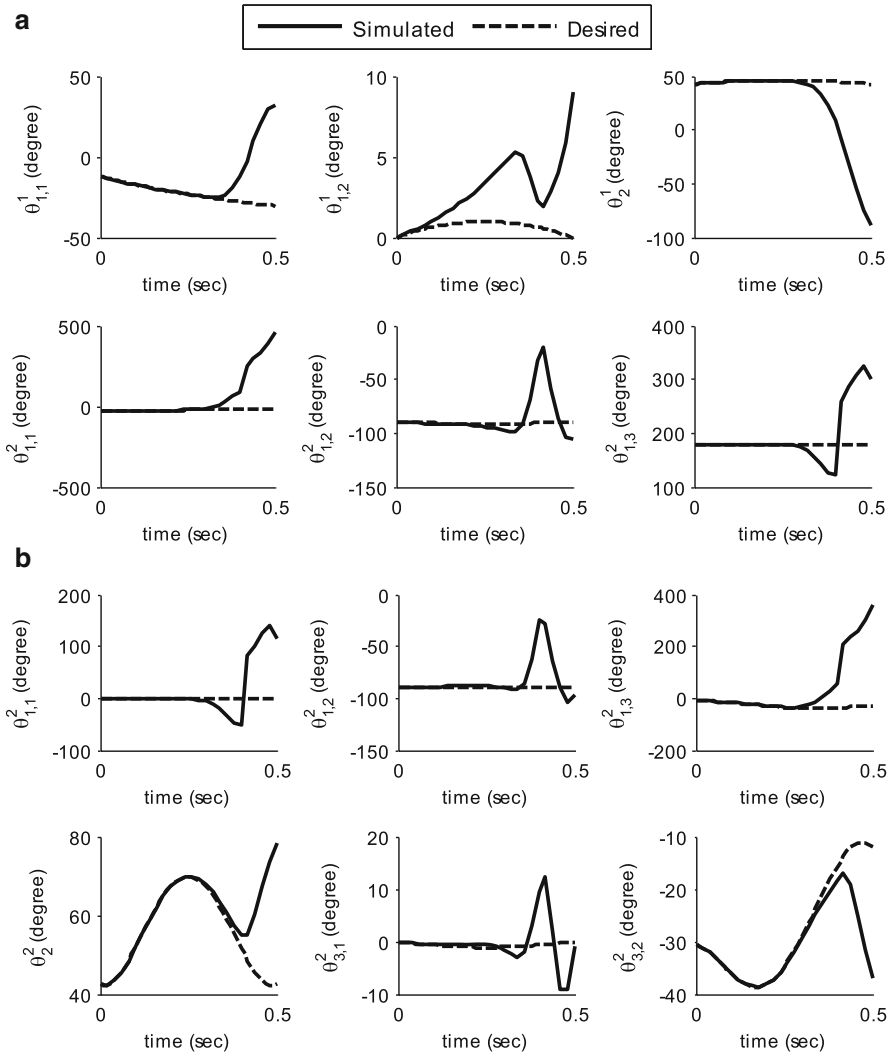


Fig. 6.13 Simulated joint angles of the biped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

The deviation in the simulated angles is mainly attributed to what is known in the literature as zero eigen-value effect (Saha and Schiehlen 2001). The real system may also not behave as expected due to disturbances caused by unmodeled parameters like friction, backlash, etc. and non-exact geometrical and inertia parameters. Hence, a control scheme must be considered, as this forms a part and parcel of achieving proper walking. The application of control scheme will be explained in Chap. 9.

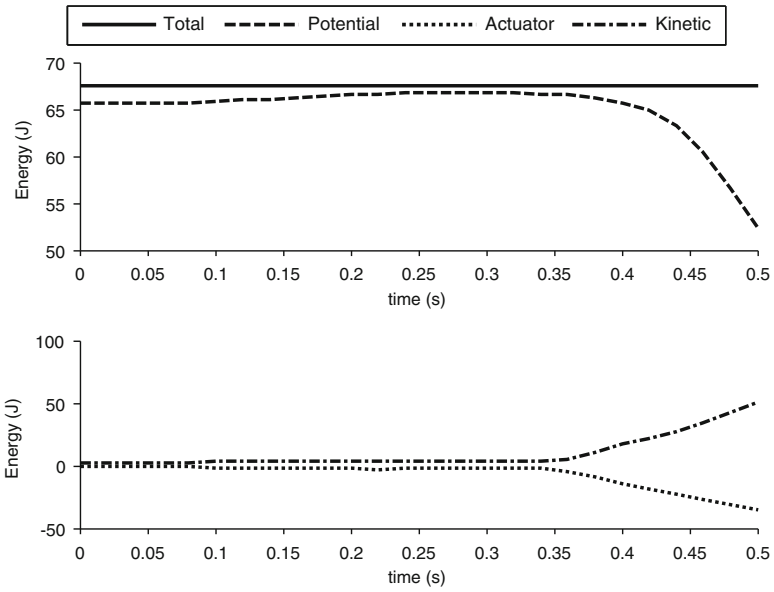


Fig. 6.14 Energy balance for the biped (Maximum error in total energy = 0.0058)

The computational complexity of the biped using the proposed algorithms with those in the literature is provided later in Sect. 6.3. In order to validate the simulation results the law of energy conservation was used (as shown in Appendix C) where the total energy of the biped while walking remains constant. The energy plots for the biped are shown in Fig. 6.14, where the total energy remains constant throughout the simulation period with maximum error in total energy of the order 10^{-3}. This validates the simulation results.

6.3 Computational Efficiency

In order to find the computational efficiency of the recursive inverse and forward dynamics algorithms given in Tables 6.1 and 6.2, complexities of the different steps are counted in terms of arithmetic operations, namely, Multiplications/Divisions (M) and Additions/Subtractions (A), as shown in Tables 6.8 and 6.9, respectively.

Note that, in Table 6.9, wherever required, a 6-dimensional vector, say, $\boldsymbol{\eta}_j$, was represented as

$$\boldsymbol{\eta}_j = \begin{bmatrix} \boldsymbol{\eta}_{j,t} \\ \boldsymbol{\eta}_{j,b} \end{bmatrix} \tag{6.15}$$

where $\boldsymbol{\eta}_{j,t}$ and $\boldsymbol{\eta}_{j,b}$ represent top and bottom three elements, respectively, of $\boldsymbol{\eta}_j$.

Table 6.8 Computational count for the recursive inverse dynamics of a fixed-base system

Forward recursion: For $k = 1{:}\eta$, $r = 1{:}\varepsilon_k$ (Given $\mathbf{I}_k$, $\mathbf{d}_k$, m_k, $m_k\mathbf{d}_k$, $\mathbf{a}_{\beta(k),k}$)			
For $r = 1$	Counts	For $r = 2{:}\varepsilon_k$	Counts
1 $\mathbf{t}_j = \mathbf{A}_{k,\beta}\mathbf{t}_{\beta_j} + \mathbf{p}_j\dot{\theta}_j$		$\mathbf{t}_j = \mathbf{t}_{j-1} + \mathbf{p}_j\dot{\theta}_j$	
$\boldsymbol{\omega}_j = \mathbf{Q}_j^T\boldsymbol{\omega}_{\beta(j)} + \mathbf{e}_j\dot{\theta}_j$	8M5A	$\boldsymbol{\omega}_j = \mathbf{Q}_j^T\boldsymbol{\omega}_{j-1} + \mathbf{e}_j\dot{\theta}_j$	4M3A
$\dot{\mathbf{o}}_j = \mathbf{Q}_j^T\left(\dot{\mathbf{o}}_{\beta(j)} + \boldsymbol{\omega}_{\beta(j)} \times \mathbf{a}_{\beta(k),k}\right)$[a]	–	$\dot{\mathbf{o}}_j = \mathbf{Q}_j^T\dot{\mathbf{o}}_{j-1}$ ++	–
2 $\dot{\mathbf{t}}_j = \mathbf{A}_{k,\beta}\dot{\mathbf{t}}_{\beta_j} + \dot{\mathbf{A}}_{k,\beta}\mathbf{t}_{\beta_j}$			
$+\boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$		$\dot{\mathbf{t}}_j = \dot{\mathbf{t}}_{j-1} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	
$\dot{\boldsymbol{\omega}}_j = \mathbf{Q}_i^T\dot{\boldsymbol{\omega}}_{\beta(j)} + \mathbf{e}_j\ddot{\theta}_j + \left(\boldsymbol{\omega}_j \times \mathbf{e}_j\right)\dot{\theta}_j$	10M7A	$\dot{\boldsymbol{\omega}}_j = \mathbf{Q}_i^T\dot{\boldsymbol{\omega}}_{j-1} + \mathbf{e}_j\ddot{\theta}_j + \left(\boldsymbol{\omega}_j \times \mathbf{e}_j\right)\dot{\theta}_j$	6M5A
$\ddot{\mathbf{o}}_j = \mathbf{Q}_i^T\left(\ddot{\mathbf{o}}_{\beta(j)} + \boldsymbol{\varpi}_{\beta(j)}\mathbf{a}_{\beta(k),k}\right)$	17M13A	$\ddot{\mathbf{o}}_j = \mathbf{Q}_j^T\ddot{\mathbf{o}}_{j-1}$	4M2A
For $r = \varepsilon_k$		For $r = 1{:}(\varepsilon_k\text{-}1)$	
3 $\boldsymbol{\varpi}_j = \tilde{\dot{\boldsymbol{\omega}}}_j + \tilde{\boldsymbol{\omega}}_j\tilde{\boldsymbol{\omega}}_j$	6M9A	$\boldsymbol{\varpi}_i = \mathbf{0}$	–
4 $\mathbf{w}_k^* = \mathbf{M}_k\dot{\mathbf{t}}_j + \boldsymbol{\Omega}_j\mathbf{M}_k\mathbf{E}_k\mathbf{t}_j\,;\tilde{\mathbf{w}}_j = \mathbf{w}_k^*$		$\tilde{\mathbf{w}}_j = \mathbf{0}$	
$\mathbf{n}' = \mathbf{I}_k\dot{\boldsymbol{\omega}}_j + \left(\boldsymbol{\omega}_j \times \mathbf{I}_k\boldsymbol{\omega}_j\right)$	15M15A	$\mathbf{n}' = \mathbf{0}$	–
$\tilde{\mathbf{n}}_j = \left[\mathbf{n}' + \left(m_k\mathbf{d}_k \times \ddot{\mathbf{o}}_j\right)\right]$	6M6A	$\tilde{\mathbf{n}}_j = \mathbf{0}$	–
$\mathbf{f}_j = \left[m_k\ddot{\mathbf{o}}_j + \boldsymbol{\varpi}_j m_k\mathbf{d}_k\right]$	12M9A	$\tilde{\mathbf{f}}_j = \mathbf{0}$	–
Backward recursion: For $k = \eta{:}1$, $r = \varepsilon_k{:}1$			
For $r = 1$		For $r = \varepsilon_k{:}2$	
1 $\tau_j = \mathbf{p}_j^T\tilde{\mathbf{w}}_j$		$\tau_j = \mathbf{p}_j^T\tilde{\mathbf{w}}_j$	
$\tau_j = \mathbf{e}_j^T\tilde{\mathbf{n}}_j$	–	$\tau_j = \mathbf{e}_j^T\tilde{\mathbf{n}}_j$	–
2 $\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\tilde{\mathbf{w}}_j$		$\tilde{\mathbf{w}}_{j-1} = \tilde{\mathbf{w}}_j$	
$\mathbf{A}_{k,\beta}^T\tilde{\mathbf{w}}_j$ (Eq. A.16)	20M12A	$\tilde{\mathbf{n}}_{j-1} = \mathbf{Q}_j\tilde{\mathbf{n}}_j$	4M2A
$\tilde{\mathbf{w}}_{\beta_j} + \mathbf{A}_{k,\beta}\mathbf{T}\tilde{\mathbf{w}}_j$	0M6A	$\tilde{\mathbf{f}}_{j-1} = \mathbf{Q}_j\tilde{\mathbf{f}}_j$	4M2A
Total	**94M82A**	+	**22M14A** **(ε_k-1)**

Note: (1) If parent of a link is fixed then only 13M7A counts are required instead of 94M82A.
(2) [a]not required in the algorithm and hence no computations.
(3) See Appendix A for detailed computational complexity.

The computational counts for the recursive inverse and forward dynamics of the k^{th} link connected by a multiple-DOF joint is summarized in Table 6.10.

It may be concluded from Table 6.10 that if a system consists of total number of n_1, n_2 and n_3 joint variables associated with 1-, 2- and 3-DOF joints, then the computational count of the recursive inverse and forward dynamics algorithms are obtained as

For inverse dynamics: $(94n_1 + 58n_2 + 46n_3 - 81)M(82n_1 + 48n_2 + 36\frac{2}{3}n_3 - 75)A$

For forward dynamics: $(135n_1 + 99n_2 + 87n_3 - 116)M(131n_1 + 92n_2 + 79\frac{2}{3}n_3 - 123)A$

The resulting computational complexities of the recursive inverse and forward dynamics algorithms are compared in Tables 6.11 and 6.12, respectively. It is interesting to note that the complexities are proportional to n $(=n_1 + n_2 + n_3)$ the total numbers of joint variables associated with 1-DOF, 2-DOF, and 3-DOF joints, respectively. It is pointed out here that the computational complexities of

Table 6.9 Computational count for the recursive forward dynamics of a fixed-base system

Backward recursion: For $k=\eta$:1, $r=\varepsilon_k$:1				
	For $r=1$	Counts	For $r=\varepsilon_k$:2	Counts
1	$\hat{\boldsymbol{\psi}}_j = \hat{\mathbf{M}}_j \mathbf{p}_j$		$\hat{\boldsymbol{\psi}}_j = \hat{\mathbf{M}}_j \mathbf{p}_j$	
	$\hat{\boldsymbol{\psi}}_{j,t} = \hat{\mathbf{I}}_j \mathbf{e}_j$	−	$\hat{\boldsymbol{\psi}}_{j,t} = \hat{\mathbf{I}}_j \mathbf{e}_j$	−
	$\hat{\boldsymbol{\psi}}_{j,b} = \hat{\mathbf{F}}_j \mathbf{e}_j$	−	$\hat{\boldsymbol{\psi}}_{j,b} = \hat{\mathbf{F}}_j \mathbf{e}_j$	−
2	$\hat{m}_j = \mathbf{p}_j^T \hat{\boldsymbol{\psi}}_j$		$\hat{m}_j = \mathbf{p}_j^T \hat{\boldsymbol{\psi}}_j$	
	$\hat{m}_j = \mathbf{e}_j^T \hat{\boldsymbol{\psi}}_{j,t}$	−	$\hat{m}_j = \mathbf{e}_j^T \hat{\boldsymbol{\psi}}_{j,t}$	−
3	$\boldsymbol{\psi}_j = \hat{\boldsymbol{\psi}}_j / \hat{m}_j$		$\boldsymbol{\psi}_j = \hat{\boldsymbol{\psi}}_j / \hat{m}_j$	
	$\boldsymbol{\psi}_{j,t} = \hat{\boldsymbol{\psi}}_{j,t} / \hat{m}_j$	2M0A[a]	$\boldsymbol{\psi}_{j,t} = \hat{\boldsymbol{\psi}}_{j,t} / \hat{m}_j$	1M0A
	$\boldsymbol{\psi}_{j,b} = \hat{\boldsymbol{\psi}}_{j,b} / \hat{m}_j$	3M0A	$\boldsymbol{\psi}_{j,b} = \hat{\boldsymbol{\psi}}_{j,b} / \hat{m}_j$	3M0A
4	$\hat{\varphi}_j = \varphi_j - \mathbf{p}_j^T \tilde{\boldsymbol{\eta}}_j$		$\hat{\varphi}_j = \varphi_j - \mathbf{p}_j^T \tilde{\boldsymbol{\eta}}_j$	
	$\hat{\varphi}_j = \varphi_j - \mathbf{e}_j^T \tilde{\boldsymbol{\eta}}_{j,t}$	0M1A	$\hat{\varphi}_j = \varphi_j - e_j^T \tilde{\boldsymbol{\eta}}_{j,t}$	0M1A
5	$\tilde{\varphi}_j = \hat{\varphi}_j / \hat{m}_j$	1M0A	$\tilde{\varphi}_j = \hat{\varphi}_j / \hat{m}_j$	1M0A
6	$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j \boldsymbol{\psi}_j^T$		$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j \boldsymbol{\psi}_j^T$	
	$\hat{\mathbf{I}}_{j,j} = \hat{\mathbf{I}}_j - \hat{\boldsymbol{\psi}}_{j,t} \boldsymbol{\psi}_{j,t}^T$	3M3A	$\hat{\mathbf{I}}_{j,j} = \hat{\mathbf{I}}_j - \hat{\boldsymbol{\psi}}_{j,t} \boldsymbol{\psi}_{j,t}^T$	1M1A
	$\hat{\mathbf{G}}_{j,j} = \hat{\mathbf{G}}_j - \hat{\boldsymbol{\psi}}_{j,b} \boldsymbol{\psi}_{j,b}^T$	6M6A	$\hat{\mathbf{G}}_{j,j} = \hat{\mathbf{G}}_j - \hat{\boldsymbol{\psi}}_{j,b} \boldsymbol{\psi}_{j,b}^T$	6M6A
	$\hat{\mathbf{F}}_{j,j} = \hat{\mathbf{F}}_j - \hat{\boldsymbol{\psi}}_{j,b} \boldsymbol{\psi}_{j,t}^T$	6M6A	$\hat{\mathbf{F}}_{j,j} = \hat{\mathbf{F}}_j - \hat{\boldsymbol{\psi}}_{j,b} \boldsymbol{\psi}_{j,t}^T$	3M3A
7	$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_j$		$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_j$	
	$\boldsymbol{\eta}_{j,t} = \boldsymbol{\psi}_{j,t} \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_{j,t}$	2M3A	$\boldsymbol{\eta}_{j,t} = \boldsymbol{\psi}_{j,t} \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_{j,t}$	1M2A
	$\boldsymbol{\eta}_{j,b} = \boldsymbol{\psi}_{j,b} \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_{j,b}$	3M3A	$\boldsymbol{\eta}_{j,b} = \boldsymbol{\psi}_{j,b} \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_{j,b}$	3M3A
8	$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{\beta_j} + \mathbf{A}_{k,\beta}^T \hat{\mathbf{M}}_{j,j} \mathbf{A}_{k,\beta}$		$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{j,j}$	
	$\mathbf{A}_{k,\beta}^T \hat{\mathbf{M}}_{j,j} \mathbf{A}_{k,\beta}$	64M63A	$\hat{\mathbf{M}}_{j,j}$	24M23A
	$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{\beta_j} + \mathbf{A}_{k,\beta}^T \hat{\mathbf{M}}_{j,j} \mathbf{A}_{k,\beta}$	15A		
9	$\tilde{\boldsymbol{\eta}}_{\beta_j} = \mathbf{A}_{k,\beta}^T \boldsymbol{\eta}_j$	20M12A	$\tilde{\boldsymbol{\eta}}_{\beta_j} = \boldsymbol{\eta}_j$	8M4A
Forward recursion: For $k=1$:η, $r=1$:ε_k				
	For $r=1$		For $r=2$:ε_k	
1	$\tilde{\boldsymbol{\mu}}_j = \mathbf{A}_{k,\beta} \boldsymbol{\mu}_{\beta_j}$	20M12A	$\tilde{\boldsymbol{\mu}}_j = \boldsymbol{\mu}_{\beta_j}$	8M4A
2	$\ddot{\theta}_j = \tilde{\varphi}_j - \boldsymbol{\psi}_j^T \tilde{\boldsymbol{\mu}}_j$		$\ddot{\theta}_j = \tilde{\varphi}_j - \boldsymbol{\psi}_j^T \tilde{\boldsymbol{\mu}}_j$	
	$\ddot{\theta}_j = \tilde{\varphi}_j - (\boldsymbol{\psi}_{j,t}^T \tilde{\boldsymbol{\mu}}_{j,t} + \boldsymbol{\psi}_{j,b}^T \tilde{\boldsymbol{\mu}}_{j,b})$	5M6A	$\ddot{\theta}_j = \tilde{\varphi}_j - (\boldsymbol{\psi}_{j,t}^T \tilde{\boldsymbol{\mu}}_{j,t} + \boldsymbol{\psi}_{j,b}^T \tilde{\boldsymbol{\mu}}_{j,b})$	4M5A
3	$\boldsymbol{\mu}_j = \mathbf{p}_j \ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_j$		$\boldsymbol{\mu}_j = \mathbf{p}_j \ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_j$	
	$\boldsymbol{\mu}_{j,t} = \mathbf{e}_j \ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_{j,t}$	0M1A	$\boldsymbol{\mu}_{j,t} = \mathbf{e}_j \ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_{j,t}$	0M1A
	$\boldsymbol{\mu}_{j,b} = \tilde{\boldsymbol{\mu}}_{j,b}$	−	$\boldsymbol{\mu}_{j,b} = \tilde{\boldsymbol{\mu}}_{j,b}$	−
	Total	**135M131A**	+	**63M53A (ε_k-1)**

Note: (1) For the terminal link (link with no child) only 19M8A counts are required instead of 135 M131.
(2) [a]The last element of $\boldsymbol{\psi}_{j,t}$ is always identity, i.e., $\boldsymbol{\psi}_{j,t} = [\times \times 1]^T$
(3) See Appendix A for detailed computational complexity.

the algorithms reported in the literature depend on total number of joint variables n, irrespective of the type of joints present in the system. Hence, number of links to be treated in the algorithm is equal to the number of joint variables, whereas in the algorithm presented here it is equal to number of joints. As a result, algorithms perform much faster when a system has multiple-DOF joints.

Table 6.10 Summary of computational counts for the k^{th} link

	Inverse dynamics 94M82A + 22M14A(ε_k-1)			Forward dynamics 135M131A + 63M53A(ε_k-1)		
	Joint type			Joint type		
	1-DOF $\varepsilon_k = 1$	2-DOF $\varepsilon_k = 2$	3-DOF $\varepsilon_k = 3$	1-DOF $\varepsilon_k = 1$	2-DOF $\varepsilon_k = 2$	3-DOF $\varepsilon_k = 3$
Count per joint (CPJ)	94M82A	116M96	138M110A	135M131A	198M184	261M237A
Count per DOF (CPJ/ ε_k)	94M82A	58M48	$46M\,36\frac{2}{3}A$	135M131A	99M92A	87M79A

Table 6.11 Computational complexity of recursive O(n) inverse dynamics algorithm for fixed-base systems

Algorithms	Computational complexity	Gripper $n_1 = 4, n_2 = 0, n_3 = 0, n = 4$	Industrial robot $n_1 = 6, n_2 = 0, n_3 = 0, n = 6$	Biped $n_1 = 2, n_2 = 4, n_3 = 6, n = 12$
Proposed	$\mathbf{(94n_1+58n_2+46n_3-81)M}$ $\mathbf{(82n_1+48n_2+36\frac{2}{3}n_3-75)A}$	**295M253A**	**615M497A**	**615M497A**
Balafoutis and Patel (1991)	(93n-69)M(81n-65)A	303M259A	1047M907A	1047M907A
Angeles et al. (1989)	(105n-109)M(90n-105)A	311M255A	1151M957A	1151M957A
Saha (1999b)	(120n-44)M(97n-55)A	436M333A	1396M1109A	1396M1109A
Featherstone (1987)	(130n-68)M(101n-56)A	452M348A	1492M1156A	1492M1156A

Note: (1) M: Multiplications/Divisions, A: Addition/Subtraction; (2) $n = n_1 + n_2 + n_3$.

This is evident from Figs. 6.15 and 6.16. Figure 6.15 shows comparisons of computational complexities for the inverse dynamics algorithms, when the system consists of only 1-DOF (Fig. 6.15a), 2-DOF (Fig. 6.15b), 3-DOF (Fig. 6.15c), and equal numbers of 1- 2- and 3-DOF joints (Fig. 6.15d). It may be seen that the inverse dynamics algorithm performs as fast as the fastest algorithm available in the literature when the system consists of only 1-DOF joints, as shown in Fig. 6.15a. However, when multiple-DOF joints are introduced in the system the algorithm out performs the algorithms available in the literature and significant improvement in the computational efficiency can be obtained as seen in Figs. 6.15b, c, d.

From Fig. 6.16, it is clear that the forward dynamics algorithm performs better than any algorithm available in the literature. More the number of multiple-DOF joints more the improvement in the computational efficiency, as shown in Fig. 6.16c, d. This is mainly due the implicit inversion of the GIM using $\mathbf{UDU}^T$ decomposition and simplification of the expressions associated with the multiple-DOF joints. Moreover, many of the computations required for the evaluation of the elements of $\mathbf{U}$ need not be repeated while performing the calculation of $\mathbf{U}^T$. As a result, a computer code developer can combine many steps to enhance efficiency. In

Table 6.12 Computational complexity of recursive O(n) forward dynamics algorithm for fixed-base systems

Algorithms	Computational complexity	Gripper $n_1=4, n_2=0, n_3=0, n=4$	Industrial robot $n_1=6, n_2=0, n_3=0, n=6$	Biped $n_1=2, n_2=4, n_3=6, n=12$
Proposed	$\mathbf{(135n_1+99n_2+87n_3-116)M}$ $\mathbf{(131n_1+92n_2+79\frac{2}{3}n_3-123)A}$	**444 M 401A**	**1072M981A**	**1072M981A**
Mohan and Saha (2007)	(173n-128)M(150n-133)A	564 M 467A	1948M1667A	1948M1667A
Saha (2003)	(191n-284)M(187n-325)A	480 M 423A	2008M1919A	2008M1919A
Featherstone (1983)	(199n-198)M(174n-173)A	598 M 523A	2190M1915A	2190M1915A
Lilly and Orin (1991)	$(\frac{1}{6}n^3+10\frac{3}{2}n^2+40\frac{1}{3}n-51)$M $(\frac{1}{6}n^3+7n^2+50\frac{5}{6}n-51)$A	305 M 275A	2377M1855A	2377M1855A

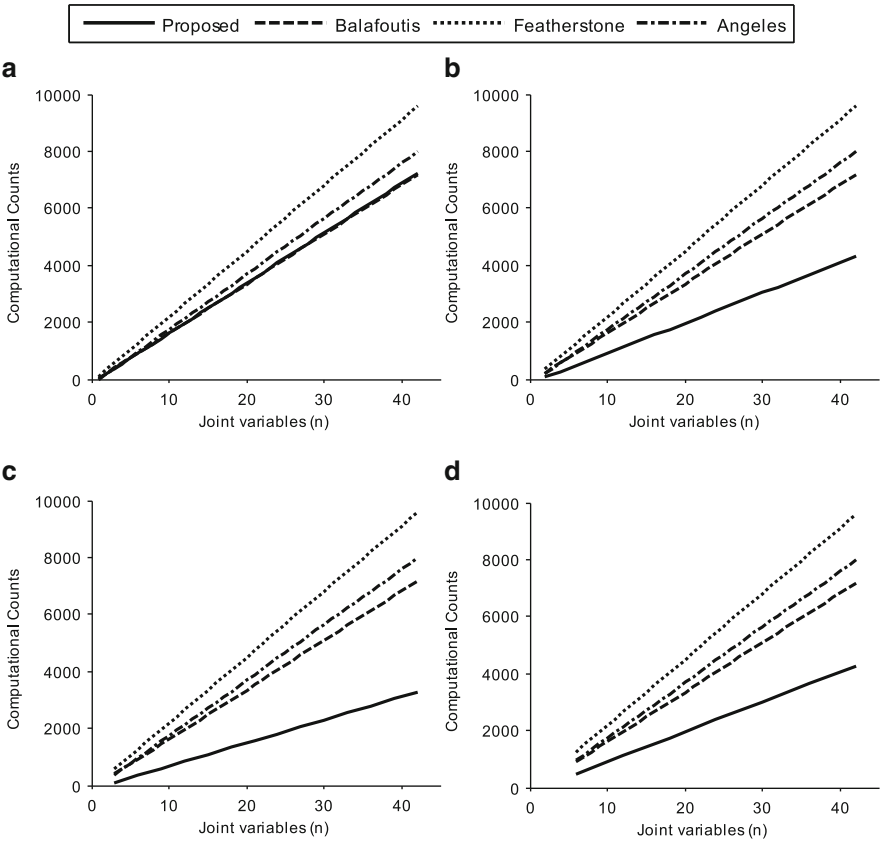


Fig. 6.15 Performance of the proposed inverse dynamics algorithm for a system with multiple-DOF joints. (**a**) All 1-DOF joints. (**b**) All 2-DOF joints. (**c**) All 3-DOF joints only. (**d**) Equal number of 1-, 2- and 3-DOF joints

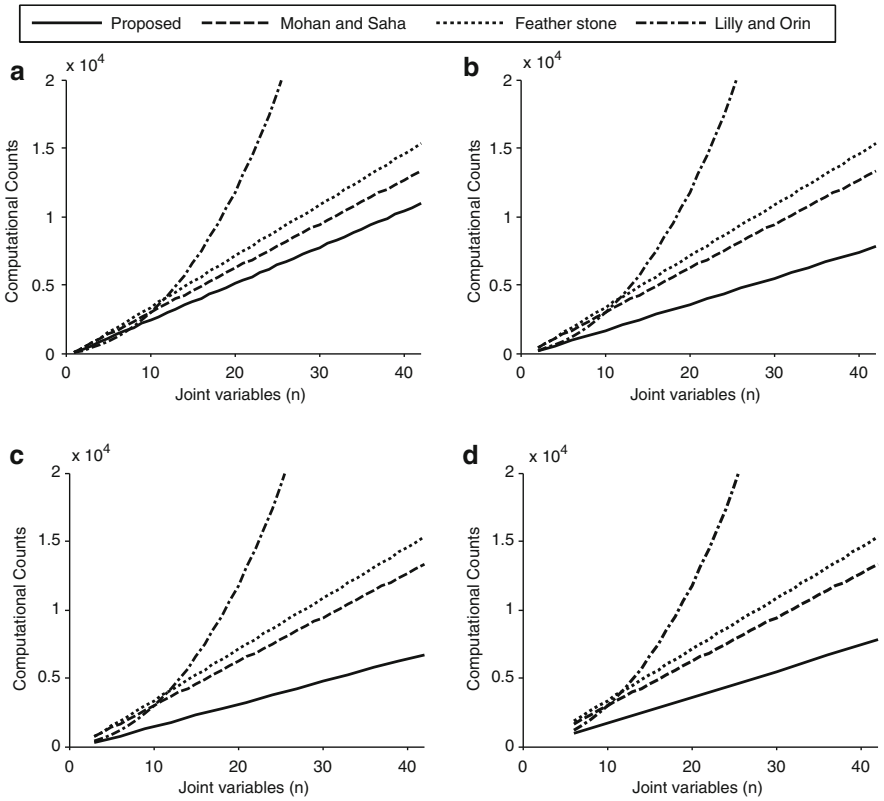


Fig. 6.16 Performance of the proposed forward dynamics algorithm for a system with multiple-DOF joints. (**a**) All 1-DOF joints. (**b**) All 2-DOF joints. (**c**) All 3-DOF joints only. (**d**) Equal number of 1-, 2- and 3-DOF joints

the tree-type robotic systems, such as biped, quadruped, etc., where the DOF of the system is more than 30, and the system consists of many multiple-DOF joints, the proposed algorithms significantly improves the computational efficiency.

6.4 Summary

In this chapter, efficient recursive inverse and forward dynamics algorithms are presented for open-chain tree-type robotic systems consisting of multiple-DOF joints. The algorithms are applicable to any serial or tree-type systems. The proposed algorithms performed better or as good as the fastest algorithm present in the literature when the systems consisted of only 1-DOF joints. However, with multiple-DOF joints in a system the algorithms performed much better. Several systems like robotic gripper, a serial manipulator, and spatial biped were analyzed using the recursive algorithms.

Chapter 7
Recursive Dynamics for Floating-Base Systems

Robotic systems studied in Chap. 6 have their bases fixed, however, in reality many robotic systems have their bases mobile or floating. In the case of a fixed-base robotic system, the base does not influence the dynamics, whereas it significantly influences the dynamics in the case of a floating-base robotic system. Space manipulators and legged robots are examples of floating-base robotic systems. Legged robots find applications in maintenance task of industrial plants, operations in dangerous and emergency environments, surveillance, maneuvering unknown terrains, human care, terrain adaptive vehicles and many more. In the case of legged robots they are either classified based on the number of legs, e.g., biped, quadruped, hexapod, etc., or the way it balances, e.g., statically or dynamically balanced. As reviewed in Chap. 2, legged robots (1) have variable topology, (2) move with high joint accelerations, (3) are dynamically not balanced if Center-of-Mass (COM) moves out of the polygon formed by the support feet, and (4) are under actuated. Hence, objective of achieving stable motion is difficult to decompose into actuator commands. Therefore, control of legged robots is intricate and dynamics plays vital role in achieving stable motion.

As mentioned above, legged robots have variable topologies. One approach for their dynamic analyses is to have separate dynamic models for different topologies or configurations as shown by Shih et al. (1993), Raibert et al. (1993), Ono et al. (2001) and others. For example, when one foot of a biped is on the ground it can be treated as a fixed-base tree-type system, as analyzed in Chap. 6, whereas it is as a closed-loop system when both the feet are on the ground. Such configuration-dependant dynamic analysis is inconvenient when the system has many configurations, which are frequently changing as the robot walk. An alternative approach is to treat a biped or a quadruped as a floating-base system as proposed by Freeman and Orin (1991), Ouezdou et al. (1998) and Vukobratovic et al. (2007). In this approach, a foot touching the ground is a contact rather than fixed as in the configuration-dependant approach of Chap. 6. Hence, it may be referred to as the configuration-independent approach. The latter approach is more generic and helps in modeling legged robots in a unified manner, which is presented in this chapter.

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_7,

Contact problem for configuration-dependent approach is essentially to solve the kinematic constrains together with the equations of motion. In this approach, a single point contact is assumed and the contact remains fixed during the support phase, which is however not valid in reality. On the contrary, contact problem for configuration-independent approach can be divided into three categories, viz., (a) analytical, (b) impulse-based, and (c) penalty-based. Analytical method of contact formulation (Baraff 1994; Stewart and Trinkle 2000; Lloyd 2005) involves equality and inequality constraints, and solution of constraint forces is an optimization problem. This approach is useful when the contact environment is composed of small number of objects of relatively simple shapes. Presence of friction makes analytical approach quite complex. In the impulse-based method (Mirtich and Canny 1995), the contact between bodies is modeled as collisions at points. Impulse-based contact has limitation, particularly, when the objects have continuous or stable contacts. In penalty-based approach (Bogert et al. 1989; Gerritsen et al. 1995; Marhefka and Orin 1996; Nigg and Herzog 1999), the constraint forces at the contact points are modeled by the deformation of linear or nonlinear visco-elastic model and its time derivatives. Penalty base approach, however, requires precise integration with refined time steps during collision. Recently, Yamane and Nakamura (2006) and Drumwright (2008) showed that the penalty based approach can be used to obtain fast and stable simulations.

It is worth mentioning that the number of Degrees-of-Freedom (DOF) of legged robots is generally high. For example, a humanoid robot has 40 DOF, as shown by Yamane and Nakamura (1999). Moreover, motion of the legged robots is very complex. Hence, the role of recursive dynamics, which leads to efficient algorithms for systems with many DOF, is inevitable for trajectory planning and control of such robots. Recursive algorithms are also known to provide numerically stable simulation results and thus make the prediction of a robot's motion more realistic.

The main objective of this chapter is to present dynamic analyses of several legged robots, which have spatial joints, e.g., a spherical or a universal joint, in addition to the revolute joints. In this chapter, a legged robot is considered as a floating-base system with intermittent contact of feet. The penalty-based contact modeling is used to take into account the intermittent foot-ground interaction. The concept of kinematic modules introduced in the Chaps. 4 and 5 is used to obtain efficient algorithms for recursive dynamics of the floating-base robotic systems like biped, quadruped and hexapod consisting of multiple-DOF joints.

7.1 Recursive Dynamics

Recursive dynamics of a floating-base robotic system, e.g., a legged robot, is important due to following reasons: (1) The DOF of the robot is as high as 40 or more.; (2) system consists of many multiple-DOF joints; (3) The inverse dynamics

problem also requires solution of differential equations for the base motion; and (4) The foot-ground interaction inherently consumes substantial time for computation. As a result, the use of recursive algorithms outperforms all other methods for solving the problem of dynamics. Two dynamics algorithms are presented next.

7.1.1 Inverse Dynamics

In contrast to the fixed-base robotic systems, inverse dynamics problem of a floating-base system involves solution of the accelerations for floating-base, followed by the computation of the joint torques. Therefore, in order to solve the inverse dynamics problem, the equations of motion given by Eq. (5.12) are rewritten after separating the floating-base accelerations, $\ddot{\mathbf{q}}_0$, from the joint accelerations, $\ddot{\mathbf{q}}_\theta$, i.e.,

$$\begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_{\theta 0}^T \\ \mathbf{I}_{\theta 0} & \mathbf{I}_\theta \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_0 \\ \ddot{\mathbf{q}}_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_\theta \end{bmatrix} - \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_\theta \end{bmatrix} \tag{7.1}$$

where $\mathbf{I}_0$ is the 6×6 matrix associated with the floating base, $\mathbf{I}_\theta$ is the $n \times n$ matrix associated with the rest of the tree-type legged robot, and $\mathbf{I}_{\theta 0} \equiv \left[\bar{\mathbf{I}}_{1,0}^T \cdots \bar{\mathbf{I}}_{s,0}^T\right]^T$ is the $n \times 6$ matrix. Moreover, $\ddot{\mathbf{q}}_\theta \equiv \left[\ddot{\bar{\mathbf{q}}}_1^T \cdots \ddot{\bar{\mathbf{q}}}_s^T\right]^T$ and $\boldsymbol{\tau}_\theta \equiv \left[\bar{\boldsymbol{\tau}}_1^T \cdots \bar{\boldsymbol{\tau}}_s^T\right]^T$ are the n-dimensional vectors and $\left[\mathbf{h}_0^T \ \mathbf{h}_\theta^T\right]^T \equiv \mathbf{h} \equiv \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}^F$ is the $(n + n_0)$-dimensional vector.

The inverse dynamics of the legged robot is then formulated by using Eq. (7.1) as

$$\begin{aligned} \ddot{\mathbf{q}}_0 &= -\mathbf{I}_0^{-1}\tilde{\mathbf{h}}_0, \text{ where } \tilde{\mathbf{h}}_0 \equiv \mathbf{I}_{\theta 0}^T \ddot{\mathbf{q}}_\theta + \mathbf{h}_0 \\ \boldsymbol{\tau}_\theta &= \mathbf{I}_{\theta 0}\ddot{\mathbf{q}}_0 + \tilde{\mathbf{h}}_\theta, \text{ where } \tilde{\mathbf{h}}_\theta \equiv \mathbf{I}_\theta \ddot{\mathbf{q}}_\theta + \mathbf{h}_\theta \end{aligned} \tag{7.2}$$

It is evident from Eq. (7.2) that the inverse dynamics of a floating-base legged robot involves the solution of accelerations for the floating-base, $\ddot{\mathbf{q}}_0$, followed by evaluation of the joint torques, $\boldsymbol{\tau}_\theta$, by using the algebraic equations. The above calculations are done using following three steps:

Step 1: Computation of $\mathbf{t}, \mathbf{t}'$, and $\mathbf{w}^* = (\mathbf{M}\mathbf{t}' + \boldsymbol{\Omega}\mathbf{M}\mathbf{E}\mathbf{t}) - \mathbf{w}^F$
Step 2: Computation of $\tilde{\mathbf{h}} = \left[\tilde{\mathbf{h}}_0^T \ \tilde{\mathbf{h}}_\theta^T\right]^T = \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^*$, $\mathbf{I}_0$ and $\mathbf{I}_{\theta 0}$
Step 3: Computation of $\ddot{\mathbf{q}}_0 = -\mathbf{I}_0^{-1}\tilde{\mathbf{h}}_0$ and $\boldsymbol{\tau}_\theta = \mathbf{I}_{\theta 0}\ddot{\mathbf{q}}_0 + \tilde{\mathbf{h}}_\theta$

where $\mathbf{t}'$ is the generalized twist-rate vector while $\ddot{\mathbf{q}}_0 = \mathbf{0}$. The algorithm to compute the above three steps require inter- and intra-modular recursions, that are explained next.

Step 1: Forward recursion: Computation of $\mathbf{t}', \mathbf{t}$, and $\mathbf{w}^*$

In this step, the generalized twist $\mathbf{t}$, twist-rate $\mathbf{t}'$, and wrench $\mathbf{w}^*$ are calculated. Inter- and intra-modular computations involved are shown below:

A. Inter-modular computations:

For module M_0

$$\begin{aligned}\bar{\mathbf{t}}_0 &= \overline{\mathbf{N}}_0\dot{\bar{\mathbf{q}}}_0, \bar{\mathbf{t}}'_0 = \dot{\overline{\mathbf{N}}}_0\dot{\bar{\mathbf{q}}}_0 + \boldsymbol{\rho}\\ \overline{\mathbf{w}}_0^* &= \overline{\mathbf{M}}_0\bar{\mathbf{t}}'_0 + \overline{\boldsymbol{\Omega}}_0\overline{\mathbf{M}}_0\overline{\mathbf{E}}_0\bar{\mathbf{t}}_0 - \overline{\mathbf{w}}_0^F\end{aligned} \tag{7.3}$$

For module M_i $(i = 1, \ldots, s)$

$$\begin{aligned}\bar{\mathbf{t}}_i &= \overline{\mathbf{A}}_{i,\beta}\bar{\mathbf{t}}_{\beta_i} + \overline{\mathbf{N}}_i\dot{\bar{\mathbf{q}}}_i\\ \bar{\mathbf{t}}'_i &= \dot{\overline{\mathbf{A}}}_{i,\beta}\bar{\mathbf{t}}_{\beta_i} + \overline{\mathbf{A}}_{i,\beta}\bar{\mathbf{t}}'_{\beta_i} + \dot{\overline{\mathbf{N}}}_i\dot{\bar{\mathbf{q}}}_i + \overline{\mathbf{N}}_i\ddot{\bar{\mathbf{q}}}_i\\ \overline{\mathbf{w}}_i^* &= \overline{\mathbf{M}}_i\bar{\mathbf{t}}'_i + \overline{\boldsymbol{\Omega}}_i\overline{\mathbf{M}}_i\overline{\mathbf{E}}_i\bar{\mathbf{t}}_i - \overline{\mathbf{w}}_i^F\end{aligned} \tag{7.4}$$

Intra-moduler steps for the above inter-modular steps are given next.

B. Intra-modular computations:

For floating base #0

$$\begin{aligned}\mathbf{t}_0 &= \mathbf{P}_0\dot{\mathbf{q}}_0, \mathbf{t}'_0 = \dot{\mathbf{P}}_0\dot{\mathbf{q}}_0 + \boldsymbol{\rho}\\ \mathbf{w}_0^* &= \mathbf{M}_0\mathbf{t}'_0 + \boldsymbol{\Omega}_0\mathbf{M}_0\mathbf{E}_0\mathbf{t}_0 - \mathbf{w}_0^F\end{aligned} \tag{7.5}$$

In Eq. (7.5), the vector $\boldsymbol{\rho} \equiv [\mathbf{0}^T\ \mathbf{g}^T]^T$, where '$\mathbf{g}$' is the 3-dimensional vector due to gravitational acceleration, has been added to the acceleration of the floating-base to take into account the effect of gravity on the links. Moreover, $\mathbf{t}'_0$ is the twist-rate of floating-base, when $\ddot{\mathbf{q}}_0 = \mathbf{0}$.

For link $\#k^i$ $(k^i = 1^i, \ldots, \eta^i)$

For $j = 1 : \varepsilon_k$

$$\begin{aligned}\mathbf{t}_{k_j} &= \mathbf{A}_{k,\beta}\mathbf{t}_{\beta_{(k_j)}} + \mathbf{p}_{k_j}\dot{\theta}_{k_j}, j = 1\\ &= \mathbf{t}_{\beta_{(k_j)}} + \mathbf{p}_{k_j}\dot{\theta}_{k_j}, \qquad j > 1\\ \mathbf{t}'_{k_j} &= \mathbf{A}_{k,\beta}\mathbf{t}'_{\beta_{(k_j)}} + \dot{\mathbf{A}}_{k,\beta}\mathbf{t}_{\beta_{(k_j)}} + \boldsymbol{\Omega}_{k_j}\mathbf{p}_{k_j}\dot{\theta}_{k_j} + \mathbf{p}_{k_j}\ddot{\theta}_{k_j}, j = 1\\ &= \mathbf{t}'_{\beta_{(k_j)}} + \boldsymbol{\Omega}_{k_j}\mathbf{p}_{k_j}\dot{\theta}_{k_j} + \mathbf{p}_{k_j}\ddot{\theta}_{k_j}, \qquad j > 1\\ \mathbf{w}_{k_j}^* &= 0, \qquad j < \varepsilon_k\\ &= \mathbf{M}_{k_j}\mathbf{t}'_{k_j} + \boldsymbol{\Omega}_{k_j}\mathbf{M}_{k_j}\mathbf{E}_{k_j}\mathbf{t}_{k_j} - \mathbf{w}_k^F, \qquad j = \varepsilon_k\end{aligned} \tag{7.6}$$

where $\beta \equiv (k-1)$, for $k > 1$.

Step 2. Backward Recursion: Computation of $\tilde{\mathbf{h}}, \mathbf{I}_0$ and $\mathbf{I}_{\theta 0}$

Having the results of Step 1 available, this step computes $\tilde{\mathbf{h}}, \mathbf{I}_0$ and $\mathbf{I}_{\theta 0}$ with the help of another set of inter- and intra-modular steps given below:

A. Inter-modular computations:

For module M_i $(i = s, \ldots, 1)$

$$\begin{aligned}
\tilde{\overline{\mathbf{h}}}_i &= \overline{\mathbf{N}}_i^T \tilde{\overline{\mathbf{w}}}_i, \text{ where } \tilde{\overline{\mathbf{w}}}_i \equiv \overline{\mathbf{w}}_i^* + \sum_{j \in \xi_i} \overline{\mathbf{A}}_{j,i}^T \tilde{\overline{\mathbf{w}}}_j \\
\tilde{\overline{\mathbf{K}}}_i &= \overline{\mathbf{A}}_{i,0}^T \overline{\mathbf{K}}_i, \text{ where } \overline{\mathbf{K}}_i = \tilde{\overline{\mathbf{M}}}_i \overline{\mathbf{N}}_i \\
&\qquad \text{and } \tilde{\overline{\mathbf{M}}}_i = \overline{\mathbf{M}}_i + \sum_{j \in \xi_i} \overline{\mathbf{A}}_{j,i}^T \tilde{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,i}
\end{aligned} \tag{7.7}$$

where ξ_i stands for array of the children of module M_i. Moreover, $\tilde{\overline{\mathbf{w}}}_i \equiv \overline{\mathbf{w}}_i^*$ and $\tilde{\overline{\mathbf{M}}}_i = \overline{\mathbf{M}}_i$ if $\xi_i = \{\}$, i.e., having no children. In Eq. (7.7), $\tilde{\overline{\mathbf{M}}}_i$ represents the mass matrix of the i^{th} composite-module which is defined in Eq. (5.19).

For module M_0

$$\begin{aligned}
\tilde{\overline{\mathbf{h}}}_0 &= \overline{\mathbf{N}}_0^T \tilde{\overline{\mathbf{w}}}_0, \text{ where } \tilde{\overline{\mathbf{w}}}_0 \equiv \overline{\mathbf{w}}_0^* + \sum_{j \in \xi_0} \overline{\mathbf{A}}_{j,0}^T \tilde{\overline{\mathbf{w}}}_j \\
\tilde{\mathbf{I}}_{0,0} &= \overline{\mathbf{K}}_0^T \overline{\mathbf{N}}_0, \text{ where } \overline{\mathbf{K}}_0 = \tilde{\overline{\mathbf{M}}}_0 \overline{\mathbf{N}}_0 \\
&\qquad \text{and } \tilde{\overline{\mathbf{M}}}_0 = \overline{\mathbf{M}}_0 + \sum_{j \in \xi_0} \overline{\mathbf{A}}_{j,0}^T \tilde{\overline{\mathbf{M}}}_j \overline{\mathbf{A}}_{j,0}
\end{aligned} \tag{7.8}$$

The above inter-modular steps require the following sets of intra-modular computations.

B. Intra-modular computations:

For link $\#k^i$ $(k = \eta^i, \ldots, 1)$

For $j = \varepsilon_k \ldots 1$

$$\begin{aligned}
\tilde{h}_{k_j} &= \mathbf{p}_{k_j}^T \tilde{\mathbf{w}}_{k_j}, \text{ where } \tilde{\mathbf{w}}_{k_j} = \mathbf{w}_{k_j}^* + \sum_{l \in \xi_{(k_j)}} \mathbf{A}_{l,k}^T \tilde{\mathbf{w}}_l, j = \varepsilon_k, k = \eta^i \\
&= \mathbf{w}_{k_j}^* + \mathbf{A}_{k+1,k}^T \tilde{\mathbf{w}}_{(k+1)_1}, j = \varepsilon_k, k < \eta^i \\
&= \tilde{\mathbf{w}}_{k_{j+1}}, j < \varepsilon_k \\
\tilde{\boldsymbol{\kappa}}_{k_j} &= \mathbf{A}_{k,0}^T \boldsymbol{\kappa}_{k_j}, \text{ where } \boldsymbol{\kappa}_{k_j} = \tilde{\mathbf{M}}_{k_j} \mathbf{p}_{k_j} \\
&\qquad \text{and } \tilde{\mathbf{M}}_{k_j} = \mathbf{M}_{k_j} + \sum_{l \in \xi_{(k_j)}} \mathbf{A}_{l,k}^T \tilde{\mathbf{M}}_l \mathbf{A}_{l,k}, j = \varepsilon_k, k = \eta^i \\
&\qquad = \mathbf{M}_{k_j} + \mathbf{A}_{k+1,k}^T \tilde{\mathbf{M}}_{(k+1)_1} \mathbf{A}_{k+1,k}, j = \varepsilon_k, k < \eta^i \\
&\qquad = \tilde{\mathbf{M}}_{k_{j+1}}, j < \varepsilon_k
\end{aligned} \tag{7.9}$$

For floating base #0

$$\mathbf{h}_0 = \mathbf{P}_0^T \tilde{\mathbf{w}}_0, \text{ where } \tilde{\mathbf{w}}_0 = \tilde{\mathbf{w}}_0^* + \sum_{l \in \boldsymbol{\xi}_0} \mathbf{A}_{l,0}^T \tilde{\mathbf{w}}_l$$

$$\mathbf{I}_{0,0} = \mathbf{K}_0^T \mathbf{P}_0, \text{ where } \mathbf{K}_0 = \tilde{\mathbf{M}}_0 \mathbf{P}_0$$

$$\text{and } \tilde{\mathbf{M}}_0 = \mathbf{M}_0 + \sum_{l \in \boldsymbol{\xi}_0} \mathbf{A}_{l,0}^T \tilde{\mathbf{M}}_l \mathbf{A}_{l,0} \tag{7.10}$$

Step 3. Forward recursion: Computation of $\ddot{\mathbf{q}}_0$ and $\boldsymbol{\tau}$

In this step, $\ddot{\mathbf{q}}_0$, and $\boldsymbol{\tau}$ are obtained by using the following inter- and intra-modular computations:

A. Inter-modular computations:

For module M_0

$$\ddot{\tilde{\mathbf{q}}}_0 = -\overline{\mathbf{I}}_{0,0}^{-1} \tilde{\tilde{\mathbf{h}}}_0$$

$$\tilde{\tilde{\mathbf{q}}}_0 = \overline{\mathbf{N}}_0 \ddot{\overline{\mathbf{q}}}_0 \tag{7.11}$$

For module M_i $(i = 1, \ldots, s)$

$$\overline{\boldsymbol{\tau}}_i = \tilde{\tilde{\mathbf{K}}}_i^T \tilde{\tilde{\mathbf{q}}}_0 + \tilde{\tilde{\mathbf{h}}}_i \tag{7.12}$$

It is worth noting that in Eq. (7.12), $\tilde{\tilde{\mathbf{K}}}_i^T \tilde{\tilde{\mathbf{q}}}_0 = (\overline{\mathbf{N}}_i^T \tilde{\overline{\mathbf{M}}}_i \overline{\mathbf{A}}_{i,0})(\overline{\mathbf{N}}_0 \ddot{\overline{\mathbf{q}}}_0) = \overline{\mathbf{I}}_{i,0} \ddot{\overline{\mathbf{q}}}_0$ where $\overline{\mathbf{I}}_{i,0} = \overline{\mathbf{N}}_i^T \tilde{\overline{\mathbf{M}}}_i \overline{\mathbf{A}}_{i,0} \overline{\mathbf{N}}_0$.

B. Intra-modular computations:

For floating base #0

$$\ddot{\tilde{\mathbf{q}}}_0 = -\mathbf{I}_{0,0}^{-1} \tilde{\mathbf{h}}_0$$

$$\tilde{\mathbf{q}}_0 = \mathbf{P}_0 \ddot{\tilde{\mathbf{q}}}_0 \tag{7.13}$$

For link #k^i (For $k = 1, \ldots, \eta^i$)

$$\text{For } j = 1 : \varepsilon_k$$

$$\tau_k = \tilde{\boldsymbol{\kappa}}_k^T \tilde{\mathbf{q}}_0 + \tilde{h}_k \tag{7.14}$$

Once again in Eq. (7.14), $\tilde{\boldsymbol{\kappa}}_k^T \tilde{\mathbf{q}}_0 = (\mathbf{p}_k^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,0})(\mathbf{P}_0 \ddot{\tilde{\mathbf{q}}}_0) = \mathbf{I}_{i,0} \ddot{\tilde{\mathbf{q}}}_0$ where $\mathbf{I}_{i,0} = \mathbf{p}_k^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,0} \mathbf{P}_0$. It may be noted that the inter-modular steps are nothing but the compact representation of the intra-modular steps. For example, the expression for $\boldsymbol{\tau}_k$ in Eq. (7.14), for $k = 1, \ldots, \eta^i$, is obtained after writing $\overline{\boldsymbol{\tau}}_i$ in Eq. (7.12) in terms of the block elements of $\tilde{\tilde{\mathbf{K}}}_i$ and $\tilde{\tilde{\mathbf{h}}}_i$. Hence, they are equivalent.

Table 7.1 Recursive O(n) inverse dynamics algorithm for floating-base robotic systems

Step 1: Compute: $\mathbf{t}_j$, $\mathbf{t}'_j$, and $\mathbf{w}^*_j$	Step 2: Compute: $\tilde{h}_j$ and $\tilde{\mathbf{M}}_j$	Step 3: Compute: $\ddot{\mathbf{q}}_0$ and $\boldsymbol{\tau}_j$
$i = 0, k = 0, j = 0$	$j = DOF$	$i = 0, k = 0, j = 0$
$\mathbf{t}_0 = \mathbf{P}_0\dot{\mathbf{q}}_0$	For $i = s : 1$ (Inter-modular)	$\ddot{\mathbf{q}}_0 = -\mathbf{I}_{0,0}^{-1}\tilde{\mathbf{h}}_0$
$\mathbf{t'}_0 = \dot{\mathbf{P}}_0\dot{\mathbf{q}}_0 + \boldsymbol{\rho}$	For $k = \eta^i : 1$(Intra-modular)	$\tilde{\mathbf{q}}_0 = \mathbf{P}_0\ddot{\mathbf{q}}_0$
$\mathbf{A}_{0,0} = \mathbf{1}$	For $r = \varepsilon_k : 2$ (Joint level)	For $i = 1 : s$ (Inter-modular)
$\mathbf{w}_0{}^* = \mathbf{M}_0\mathbf{t'}_0 + \boldsymbol{\Omega}_0\mathbf{M}_0\mathbf{E}_0\mathbf{t}_0 - \mathbf{w}_0^F$	call function_1	For $k = 1 : \eta^i$ (Inter-modular)
$\tilde{\mathbf{w}}_0 = \mathbf{w}_0^*, \tilde{\mathbf{M}}_0 = \mathbf{M}_0$	$\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_j$	For $r = 1 : \varepsilon_k$ (Joint level)
For $i = 1 : s$ (Inter-modular)	$\tilde{\mathbf{M}}_{\beta_j} = \tilde{\mathbf{M}}_j$	$j = j + 1$
For $k = 1 : \eta^i$ (Intra-modular)	$j = j - 1$	$\tau_j = \tilde{\boldsymbol{\kappa}}_j^T\tilde{\mathbf{q}}_0 + \tilde{h}_j$
$r = 1$ (Joint level)	end	end
$j = j + 1$	$r = 1$ (Joint level)	end
$\mathbf{t}_j = \mathbf{A}_{k,\beta}\mathbf{t}_{\beta_j} + \mathbf{p}_j\dot{\theta}_j$	call function_1	end
$\mathbf{t'}_j = \mathbf{A}_{k,\beta}\mathbf{t'}_{\beta_j} + \dot{\mathbf{A}}_{k,\beta}\mathbf{t}_{\beta_j} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	$\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\tilde{\mathbf{w}}_j$	
$\mathbf{A}_{k,0} = \mathbf{A}_{k,\beta}\mathbf{A}_{k,0}$	$\tilde{\mathbf{M}}_{\beta_j} = \tilde{\mathbf{M}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\tilde{\mathbf{M}}_j\mathbf{A}_{k,\beta}$	
$\tilde{\mathbf{w}}_j = \mathbf{0}, \tilde{\mathbf{M}}_j = \mathbf{O}$	$j = j - 1$	
For $r = 2 : \varepsilon_k$	end	
$j = j + 1$	end	
$\mathbf{t}_j = \mathbf{t}_{j-1} + \mathbf{p}_j\dot{\theta}_j$	$i = 0, k = 0, j = 0$	
$\mathbf{t'}_j = \mathbf{t'}_{j-1} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	$\tilde{\mathbf{h}}_0 = \mathbf{P}_0^T\tilde{\mathbf{w}}_0,$	
$\mathbf{A}_{k,0} = \mathbf{A}_{k,0}$	$\mathbf{K}_0 = \tilde{\mathbf{M}}_0\mathbf{P}_0$	
$\tilde{\mathbf{w}}_j = \mathbf{0}, \tilde{\mathbf{M}}_j = \mathbf{O}$	$\mathbf{I}_{0,0} = \mathbf{K}_0^T\mathbf{P}_0$	
end	$---$	
$\mathbf{w}_k^* = \mathbf{M}_k\mathbf{t'}_j + \boldsymbol{\Omega}_j\mathbf{M}_k\mathbf{E}_k\mathbf{t}_j - \mathbf{w}_j^F$	**function_1**	
$\tilde{\mathbf{w}}_j = \mathbf{w}_k^*, \tilde{\mathbf{M}}_j = \mathbf{M}_k$	$\tilde{h}_j = \mathbf{p}_j^T\tilde{\mathbf{w}}_j$	
end	$\boldsymbol{\kappa}_j = \tilde{\mathbf{M}}_j\mathbf{p}_j$	
end	$\tilde{\boldsymbol{\kappa}}_j = \mathbf{A}_{k,0}^T\boldsymbol{\kappa}_j$	

The algorithmic implementation of the above steps is shown in Table 7.1. The computational counts for the various steps in Table 7.1 and a detailed comparison of the inverse dynamics algorithm with those existing in the literature will be provided in Sect. 7.5.

7.1.2 Forward Dynamics

Main objective of the forward dynamics is to find independent joint accelerations for a given set of actuator torques, which are integrated twice to obtain the joint velocities and positions. In order to perform the forward dynamics, the equations of motion obtained in Eq. (5.12) are rewritten as follows:

$$\mathbf{I}\ddot{\mathbf{q}} = \boldsymbol{\tau} - \mathbf{h}, \text{ where } \mathbf{h} \equiv \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}^F \equiv \mathbf{N}_d^T \mathbf{N}_l^T \mathbf{w}^* \tag{7.15}$$

where $\mathbf{w}^* \equiv (\mathbf{M}\mathbf{t}' + \boldsymbol{\Omega}\mathbf{M}\mathbf{E}\mathbf{t}) - \mathbf{w}^F$. In contrast to the inverse dynamics algorithm, where $\mathbf{t}'$ corresponds to the twist-rate when $\ddot{\mathbf{q}}_0 = \mathbf{0}$, here, $\mathbf{t}'$ corresponds to the twist rate when $\ddot{\mathbf{q}}_0 = \mathbf{0}$. Similar to the forward dynamics of a fixed-base system proposed in Sect. 6.1.2, the accelerations of the floating-base system are solved recursively following the $\mathbf{U}\mathbf{D}\mathbf{U}^T$ decomposition of the GIM given by Eq. (7.15). Based on the block $\mathbf{U}\mathbf{D}\mathbf{U}^T$ decomposition of the GIM, the equations of motion, Eq. (7.15), are presented as

$$\mathbf{U}\mathbf{D}\mathbf{U}^T\ddot{\mathbf{q}} = \boldsymbol{\tau} - \mathbf{h} \tag{7.16}$$

where $\mathbf{U}$ and $\mathbf{D}$ are the $(n+n_0)\times(n+n_0)$ block upper-triangular and diagonal matrices, respectively. It may be noted that vector $\mathbf{h}$ can be obtained by using the inverse dynamics algorithm of Sect. 7.1.1. However, the use of inverse dynamics for the computation of $\mathbf{h}$ is not computationally efficient as it involves computation of the terms associated with GIM, i.e., $\tilde{\mathbf{M}}_k$ and $\mathbf{I}_{k,0}$. On the contrary, here, $\mathbf{h}$ is computed efficiently together with the computation of the joint accelerations. Next, the generalized acceleration, $\ddot{\mathbf{q}}$, is calculated by using three sets of linear algebraic equations, namely,

$$\begin{aligned} &\text{(i)} \quad \mathbf{U}\hat{\boldsymbol{\varphi}} = \boldsymbol{\tau} - \mathbf{h}, \text{ where } \hat{\boldsymbol{\varphi}} = \mathbf{D}\mathbf{U}^T\ddot{\mathbf{q}} \\ &\text{(ii)} \quad \mathbf{D}\tilde{\boldsymbol{\varphi}} = \hat{\boldsymbol{\varphi}}, \text{ where } \tilde{\boldsymbol{\varphi}} = \mathbf{U}^T\ddot{\mathbf{q}} \\ &\text{(iii)} \quad \mathbf{U}^T\ddot{\mathbf{q}} = \tilde{\boldsymbol{\varphi}} \end{aligned} \tag{7.17}$$

Equations (7.16) or the above steps actually require inter- and intra-modular recursions as given next.

Step 1. Forward recursion: Computation of $\mathbf{t}, \mathbf{t}'$, and $\mathbf{w}^*$

This step has similar inter- and intra-modular recursions as shown in step 1 of the inverse dynamics algorithm given in Sect. 7.1.1. However, here $\mathbf{t}'$ corresponds to the generalized twist-rate when $\ddot{\mathbf{q}} = \mathbf{0}$.

Step 2. Backward recursion: Computation of $\hat{\boldsymbol{\varphi}}$, and $\tilde{\boldsymbol{\varphi}}$

In this step, $\hat{\boldsymbol{\varphi}}$ and $\tilde{\boldsymbol{\varphi}}$ are obtained using backward substitutions based on inter- and intra-modular recursions, which are given below:

A. Inter-modular computations:
For module M_i $(i = \mathrm{s}, \ldots, 0)$

$$\begin{aligned}
&\text{(i) } \hat{\overline{\boldsymbol{\varphi}}}_i = \overline{\boldsymbol{\tau}}_i - \overline{\mathbf{N}}_i^T(\tilde{\overline{\boldsymbol{\eta}}}_i + \overline{\mathbf{w}}_i^*), \text{ where, } \tilde{\overline{\boldsymbol{\eta}}}_i = \sum_{j \in \xi_i} \overline{\mathbf{A}}_{j,i}^T \overline{\boldsymbol{\eta}}_j, \text{ and} \\
&\qquad \overline{\boldsymbol{\eta}}_j = \overline{\boldsymbol{\Psi}}_j \hat{\overline{\boldsymbol{\varphi}}}_j + (\tilde{\overline{\boldsymbol{\eta}}}_j + \overline{\mathbf{w}}_i^*) \qquad (7.18) \\
&\text{(ii) } \tilde{\overline{\boldsymbol{\varphi}}}_i = \hat{\overline{\mathbf{I}}}_i^{-1} \hat{\overline{\boldsymbol{\varphi}}}_i
\end{aligned}$$

In Eq. (7.18), $\tilde{\overline{\boldsymbol{\eta}}}_i$ and $\overline{\boldsymbol{\eta}}_i$ are the $6n^i$-dimensional vectors, and $\tilde{\overline{\boldsymbol{\eta}}}_i = \mathbf{0}$, if $\boldsymbol{\xi}_i = \{\}$. Also, $\overline{\mathbf{w}}_i^*$ is added to $\tilde{\overline{\boldsymbol{\eta}}}_i$ in order to take into account the effect of $\mathbf{h}$, appearing in Eq. (7.17). Moreover, $\hat{\overline{\mathbf{I}}}_i$ and $\overline{\boldsymbol{\Psi}}_i$ are already obtained in Eqs. (5.34) and (5.35), respectively. Intra-modular computations for the above step are carried out as follows:

B. Intra-modular computations:
For link $\#k^i$ $(k = \eta^i, \ldots, 1)$

For $j = \varepsilon_k ... 1$

$$\begin{aligned}
\text{(i) } \hat{\varphi}_{k_j} &= \tau_{k_j} - \mathbf{p}_{k_j}^T(\tilde{\boldsymbol{\eta}}_{k_j} + \mathbf{w}_{k_j}^*), \text{where } \tilde{\boldsymbol{\eta}}_{k_j} = \sum_{l \in \xi_k} \mathbf{A}_{l,k}^T \boldsymbol{\eta}_l, j = \varepsilon_k, k = \eta^i \\
&= \mathbf{A}_{k+1,k}^T \boldsymbol{\eta}_{(k+1)_1}, j = \varepsilon_k, k < \eta^i \\
&\text{and } \boldsymbol{\eta}_l = \boldsymbol{\psi}_l \hat{\varphi}_l + (\tilde{\boldsymbol{\eta}}_l + \mathbf{w}_l^*) \\
&= \tau_{k_j} - \mathbf{p}_{k_j}^T \tilde{\boldsymbol{\eta}}_{k_j}, \text{ where } \tilde{\boldsymbol{\eta}}_{k_j} = \boldsymbol{\eta}_{k_{j+1}} j < \varepsilon_k \\
&\text{and } \boldsymbol{\eta}_{k_{j+1}} = \boldsymbol{\psi}_{k_{j+1}} \hat{\varphi}_{k_{j+1}} + \tilde{\boldsymbol{\eta}}_{k_{j+1}} \qquad (7.19) \\
\text{(ii) } \tilde{\varphi}_{k_j} &= \hat{\varphi}_{k_j} / \hat{m}_{k_j}
\end{aligned}$$

For floating base #0

$$\begin{aligned}
&\text{(i) } \hat{\boldsymbol{\varphi}}_0 = \boldsymbol{\tau}_0 - \mathbf{P}_0^T(\tilde{\boldsymbol{\eta}}_0 + \mathbf{w}_0^*), \\
&\qquad \text{where } \tilde{\boldsymbol{\eta}}_0 = \sum_{l \in \xi_0} \mathbf{A}_{l,0}^T \boldsymbol{\eta}_l, \text{and } \boldsymbol{\eta}_l = \boldsymbol{\Psi}_l \hat{\boldsymbol{\varphi}}_l + (\tilde{\boldsymbol{\eta}}_l + \mathbf{w}_l^*) \qquad (7.20) \\
&\text{(ii) } \tilde{\boldsymbol{\varphi}}_0 = \hat{\mathbf{I}}_0^{-1} \hat{\boldsymbol{\varphi}}_0
\end{aligned}$$

Step 3. Forward recursion: Computation of $\ddot{\mathbf{q}}$

Finally, the generalized independent accelerations ($\ddot{\mathbf{q}}$) are computed using inter- and intra-modular steps, which are given below:

A. Inter-modular computations:

For module M_0

$$\ddot{\overline{\mathbf{q}}}_0 = \tilde{\overline{\boldsymbol{\varphi}}}_0 \tag{7.21}$$

For module M_i ($i = 1, \ldots, s$)

$$\ddot{\overline{\mathbf{q}}}_i = \tilde{\overline{\boldsymbol{\varphi}}}_i - \overline{\boldsymbol{\Psi}}_i^T \tilde{\overline{\boldsymbol{\mu}}}_i, \text{ where } \tilde{\overline{\boldsymbol{\mu}}}_i = \overline{\mathbf{A}}_{i,\beta} \overline{\boldsymbol{\mu}}_{\beta_i},$$
$$\text{and } \overline{\boldsymbol{\mu}}_{\beta_i} = \overline{\mathbf{N}}_{\beta_i} \ddot{\overline{\mathbf{q}}}_{\beta_i} + \tilde{\overline{\boldsymbol{\mu}}}_{\beta_i} \tag{7.22}$$

in which $\overline{\boldsymbol{\mu}}_i$ and $\tilde{\overline{\boldsymbol{\mu}}}_i$ are the $6n^i$-dimensional vectors. Intra-modular computations for the above step are given next.

B. Intra-modular computations:

For floating base #0

$$\ddot{\mathbf{q}}_0 = \tilde{\boldsymbol{\varphi}}_0 \tag{7.23}$$

For link #k^i ($k = 1, \ldots, \eta^i$)

For $j = 1 : \varepsilon_k$

$$\ddot{\theta}_{k_j} = \tilde{\boldsymbol{\varphi}}_{k_j} - \boldsymbol{\psi}_{k_j}^T \tilde{\boldsymbol{\mu}}_{k_j}, \text{ where } \tilde{\boldsymbol{\mu}}_{k_j} = \mathbf{A}_{k,\beta} \boldsymbol{\mu}_{\beta_{(k_j)}}, j = 1$$
$$= \boldsymbol{\mu}_{k_{j-1}}, j > 1$$
$$\text{and } \boldsymbol{\mu}_{\beta_{(k_j)}} = \mathbf{p}_{\beta_{(k_j)}} \ddot{\theta}_{\beta_{(k_j)}} + \tilde{\boldsymbol{\mu}}_{\beta_{(k_j)}} \tag{7.24}$$

In the above step $\beta_k = (k-1)$, for $k > 1$.

It may be noted that the derivation of the intra-modular steps from the inter-modular steps are not straightforward as in the case of inverse dynamics. This essentially requires analytical $\mathbf{U}_i \mathbf{D}_i \mathbf{U}_i^T$ decomposition of $\hat{\overline{\mathbf{I}}}_i$. Next, writing $\ddot{\overline{\mathbf{q}}}_i$ of Eq. (7.22) in terms of the block expressions of $\tilde{\overline{\boldsymbol{\varphi}}}_i$, $\overline{\boldsymbol{\Psi}}_i$ and $\tilde{\overline{\boldsymbol{\mu}}}_i$, the expressions of $\ddot{\theta}_k$, for $k = 1, \ldots, \eta^i$, as in Eq. (7.24) are obtained.

Algorithmic implementation of the recursive forward dynamics is shown in Table 7.2. Computational counts of different steps in Table 7.2 and comparison of the forward dynamics algorithm with those existing in literature will be provided in Sect. 7.5.

Table 7.2 Recursive O(n) forward dynamics algorithm for floating-base robotic systems

Step 1: Compute $\mathbf{t}$, $\mathbf{t}'$, and $\mathbf{w}^*$	Step 2: Compute $\hat{\varphi}_k$ and $\hat{\varphi}_k$	Step 3: Compute $\ddot{\theta}_k$
$i = 0,\ k = 0,\ j = 0$	$j = DOF$	$i = 0, k = 0, j = 0$
$\mathbf{t}_0 = \mathbf{P}_0\dot{\mathbf{q}}_0$	For $i = s : 1$ (Inter-modular)	$\ddot{\mathbf{q}}_0 = \tilde{\boldsymbol{\varphi}}_0$
$\mathbf{t}'_0 = \boldsymbol{\Omega}_0\mathbf{P}_0\dot{\mathbf{q}}_0 + \rho$	For $k = \eta^i : 1$ (Inter-modular)	$\boldsymbol{\mu}_0 = \mathbf{P}_0\ddot{\mathbf{q}}_0$
$\mathbf{w}_0^* = \mathbf{M}_0\mathbf{t}'_0 + \boldsymbol{\Omega}_0\mathbf{M}_0\mathbf{t}_0 - \mathbf{w}_0^F$	For $r = \varepsilon_k : 2$(Joint level)	For $i = 1 : s$ (Inter-modular)
$\hat{\mathbf{M}}_0 = \mathbf{M}_0$	call function_2	For $k = 1 : \eta^i$ (Intra-modular)
For $i = 1 : s$ (Inter-modular)	$\hat{\varphi}_j = \tau_j - \mathbf{p}_j^T\tilde{\boldsymbol{\eta}}_j$	$r = 1$ (Joint level)
For $k = 1 : \eta^i$ (Intra-modular)	$\hat{\varphi}_j = \hat{\varphi}_j/\hat{m}_j$	$j = j + 1$
$r = 1$ (Joint level)	$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j\boldsymbol{\psi}_j^T$	$\tilde{\boldsymbol{\mu}}_j = \mathbf{A}_{k,\beta}\boldsymbol{\mu}_{\beta j}$
$j = j + 1$	$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j\hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_j$	call function_4
$\mathbf{t}_j = \mathbf{A}_{k,\beta}\mathbf{t}_{\beta_j} + \mathbf{p}_j\dot{\theta}_j$	$\hat{\mathbf{M}}_{j-1} = \hat{\mathbf{M}}_{j,j}$	For $r = 2 : \varepsilon_k$
$\mathbf{t}'_j = \mathbf{A}_{k,\beta}\mathbf{t}'_{\beta_j} + \dot{\mathbf{A}}_{k,\beta}\mathrm{t}_{\beta_j} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j$	$\hat{\boldsymbol{\eta}}_{j-1} = \boldsymbol{\eta}_j$	$j = j + 1$
$\hat{\mathbf{M}}_j = \mathbf{O}$	$j = j - 1$	$\tilde{\boldsymbol{\mu}}_j = \boldsymbol{\mu}_{j-1}$
For $r = 2 : \varepsilon_k$	end	call function_4
$j = j + 1$	$r = 1$	end
$\mathbf{t}_j = \mathbf{t}_{j-1} + \mathbf{p}_j\dot{\theta}_j$	call function_2	end
$\mathbf{t}'_j = \mathbf{t}'_{j-1} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j$	$\hat{\varphi}_j = \tau_j - \mathbf{p}_j^T(\tilde{\boldsymbol{\eta}}_j + \mathbf{w}_k^*)$	– – – – – – – – – – – – – – –
$\hat{\mathbf{M}}_j = \mathbf{O}$	$\hat{\varphi}_j = \hat{\varphi}_j/\hat{m}_j$	**function_4**
end	$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j\boldsymbol{\psi}_j^T$	$\ddot{\theta}_j = \tilde{\boldsymbol{\varphi}}_j = \boldsymbol{\Psi}_j^T\tilde{\boldsymbol{\mu}}_j$
$\mathbf{w}_k^* = \mathbf{M}_k\mathbf{t}'_j + \boldsymbol{\Omega}_j\mathbf{M}_k\mathbf{E}_k\mathbf{t}_j - \mathbf{w}_j^F$	$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j\hat{\varphi}_j + (\tilde{\boldsymbol{\eta}}_j + \mathbf{w}_k^*)$	$\boldsymbol{\mu}_j = \mathbf{p}_j\ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_j$
$\hat{\mathbf{M}}_j = \mathbf{M}_k$	$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\hat{\mathbf{M}}_{j,j}\mathbf{A}_{k,\beta}$	
end	$\tilde{\boldsymbol{\eta}}_{\beta_j} = \tilde{\boldsymbol{\eta}}_{\beta_j} + \mathbf{A}_{k,\beta}^T\boldsymbol{\eta}_j$	
end	$j = j - 1$	
	end	
	end	
	$i = 0, k = 0, j = 0$	
	$\hat{\boldsymbol{\psi}}_0 = \hat{\mathbf{M}}_0\mathbf{P}_0$	
	$\hat{\mathbf{I}}_0 = \mathbf{P}_0^T\hat{\boldsymbol{\psi}}_0$	
	$\hat{\boldsymbol{\varphi}}_0 = \tau_0 - \mathbf{P}_0^T(\tilde{\boldsymbol{\eta}}_0 + \mathbf{w}_0^*)$	
	$\tilde{\boldsymbol{\varphi}}_0 = \hat{\mathbf{I}}_0^{-1}\hat{\boldsymbol{\varphi}}_0$	
	– – – – – – – – – – – –	
	function_2	
	$\hat{\boldsymbol{\psi}}_j = \hat{\mathbf{M}}_j\mathbf{p}_j$	
	$\hat{m}_j = \mathbf{p}_j^T\hat{\boldsymbol{\psi}}_j$	
	$\boldsymbol{\psi}_j = \hat{\boldsymbol{\psi}}_j/\hat{m}_j$	

The recursive algorithms for inverse and forward dynamics of floating-base systems are implemented in MATLAB. These form the sub-modules of Recursive Dynamics Simulator (ReDySim). The ReDySim has been used for the analysis of floating-base systems presented in this chapter. Detailed discussion on ReDySim is provided in Chap. 10.

7.2 Biped

There have been significant contributions in the fields of dynamics, control, and gait planning of biped robots over the last two decades. Research in the area of biped walking can be categorized into active (Sakagami et al. 2002; Kuroki et al. 2003; Kurazume et al. 2003; Kaneko et al. 2008) and passive walking (McGeer 1990; Collins and Ruina 2005; Wisse et al. 2005). Active walking further can be categorized into static and dynamic walking. In static walking, Center-Of-Mass (COM) remains within the convex hull of the support feet, whereas in dynamics walking, Zero-Moment-Point (ZMP) (Vukobratovic et al. 1989) stays within the convex hull. Concept of the ZMP has played a significant role in achieving dynamic walking for biped. The ZMP-based walking pattern generation of a biped can mainly be divided into two. In the first approach, walking pattern is obtained from the equation of the ZMP, as shown by Hirai et al. (1998), Yamaguchi et al. (1999), Kagami et al. (2002), and Huang et al. (2001). Trajectory generation with the help of equations of ZMP results into computationally expensive procedure as it is a problem of solving complex differential equations involving many dynamic parameters. Second approach is based on the Inverted Pendulum Model (IPM), as shown by Kajita and Tani (1991). This approach assumes that the mass of the biped is concentrated at the hip. The IPM based trajectory generation is much simpler and relies on the feedback control. It has been successfully implemented to attain dynamic walking by Park and Kim (1998), Kajita et al. (2003), Harada et al. (2004) and Morisawa et al. (2005). The discussion on ZMP and trajectory generation for a 3-dimenisonal walk using IPM is provided in Appendix B. In this section, an IPM based trajectory is used to obtain planar and spatial bipedal dynamically stable walking. In contrast to the work by Park and Kim (1998) and Kajita et al. (2003), where a controller plays the major role in achieving the stable gait, here, the input joint torques are obtained purely based on the inverse dynamics model of the biped robot under study.

Note that a biped negotiates three topologies, viz., double support, single support, and flight, during different phases of its walking or running. Analysis of biped dynamics may be either configuration-dependent or configuration-independent. The former approach uses different sets of equations of motion for different configurations, while the latter uses a single set of equations of motion for the complete dynamic analysis. Here, the configuration-independent approach, based on the dynamics of floating-base systems presented in Sect. 7.1 is followed. For the feet contact, penalty-based model, as shown in Appendix D is used.

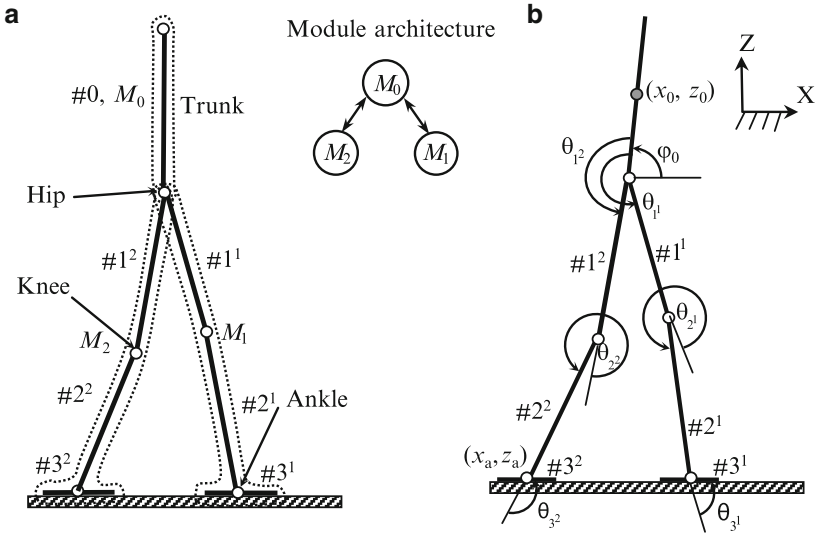


Fig. 7.1 A 7-link planar biped. (**a**) Module architecture. (**b**) Joint variables

7.2.1 A Planar Biped

In the configuration-independent approach, a floating-base is identified first. Knowing the motions of the floating-base, and other joint motions, dynamic analyses may be performed. Figure 7.1 shows a 7-link planar biped. Trunk of the biped is assumed to be the floating-base that forms module M_0, and two legs form modules M_1 and M_2, as shown in Fig. 7.1a. Note that the expression of the DeNOC matrices for the system under study is obtained in Chap. 4, whereas the GIM, its inverse and decomposition are shown in Chap. 5. The coordinates of the Center-of-Mass (COM) (x_0, y_0, z_0) and the YXZ Euler angles $(\varphi_0, \theta_0, \psi_0)$ are assumed to be the generalized independent coordinates for the trunk #0, as shown in Fig. 7.1b. It is assumed that the biped moves in a sagittal plane and hence x_0, y_0, and φ_0 are the only variable coordinates and others are constant. Moreover, $\theta_{1^1}, \theta_{2^1}, \theta_{3^1}, \theta_{1^2}, \theta_{2^2},$ andθ_{3^2} are the generalized coordinates associated with the joint variables, which are also shown in Fig. 7.1b.

The length and mass of the links are taken as $l_0 = 0.5$ m, $l_{1^1} = l_{2^1} = l_{1^2} = l_{2^2} = 0.5$m, $l_{3^1} = l_{3^2} = 0.15$m, $m_0 = 5$Kg, and $m_{1^1} = m_{2^1} = m_{1^2} = m_{2^2} = 1$ Kg, and $m_{3^1} = m_{3^2} = 0.2$ Kg. The trajectories for the COM of trunk and the swing foot are synthesized first. The COM trajectories are designed based on the Inverted Pendulum Model (IPM), whereas the swing foot trajectory is defined as cosine function. These trajectories are function of cycle time (T), co-ordinates of COM $x_0(0)$, $y_0(0)$ and $z_0(0)$ at $T = 0$, stride length (l_s,) and maximum foot height (h_f) which are synthesized in Appendix B. For the planar biped under study, T, $x_0(0)$, $y_0(0)$, $z_0(0)$, l_s, and h_f are assumed as 1 s, −0.15, 0, 0.96, 0.3 and 0.1 m,

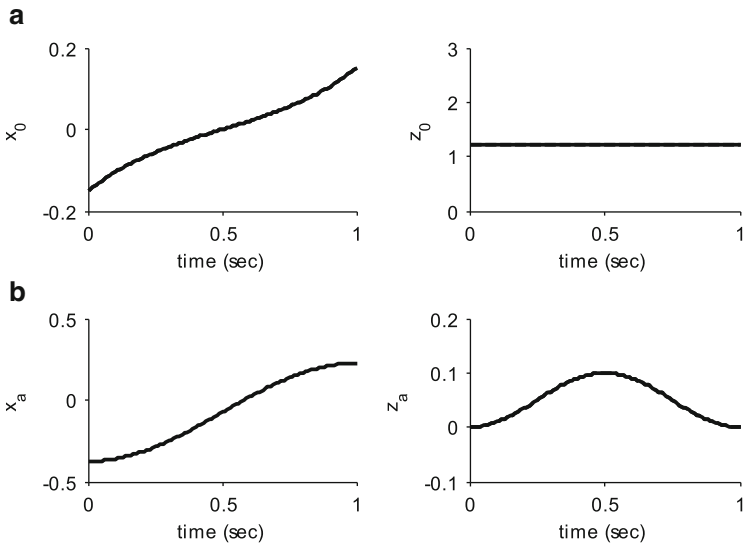


Fig. 7.2 Designed trajectories of the trunk COM and ankle of the planar biped. (**a**) Trunk's COM. (**b**) Ankle of the swing foot

respectively. The resulting trajectories of the COM of the trunk and the ankle of the swing foot are shown in Fig. 7.2. The COM and feet trajectories are then used to obtain the joint trajectories using the inverse kinematics relationships provided in Appendix B. The joint trajectories thus obtained are shown in Fig. 7.3.

Recursive inverse dynamics algorithm derived in Sect. 7.1.1 is then used to obtain the motion of the trunk and the driving toques for one cycle of the bipedal walking. The motion of the trunk in Fig. 7.4 depicts that the biped travels distance of 0.3 m in one second. During the waking cycle, variation of the angle φ_0 is found to be 3°, which is acceptable. The corresponding joint torques are shown in Fig. 7.5. The torques are continuous in nature. However, the discontinuity appears after 0.83 s that is due to the touchdown of the swing foot. This is evident from Fig. 7.6, where the vertical and horizontal reactions for the swing and support feet are shown. The swing foot touches the ground after 0.83 s. As a result, vertical reaction on foot changes from 0 to 100 N and the horizontal reaction changes from 0 to −50 N. Two types of ground models, namely, dusty-ground model and firm-ground, have been used for modeling the foot-ground interactions. Biped walking is successfully achieved using both the models, however the results are reported for the firm ground model. The ground model ensures that the vertical reaction force remains always positive throughout the walking cycle. Hence, ground never pulls the legs. Moreover, the horizontal friction force was approximated by using a pseudo-Coulomb friction model. Details of the ground models are provided in Appendix D.

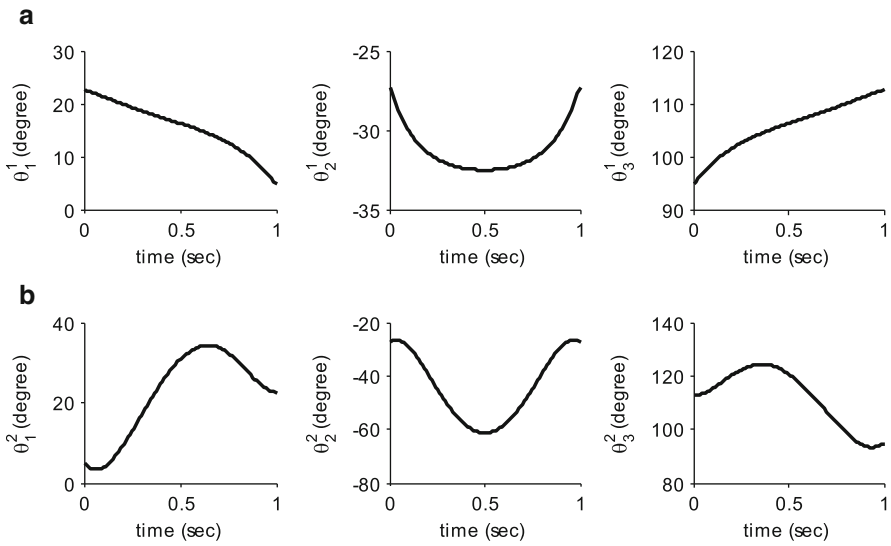


Fig. 7.3 Joint trajectories of the planar biped obtained from trunk and ankle trajectories. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

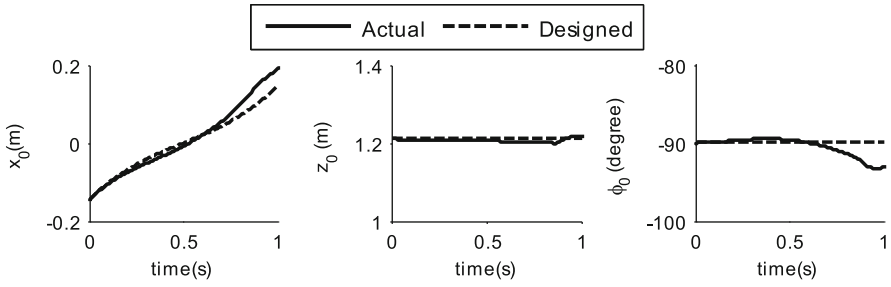


Fig. 7.4 Motions of trunk (Floating-base) of the planar biped. (**a**) x_0 (COM). (**b**) z_0 (COM). (**c**) φ_0

Next, the forward dynamics is performed to simulate the motion of the biped by using the torque obtained from the inverse dynamics algorithm. The recursive forward dynamics algorithm given in the Sect. 7.1.2 was used to simulate the biped shown in Fig. 7.1a. It is evident from Fig. 7.7 that the biped moves in the forward direction (i.e., X) with stable periodic motion. Simulation results in Fig. 7.8 show that the biped follows all the desired joint trajectories without any feedback control.

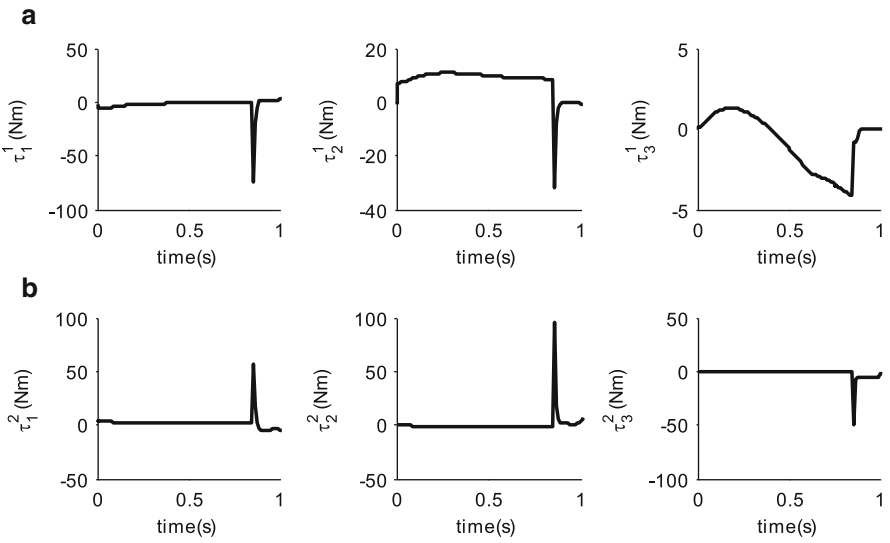


Fig. 7.5 Joint torques at different joints of the planar biped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

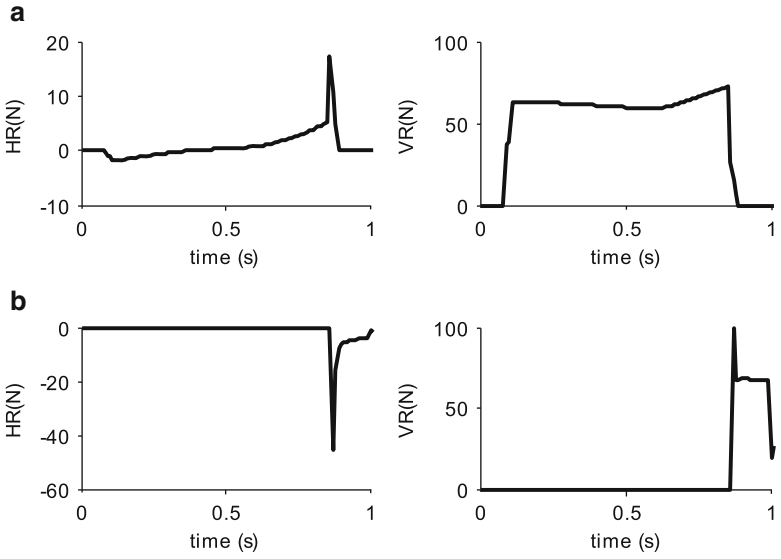


Fig. 7.6 Horizontal Reaction (*HR*) and Vertical Reaction (*VR*) on the feet of the planar biped. (**a**) Support foot (Module 1). (**b**) Swing foot (Module 2)

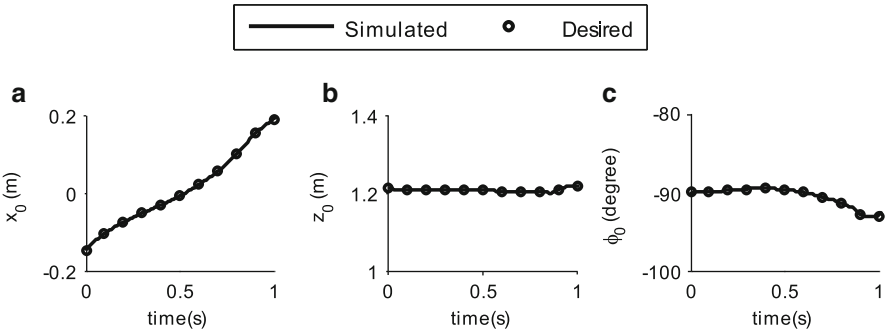


Fig. 7.7 Simulated motions of trunk of the planar biped. (**a**) x_0 (COM). (**b**) z_0 (COM). (**c**) φ_0

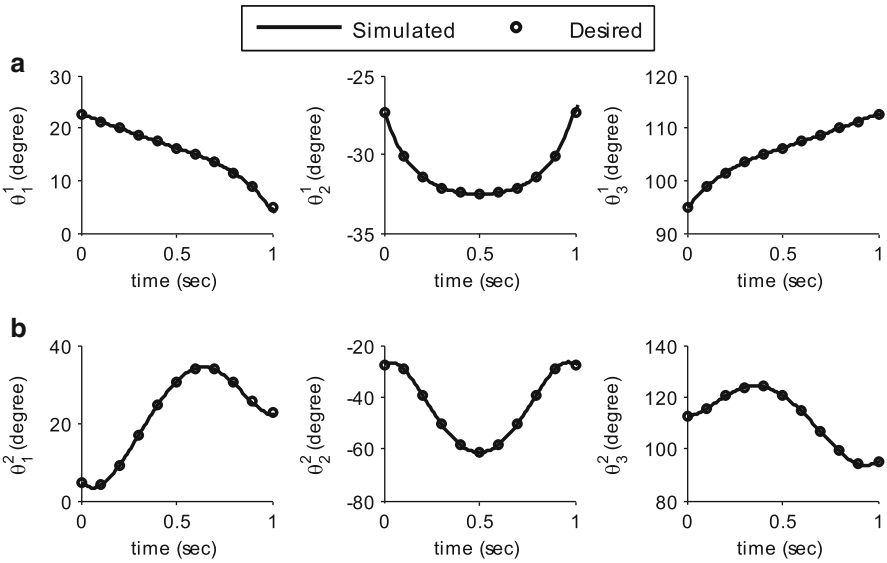


Fig. 7.8 Simulated joint motions of the planar biped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

7.2.2 Spatial Biped

In the previous subsection, dynamics of planar biped was studied, where all the joints are 1-DOF revolute joints, however, a real-life biped, e.g., a human, has many multi-DOF joints as shown in Fig. 7.9. The ankle, for example, allows 2-DOF motion which can be modeled as a universal joint. Similarly, each leg at the hip has a spherical joint, which may be modeled as Euler-Angles-Joints as proposed in Chap. 3. The spatial biped in Fig. 7.9 has seven links and 18-DOF.

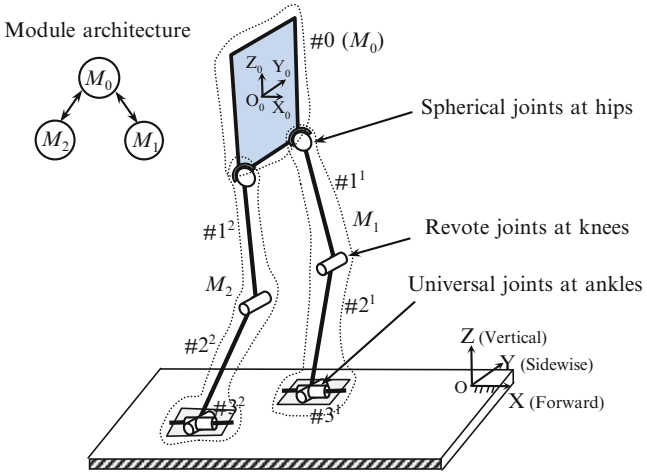


Fig. 7.9 A spatial biped and its modularization

The biped is divided into three modules, as shown in Fig. 7.9. Inverted Pendulum Model is again used to obtain the motion of the trunk, whereas trajectory of swing foot is obtained as a cosine function. For this, T, $x_0(0)$, $y_0(0)$, $z_0(0)$, l_s, and h_f are assumed as 0.5 s, −0.15, 0.08, 0.92, 0.3 and 0.1 m, respectively. The trajectories are designed to achieve dynamic walk of the biped with the forward velocity of 0.6 m/s. The trajectories of trunk's COM and swing foot are shown in Fig. 7.10. Note that (x_0, y_0, z_0) represent the co-ordinates of the Trunk's COM, i.e., O_0 in Fig. 7.9, in the inertial frame (O-XYZ), whereas (φ_0, θ_0, ψ_0) represent the YXZ Euler Angles between frames O-XYZ and O_0-$X_0Y_0Z_0$.

The desired joint trajectories, i.e., $\boldsymbol{\theta}_{1^1} = [\theta_{1_1^1}\ \theta_{1_2^1}\ \theta_{1_3^1}]^T, \theta_{1_2^1}, \boldsymbol{\theta}_{3^1} = [\theta_{3_1^1}\ \theta_{3_2^1}]^T$, $\boldsymbol{\theta}_{1^2} = [\theta_{1_1^2}\ \theta_{1_2^2}\ \theta_{1_3^2}]^T, \theta_{3^2}$, and $\boldsymbol{\theta}_{3^2} = [\theta_{3_1^2}\ \theta_{3_2^2}]^T$, calculated from the trunk and feet trajectories using inverse kinematics relationships, are shown Fig. 7.11. The length and mass of the links are taken as $l_0 = 0.1$ m, $w_0 = 0.1$m $l_{1^1} = l_{2^1} = l_{1^2} = l_{2^2} = 0.5$ m, $l_{3^1} = l_{3^2} = 0.15$ m, $m_0 = 5$ Kg, and $m_{1^1} = m_{2^1} = m_{1^2} = m_{2^2} = 1$ Kg, and $m_{3^1} = m_{3^2} = 0.2$ Kg. It may be noted that four points of the foot plane are monitored to check the status of contact in contrast to two points as in the case of planar biped. This is outlined in Appendix D.

Inverse dynamics of the biped was then performed to obtain the motion of the trunk and the joint torques for given input joint motions. Figure 7.12 shows that for the given joint motions the biped travels the distance of 0.3 m in 0.5 s in the forward direction (i.e., X), whereas motion along Z remains unchanged. This happened due to the assumption that the COM always maintains a specified height. The orientation of the trunk represented in terms of Euler angles is limited to 2°. Corresponding torques are shown in Fig. 7.13. It is worth noting that at touch down the joint torques

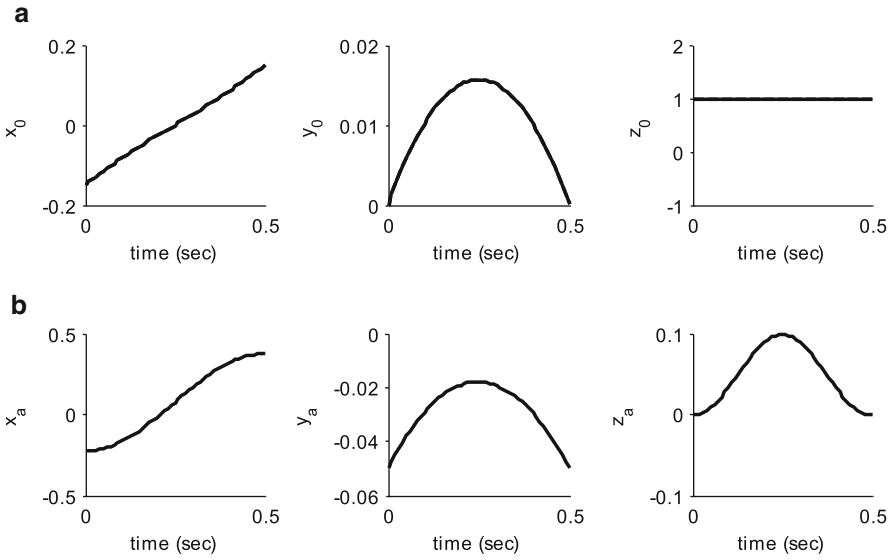


Fig. 7.10 Designed trajectories of the trunk's COM and ankle of the spatial biped. (**a**) Trunk's COM. (**b**) Ankle of the swing foot

are smoother than that of the planar biped. This is mainly due to the spatial motion of the biped and the presence of the multiple-DOF joints at the ankles and hips.

It may be pointed out that the single support phase of a spatial biped with the same model parameters was also analyzed in Chap. 6. There, the biped was assumed to be a fixed-base robotic system, where the foot is the fixed-base. For the sake of comparison, the joint torques $\boldsymbol{\tau}_{1^1} = [\tau_{1_1^1} \;\; \tau_{1_2^1} \;\; \tau_{1_3^1}]^T, \tau_{2^1}, \boldsymbol{\tau}_{3^1} = [\tau_{3_1^1} \;\; \tau_{3_2^1}]^T, \boldsymbol{\tau}_{1^2} = [\tau_{1_1^2} \;\; \tau_{1_2^2} \;\; \tau_{1_3^2}]^T, \tau_{2^2}$ and $\boldsymbol{\tau}_{3^2} = [\tau_{3_1^2} \;\; \tau_{3_2^2}]^T$ as shown in Fig. 7.13 for the floating-base system, may be compared with $\boldsymbol{\tau}_{3^1} = [\tau_{3_1^1} \;\; \tau_{3_2^1} \;\; \tau_{3_3^1}]^T, \tau_{2^1}, \boldsymbol{\tau}_{1^1} = [\tau_{1_1^1} \;\; \tau_{1_2^1}]^T, \boldsymbol{\tau}_{1^2} = [\tau_{1_1^2} \;\; \tau_{1_2^2} \;\; \tau_{1_3^2}]^T, \tau_{2^2}$, and $\boldsymbol{\tau}_{3^2} = [\tau_{3_1^2} \;\; \tau_{3_2^2}]^T$ given in Fig. 6.6 for the fixed-base system. The comparison of the figures, namely, Figs. 6.6 and 7.13, leads to the following observations:

- Joint torques for the swing leg match immediately after the lift-off and before the touchdown.
- For the support leg, torques match at the hip and the knee, but torques at the ankle differ. The difference in torques at the ankle is mainly due to the assumption of the foot remaining fixed during the support phase.
- The dynamic modeling of the biped as a floating-base system not only gives rise to unified approach to model different phases of motion but also allows one to simulate the biped on a variety of surfaces just by changing the parameters of the ground model.

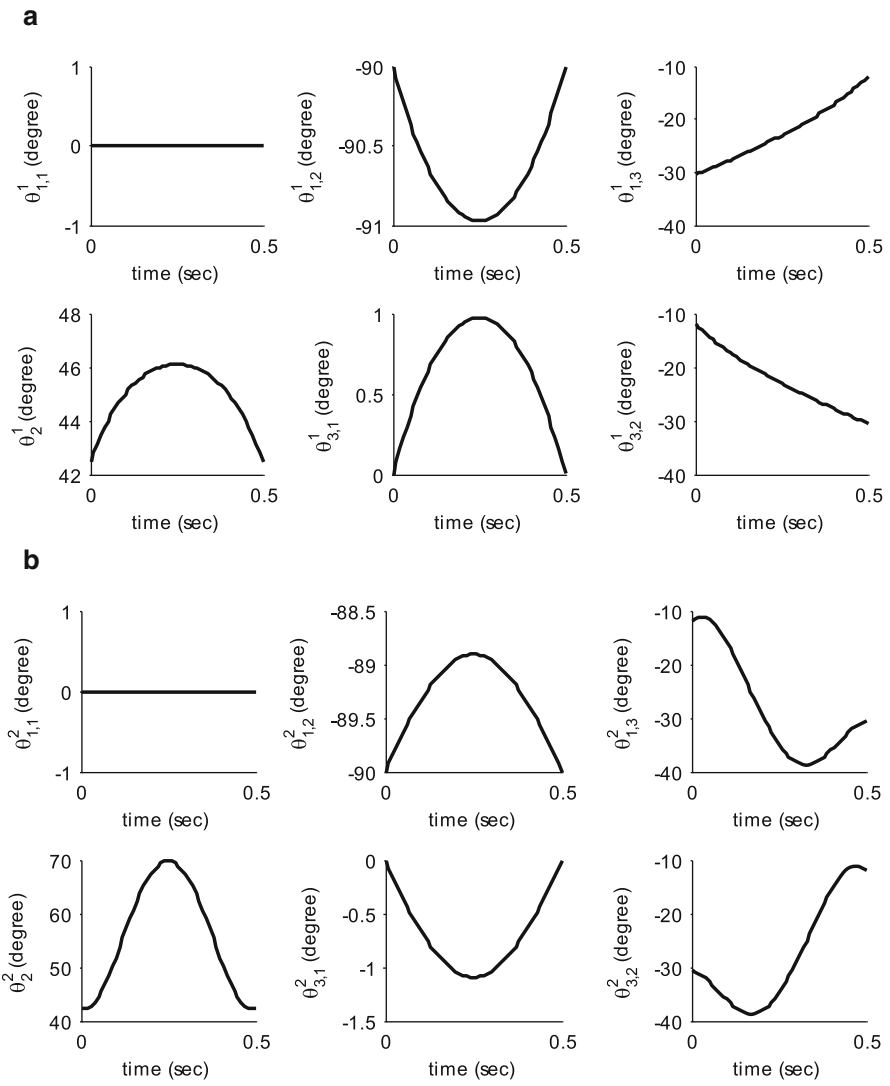


Fig. 7.11 Joint trajectories of the spatial biped obtained from the trunk and ankle trajectories. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

Forward dynamics was then performed with the values of the torques obtained in inverse dynamics calculations. Simulated motions of the trunk are shown in Fig. 7.14. Here, the attitude of the biped is uncontrollable after 0.1 s, as depicted in Fig. 7.14b. Corresponding simulated joint angles are shown in Fig. 7.15 from which it may be seen that the biped is unable to follow the desired trajectory after 0.1 s. This is attributed to the propagation of numerical errors in computation,

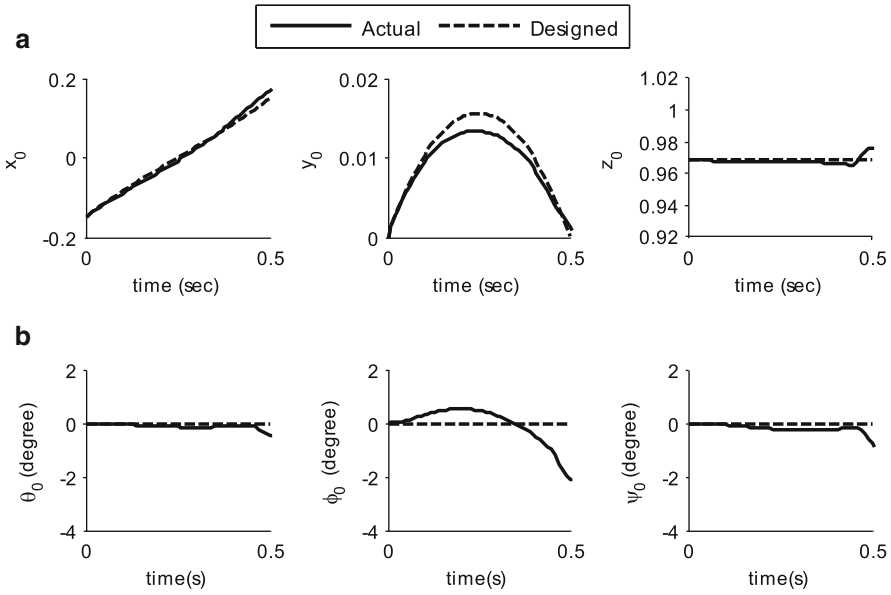


Fig. 7.12 Motions of the trunk of the spatial biped. (**a**) Center-of-Mass. (**b**) Euler angles

which is expected for complex systems. In an actual system, it may not happen if all parameters were identified accurately. However, in real systems inaccuracy is inevitable due to the manufacturing errors in the components of the biped. Moreover, many unmodeled phenomenon like backlash in the actuators, joint frictions, etc. may lead to the deviation of the system's behavior.

Hence, a system like the biped must be controlled. The issue of control will be taken up in Chap. 9.

7.3 Quadruped

Apart from the biped robots, extensive research work in the field of quadruped legged robots has been reported. Muybridge (1957) was first to present a systematic study on different gaits of a quadruped horse. Later, Gambaryan (1974) provided the detailed study of different gaits and mechanics of quadruped mammals. Raibert (1990) used symmetry in quadruped motion and built a hydraulically actuated quadruped. Inspired by the work of Raibert, Furusho et al. (1995), Kimura et al. (1999), Ridderström et al. (2000), Marhefka et al. (2003) and Poulakakis et al. (2006) built several quadruped machines. The major research in the field of quadruped is on achieving dynamic walking and running. This work mainly attempts dynamic walking of the quadruped. Early work on quadruped walking can

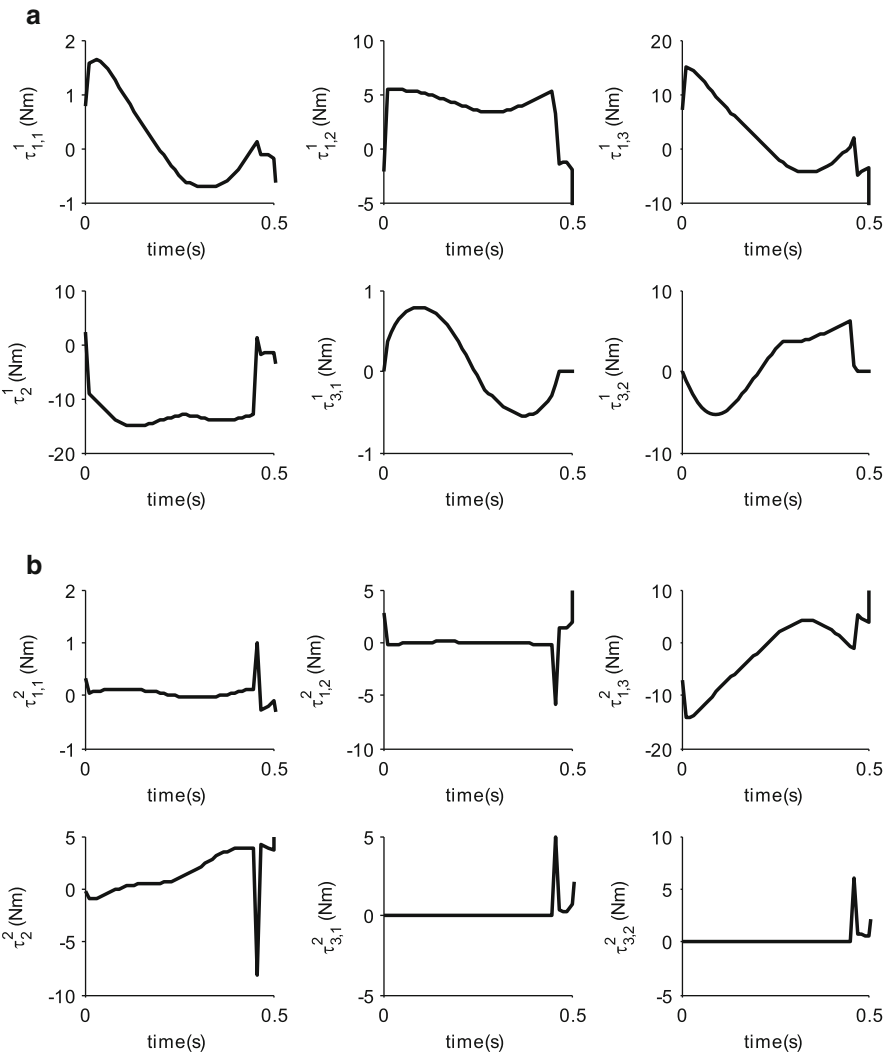


Fig. 7.13 Joint torque at different joints of the spatial biped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

be found in the reference of Miura et al. (1985). Buehler et al. (1999) presented the open loop dynamic walking, where quadruped with one actuator per leg was considered. The work done by Ridderström et al. (2000) mainly pertains to achieve statically balanced gait where three feet always remain on the ground. Kurazume et al. (2001) presented an adaptive attitude control scheme of a quadruped during support phase. Later, Doi et al. (2005) presented quadruped walking on steep slope.

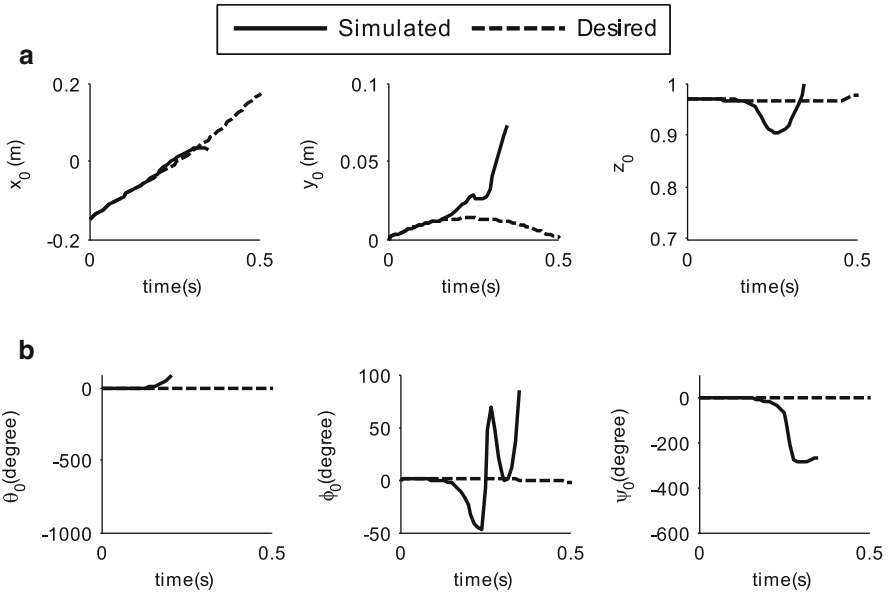


Fig. 7.14 Simulated motions of trunk of the spatial biped. (**a**) Center-of-Mass (**b**) Euler angles

Here, the dynamic analysis of a spatial quadruped is reported based on trotting gait, where diagonal legs move in synchronized manner. In order to study the spatial behavior, a 9-link, 18-DOF spatial quadruped as shown in Fig. 7.16 is considered. It has universal joints at the hips, whereas knees contain revolute joints. The quadruped is divided into five modules, as shown in Fig. 7.16b. The Inverted Pendulum Model (IPM) and cosine function were again used to attain the trajectory of the COM of trunk and swing foot, respectively. For this, the cycle time (T), the co-ordinates of COM $x_0(0)$, $y_0(0)$ and $z_0(0)$, the stride length (l_s,) and the maximum foot height (h_f) are taken as 1 s, −0.15, 0, 0.5, 0.3 and 0.08 m, respectively. The trajectory was designed to attain the forward velocity of 0.4 m/s. The trajectories are show in Fig. 7.17.

The desired trajectories at the joint levels, i.e., $\boldsymbol{\theta}_{1^1} = [\theta_{1_1^1}\ \theta_{1_2^1}]^T, \theta_{2^1}, \boldsymbol{\theta}_{1^2} = [\theta_{1_1^2}\ \theta_{1_2^2}]^T, \theta_{2^2}, \boldsymbol{\theta}_{1^3} = [\theta_{1_1^3}\ \theta_{1_2^3}]^T, \theta_{2^3}, \boldsymbol{\theta}_{1^4} = [\theta_{1_1^4}\ \theta_{1_2^4}]^T$, and θ_{2^4}, calculated from the COM and the feet trajectories using inverse kinematics, are shown in Fig. 7.18. The length and mass of the links were taken as $l_0 = 1$ m, $l_{1^1} = l_{2^1} = l_{1^2} = l_{2^2} = l_{1^3} = l_{2^3} = l_{1^4} = l_{2^4} = 0.3$m, $m_0 = 4$Kg, and $m_{1^1} = m_{2^1} = m_{1^2} = m_{2^2} = m_{1^3} = m_{2^3} = m_{1^4} = m_{2^4} = 1$Kg.

Trunk's motions were obtained from the joint motions using inverse dynamics algorithm shown in Table 7.1. The motions of trunk are shown in Fig. 7.19. Note that the given joint motions provide finer attitude control as the maximum change in the Euler angles as seen in Fig. 7.19b, associated with trunk is limited to 2°.

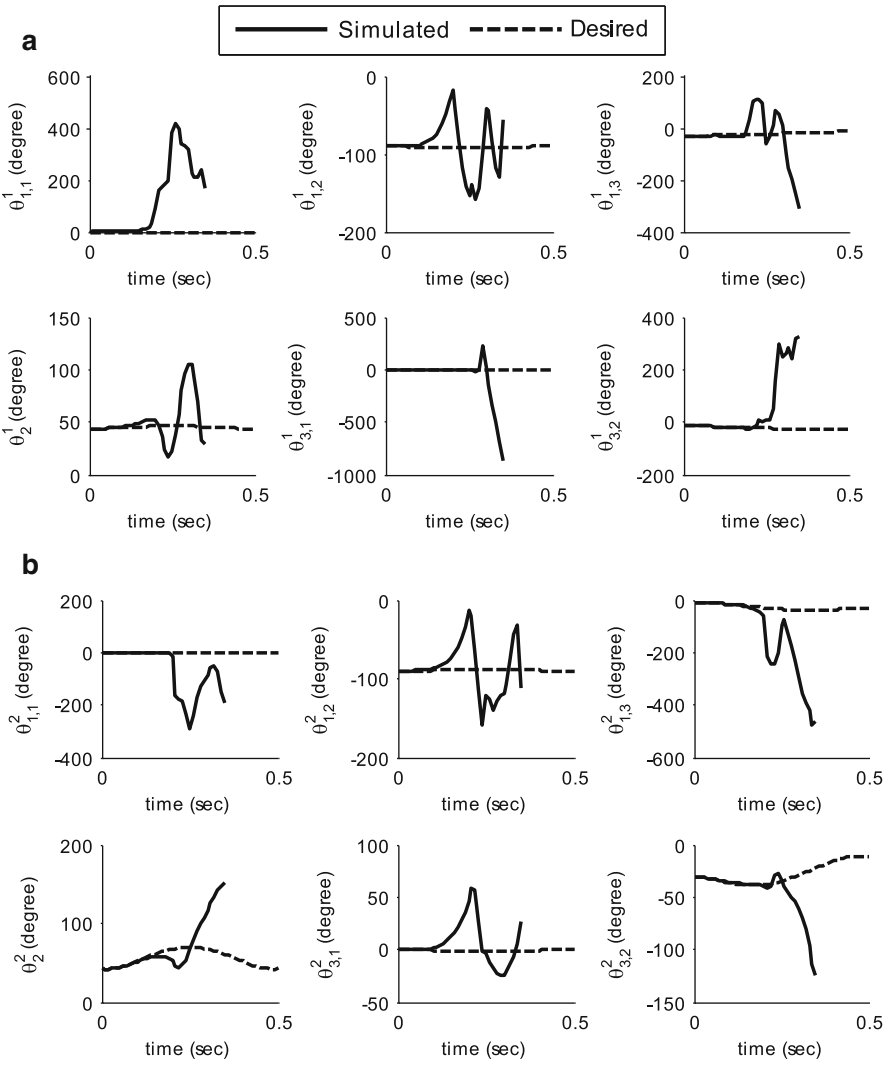


Fig. 7.15 Simulated joint motions of the spatial biped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

The joint torques were then calculated for the prescribed trunk and joint motions. Figure 7.20 shows joint torques at hip and knee for swing and support legs. The discontinuity in the torque after 0.8 s is due to the touchdown of the swing legs.

The simulation was performed, based on the torques obtained in Fig. 7.20. The forward dynamics algorithm used for simulation is presented in Table 7.2. The variation of the COM of the trunk and the Euler angles are shown in Fig 7.21, which depicts that the quadruped moves in the forward direction (X). It is unable

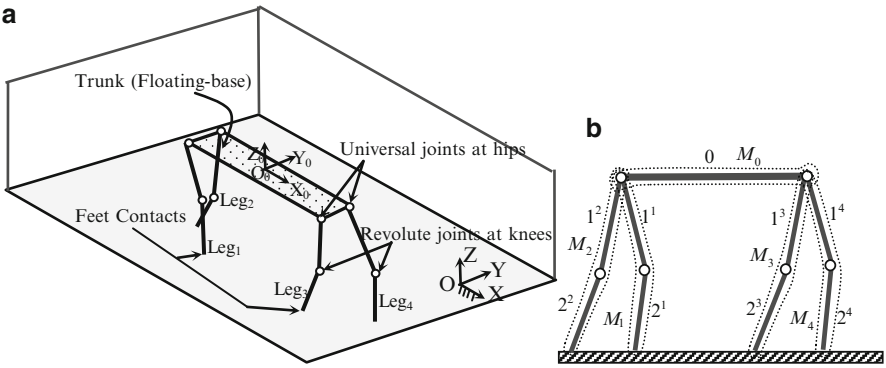


Fig. 7.16 A quadruped and its modularization

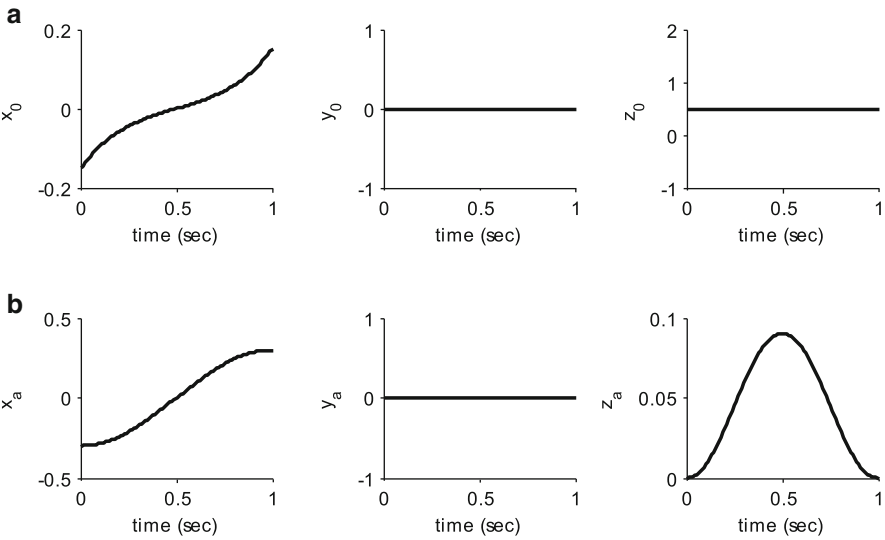


Fig. 7.17 Designed trajectories of the trunk's COM and ankle of the quadruped. (**a**) Trunk's COM. (**b**) Ankle of the swing foot

to follow the desired trajectory after 0.8 s. Figure 7.22 shows the variation of the joint angles, which also follow the desired trajectories till 0.8 s. Once again, the deviation of the actual trajectories from the desired one is due to the propagation of numerical errors. As discussed at the end of Sect. 7.2.2, these deviations may not actually happen in real system if the quadruped is correctly modeled. However, uncertainty is inevitable and a control strategy must be introduced, which is done in Chap. 9.

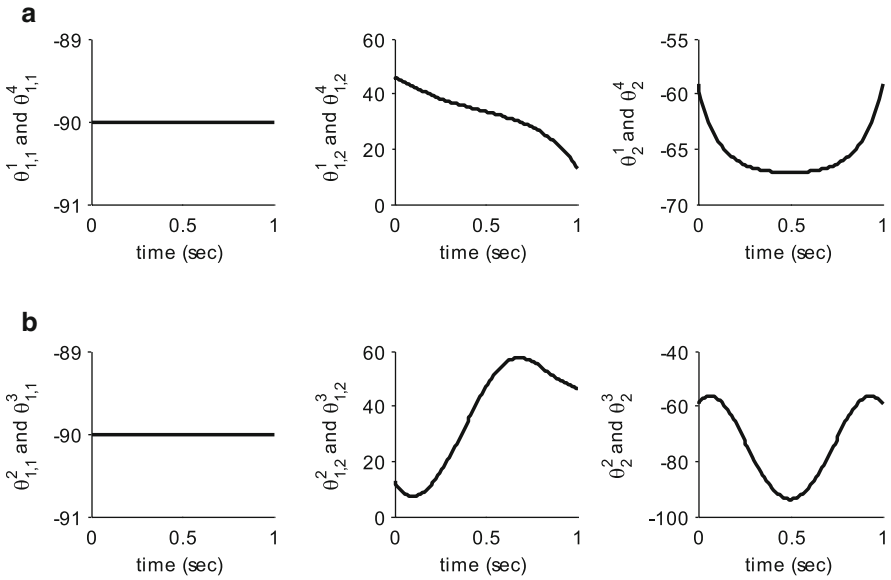


Fig. 7.18 Joint trajectories of the quadruped obtained from trunk and ankle trajectories. (**a**) Support leg (Modules 1 and 4). (**b**) Swing leg (Modules 2 and 3)

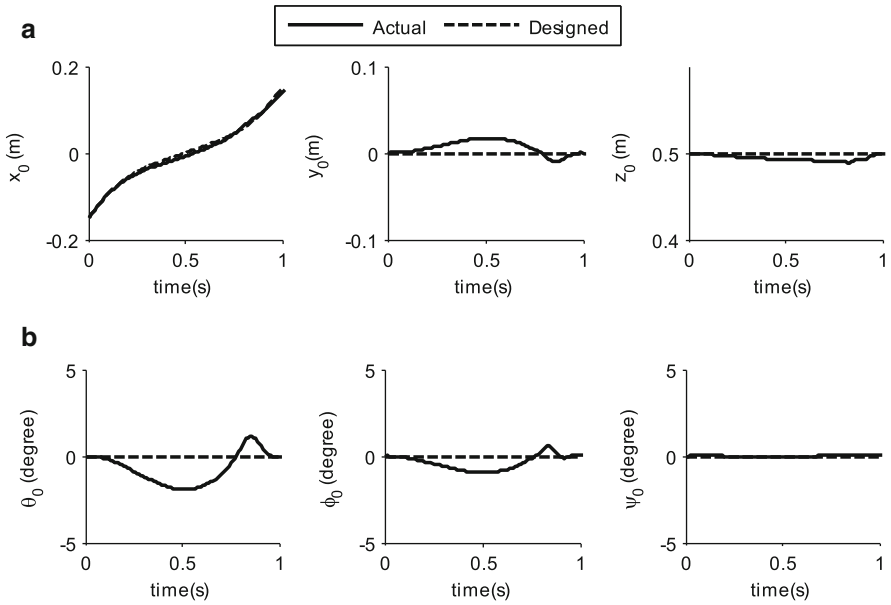


Fig. 7.19 Motions of trunk of the quadruped. (**a**) Center-of-Mass. (**b**) Euler angles

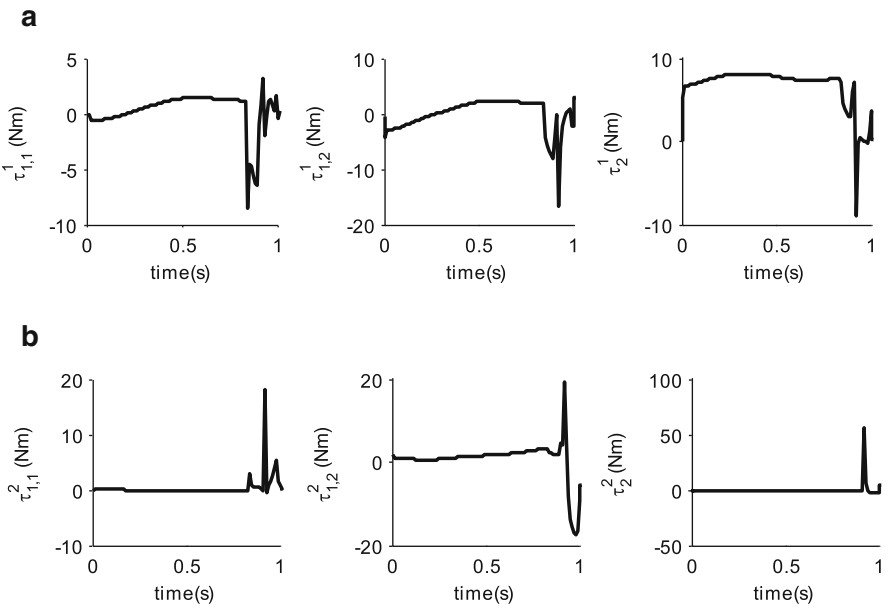


Fig. 7.20 Joint torques at different joints of the quadruped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

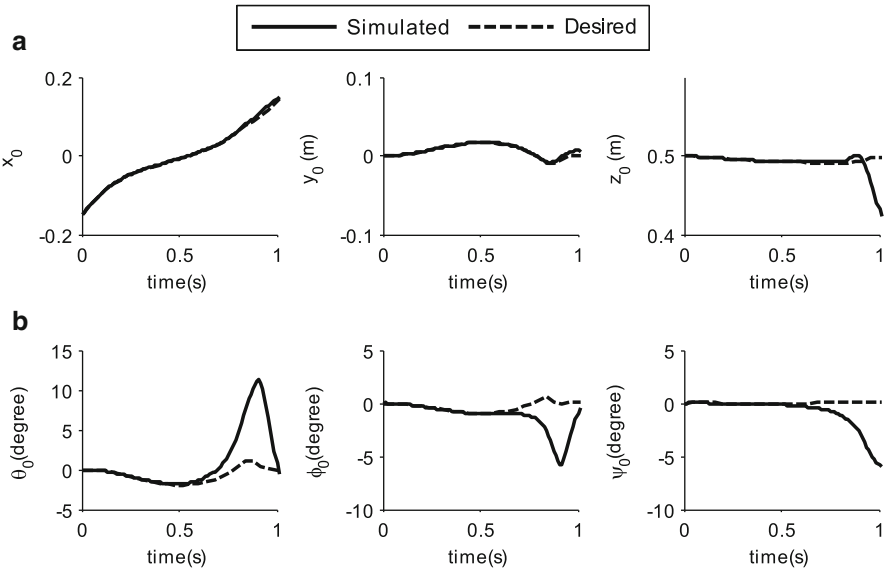


Fig. 7.21 Simulated motions of trunk of the quadruped. (**a**) Center-of-Mass. (**b**) Euler angles

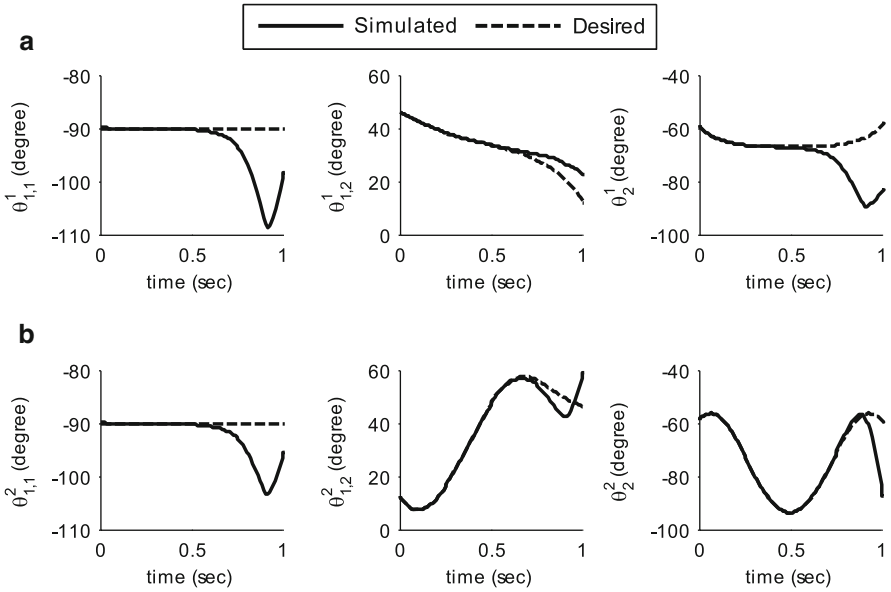


Fig. 7.22 Simulated joint motions of the quadruped. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

7.4 Hexapod

Dynamic analyses of a hexapod walking are carried out next to understand its behavior by using algorithms presented in Sects. 7.1.1 and 7.1.2. For such systems, tripod gait is the most popular one, where the triangularly placed legs move in synchronization. Hence, if the COM of the hexapod remains within the polygon formed by the support feet, the hexapod walking is statically balanced. Any trajectory that follows the above condition will be able to achieve a stable gait. However, this limits the speed of walking. Moreover, walking on rough terrain, self-righting (Saranli et al. 2004), etc. lead to dynamic walking. Figure 7.23 shows 13-link 18-DOF spatial hexapod, where all the joints are assumed to be revolute. In order to use the modular dynamics algorithm presented in this chapter, the hexapod was divided into seven modules, as shown in Fig. 7.23b. Using IPM approach, trajectory of the COM of the trunk was obtained in order to make the hexapod move at a speed of 0.8 m/s. For this, T, $x_0(0)$, $y_0(0)$, $z_0(0)$, l_s, and h_f are assumed as 0.5 s, −0.20, 0, 0.55, 0.4 and 0.08 m, respectively. The trajectories of trunk's COM and the ankle of swing foot thus obtained are shown in Fig. 7.24.

The desired trajectories at the joint levels, i.e., θ_{1^1}, θ_{2^1}, θ_{1^2}, θ_{2^2}, θ_{1^3}, θ_{2^3}, θ_{1^4}, θ_{2^4}, θ_{1^5}, θ_{2^5}, θ_{1^6}, and θ_{2^6}, are shown in Fig. 7.25. The length and mass of the links were taken as $l_0 = 1$ m, $l_{1^1} = l_{2^1} = l_{1^2} = l_{2^2} = l_{1^3} = l_{2^3} = l_{1^4} = l_{2^4} = l_{1^5} = l_{2^5} = l_{1^6} = l_{2^6} = 0.3$m, $m_0 = 4$ Kg, and $m_{1^1} = m_{2^1} = m_{1^2} = m_{2^2} = m_{1^3} = m_{2^3} = m_{1^4} = m_{2^4} = m_{1^5} = m_{2^5} = m_{1^6} = m_{2^6} = 1$ Kg.

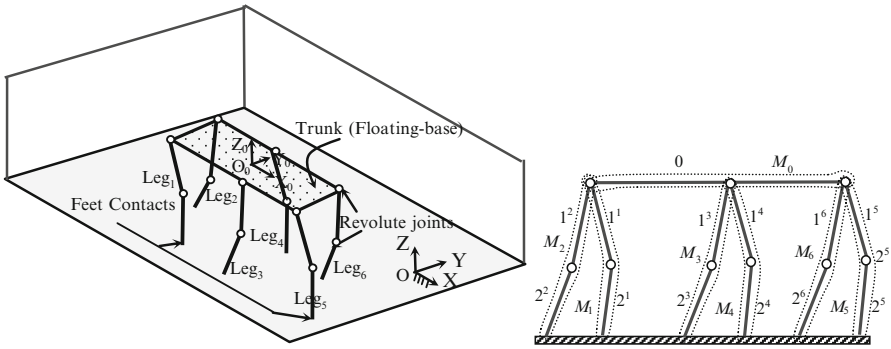


Fig. 7.23 A hexapod and its modularization

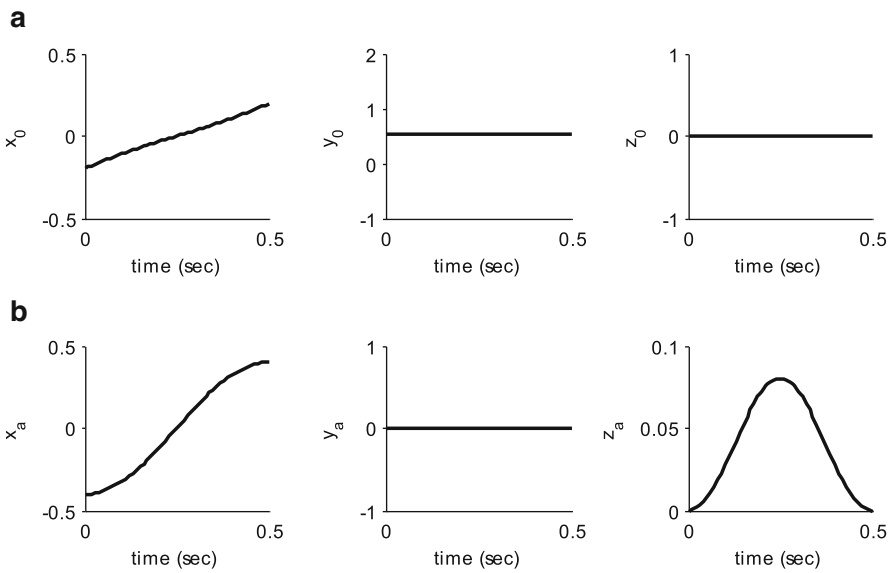


Fig. 7.24 Designed trajectories of the trunk COM and ankle of the hexapod. (**a**) Trunk COM. (**b**) Ankle of the swing foot

Inverse dynamics results are shown in Figs. 7.26 and 7.27. The motions of the trunk (Fig. 7.26) show that the hexapod travels 0.4 m in 0.5 s in the forward direction (X), whereas the orientations of trunk shown by Euler Angles vary within 0.3° only. This shows that fine control over attitude of the trunk can be achieved by appropriate selection of the parameters of the trajectory. Figure 7.27 shows that the joint torques needed to achieve the desired motion of the trunk that was synthesized using IPM. It is worth noting that for the given joint motions the joint torques are continuous, and the tendency of discontinuity is much less pronounced in comparison with the biped and quadruped even after the touchdown at 0.45 s.

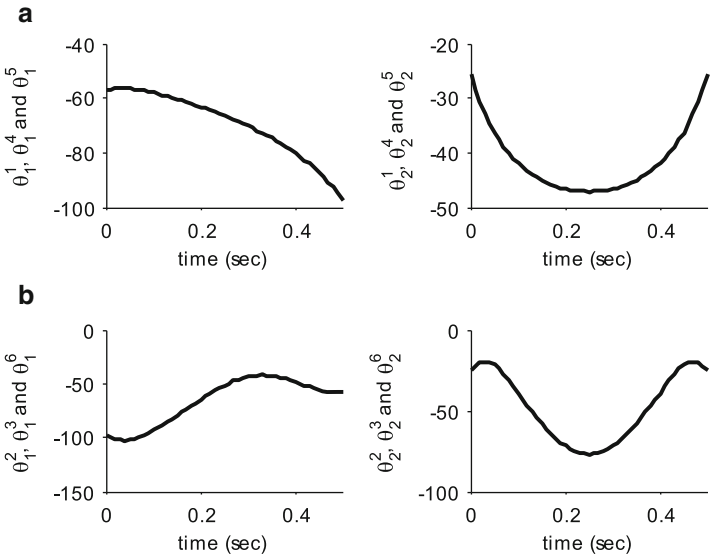


Fig. 7.25 Joint trajectories of the hexapod obtained from trunk and ankle trajectories. (**a**) Support leg (Modules 1, 4 and 5). (**b**) Swing leg (Modules 2, 3 and 6)

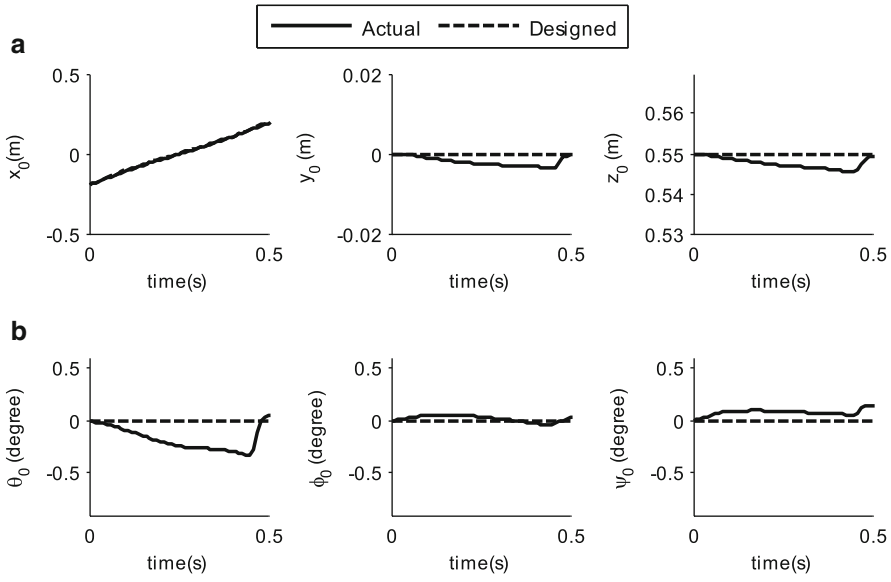


Fig. 7.26 Motions of trunk of the hexapod. (**a**) Center-of-Mass. (**b**) Euler angles

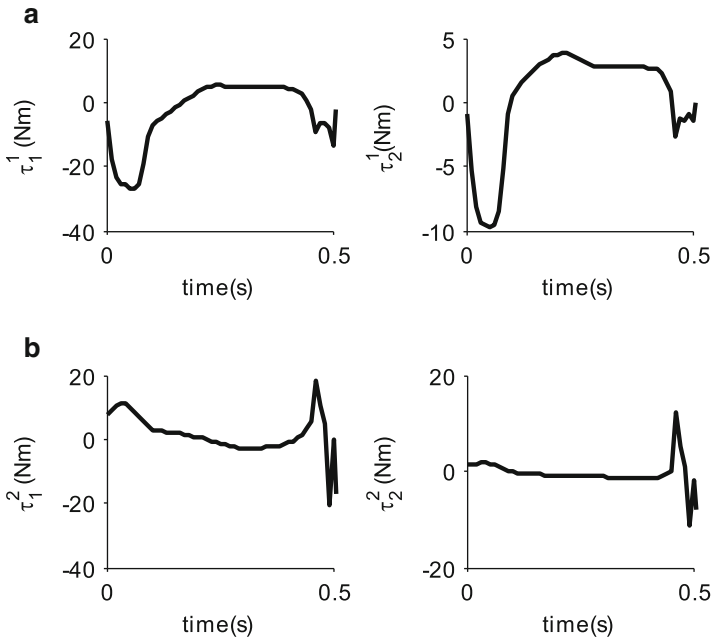


Fig. 7.27 Joint torque at different joints of the hexapod. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

Like the other walking robots in the previous section, the motion of hexapod was simulated for the joint torques shown in Fig. 7.27. Simulation results, in Fig. 7.28, show that the hexapod moves in the forward direction (X), while its orientation shown by the Euler angles deviate only at the end of the walking cycle. Similarly, the hexapod is able to follow the desired joint angles with little variations at the end as depicted in Fig. 7.29.

7.5 Computational Efficiency

Since recursive algorithms are well known for computational efficiencies when the Degrees-of-Freedom (DOF) are many, it was decided to look into this aspect for the floating-base system. Computational counts for different steps of recursive inverse and forward dynamics algorithms of floating-base system, obtained in Tables 7.1 and 7.2, are shown in Tables 7.3 and 7.4, respectively.

The computational count for the recursive inverse and forward dynamics of the *k*th link connected by a multiple-DOF joint is summarized in Table 7.5.

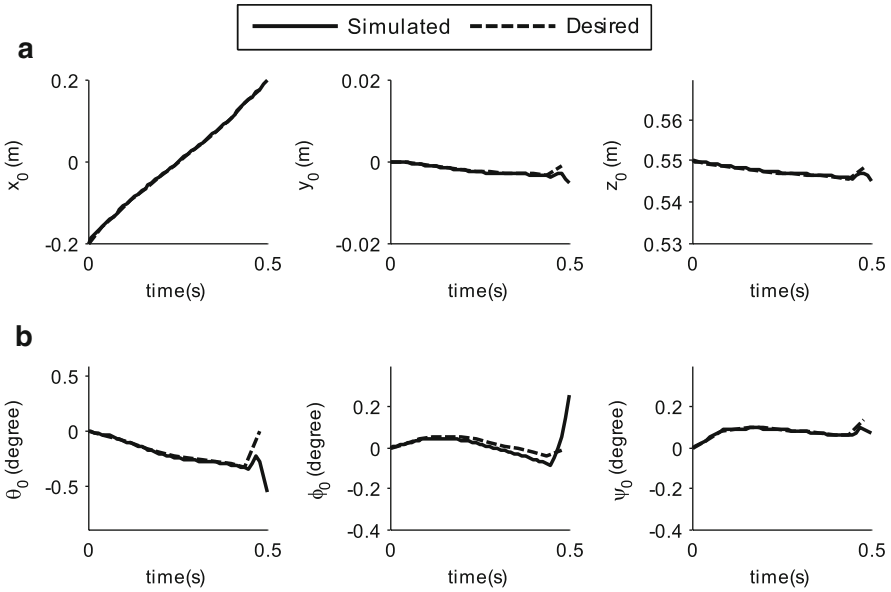


Fig. 7.28 Simulated motions of the trunk of the hexapod. (**a**) Center-of-Mass. (**b**) Euler angles

Based on Table 7.5, the computational count for the recursive inverse and forward dynamics for floating-base system is obtained as

For inverse dynamics: $(164n_1+113n_2+96n_3+60)M(156n_1+113n_2+82\frac{2}{3}n_3+66)A$

For forward dynamics: $(209n_1 + 139\frac{1}{2}n_2 + 116\frac{1}{3}n_3 + 60)M(201n_1 + 130n_2 + 106\frac{1}{3}n_3 + 72)A$

where n_1, n_2 and n_3 are the total number of joint variables associated with 1-, 2- and 3-DOF joints, respectively. Next, the computational efficiencies for the inverse and forward dynamics algorithms are compared in Tables 7.6 and 7.7, respectively, with those available in the literature.

As expected, the computational complexities depend on the total number of joint variables, n $(=n_1 + n_2 + n_3)$. As evident form Tables 7.6 and 7.7 and Figs. 7.30 and 7.31, the algorithms obtained are computationally more efficient than those available in the literature. The efficiencies were mainly achieved from novel modeling of the multiple-DOF joints proposed in Chap. 3. On this aspect, Yamane and Nakamura (1999) showed that 40 DOF humanoid has 12 spherical joints and 4 revolute joints. Each spherical joint can be represented as three intersecting revolute joints. Such an approach will require one to have 41 links connected by 40 revolute joints. On the contrary, the use of Euler Angles Joints (EAJs) in Chap. 3 treats each spherical joint as three intersecting 1-DOF revolute joints, but no physical link between 1st and 2nd, and 2nd and 3rd joints. As a result, only 17 links, instead of 41 links, are

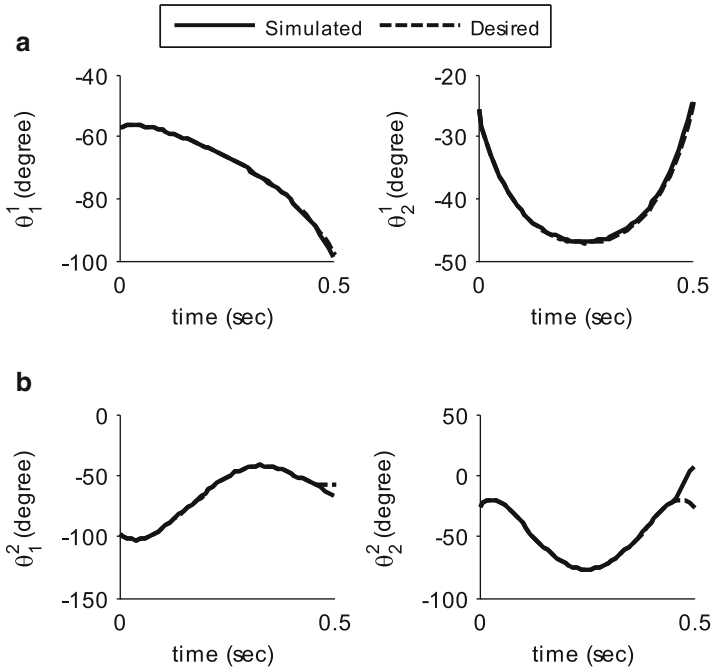


Fig. 7.29 Simulated joint motions of the hexapod. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

required for the analysis of the above-mentioned biped, which has 12 spherical and 4 revolute joints. This brings significant savings in the computational complexity. Similar benefits were achieved in other walking robots reported in this chapter. This was not possible in the work reported by Sugihara et al. (2002), Kurazume et al. (2003), Vukobratovic et al. (2007), Kwon and Park (2009) and others.

It may be noted that biped, quadruped and hexapod analyzed in the previous subsection contain different combinations of multiple-DOF joints, however, have a total of 12 joint variables. Hence, the use of any existing algorithms for the analysis will require the same computational counts as shown in Tables 7.4 and 7.5. However, with the use of the proposed algorithms the quadruped requires less computational counts than the hexapod. This is due to the presence of universal joints in the quadruped. Moreover, the biped requires even less computational count than the quadruped, as shown in Tables 7.4 and 7.5, due to presence of spherical joints. This is also evident from Figs. 7.30 and 7.31, where the slopes for the systems with multiple-DOF joints are much less in both inverse and forward dynamics. It is pointed out here that in contrast to the inverse dynamics algorithm for fixed-base system, the same for floating-base systems requires some additional computations, mainly, to compute elements of the Generalized Inertia Matrix (GIM) as evident from Eq. (7.1). Computational counts for such inverse

Table 7.3 Computational count for recursive inverse dynamics of floating-base system

		Counts		Counts
Forward recursion:				
For $k=0$ (Floating-base #0)				
1	$\mathbf{t}_0 = \mathbf{P}_0\mathbf{q}_0 = \mathbf{q}_0(\mathbf{P}_0 = \mathbf{1})$	−		
2	$\mathbf{t}'_0 = \dot{\mathbf{P}}_0\dot{\mathbf{q}}_0 + \boldsymbol{\rho} = \boldsymbol{\rho}(\dot{\mathbf{P}}_0 = \mathbf{O})$			
	$\mathbf{A}_{0,0} = 1$	−		
3	$\mathbf{w}_0^* = \mathbf{M}_0\dot{\mathbf{t}}_0 + \boldsymbol{\Omega}_0\mathbf{M}_0\mathbf{E}_0\mathbf{t}_0 - \mathbf{w}_0^F$; $\tilde{\mathbf{w}}_0 = \mathbf{w}_0^*, \tilde{\mathbf{M}}_0 = \mathbf{M}_0$	24M30		
For $k=1{:}\eta$, $r=1{:}\varepsilon_k$				
	For $r=1$		For $r=2{:}\varepsilon_k$	
4	$\mathbf{t}_j = \mathbf{A}_{k,\beta_k}\mathbf{t}_{\beta_j} + \mathbf{p}_j\dot{\theta}_j$	8M5A	$\mathbf{t}_j = \mathbf{t}_{j-1} + \mathbf{p}_j\dot{\theta}_j$	4M3A
5	$\mathbf{t'}_j = \mathbf{A}_{k,\beta_k}\mathbf{t'}_{\beta_j} + \dot{\mathbf{A}}_{k,\beta_k}\mathbf{t}_{\beta_j} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	27M20A	$\mathbf{t'}_j = \mathbf{t'}_{j-1} + \boldsymbol{\Omega}_j\mathbf{p}_j\dot{\theta}_j + \mathbf{p}_j\ddot{\theta}_j$	10M7A
6	$\mathbf{A}_{k,0} = \mathbf{A}_{k,\beta}\mathbf{A}_{\beta,0}$		$\mathbf{A}_{k,0} = \mathbf{A}_{k,0}$	
	$[\mathbf{a}_{0,k}]_j = \mathbf{Q}_j^T[\mathbf{a}_{0,\beta(k)} + \mathbf{a}_{\beta(k),k}]_{j-1}$	8M7A	$[\mathbf{a}_{0,k}]_j = \mathbf{Q}_j^T[\mathbf{a}_{0,k}]_{j-1}$	4M2A
	For $r=\varepsilon_k$		For $r=1{:}(\varepsilon_k-1)$	
7	$\boldsymbol{\varpi}_j = \tilde{\tilde{\boldsymbol{\omega}}}_j + \tilde{\boldsymbol{\omega}}_j\tilde{\boldsymbol{\omega}}_j$	6M9A	$\boldsymbol{\varpi}_i = \mathbf{0}$	0M0A
8	$\mathbf{w}_k^* = \mathbf{M}_k\mathbf{t'}_j + \boldsymbol{\Omega}_j\mathbf{M}_k\mathbf{E}_k\mathbf{t}_j - \mathbf{w}_k^F$; $\tilde{\mathbf{w}}_j = \mathbf{w}_k^*, \tilde{\mathbf{M}}_j = \mathbf{M}_k$	39M45	$\tilde{\mathbf{w}}_j = \mathbf{0}, \tilde{\mathbf{M}}_j = \mathbf{O}$	0M0A
Backward recursion:				
For $k=\eta{:}1$, $r=\varepsilon_k{:}1$				
	For $r=1$		For $r=\varepsilon_k{:}2$	
1	$\tilde{h}_j = \mathbf{p}_j^T\tilde{\mathbf{w}}_j$		$\tilde{h}_j = \mathbf{p}_j^T\tilde{\mathbf{w}}_j$	
	$\tilde{h}_j = \mathbf{e}_j^T\tilde{\mathbf{n}}_j$	−	$\tilde{h}_j = \mathbf{e}_j^T\tilde{\mathbf{n}}_j$	−
2	$\boldsymbol{\kappa}_j = \tilde{\mathbf{M}}_j\mathbf{p}_j$		$\boldsymbol{\kappa}_j = \tilde{\mathbf{M}}_j\mathbf{p}_j$	
	$\boldsymbol{\kappa}_{j,t} = \tilde{\mathbf{I}}_j\mathbf{e}_j$	−	$\boldsymbol{\kappa}_{j,t} = \tilde{\mathbf{I}}_j\mathbf{e}_j$	−
	$\boldsymbol{\kappa}_{j,b} = -(\tilde{\boldsymbol{\delta}}_j \times 1)\mathbf{e}_j$	−	$\boldsymbol{\kappa}_{j,b} = -(\tilde{\boldsymbol{\delta}}_j \times 1)\mathbf{e}_j$	−
3	$\tilde{\boldsymbol{\kappa}}_j = \mathbf{A}_{j,0}^T\boldsymbol{\kappa}_j$		$\tilde{\boldsymbol{\kappa}}_j = \mathbf{A}_{j,0}^T\boldsymbol{\kappa}_j$	
	$\tilde{\boldsymbol{\kappa}}_{j,b} = \boldsymbol{\kappa}_{j,b}$	−	$\tilde{\boldsymbol{\kappa}}_{j,b} = \boldsymbol{\kappa}_{j,b}$	−
	$\tilde{\boldsymbol{\kappa}}_{j,t} = \boldsymbol{\kappa}_{j,t} + \left[\mathbf{a}_{k,0}^T\right]_j\boldsymbol{\kappa}_{j,b}$	6M6A	$\tilde{\boldsymbol{\kappa}}_{j,t} = \boldsymbol{\kappa}_{j,t} + \left[\mathbf{a}_{k,0}^T\right]_j\boldsymbol{\kappa}_{j,b}$	6M6A
4	$\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_{\beta_j} + \mathbf{A}_{k,\beta_j}^T\tilde{\mathbf{w}}_j$	20M18A	$\tilde{\mathbf{w}}_{\beta_j} = \tilde{\mathbf{w}}_j$	8M4A
5	$\tilde{\mathbf{M}}_{\beta_j} = \mathbf{M}_{\beta_j} + \mathbf{A}_{k,\beta}^T\tilde{\mathbf{M}}_j\mathbf{A}_{k,\beta}$	34M40A	$\tilde{\mathbf{M}}_{\beta_j} = \tilde{\mathbf{M}}_j$	16M10A
For $k=0$ (Floating-base #0)				
6	$\tilde{\mathbf{h}}_0 = \mathbf{P}_0^T\tilde{\mathbf{w}}_0$	−		
7	$\mathbf{K}_0 = \tilde{\mathbf{M}}_0\mathbf{P}_0; \mathbf{I}_{0,0} = \mathbf{K}_0^T\mathbf{P}_0 = \tilde{\mathbf{M}}_0$	−		
Forward recursion:				
For $k=0$ (Floating-base #0)				
1	$\ddot{\mathbf{q}}_0 = -\mathbf{I}_{0,0}^{-1}\tilde{\mathbf{h}}_0$	36M36A		
2	$\tilde{\mathbf{q}}_0 = \mathbf{P}_0\ddot{\mathbf{q}}_0 = \ddot{\mathbf{q}}_0$	−		
For $k=1{:}\eta$, $r=1{:}\varepsilon_k$				
	For $r=1$		For $r=2{:}\varepsilon_k$	
3	$[\tilde{\mathbf{q}}_0]_j = \begin{bmatrix} \mathbf{Q}_k^T[\tilde{\mathbf{q}}_{0,t}]_{j-1} \\ \mathbf{Q}_k^T[\tilde{\mathbf{q}}_{0,b}]_{j-1} \end{bmatrix}$	16M8A	$[\tilde{\mathbf{q}}_0]_j = \begin{bmatrix} \mathbf{Q}_k^T[\tilde{\mathbf{q}}_{0,t}]_{j-1} \\ \mathbf{Q}_k^T[\tilde{\mathbf{q}}_{0,b}]_{j-1} \end{bmatrix}$	8M4A
4	$\boldsymbol{\tau}_j = \tilde{\boldsymbol{\kappa}}_j^T[\tilde{\mathbf{q}}_0]_j + \tilde{h}_j$	6M7A	$\boldsymbol{\tau}_j = \tilde{\boldsymbol{\kappa}}_j^T[\tilde{\mathbf{q}}_0]_j + \tilde{h}_j$	6M7A
	Total: (1) *k*th link	**164M156A**	+	**62M46A**(ε_k-1)
	(2) Floating-base (#0)	**60M66A**		

Table 7.4 Computational count for recursive forward dynamics of the floating-base system

Forward recursion				
(In this step $\mathbf{t}_j, \mathbf{t}'_j (\ddot{\theta}_j = \mathbf{0})$ **and** $\mathbf{w}_k^*$ are calculated similarly as in forward recursion of inverse dynamics)				
		Counts		Counts
For $k=0$ (Base #0): $\mathbf{t}_0, \mathbf{t}'_0$ **and** $\mathbf{w}_0^*$		24M30A		
For $k=1{:}\eta$, $r=1{:}\varepsilon_k$				
For $r=1$: $\mathbf{t}_j, \mathbf{t}'_j$ **and** $\mathbf{w}_k^*$		74M66A	For $r=2{:}\varepsilon_k$: $\mathbf{t}_j, \mathbf{t}'_j$	14M10A
Backward recursion				
For $k=\eta{:}1$, $r=\varepsilon_k{:}1$				
	For $r=1$		For $r=\varepsilon_k{:}2$	
1	$\hat{\boldsymbol{\psi}}_j = \hat{\mathbf{M}}_j \mathbf{p}_j$	–	$\hat{\boldsymbol{\psi}}_j = \hat{\mathbf{M}}_j \mathbf{p}_j$	–
2	$\hat{m}_j = \mathbf{p}_j^T \hat{\boldsymbol{\psi}}_j$	–	$\hat{m}_j = \mathbf{p}_j^T \hat{\boldsymbol{\psi}}_j$	–
3	$\boldsymbol{\psi}_j = \hat{\boldsymbol{\psi}}_j / \hat{m}_j$	5M0A	$\boldsymbol{\psi}_j = \hat{\boldsymbol{\psi}}_j / \hat{m}_j$	4M0A
4	$\hat{\varphi}_j = \boldsymbol{\tau}_j - \mathbf{p}_j^T (\tilde{\boldsymbol{\eta}}_j + \mathbf{w}_k^*)$	0M2A	$\hat{\varphi}_j = \boldsymbol{\tau}_j - \mathbf{p}_j^T \tilde{\boldsymbol{\eta}}_j$	0M2A
5	$\bar{\varphi}_j = \hat{\varphi}_j / \hat{m}_j$	1M0A	$\bar{\varphi}_j = \hat{\varphi}_j / \hat{m}_j$	1M0A
6	$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j \boldsymbol{\psi}_j^T$	18M18A	$\hat{\mathbf{M}}_{j,j} = \hat{\mathbf{M}}_j - \hat{\boldsymbol{\psi}}_j \boldsymbol{\psi}_j^T$	10M10A
7	$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j \hat{\varphi}_j + (\tilde{\boldsymbol{\eta}}_j + \mathbf{w}_k^*)$	5M18A	$\boldsymbol{\eta}_j = \boldsymbol{\psi}_j \hat{\varphi}_j + \tilde{\boldsymbol{\eta}}_j$	4M11A
8	$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{\beta_j} + \mathbf{A}_{k,\beta}^T \hat{\mathbf{M}}_{j,j} \mathbf{A}_{k,\beta}$	64M78A	$\hat{\mathbf{M}}_{\beta_j} = \hat{\mathbf{M}}_{j,j}$	24M23A
9	$\tilde{\boldsymbol{\eta}}_{\beta_j} = \mathbf{A}_{k,\beta}^T \boldsymbol{\eta}_j$	20M12A	$\tilde{\boldsymbol{\eta}}_{\beta_j} = \boldsymbol{\eta}_j$	8M4A
For $k=0$ (Floating-base #0)				
10	$\hat{\boldsymbol{\psi}}_0 = \hat{\mathbf{M}}_0 \mathbf{P}_0 = \hat{\mathbf{M}}_0$	0M0A		
11	$\hat{\mathbf{I}}_0 = \mathbf{P}_0^T \hat{\boldsymbol{\psi}}_0 = \hat{\boldsymbol{\psi}}_0$	0M0A		
12	$\hat{\boldsymbol{\varphi}}_0 = \boldsymbol{\tau}_0 - \mathbf{P}_0^T (\tilde{\boldsymbol{\eta}}_0 + \mathbf{w}_0^*)$ $= -(\tilde{\boldsymbol{\eta}}_0 + \mathbf{w}_0^*)$	0M6A		
13	$\tilde{\boldsymbol{\varphi}}_0 = \hat{\mathbf{I}}_0^{-1} \hat{\boldsymbol{\varphi}}_0$	36M36A		
Forward recursion				
For $k=0$ (Floating-base #0)				
1	$\ddot{\mathbf{q}}_0 = \tilde{\boldsymbol{\varphi}}_0$	0M0A		
2	$\boldsymbol{\mu}_0 = \mathbf{P}_0 \ddot{\mathbf{q}}_0 = \ddot{\mathbf{q}}_0$			
For $k=1{:}\eta$, $r=1{:}\varepsilon_k$				
	For $r=1$		For $r=2{:}\varepsilon_k$	
3	$\tilde{\boldsymbol{\mu}}_j = \mathbf{A}_{k,\beta} \boldsymbol{\mu}_{\beta_j}$	20M12A	$\tilde{\boldsymbol{\mu}}_j = \boldsymbol{\mu}_{\beta_j}$	8M4A
4	$\ddot{\theta}_j = \tilde{\varphi}_j - \boldsymbol{\psi}_j^T \tilde{\boldsymbol{\mu}}_j$	5M6A	$\ddot{\theta}_j = \tilde{\varphi}_j - \boldsymbol{\psi}_j^T \tilde{\boldsymbol{\mu}}_j$	4M5A
5	$\boldsymbol{\mu}_j = \mathbf{p}_j \ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_j$	0M1A	$\boldsymbol{\mu}_j = \mathbf{p}_j \ddot{\theta}_j + \tilde{\boldsymbol{\mu}}_j$	0M1A
	Total: 1) kth link	**209M201A**	+	**70M59A(ε_k–1)**
	2) Floating-base (#0)	**60M72A**		

Table 7.5 Summary of computational count for the kth link

	Inverse dynamics 164M156A+62M46A(ε_k-1)			Forward dynamics 209M201A+70M59A(ε_k-1)		
	1-DOF $\varepsilon_k=1$	2-DOF $\varepsilon_k=2$	3-DOF $\varepsilon_k=3$	1-DOF $\varepsilon_k=1$	2-DOF $\varepsilon_k=2$	3-DOF $\varepsilon_k=3$
Count per joint (CPJ)	164M156A	226M202	288M248A	209M201A	279M260	349M319A
Count per DOF (CPJ/ε_k)	164M156A	113M101	$96M82\frac{2}{3}A$	209M201A	$139\frac{1}{2}M130A$	$116\frac{1}{3}M106\frac{1}{3}A$

Table 7.6 Computational complexity of the recursive O(n) inverse dynamics algorithm for floating-base systems

Algorithms	Computational complexity in terms of $n = n_1 + n_2 + n_3$	Floating base $n_0 = 6$	Biped $n_1 = 2$, $n_2 = 4$, $n_3 = 6$, $n = 12$, $n_0 = 6$	Quadruped $n_1 = 4$, $n_2 = 8$, $n_3 = 0$, $n = 12$, $n_0 = 6$	Hexapod $n_1 = 12$, $n_2 = 0$, $n_3 = 0$, $n = 12$, $n_0 = 6$
Proposed	$\mathbf{(164n_1 + 113n_2 + 96n_3)M}$	**60M**	**1416M**	**1620M**	**2028M**
	$\mathbf{(156n_1 + 113n_2 + 82\frac{2}{3}n_3)A}$	**66A**	**1322A**	**1594A**	**1938A**
[a]Balafoutis and Patel (1991)	$(93n - 69)\mathbf{M}(\mathbf{81}n-65)\mathrm{A}+$ $\left(10\frac{3}{2}n^3 + 11n - 42\right)\mathrm{M}$ $\left(\frac{11}{2}n^2 + \frac{65}{2}n - 54\right)\mathrm{A}$	60M 66A	3011M 2222A	3011M 2222A	3011M 2222A
[a]Featherstone (1987)	$(130n-68)\mathrm{M}(101n-56)\mathrm{A}+$ $(10n^2+31n-41)\mathrm{M}$ $(6n^2+40n-46)\mathrm{A}$	60M 66A	3479M 2644A	3479M 2644A	3479M 2644A
[a]Saha (1999b), Bhangale et al. (2004)	$(120n-44)\mathrm{M}(97n-55)\mathrm{A}+$ $(11\mathrm{n}^2+42\mathrm{n}-18)\mathrm{M}$ $(7n^2+44n-53)\mathrm{A}$	60M 66A	3682M 2782A	3682M 2782A	3682M 2782A

Note: *M* Multiplications/Divisions, *A* Addition/Subtraction
[a]Complexity assumed as the computation of Recursive Newton Euler Algorithm (*RNEA*) and Composite Rigid Body Algorithm (*CRBA*) proposed by the same author

Table 7.7 Computational complexity of the recursive O(n) forward dynamics algorithm for floating-base systems

Algorithms	Computational complexity ($n = n_1 + n_2 + n_3$)	Floating base $n_0 = 6$	Biped $n_1 = 2$, $n_2 = 4$, $n_3 = 6$, $n = 12$, $n_0 = 6$	Quadruped $n_1 = 4$, $n_2 = 8$, $n_3 = 0$, $n = 12$, $n_0 = 6$	Hexapod $n_1 = 12$, $n_2 = 0$, $n_3 = 0$, $n = 12$, $n_0 = 6$
Proposed	$\mathbf{\left(209n_1 + 139\frac{1}{2}n_2 + 116\frac{1}{3}n_3\right)M}$	**60M**	**1430M**	**2008M**	**2568M**
McMillan and Orin (1998)	$\mathbf{\left(201n_1 + 130n_2 + 106\frac{1}{3}n_3\right)A}$ $(224n)\mathrm{M}$ $(205n)\mathrm{A}$	**66A** 158M 131	**1624A** 2846M 2591A	**1910A** 2846M 2591A	**2478A** 2846M 2591A
Stejskal and Valasek (1996)[a]	$(226n)\mathrm{M}$ $(206n)\mathrm{A}$	158M 131[b]	2870M 2603A	2870M 2603A	2870M 2603A
McMillan and Orin (1998)	$\left(\frac{1}{6}n^3 + 17\frac{1}{2}n^3 + 175\frac{1}{3}n\right)\mathrm{M}$ $\left(\frac{1}{6}n^3 + 13n^3 + 160\frac{5}{6}n\right)\mathrm{A}$	137M 91A	5081M 4181A	5081M 4181A	5081M 4181A

Note: (1) *M* Multiplications/Divisions, *A* Addition/Subtraction; (2) DOF $= n_0 + n$.
[a]When implemented on floating base systems
[b]Assumed same as that of McMillan et al. (1995) as both use articulated body algorithm

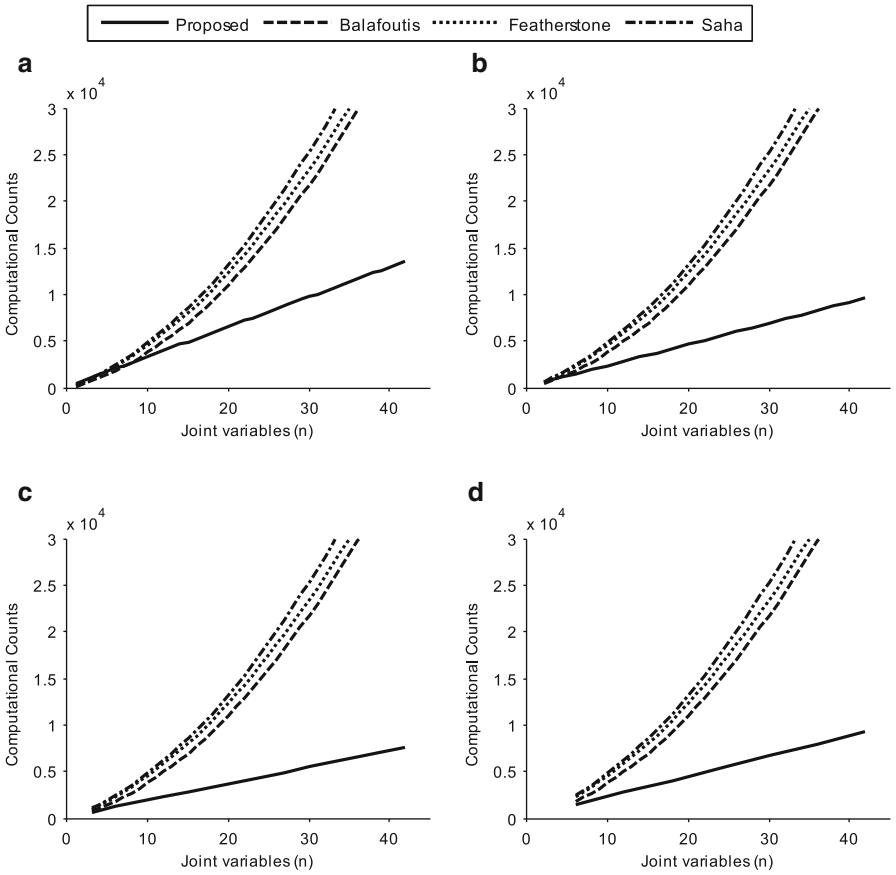


Fig. 7.30 Performance of the proposed inverse dynamics algorithm for a system with multiple-DOF joints. (**a**) All 1-DOF. (**b**) All 2-DOF joints. (**c**) All 3-DOF. (**d**) Equal number of 1-, 2- and 3-DOF joints

dynamics algorithm for floating-base system are not found in the literature. Hence, the computational count of the proposed inverse dynamics algorithms is compared with the total computational count of Recursive Newton-Euler Algorithm (RNEA) and Composite-Rigid-Body Algorithm (CRBA) proposed by the same author.

7.6 Summary

In this chapter, efficient recursive Order (n) inverse and forward dynamics algorithms were presented for the analyses of floating-base robotic systems consisting of multiple-DOF joints. Several important legged robots were analyzed using these

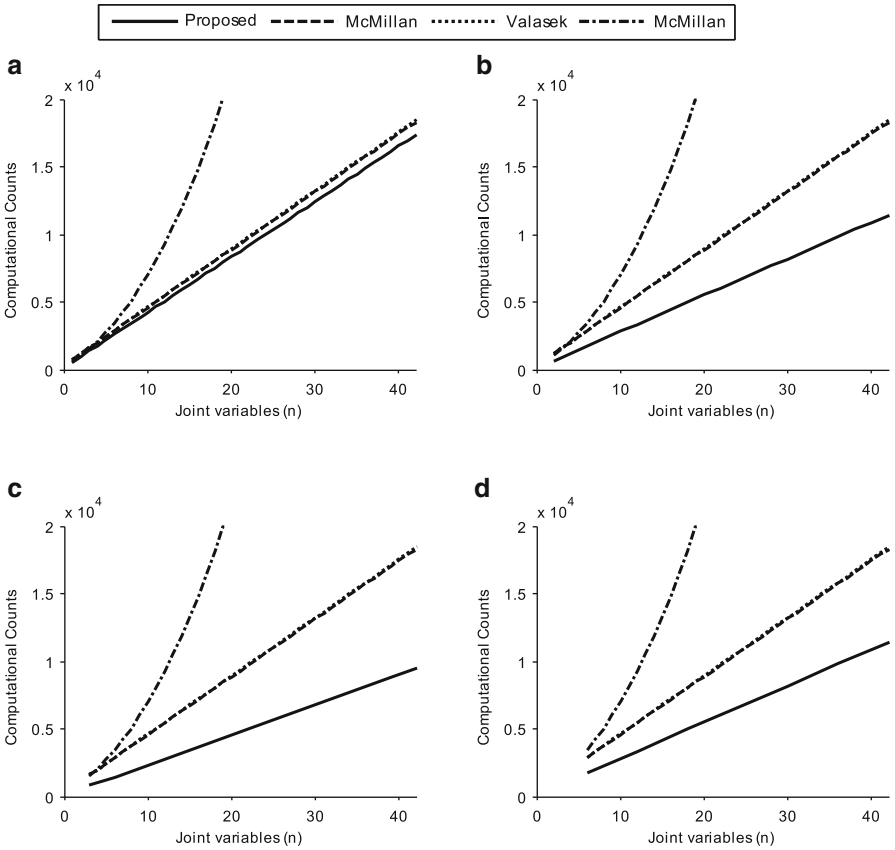


Fig. 7.31 Performance of the proposed forward dynamics algorithm for a system with multiple-DOF joints. (**a**) All 1-DOF. (**b**) All 2-DOF joints. (**c**) All 3-DOF. (**d**) Equal number of 1-, 2- and 3-DOF joints

algorithms, which required for their design and control. The algorithms performed little better than the fastest algorithm available in the literature when the robots consist of only 1-DOF joints but performed much better when the robot have many multiple-DOF joints.

Chapter 8
Closed-Loop Systems

Closed-loops are inherent in many practical robotic systems. In this chapter, analyses of closed-loop systems are presented using the dynamic formulation given in Chaps. 5, 6, and 7.

8.1 Tree-Type Representation of Closed-Loop Systems

Figure 8.1a shows schematic diagram of a general closed-loop system. The closed-loop system has η links and η_c joints, and there exists $\eta_l = (\eta_c - \eta)$ independent kinematic loops. One of the approaches to analyze a closed-loop system is to convert it into equivalent tree-type architecture by cutting suitable joints. A closed-loop system with η_l independent loops is required to cut at η_l joints in order to convert it into a tree-type system as shown in Fig. 8.1b. For complex systems, the joints to be cut may be identified using the concept of Cumulative DOF (CDOF) in graph theory (Deo 1974) as suggested by Chaudhary and Saha (2007). The concept will be illustrated in Sect. 8.4. The cut opened joints are then substituted with suitable constraint forces denoted with λ's, which are also known as Lagrange multipliers. These multipliers are treated as external forces to the resulting open tree-type system. As a result, the problem is converted into solving a tree-type system with externally applied constraint forces. Therefore equations of motion of the closed-loop system in terms of the externally loaded open-loop system are written as

$$\mathbf{I}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} = \boldsymbol{\tau} + \boldsymbol{\tau}^F + \mathbf{J}^T\boldsymbol{\lambda} \tag{8.1}$$

In Eq. (8.1), $\mathbf{J}$ represents the $l \times n$ constrained Jacobian matrix (Nikravesh 1988) for the closed-loop system, where l is the total number of constraints imposed by the joints of the η_l loops and n is total number of joint variables of the open tree-type system. It is defined in a way so that $\mathbf{J}\dot{\mathbf{q}} = \mathbf{0}$. Moreover, $\boldsymbol{\lambda}$ is the l-dimensional vector of Lagrange multipliers representing the constraint forces at the cut joints.

It is worth noting here that Eq. (8.1) can further be written in terms of the independent coordinates, i.e., DOF of the closed-loop system at hand, by eliminating

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_8,

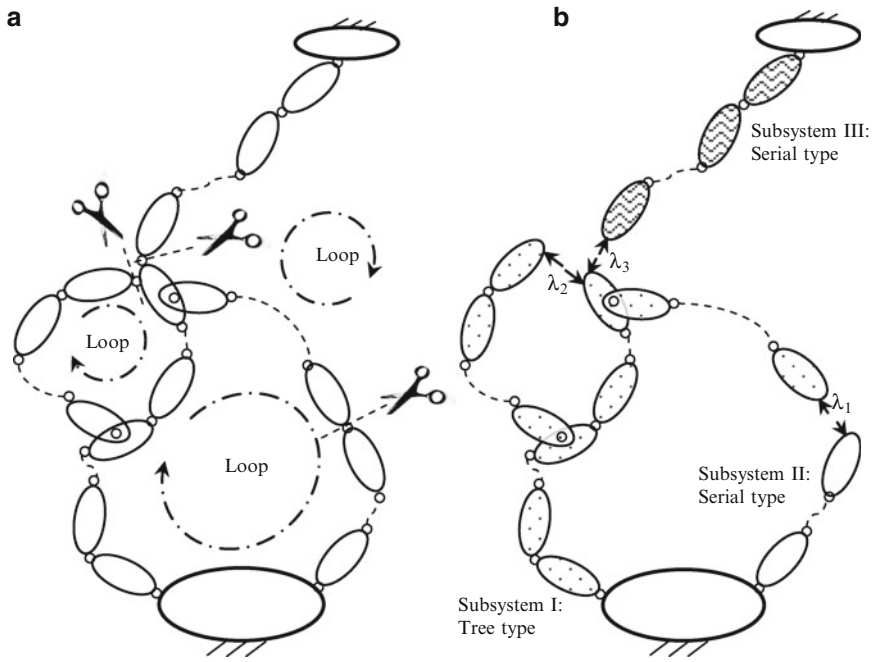


Fig. 8.1 Tree-type representation of a closed-loop system. (**a**) Closed-loop system. (**b**) Tree-type representation of (**a**)

the Lagrange multipliers (Shabana 2001). This, however, will not allow one to use the recursive algorithms obtained in Chaps. 6 and 7 for solving the closed-loop system. Hence, the latter approach will not be followed in this chapter.

8.2 Dynamic Formulation

Dynamic formulation of a closed-loop system, namely, inverse and forward dynamics formulation, is explained in the following sections.

8.2.1 Inverse Dynamics

Once the equivalent tree-type system of a closed-loop system is obtained, one can solve for the Lagrange multipliers and the actuated forces or torques together. For that, Eq. (8.1) is written as

$$\boldsymbol{\tau} + \mathbf{J}^T \boldsymbol{\lambda} = \mathbf{I}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}^F \tag{8.2}$$

Next, the vector of generalized forces $\boldsymbol{\tau}$ can be written in terms of the torques required at the actuated joints, namely, $\boldsymbol{\tau} \equiv \mathbf{J}_a^T \boldsymbol{\tau}_a$. Hence, substituting $\boldsymbol{\tau} \equiv \mathbf{J}_a^T \boldsymbol{\tau}_a$ into Eq. (8.2), one obtains

$$\mathbf{J}_a^T \boldsymbol{\tau}_a + \mathbf{J}^T \boldsymbol{\lambda} = \boldsymbol{\tau}^{\varphi}, \text{where } \boldsymbol{\tau}^{\varphi} = \mathbf{I}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}^F \tag{8.3}$$

In Eq. (8.3), $\mathbf{J}_a$ is the $n_a \times n$ matrix for n_a joint variables associated with the actuated joints. Rearranging Eq. (8.3), one can solve for $\boldsymbol{\tau}_a$ and $\boldsymbol{\lambda}$ as

$$\begin{bmatrix} \boldsymbol{\tau}_a \\ \boldsymbol{\lambda} \end{bmatrix} = \mathbf{F}^{-1} \boldsymbol{\tau}^{\varphi}, \text{where } \mathbf{F} \equiv \begin{bmatrix} \mathbf{J}_a^T & \mathbf{J}^T \end{bmatrix} \tag{8.4}$$

In (8.4), $\boldsymbol{\tau}^{\varphi}$ is obtained using any recursive inverse dynamics algorithm, say, the one proposed earlier in Chap. 6, whereas $\mathbf{F}$ is the $n \times n$ matrix.

Alternatively, Chaudhary and Saha (2007) proposed a concept of "determinate" and "indeterminate" subsystems, which requires inversion of several smaller matrices corresponding to the number of subsystems in the large tree. This approach has been proven to be more powerful and elegant in comparison to the inversion of the matrix $\mathbf{F}$ given by Eq. (8.4). Hence, the methodology of Chaudhary and Saha (2007) will be adopted in Sects. 8.3, 8.4, and 8.5 for the inverse dynamic of closed-loop systems.

8.2.2 *Forward Dynamics*

In forward dynamics, independent joint acceleration ($\ddot{\mathbf{q}}$) and constraint forces ($\boldsymbol{\lambda}$) are the unknown. As the system has n unknowns for $\ddot{\mathbf{q}}$ and l unknown for $\boldsymbol{\lambda}$, forward dynamics problem requires solution of $(n + l)$ unknowns using n equations of motion and l kinematic constraint equations resulting out of the joints of the loops. Conventionally (Shabana 2001), $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$ are solved together by combining the equations of motion given by Eq. (8.1), and the second derivative of constraint equations given by $\mathbf{J}\ddot{\mathbf{q}} = -\dot{\mathbf{J}}\dot{\mathbf{q}}$. They are put together as

$$\begin{bmatrix} \mathbf{I} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\varphi} \\ -\dot{\mathbf{J}}\dot{\mathbf{q}} \end{bmatrix}, \text{where } \boldsymbol{\varphi} \equiv \boldsymbol{\tau} + \boldsymbol{\tau}^G - \mathbf{C}\dot{\mathbf{q}} \tag{8.5}$$

The formulation in Eq. (8.5) is popularly known as Differential Algebraic Equations (DAE). Using Eq. (8.5), $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$ can be solved simultaneously, which requires $O(n + l)^3$ computations. Alternatively, $\boldsymbol{\lambda}$ can also be obtained first by rearranging Eq. (8.5) as

$$\boldsymbol{\lambda} = -(\mathbf{J}\mathbf{I}^{-1}\mathbf{J}^T)^{-1}(\mathbf{J}\mathbf{I}^{-1}\boldsymbol{\varphi} + \dot{\mathbf{J}}\dot{\mathbf{q}}) \tag{8.6}$$

Next, the solution of the joint accelerations is obtained as

$$\ddot{\mathbf{q}} = \mathbf{I}^{-1}(\mathbf{J}^T \boldsymbol{\lambda} + \boldsymbol{\varphi}) \tag{8.7}$$

As a result, the solution of $\boldsymbol{\lambda}$ requires at the worst $O(l^3)$ computations, whereas the joint accelerations can be solved recursively using the recursive forward dynamics algorithm of the tree-type systems presents in Chap. 6. The latter requires only $O(n)$ computations. Hence, the overall efficiency is expected to be better compared to the $O(n+l)^3$ computations. Moreover, numerical stability associated with the recursive algorithm is expected to provide more realistic behavior of the closed-loop systems under study.

8.3 Four-Bar Mechanism

Four-bar mechanism is the simplest closed-loop system, which constitutes subsystem of many robotic and industrial systems. Hence, the dynamic analysis of a four-bar mechanism is undertaken in this section. Figure 8.2 shows a four-bar mechanism (links #0 – #1 – #3 – #2 – #0). In order to analyze the four-bar mechanism using the methodology prescribed in Sect. 8.2, its equivalent tree-type representation is first obtained. As the four-bar mechanism has only one loop, it needs to be cut at one joint. Figure 8.2b, c illustrate two different methods of cutting the loops. In Fig. 8.2b, the mechanism is cut at joint 3, whereas joint 4 is cut in Fig. 8.2c. It is worth noting that both the approaches lead to two subsystems.

Referring to Fig. 8.2b, subsystem-I has two links. Hence, it will have two constrained equations of motion. However, it has three unknowns, namely, two components of the constraint forces at the cut joint denoted by $\boldsymbol{\lambda}$, and the driving torque τ_a. As a result, subsystem-I is indeterminate subsystem (Chaudhary and Saha 2007). Similarly, subsystem-II is also indeterminate as it has one constrained equation of motion with two unknowns, i.e., two components of $\boldsymbol{\lambda}$. For the solution of inverse dynamics problem one needs to combine both the indeterminate systems, resulting into three equations with three unknowns and thus making it a determinate system.

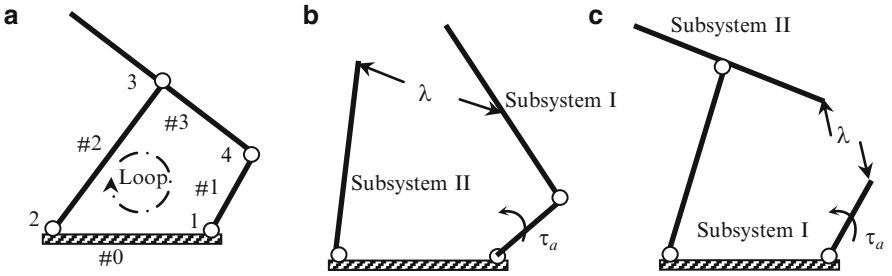


Fig. 8.2 A four-bar mechanism and its tree-type representations

On the contrary, in Fig. 8.2c, subsystem-I has one hinged link. It has one equation of motion and three unknowns, namely, two components of $\boldsymbol{\lambda}$, and τ_a. As a result, subsystem-I is indeterminate. However, subsystem-II is determinate as it has two equations of motion with two unknowns. Hence, one can solve for $\boldsymbol{\lambda}$ first. With two components of $\boldsymbol{\lambda}$ are known, the subsystem-I becomes determinate which will allow to solve for τ_a from its only constrained equation of motion. This way, a subsystem-level recursion has been formed that makes the resultant algorithm computationally very efficient (Chaudhary and Shah 2009). The following steps explain the inverse dynamics algorithm for the four-bar mechanism:

- The one independent loop (Fig. 8.2a) is identified as joints (1-2-3-4-1). Based on the loop closure equations, the kinematic constraints are obtained in the form $\mathbf{J}\dot{\mathbf{q}} = \mathbf{0}$ as

$$\begin{bmatrix} \mathbf{J}_{I,1} & \mathbf{J}_{I,2} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_I \\ \dot{\mathbf{q}}_{II} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{8.8}$$

  where $\mathbf{J}_{I,1}$, and $\mathbf{J}_{I,2}$ are of sizes 2×1 and 2×2, respectively, whereas $\ddot{\mathbf{q}}_I \equiv \ddot{q}_{1^1}$ and $\ddot{\mathbf{q}}_{II} \equiv \begin{bmatrix} \ddot{q}_{1^2} & \ddot{q}_{2^2} \end{bmatrix}^T$ are the 1- and 2-dimensional vectors.
- Using Eqs. (8.3) and (8.8), the equations of motion for the determinate subsystem-II can be written as

$$\mathbf{J}_{I,2}^T \boldsymbol{\lambda} = \boldsymbol{\tau}_{II}^{\varphi}, \text{where } \boldsymbol{\tau}_{II}^{\varphi} \equiv \mathbf{I}_{II}\ddot{\mathbf{q}}_{II} + \mathbf{C}_{II}\dot{\mathbf{q}}_{II} - \boldsymbol{\tau}_{II}^{F} \tag{8.9}$$

  In the above equation $\boldsymbol{\tau}_{II}^{\varphi}$ is known from the input joint motions and $\boldsymbol{\lambda}$ is calculated subsequently.
- Knowing the $\boldsymbol{\lambda}$ the subsystem-I becomes determinate. As a result, the driving torque is calculated using the only equation of motion which is

$$\boldsymbol{\tau}_a = \boldsymbol{\tau}_I^{\varphi} - \mathbf{J}_{I,1}^T \boldsymbol{\lambda}_1, \text{where } \boldsymbol{\tau}_I^{\varphi} \equiv \mathbf{I}_I \ddot{\mathbf{q}}_I + \mathbf{C}_I \dot{\mathbf{q}}_I - \boldsymbol{\tau}_I^{F} \tag{8.10}$$

It may be noted that $\boldsymbol{\tau}_{II}^{\varphi}$, and $\boldsymbol{\tau}_I^{\varphi}$ in Eqs. (8.9) and (8.10) were obtained using the recursive inverse dynamics algorithm, as shown in Table 6.1 of Chap. 6. In order to use the dynamic formulation presented in Chaps. 5, 6, and 7, the open architecture in Fig. 8.2 is modularized using the definition of Chap. 4. This way, the system has two modules M_1 and M_2 as shown in Fig. 8.3. The resulting joint variables are also shown in Fig. 8.3. The model parameters of the four-bar mechanism are given in Table 8.1.

For the generation of the numerical results, the input motion was provided as one complete rotation of the link $\#1^1$ with angular velocity of $3\pi/2$ rad/s. The motions for the other dependent joint angles were obtained using the loop-closure equations. The independent and dependent joint angles are shown in Fig. 8.4. Numerical values for the two component of the Lagrange multiplier $\boldsymbol{\lambda}$ are plotted in Fig. 8.5a, whereas driving torque τ_a is shown in Fig. 8.5b.

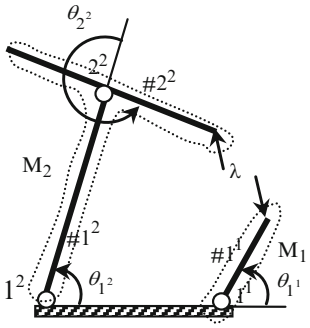


Fig. 8.3 Module architecture and the joint variables for four-bar mechanism

Table 8.1 Model parameters of the four-bar linkage

	Mass (Kg)	Length (m)
1^1	1.5	0.038
1^2	3	0.1152
2^2	5	0.2304

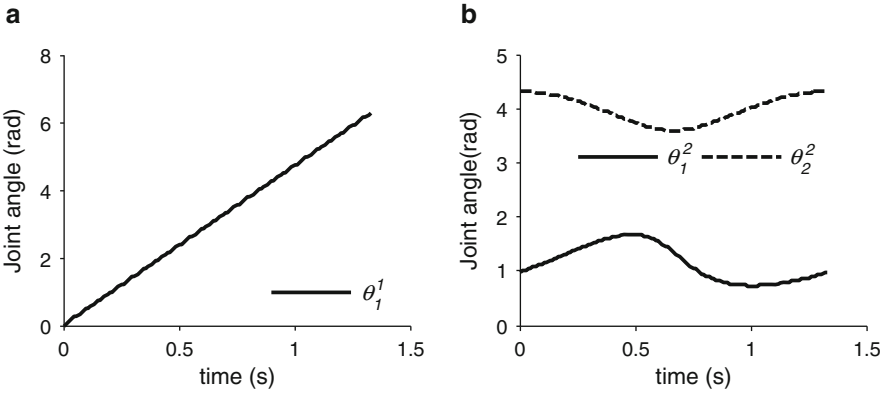


Fig. 8.4 Input joint trajectory of four-bar linkage (**a**) θ_{1^1}. (**b**) θ_{1^2} and θ_{2^2}

Next, the forward dynamics of the four-bar mechanism was performed using Eqs. (8.6) and (8.7). The torque obtained using the inverse dynamics was assumed to be the input. The joint acceleration $\ddot{\mathbf{q}}$ was numerically integrated using ode45 with relative and absolute tolerance of 10^{-5} and 10^{-7}, respectively. Simulation results for the actuated joint angle (θ_{1^1}) are plotted in Fig. 8.6a. Comparison of the simulated joint angle with the desired one, i.e., Fig. 8.4a, as shown in Fig. 8.6b depicts negligible error. Hence, the correctness of the results is validated.

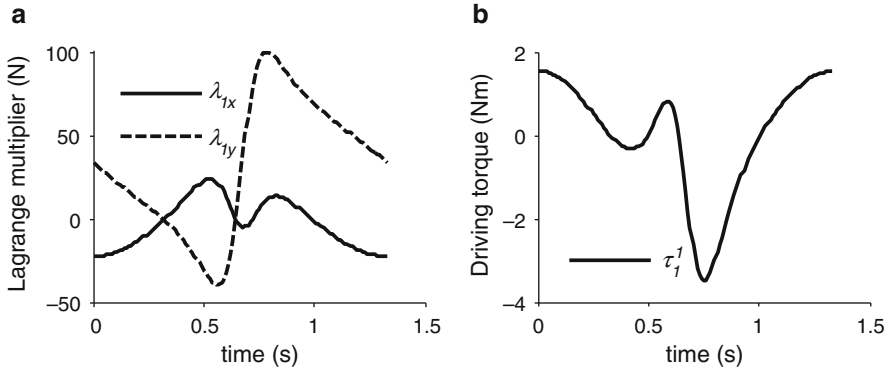


Fig. 8.5 Inverse dynamics of four-bar linkage. (**a**) Components of $\boldsymbol{\lambda}$. (**b**) Driving torque $\tau_a (\equiv \tau_{1^1})$

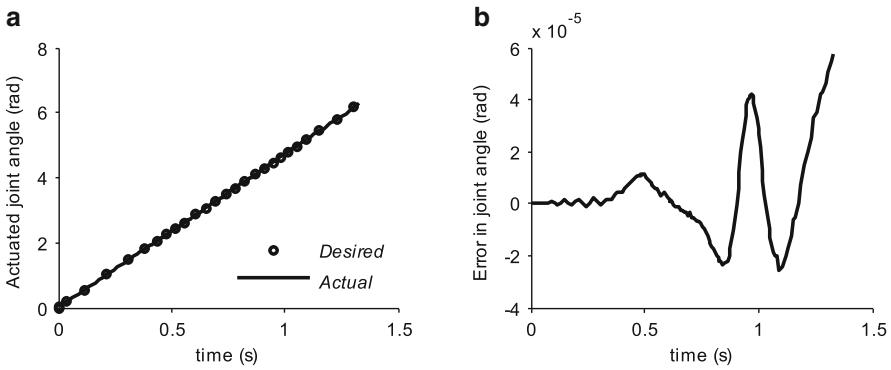


Fig. 8.6 Simulation of four-bar linkage. (**a**) Joint angle of the input link. (**b**) Error in joint angle

8.4 A Robotic Leg

A one-DOF closed-loop robotic leg (Ottaviano et al. 2004) shown in Fig. 8.7a is under taken here to demonstrate the proposed concepts to multiple-loop systems. The four-bar mechanism (#0 – #1 – #3 – #2) generates the approximate straight line which is amplified by the Pantograph mechanism (#0 – #4 – #5 – #7 – #6). The robotic leg has seven links and ten joints and, thus, it has $10 - 7 = 3$ independent loops. Hence, in order to convert it into a tree-type system it has to be opened at three joints. In order to cut the closed-loop system, its graph representation is drawn in Fig. 8.7b, where the links or bodies are indicated with circles while the lines joining the circles indicate the joints. The joints to be cut are then identified using

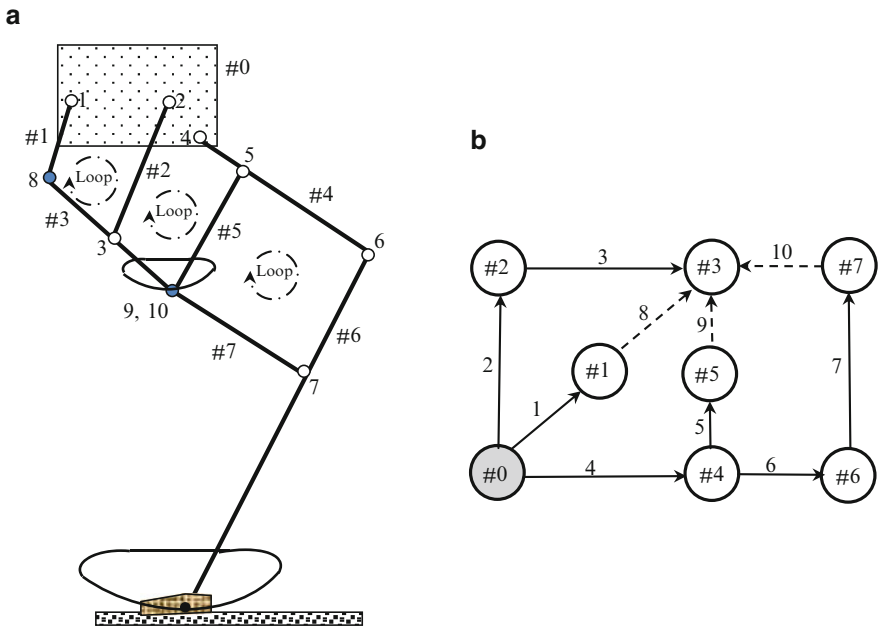


Fig. 8.7 A robotic leg and its graph representation. (**a**) Robotic leg. (**b**) Graph representation

the concept of Cumulative DOF (CDOF) provided in graph theory (Deo 1974). The CDOF is defined as the total DOF of all the joints lying between the base link (#0) and a link (say *k*th) of the system along any path. The path giving minimum CDOF is chosen as the way for cutting a close-loop system. Based on the CDOF, the graph in Fig. 8.7b is cut at joints 8, 9 and 10. The resulting tree-type system is then shown in Fig. 8.8a, where subsystem-I has one link, subsystem-II has two links and the tree-type subsystem-III has four links. The module architecture of the resulting system and associated joints variables are shown in Fig. 8.8b.

Once the tree-type system is formed, the determinate and indeterminate subsystems are identified in order to perform inverse dynamics. Referring to Fig. 8.8a, subsystem-I has only one hinged link. Hence, it will have one constrained equation of motion. However, it has three unknowns, i.e., two components of the constraint forces at the cut joint denoted by $\boldsymbol{\lambda}_1$, and the driving torque τ_a. As a result, subsystem-I is indeterminate. Similarly, subsystem-II has two constrained equations of motion with four unknowns, i.e., two components of $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ each. Hence, it is also an indeterminate subsystem. On the other hand, subsystem-III has four constrained equations of motion with four unknowns, i.e., two components of $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_3$ each. Hence, subsystem-III is determinate as the four constraint equations of motion allow one to solve for all four unknowns of the constraint forces due to the cut at the joints 9 and 10. So, the subsystem III is solved first. With four

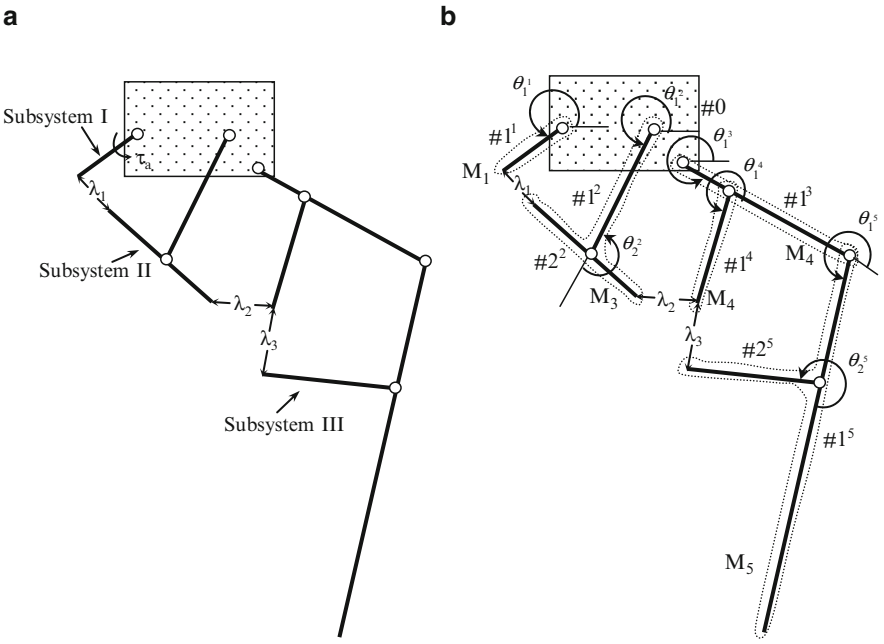


Fig. 8.8 Tree-type representation and module architecture and of the robotic leg. (**a**) Tree-type representation. (**b**) Module architecture

components of $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_3$ together known, the subsystem-II becomes determinate which then allows to solve for $\boldsymbol{\lambda}_1$ from its two constrained dynamic equations of motion. Finally, with the knowledge of the two component of $\boldsymbol{\lambda}_1$, the driving torque (τ_a) at the actuated joints can immediately be computed from the only constrained equation of motion for determinate subsystem-I. The following steps explain the inverse dynamics algorithm:

- The three independent loops (Fig. 8.7a) for the robotic leg are identified as (1-2-3-8-1), (2-3-9-5-4-2), and (4-5-6-7-10-5-4). Based on the three loop closure equations, the kinematic constraints are obtained in the form $\mathbf{J}\dot{\mathbf{q}} = \mathbf{0}$ as

$$\begin{bmatrix} \mathbf{J}_{I,1} & \mathbf{J}_{I,2} & \mathbf{O} \\ \mathbf{O} & \mathbf{J}_{II,1} & \mathbf{J}_{II,2} \\ \mathbf{O} & \mathbf{O} & \mathbf{J}_{III} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_I \\ \dot{\mathbf{q}}_{II} \\ \dot{\mathbf{q}}_{III} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{8.11}$$

  where $\mathbf{J}_{I,1}$, $\mathbf{J}_{I,2}$, $\mathbf{J}_{II,1}$, $\mathbf{J}_{II,2}$, $\mathbf{J}_{III}$ are of sizes 2×1, 2×2, 2×2, 2×4 and 2×4 respectively, whereas $\ddot{\mathbf{q}}_I \equiv \ddot{q}_{1^1}$, $\ddot{\mathbf{q}}_{II} \equiv \begin{bmatrix} \ddot{q}_{1^2} & \ddot{q}_{2^2} \end{bmatrix}^T$, and $\ddot{\mathbf{q}}_{III} \equiv \begin{bmatrix} \ddot{q}_{1^3} & \ddot{q}_{1^4} & \ddot{q}_{1^5} & \ddot{q}_{2^5} \end{bmatrix}^T$ are the 1-, 2- and 4-dimensional vectors.

Table 8.2 Model parameters of the robotic leg

	Mass (gms)	Length (mm)
1^1	10	25
1^2	20	62.5
2^2	50	125
1^3	80	200
1^4	30	75
1^5	120	300
2^5	60	150

- Using Eqs. (8.3) and (8.11), the equations of motion for the tree-type determinate subsystem-III can be written as

$$\mathbf{J}_{II,2}^T \boldsymbol{\lambda}_2 + \mathbf{J}_{III}^T \boldsymbol{\lambda}_3 = \boldsymbol{\tau}_{III}^{\varphi}, \text{ where } \boldsymbol{\tau}_{III}^{\varphi} \equiv \mathbf{I}_{III}\ddot{\mathbf{q}}_{III} + \mathbf{C}_{III}\dot{\mathbf{q}}_{III} - \boldsymbol{\tau}_{III}^{F} \tag{8.12}$$

  In the above equation $\boldsymbol{\tau}_{III}^{\varphi}$ is known from the input joint motions and $\boldsymbol{\lambda}_{III} \equiv \left[\boldsymbol{\lambda}_2^T \ \boldsymbol{\lambda}_3^T\right]^T$ is calculated subsequently.
- Knowing the $\boldsymbol{\lambda}_2$, subsystem-II becomes determinate. Hence $\boldsymbol{\lambda}_1$ is calculated from the following equation of motion:

$$\mathbf{J}_{I,2}^T \boldsymbol{\lambda}_1 = \boldsymbol{\tau}_{II}^{\varphi} - \mathbf{J}_{II,1}^T \boldsymbol{\lambda}_2, \text{ where } \boldsymbol{\tau}_{II}^{\varphi} \equiv \mathbf{I}_{II}\ddot{\mathbf{q}}_{II} + \mathbf{C}_{II}\dot{\mathbf{q}}_{II} - \boldsymbol{\tau}_{II}^{F} \tag{8.13}$$

- Knowing $\boldsymbol{\lambda}_1$ the subsystem-I becomes determinate. As a result, the driving torque is calculated using the only equation of motion which is

$$\boldsymbol{\tau}_a = \boldsymbol{\tau}_I^{\varphi} - \mathbf{J}_{I,1}^T \boldsymbol{\lambda}_1, \text{ where } \boldsymbol{\tau}_I^{\varphi} \equiv \mathbf{I}_I\ddot{\mathbf{q}}_I + \mathbf{C}_I\dot{\mathbf{q}}_I - \boldsymbol{\tau}_I^{F} \tag{8.14}$$

It may be noted that $\boldsymbol{\tau}_{III}^{\varphi}, \boldsymbol{\tau}_{II}^{\varphi}$, and $\boldsymbol{\tau}_I^{\varphi}$ in Eqs. (8.12), (8.13) and (8.14) were obtained using the proposed recursive inverse dynamics algorithm of Chap. 6. The model parameters of the robotic leg are given in Table 8.2. The input motion provided was one complete rotation of the input link #1^1 with angular velocity of $3\pi/2$ rad/s. The other dependant joint angles are obtained from loop-closure equations. The independent and dependent joint angles are shown in Fig. 8.9.

Numerical values for the Lagrange multipliers $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_3$, and the torque $\boldsymbol{\tau}_a$ at the actuated joint 1^1 are calculated using Eq. (8.12), (8.13), and (8.14), which are plotted in Fig. 8.10. Such a multi-loop mechanism was also used as a device for carpet scrapping mechanism (Saha 2003). The results of inverse dynamics for the carpet scrapping mechanism reported in Chaudhary and Saha (2009) were used here as a reference to validate the results generated for the robotic leg.

Now, the forward dynamics of the robotic leg is performed using Eqs. (8.6) and (8.7). Simulation results of the actuated joint angle (θ_{1^1}) for the robotic leg are given

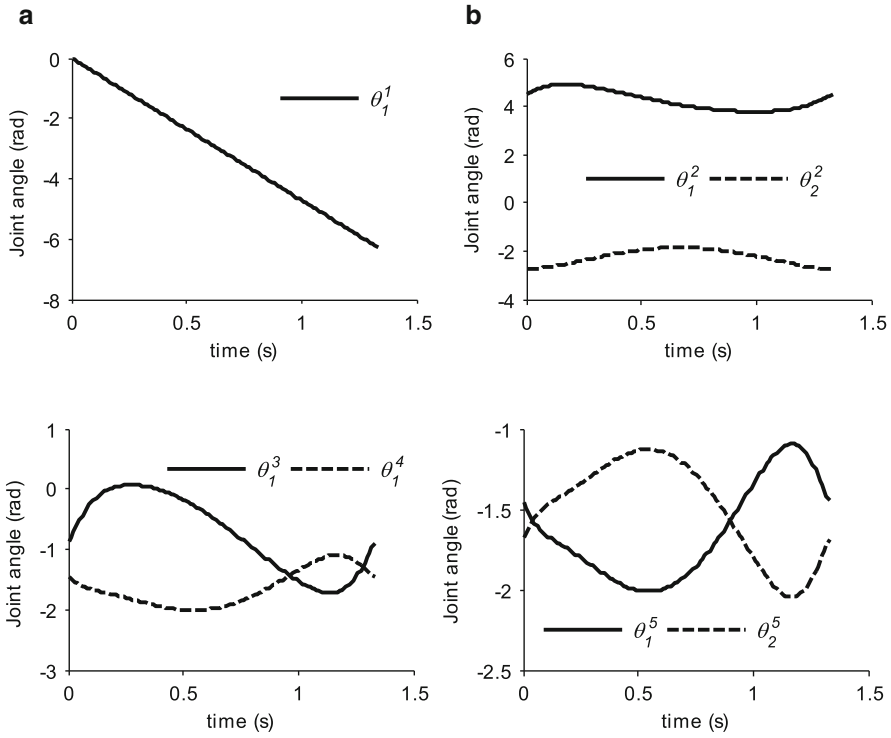


Fig. 8.9 Input joint trajectory of robotic leg. (**a**) θ_{11}. (**b**) θ_{12} and θ_{22} (**c**) θ_{13} and θ_{14} (**d**) θ_{15} and θ_{25}

in Fig. 8.11a by using the torque obtained in Fig. 8.10a as a solution of the inverse dynamics problem. Comparison of the joint angle with the desired one is also shown in Fig. 8.11b. The error between the desired and the actual joint angle is found to be small, as shown in the Fig. 8.11b. Hence, the correctness of the results is validated.

8.5 3-RRR Parallel Manipulator

Another important robotic application consisting of multiple closed-loops is a parallel manipulator. Figure 8.12 shows one such planar 3-DOF parallel manipulator, which is also popularly known as 3RRR manipulator. The 3RRR manipulator has 7 moving links, 9 joints, and 2 $(=9-7)$ independent loops, as shown in Fig. 8.12. Therefore, in order to obtain an equivalent tree-type representation, it is opened at joints 7 and 8 using the concept of minimum CDOF. The resulting tree-type system is shown in Fig. 8.12b, where subsystem-I has five link, and subsystem-II and III

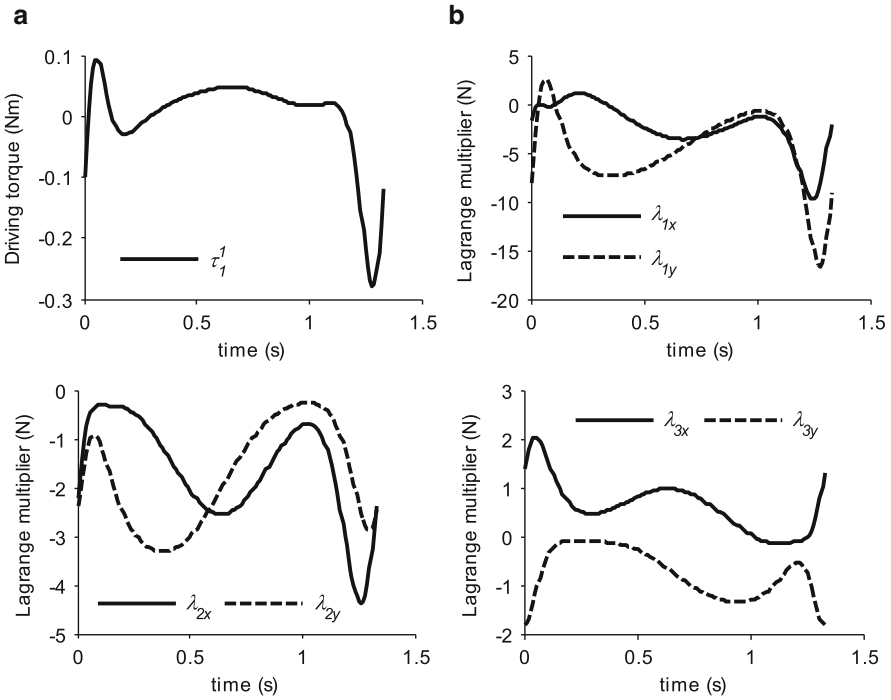


Fig. 8.10 Inverse dynamics of robotic leg. (**a**) Driving torque τ_a ($\equiv \tau_{1^1}$). (**b**) Components of $\boldsymbol{\lambda}_1$. (**c**) Components of $\boldsymbol{\lambda}_2$. (**d**) Components of $\boldsymbol{\lambda}_3$

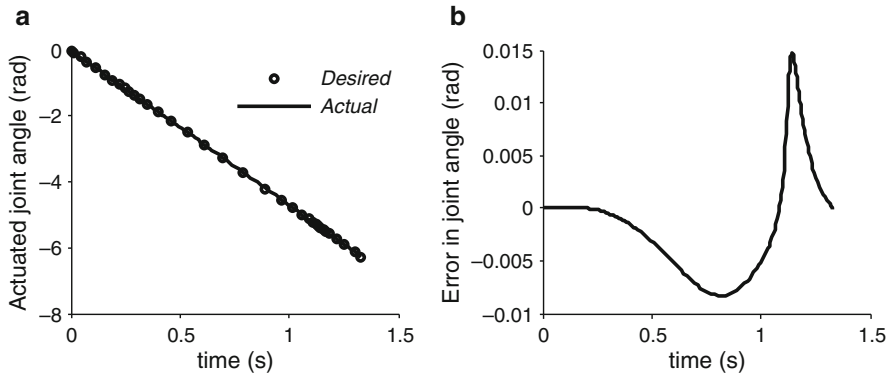


Fig. 8.11 Simulation of robotic leg. (**a**) Joint angle of the input link (**b**) Error in joint angle

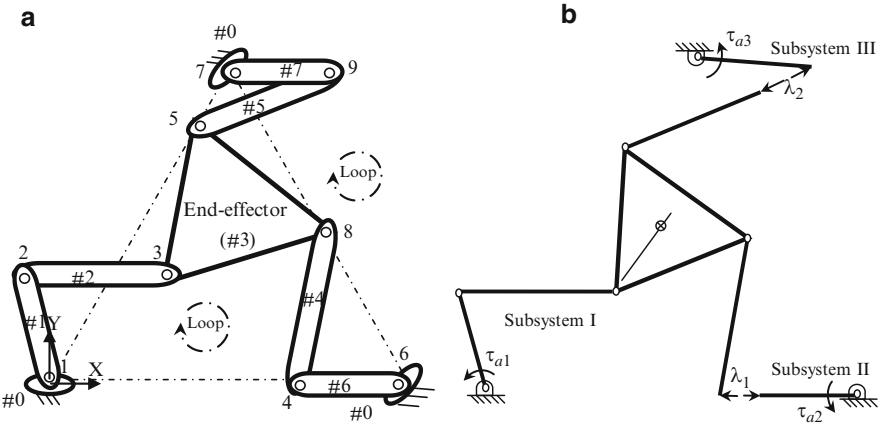


Fig. 8.12 Tree-type planar manipulator

has one link each. It is worth noting that the subsystem-I is determinate as it has five equations of motion with five unknowns (τ_{a1} and two components of $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ each). On the other hand subsystems II and III are indeterminate as both have one equation of motion with three unknown (τ_{a2} and two components of $\boldsymbol{\lambda}_1$ or τ_{a3} and two components of $\boldsymbol{\lambda}_2$). Inverse dynamics of 3RRR manipulator is solved as follows:

- The two independent loops (Fig. 8.12a) for the 3-RRR manipulator are identified as (1-2-3-8-4-6-1) and (6-4-8-5-9-7-6). Based on the two loop closure equations, the kinematic constraints are obtained in the form $\mathbf{J}\dot{\mathbf{q}} = \mathbf{0}$ as

$$\begin{bmatrix} \mathbf{J}_{I,1} & \mathbf{J}_{I,2} & \mathbf{O} \\ \mathbf{J}_{II,1} & \mathbf{O} & \mathbf{J}_{II,3} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_I \\ \dot{\mathbf{q}}_{II} \\ \dot{\mathbf{q}}_{III} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{8.15}$$

  where $\mathbf{J}_{I,1}$, $\mathbf{J}_{I,2}$, $\mathbf{J}_{II,1}$, and $\mathbf{J}_{II,3}$ are of sizes 2×5, 2×1, 2×5 and 2×1, respectively, and $\ddot{\mathbf{q}}_I \equiv \begin{bmatrix} \ddot{q}_{1^1} & \ddot{q}_{2^1} & \ddot{q}_{3^1} & \ddot{q}_{1^2} & \ddot{q}_{1^3} \end{bmatrix}^T$, $\ddot{\mathbf{q}}_{II} \equiv \ddot{q}_{1^4}$, and $\ddot{\mathbf{q}}_{III} \equiv \ddot{q}_{1^5}$ are the 5-, 1- and 1-dimensional vectors.
- Using Eqs. (8.3) and (8.15), the equations of motion for the tree-type determinate subsystem-I can be written as

$$\mathbf{J}_{I,1}^T\boldsymbol{\lambda}_1 + \mathbf{J}_{II,1}^T\boldsymbol{\lambda}_2 + \mathbf{J}_{a1}^T\tau_{a1} = \boldsymbol{\tau}_I^{\varphi}, \text{ where } \boldsymbol{\tau}_I^{\varphi} \equiv \mathbf{I}_I\ddot{\mathbf{q}}_I + \boldsymbol{C}_I\dot{\mathbf{q}}_I - \boldsymbol{\tau}_I^F \tag{8.16}$$

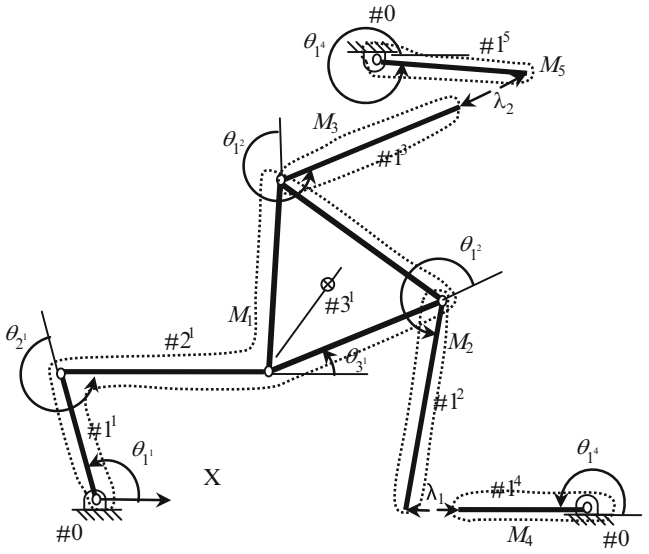


Fig. 8.13 Module architecture and joint variables for 3-RRR manipulator

where $\mathbf{J}_{a1}$ is of sizes 1×5 and $\boldsymbol{\tau}_I^{\varphi}$ is known from the input joint motions. Using the above equation five unknowns, namely, τ_{a1}, and two components of $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ each, are calculated.

- Knowing $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$, subsystem-II and III become "determinate". Hence τ_{a2} and τ_{a3} are calculated from the following equations of motion:

$$\begin{aligned}\tau_{a2} &= \boldsymbol{\tau}_{II}^{\varphi} - \mathbf{J}_{I,2}^T \boldsymbol{\lambda}_1, \text{ where } \boldsymbol{\tau}_{II}^{\varphi} \equiv \mathbf{I}_{II}\ddot{\mathbf{q}}_{II} + \mathbf{C}_{II}\dot{\mathbf{q}}_{II} - \boldsymbol{\tau}_{II}^{F} \\ \tau_{a3} &= \boldsymbol{\tau}_{III}^{\varphi} - \mathbf{J}_{II,2}^T \boldsymbol{\lambda}_2, \text{ where } \boldsymbol{\tau}_{III}^{\varphi} \equiv \mathbf{I}_{III}\ddot{\mathbf{q}}_{III} + \mathbf{C}_{III}\dot{\mathbf{q}}_{III} - \boldsymbol{\tau}_{III}^{F}\end{aligned} \tag{8.17}$$

In order to perform the inverse dynamics using the dynamic formulation presented in Chaps. 5, 6, and 7, the open architecture in Fig. 8.12b is modularized as shown in Fig. 8.13. The associated joint variables are shown in Fig. 8.13. The model parameters of the 3-RRR manipulator are given in Table 8.3.

Circular trajectory was provided to end-effector (#3). The joint trajectories were then obtained from end-effector's trajectory using inverse kinematics and shown in Fig. 8.14. Numerical values for the Lagrange multiplier $\boldsymbol{\lambda}_i$, and joint torque τ_{ai} at the actuated joints are shown in Fig. 8.15.

Forward dynamics is performed next for the free fall under gravity. The initial joint position and velocity is shown in Table 8.4. The simulated joint angles are shown in Fig. 8.16.

Table 8.3 Model parameters of the3-RRR manipulator

	Mass (Kg)	Length (m)
1^1	3	0.4
2^1	4	0.6
3^1	8	0.4[a]
1^2	4	0.6
1^3	4	0.6
1^4	3	0.4
1^5	3	0.4

[a]represents the side of equilateral triangular link

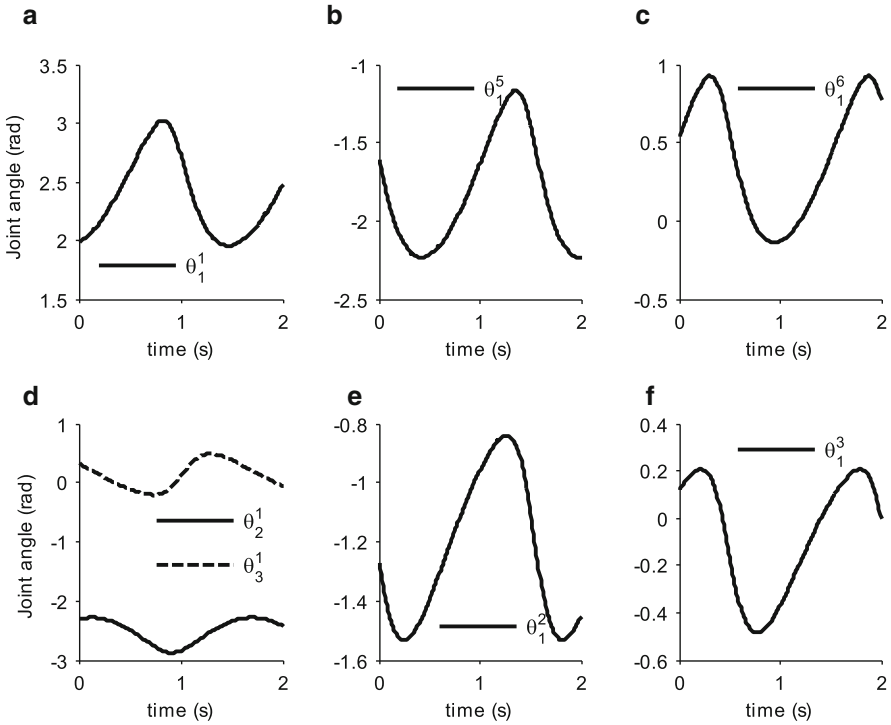


Fig. 8.14 Input joint trajectory of 3-RRR manipulator. (**a**) θ_{1^1}. (**b**) θ_{1^4}. (**c**) θ_{1^5}. (**d**) θ_{2^1} and θ_{3^1}. (**e**) θ_{1^2}. (**f**) θ_{1^3}

8.6 Summary

Dynamic analysis of closed-loop systems has been presented using the dynamic formulation presented in Chaps. 5, 6, and 7. In order to analyze closed-loop system it is opened at several joints and these joints are replaced with Lagrange multipliers.

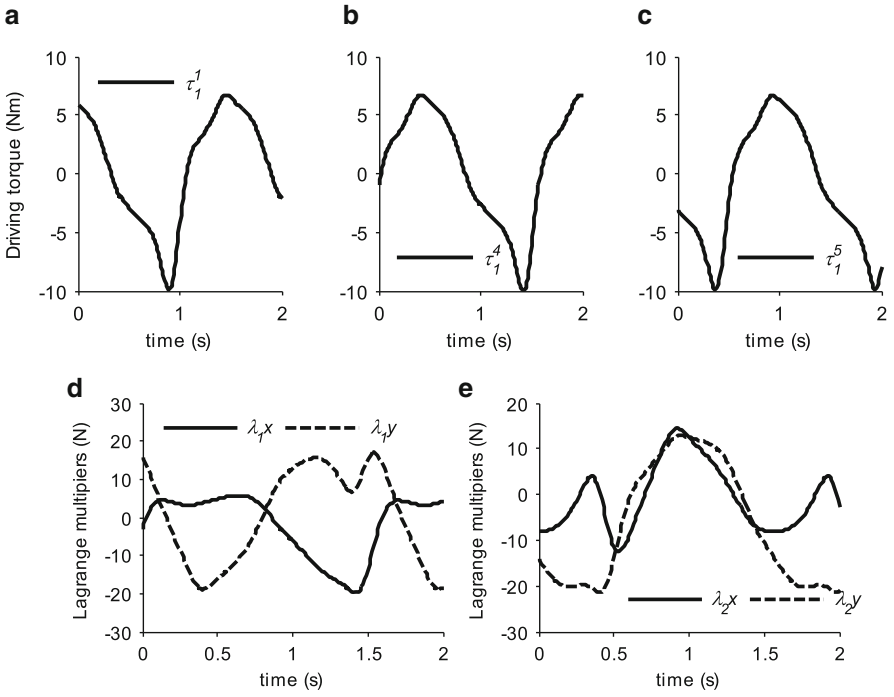


Fig. 8.15 Inverse dynamics of 3-RRR manipulator (Joint torques and Lagrange multipliers). (**a**) $\tau_{a1}(\equiv \tau_{1^1})$. (**b**) $\tau_{a2}(\equiv \tau_{1^4})$. (**c**) $\tau_{a3}(\equiv \tau_{1^5})$. (**d**) $\boldsymbol{\lambda}_2$. (**e**) $\boldsymbol{\lambda}_3$

Table 8.4 Initial condition for free fall

Joint	θ (rad)	$\dot{\theta}$ (rad/s)
1^1	1.0472	0
2^1	−0.8727	0
3^1	3.7525	0
1^2	1.3090	0
1^3	2.9322	0
1^4	4.1888	0
1^5	5.7596	0

As a result the problem of solving closed-loop system boils down to that of solving tree-type system with externally applied forces due to cutting of the joints. Several systems were analyzed for the better comprehension of the methodology.

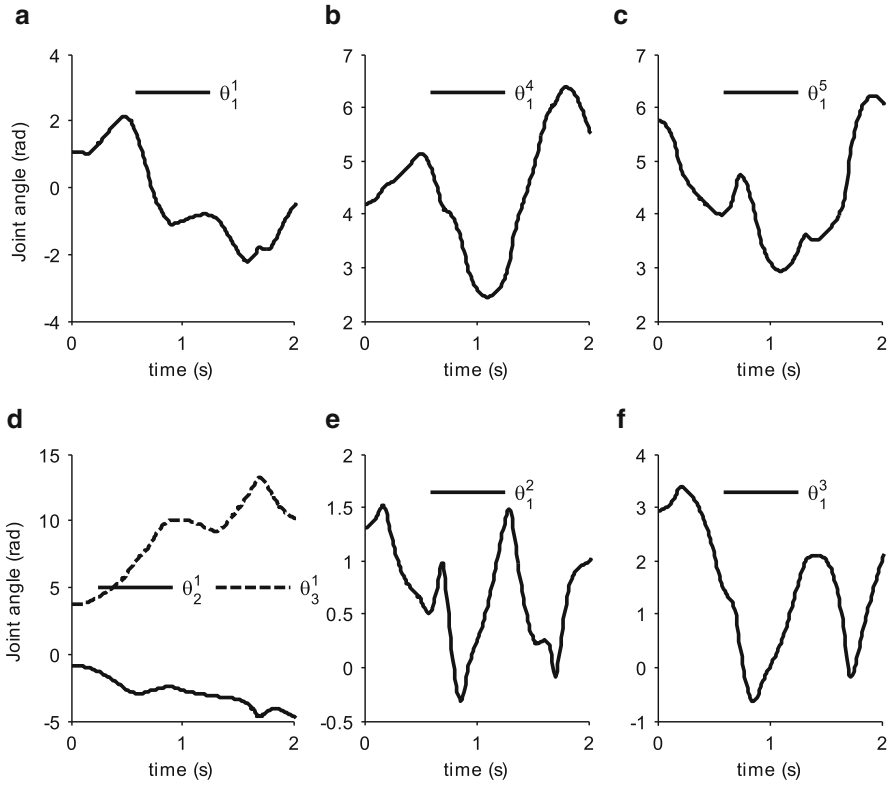


Fig. 8.16 Simulation of 3-RRR manipulator. (**a**) θ_{11}. (**b**) θ_{14}. (**c**) θ_{15}. (**d**) θ_{21} and θ_{31}. (**e**) θ_{12}. (**f**) θ_{13}

Chapter 9
Controlled Robotic Systems

A controller is an essential part of a robotic system in achieving desired motion. A Proportional Integral Derivative (PID) controller is the simplest form of controller used for this purpose. A PID controller is widely used in industries for the control of processes or machines. In a robotic system, e.g., an industrial robot, a PID controller independently controls motion of each joint ignoring the effects of the system's dynamics. However, for accomplishing complex motion or task, a PID controller does not always result into best performance, as shown by Lewis et al. (2004), Kelly et al. (2005), and Craig (2006). The legged robots discussed in Chaps. 6 and 7 are meant to perform a variety of complex tasks. As a result, the use of a PID controller without taking into account dynamics of the legged robots would result into poor control performance. On the other hand, the use of model-based controllers (Lewis et al. 2004; Kelly et al. 2005) has become popular in order to improve the performance of the conventional PID controllers. The model-based controllers work based on the information of the dynamic model of a system. If the dynamic model of a robot is not very accurate, the model-based control approach will still be able to eliminate major nonlinearities due to the robot's inertia. In this chapter, simulation of model-based control of several robotic systems will be carried out.

9.1 Model-Based Control

Since model-based controllers work well on precise information of the dynamic model of a robot, the recursive algorithms for inverse and forward dynamics proposed in Chaps. 6 and 7 are valuable in the control of robots due to their efficiency, computational uniformity, and less numerical errors produced during the dynamic computations. While the inverse dynamics algorithm helps in calculating the controlling torques of the actuators located at different joints, the forward dynamics algorithm predicts the behavior of the robot. The latter also allows real time simulation, which helps in predicting the future state of a system for

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_9,

corrective control measures. Model-based controller, e.g., computed-torque control and feedforward control, are developed next for the fixed and floating-base robotic systems.

9.1.1 Computed-Torque Control

The dynamic equations of motion obtained in Eq. (5.12) are rewritten as

$$\mathbf{I}\ddot{\mathbf{q}} + \mathbf{h} = \boldsymbol{\tau}, \text{ where } \mathbf{h} = \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}^F \tag{9.1}$$

Equation (9.1) is nonlinear as the elements of the Generalized Inertia Matrix (**I**) are nonlinear functions of the state variable **q**, and the elements of Matrix of Convective Inertia (**C**) are nonlinear functions of the state variables **q** and $\dot{\mathbf{q}}$. Hence, the use of a classical PID controller will lead to a set of nonlinear differential equations for the closed-loop system that will still have the nonlinear terms **I** and **h** as shown below:

$$\mathbf{I}\ddot{\mathbf{q}} + \mathbf{h} = \mathbf{K}_p\mathbf{e} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_I \int \mathbf{e}\, dt \tag{9.2}$$

where $\mathbf{K}_P$, $\mathbf{K}_D$ and $\mathbf{K}_I$ are diagonal matrices with constant gains on the diagonal, whereas **e** and $\dot{\mathbf{e}}$ are the vectors of the error in position and velocity, respectively. On the contrary, the computed-torque control works on the principle of feedback linearization of the nonlinear system under study. Accordingly, the controller torque $\boldsymbol{\tau}$ is given by

$$\boldsymbol{\tau} = \mathbf{I}\alpha + \mathbf{h} \tag{9.3}$$

Substituting Eq. (9.3) into Eq. (9.1) the resulting equations of motion of the closed-loop control system are represented as $\ddot{\mathbf{q}} = \boldsymbol{\alpha}$. For $\boldsymbol{\alpha}$ being the linear function of the state variables, say,

$$\boldsymbol{\alpha} = \ddot{\mathbf{q}}_{des} + \mathbf{K}_p(\mathbf{q}_{des} - \mathbf{q}) + \mathbf{K}_D(\dot{\mathbf{q}}_{des} - \dot{\mathbf{q}}) \tag{9.4}$$

the closed-loop equations are rewritten as

$$\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e} = 0 \tag{9.5}$$

where $\mathbf{e} = \mathbf{q}_{des} - \mathbf{q}$. Equation (9.10) represents the resulting close-loop control system, which is described by a set of linear differential equations. It may be shown that the origin of the closed loop system, $[\ \mathbf{e}^T \ \ \dot{\mathbf{e}}^T\ ]^T = \mathbf{0}$, is asymptotically stable (Kelly et al. 2005), i.e.,

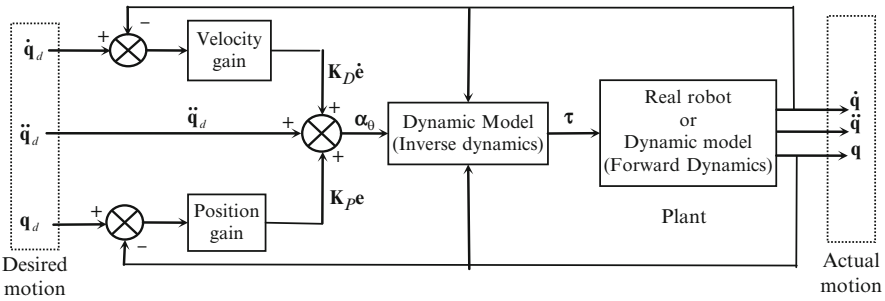


Fig. 9.1 Computed-torque control scheme

$$\lim_{t\to\infty} \mathbf{e} = \mathbf{0} \text{ and } \lim_{t\to\infty} \dot{\mathbf{e}} = \mathbf{0} \tag{9.6}$$

The resulting closed-loop control system in Eq. (9.10) is multivariable linear system where error in each joint position is governed by an independent linear differential equation. Hence, the gains $\mathbf{K}_P$ and $\mathbf{K}_D$ are chosen as $\mathbf{K}_P = diag\left[k_{p_1} \cdots k_{p_n}\right]\mathbf{K}_D = diag\left[k_{d_1} \cdots k_{d_n}\right]$. The gains k_{p_i} and k_{d_i} can be obtained by assuming critically damped response, i.e., $k_{d_i} = 2\sqrt{k_{p_i}}$. Equations (9.3) and (9.4) together form the computed torque law for fixed-base systems (Craig 2006). The schematic diagram of computed-torque control scheme is shown in Fig. 9.1. It may be noted that in Fig. 9.1, the robot can be represented by its dynamic equations of the motion.

For the floating-base system, the components of $\boldsymbol{\tau}$ associated with the generalized forces of the floating-base are zeros. Hence, Eq. (9.3) is rewritten by using Eq. (7.1) as

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_{\theta 0}^T \\ \mathbf{I}_{\theta 0} & \mathbf{I}_\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\alpha}_\theta \end{bmatrix} + \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_\theta \end{bmatrix} \tag{9.7}$$

which is further simplified as

$$\boldsymbol{\alpha}_0 = -\mathbf{I}_0^{-1}(\mathbf{I}_{\theta 0}^T \boldsymbol{\alpha}_\theta + \mathbf{h}_0) \tag{9.8}$$

$$\boldsymbol{\tau}_\theta = \mathbf{I}_{\theta 0}\boldsymbol{\alpha}_0 + (\mathbf{I}_\theta \boldsymbol{\alpha}_\theta + \mathbf{h}_\theta) \tag{9.9}$$

Substituting Eq. (9.8) into Eq. (9.9), the computed-torque control law for the floating-base system is obtained as

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ (\mathbf{I}_\theta - \mathbf{I}_{\theta 0}\,\mathbf{I}_0^{-1}\mathbf{I}_{\theta 0}^T)\boldsymbol{\alpha}_\theta + \left(\mathbf{h}_\theta - \mathbf{I}_{\theta 0}\mathbf{I}_0^{-1}\mathbf{h}_0\right) \end{bmatrix} \tag{9.10}$$

In. Eq. (9.10), $\boldsymbol{\tau}_\theta$ is the required actuator torque, whereas $\boldsymbol{\alpha}_\theta$ representing the servo part is given by

$$\boldsymbol{\alpha}_\theta = \ddot{\mathbf{q}}_{\theta\_d} + \mathbf{K}_P(\mathbf{q}_{\theta\_des} - \mathbf{q}_\theta) + \mathbf{K}_D(\dot{\mathbf{q}}_{\theta\_des} - \dot{\mathbf{q}}_\theta) \tag{9.11}$$

It is worth noting that for floating-base system the direct use of Eq. (9.7) to compute the driving torque, $\boldsymbol{\tau}_\theta$, is computationally inefficient as the equations cannot be solved recursively due to the presence of the terms $\mathbf{I}_{\theta 0}\ \mathbf{I}_0^{-1}\mathbf{I}_{\theta 0}^T$ and $\mathbf{I}_{\theta 0}\mathbf{I}_0^{-1}\mathbf{h}_0$. On the contrary, Eqs. (9.8) and (9.9) have representations similar to Eq. (7.2). As a result, they can be solved recursively using the proposed recursive inverse dynamics algorithm by substituting $\ddot{\mathbf{q}}_\theta = \boldsymbol{\alpha}_\theta$ and $\ddot{\mathbf{q}}_0 = \boldsymbol{\alpha}_0$ in Eq. (7.2).

9.1.2 Feedforward Control

In practice, any robotic system including legged robots is digitally controlled. Hence, sampling of the state variables, calculation of the joint torques, and communication of the appropriate commands to the actuators in a proper sequence form three essential tasks of a controller. For legged robots, where the system moves very fast, the above tasks have to be performed at a high rate. Out of the three tasks, computation of the joint torque is the most time consuming one. Hence, an efficient computation of the joint torques is of primary importance. In the case of known path to be followed by the robot, one may compute the joint torques in advance. These may then be stored in the controller's memory in the form of a look-up table. As a result, when the control actions need to be performed, these values are read out from the controller memory. This type of control is referred to as feedforward control and this reduces computational burden of the controller significantly. Simply forwarding the torque obtained from an inverse dynamics algorithm to the robot, as shown in Chap. 7, is the simplest form of the feedforward control. However, this suffers from several disadvantages caused by unmodeled parameters, backlash, uncertainty, external disturbances, etc. Hence, it is commonly used along with a PD controller. Thus, feedforward control scheme consists of a linear servo feedback and a feedforward of the nonlinear robot dynamic model, as shown in Fig. 9.2. It is the simplest form of the non-linear controller, which can be employed for motion control of robots. The control law for fixed-base system is given by

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{ID} + \mathbf{K}_p(\mathbf{q}_{des} - \mathbf{q}) + \mathbf{K}_d(\dot{\mathbf{q}}_{des} - \mathbf{q}) \tag{9.12}$$

whereas the same for floating-base system is given by

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_{ID} + \mathbf{K}_p(\mathbf{q}_{\theta\_des} - \mathbf{q}_\theta) + \mathbf{K}_d(\dot{\mathbf{q}}_{\theta\_des} - \mathbf{q}_\theta) \end{bmatrix} \tag{9.13}$$

where $\mathbf{K}_p \equiv diag\left[k_{p_1} \cdots k_{p_n}\right]$ and $\mathbf{K}_d \equiv diag\left[k_{d_1} \cdots k_{d_n}\right]$, and torques, $\boldsymbol{\tau}_{ID}$, are obtained from the recursive inverse dynamics algorithm. It is worth noting that the

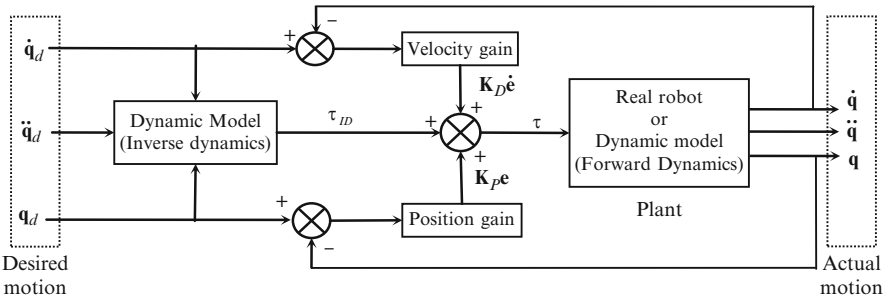


Fig. 9.2 Feedforward control scheme

feedforward control can achieve performance as good as computed-torque control with the proper tuning of the control gains. The gains can be calculated using, for example, a design procedure proposed by Kelly et al. (2005).

9.2 Biped

The computed-torque and feedforward control schemes presented in the previous sections are applied to simulate the controlled motion of the floating-base planar and spatial bipeds presented in Sect. 7.2. It may be noted that the recursive inverse dynamics algorithm proposed in Sect. 7.1.1 is used to implement the control law, whereas the recursive forward dynamics algorithm obtained in Sect. 7.1.2 is used for the simulation studies. Two control schemes are used mainly to study the effectiveness of one control scheme over the other.

9.2.1 Planar Biped

As presented in Sect. 7.2, it was observed that for the given joint torques, calculated using the inverse dynamics algorithm, the biped was able to walk for one cycle as shown in Figs. 7.7 and 7.8. However, the biped was unable to follow the desired trajectories after one cycle. This is due to zero Eigen value problem. However, in real life such deviation occurs due to unmodeled parameters, uncertainty, backlash, etc. Hence, control is inevitable in practical systems. Simulation of the controlled motion of the planar biped is presented next using the computed-torque and feedforward control schemes.

9.2.1.1 Computed-Torque Control

In order to achieve the motion of the biped using the computed-torque control scheme, the expressions in Eqs. (9.8), (9.9), and (9.11) were used. Control gains k_{p_i},

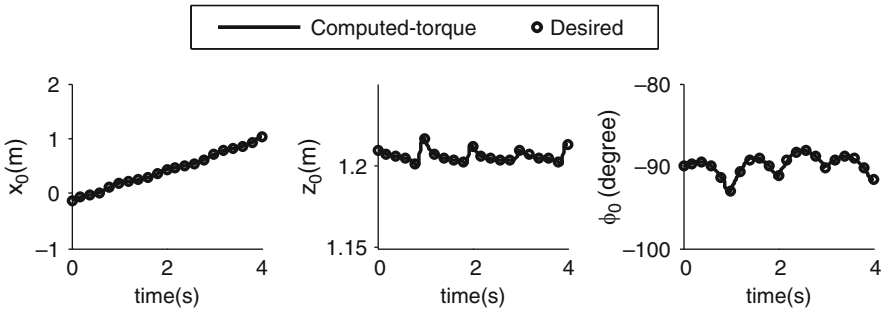


Fig. 9.3 Simulated motion of the trunk of the planar biped under computed-torque control

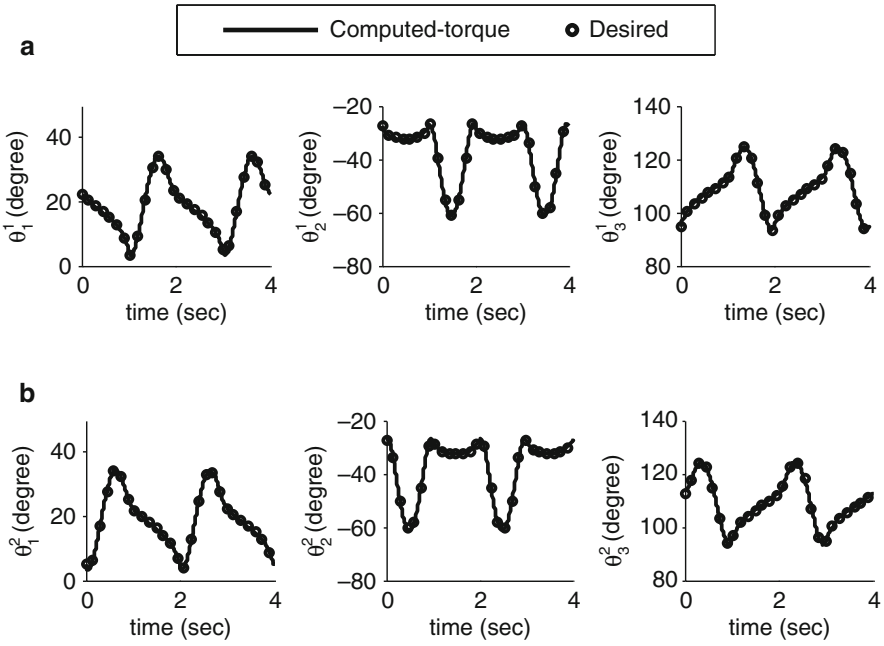


Fig. 9.4 Simulated joint motions of the planar biped under computed-torque control. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

and k_{d_i} are chosen as 49 and 14, respectively, such that $k_{d_i} = 2\sqrt{k_{p_i}}$. Joint torques are solved recursively by substituting $\ddot{\mathbf{q}}_\theta = \boldsymbol{\alpha}_\theta$ and $\ddot{\mathbf{q}}_0 = \boldsymbol{\alpha}_0$ in the inverse dynamics algorithm presented in Sect. 7.1.1. Recursive forward dynamics was then used to obtain the motions of the trunk (floating-base) and the joint angles. Figures 9.3 and 9.4 show simulated motion of the trunk and the joint angles obtained for the time span of 4 s, chosen tacitly. Comparison of the results with desired one is also shown in Figs. 9.3 and 9.4. Figure 9.3 shows that the biped moves in the forward direction

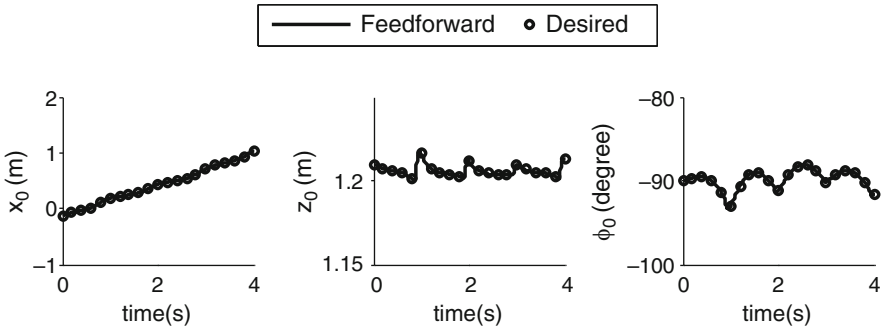


Fig. 9.5 Simulated motion of the trunk of the planar biped under feedforward control

(X) without falling for four steps. The slight variation in φ_0 is due to the assumption that the biped mass is concentrated at the trunk. The joint angles associated with the legs are also shown in Fig. 9.4. The joint angles follow the desired trajectory using computed-torque control scheme. It may be noticed that without the control action the simulated and desired trajectories was found to deviate after a time of 1 s, however in comparison, the control action ensures a close match of the trajectories (shown in Fig. 9.4) even after 4 s. The simulation was run for times more than 4 s and the same close match between the trajectories was noticed. This shows the importance and necessity of implementing control scheme for achieving desired motion of the biped robot.

9.2.1.2 Feedforward Control

Next, the feedforward control scheme was applied for the controlled simulation of the planar biped under study. For this, the joint torques, obtained from the inverse dynamics algorithm, were fed forward to the controller by using the control law given in Eq. (9.13). The gains were taken as $k_{p_i} = 49$ and $k_{d_i} = 7$. Figures 9.5 and 9.6 depict the motion of the trunk and joint angles. Simulation results in Fig 9.5 show that the biped moves in the forward direction with periodic motion. Moreover, the controller is able to follow the desired joint motion as evident form Fig. 9.6. Hence, the feedforward control scheme performs as good as computed torque control for a given set of gains.

9.2.2 Spatial Biped

Next, the computed-torque and feedforward control schemes were applied to simulate the controlled motion of the spatial biped discussed in Sect. 7.2.2. The proportional and derivative gains were taken as $k_{p_i} = 200$ and $k_{d_i} = 40$ for both

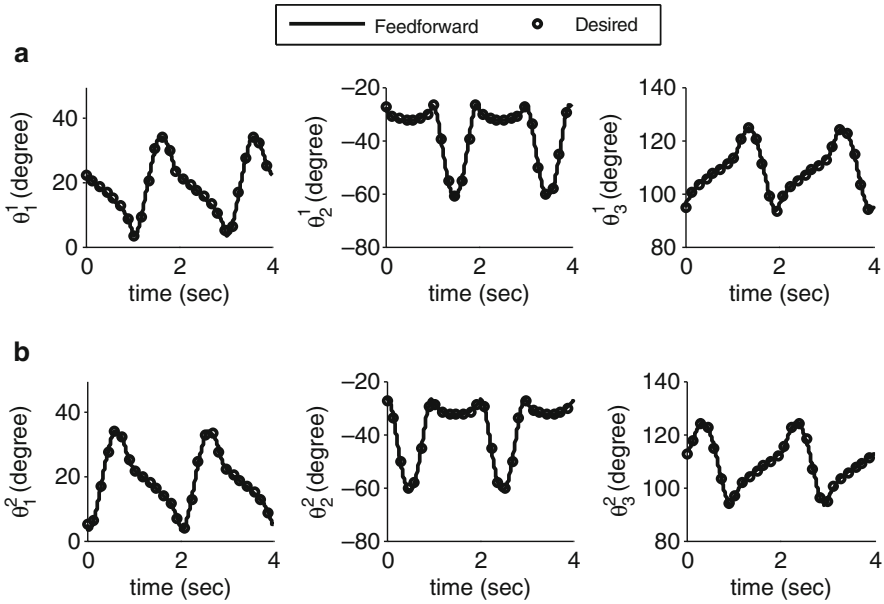


Fig. 9.6 Simulated joint motions of the planar biped under feedforward control. (**a**) Support leg (Module 1). (**b**) Swing leg (Module 2)

the control schemes. Simulated motions of the trunk and the joint angles using both the control schemes are shown in Figs. 9.7 and 9.8, respectively. The desired motions are also depicted in Figs. 9.7 and 9.8. It can be seen that the biped moves in the forward direction (X) under both the control schemes, however, feedforward control scheme performs little better than computed torque scheme, as evident from Figs. 9.7 and 9.8. It is worth noting that without the control action the simulated and desired trajectories were found to deviate after a time of 0.1 s, as shown in Figs. 7.14 and 7.15, however in comparison the control action ensures a close match of the trajectories even after 2 s. For the given set of gains feedforward scheme shows better tracking of the trajectories than the computed torque control.

9.3 Quadruped

Simulation of controlled motion of the spatial quadruped is attempted next. Dynamic simulation of quadruped was also presented in Sect. 7.3 (Figs. 7.21 and 7.22), where the torques obtained using the inverse dynamics algorithm were inputs to the actuators. The results showed that the quadruped was able to follow the trajectory for 0.8 s only before deviations occur. This is due to zero Eigen value problems. However, in real life such deviation occurs due to unmodeled parameters,

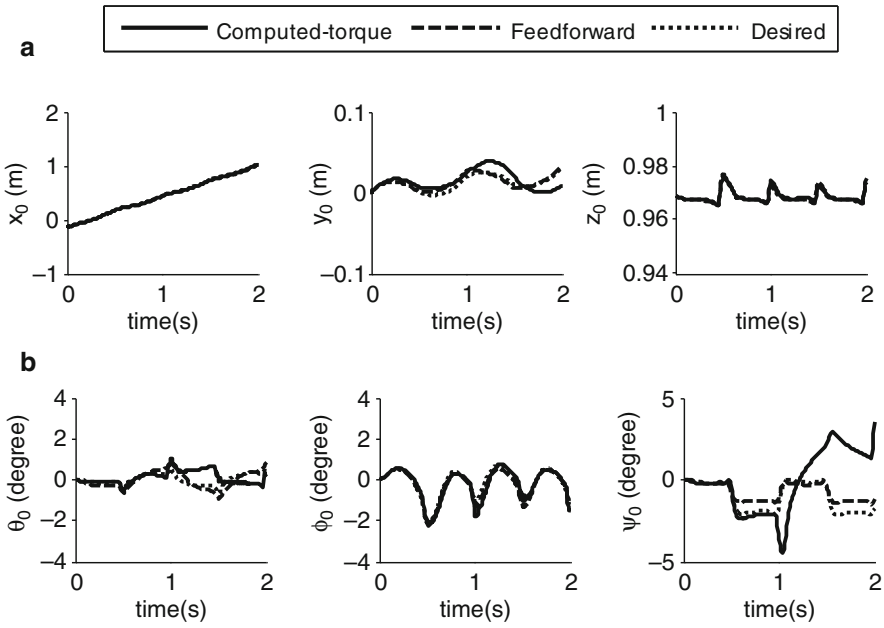


Fig. 9.7 Simulated motion of the trunk of the spatial biped under control. (**a**) Center-of-mass. (**b**) Euler Angles

uncertainty, backlash, etc. Hence, appropriate control is required. Here, the motions of the quadruped under computed-torque and feedforward control schemes are studied for the periodic walking. Values of the gains are taken as $k_{p_i} = 49$ and $k_{d_i} = 14$ for both the feedforward and computed-torque control schemes. The variations of the COM of the trunk and the Euler angles are shown in Fig. 9.9, which depict that the quadruped moves in the forward direction (X) with stable periodic motion. Figure 9.10 shows the variations of joint angles of the legs, where the quadruped follows the desired trajectories. The simulation was run for more than 4 s and the same close match between the trajectories was noticed. It is worth mentioning that for selected values of the gains, feedforward control performs little better than the computed-torque control scheme.

9.4 Hexapod

As discussed in Sect. 7.4, the dynamics plays important role in hexapod walking. In this section, simulation of hexapod walking is presented using feedforward and computed-torque control schemes. The gains were taken as $k_{p_i} = 49$ and $k_{d_i} = 14$. Hexapod was simulated for the periodic walking for the time duration of 2 s. The simulated motions of the trunk are shown in Fig. 9.11. The hexapod moves along the forward direction (X) with the velocity of 0.8 m/s. For both the control scheme, the

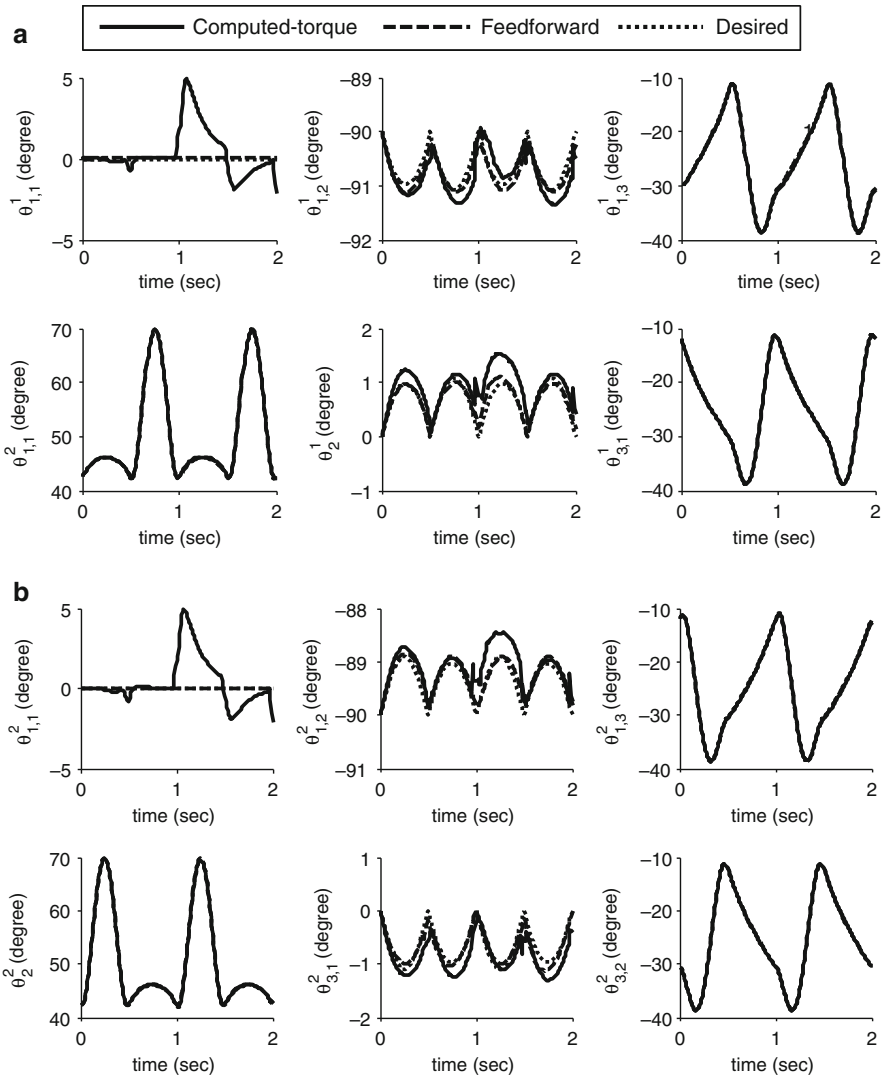


Fig. 9.8 Simulated joint motions of the spatial biped under control. (**a**)Support leg (Module 1). (**b**) Swing leg (Module 2)

Euler angles associated with the trunk are in close match with those of the desired one. Figure 9.12 shows the variation of the joint angles, which are in close match with the desired ones for both the control schemes. It is pointed out that without the control action the simulated and desired trajectories as shown in Figs. 7.28 and 7.29 was found to deviate after a time of 0.9 s, however in comparison, the control action ensures a close match of the trajectories even after 2 s.

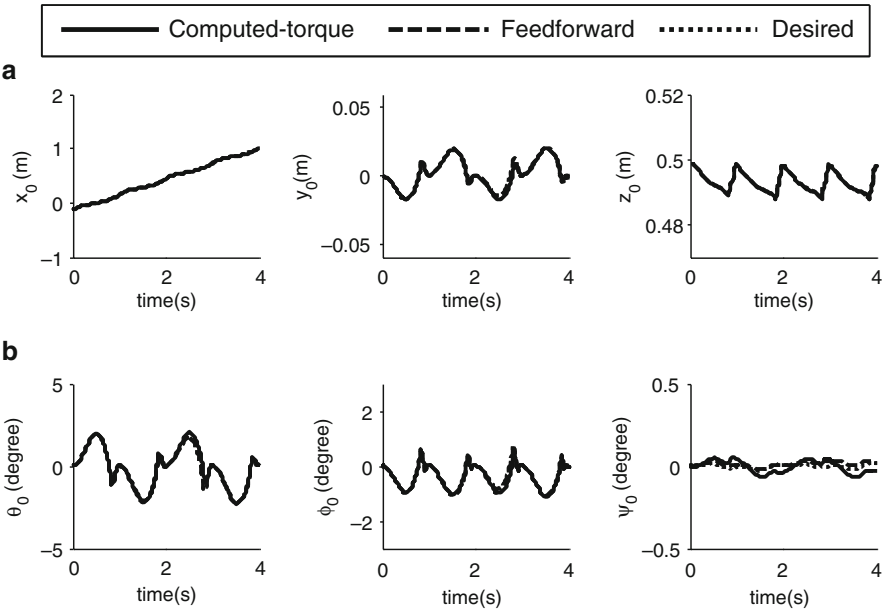


Fig. 9.9 Simulated motion of the trunk of the quadruped under control. (**a**) Center-of-mass. (**b**) Euler Angles

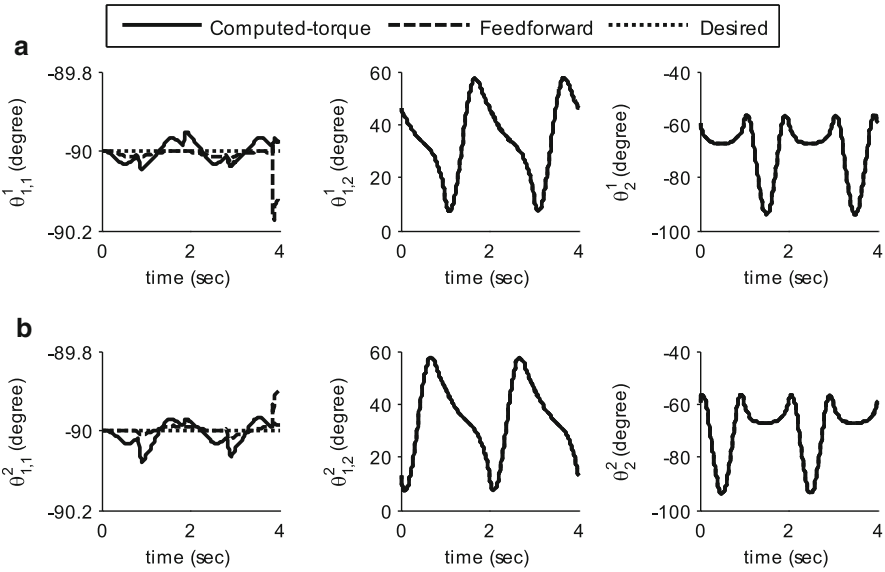


Fig. 9.10 Simulated joint motions of the quadruped under control. (**a**) Support legs (Module 1). (**b**) Swing legs (Module 2)

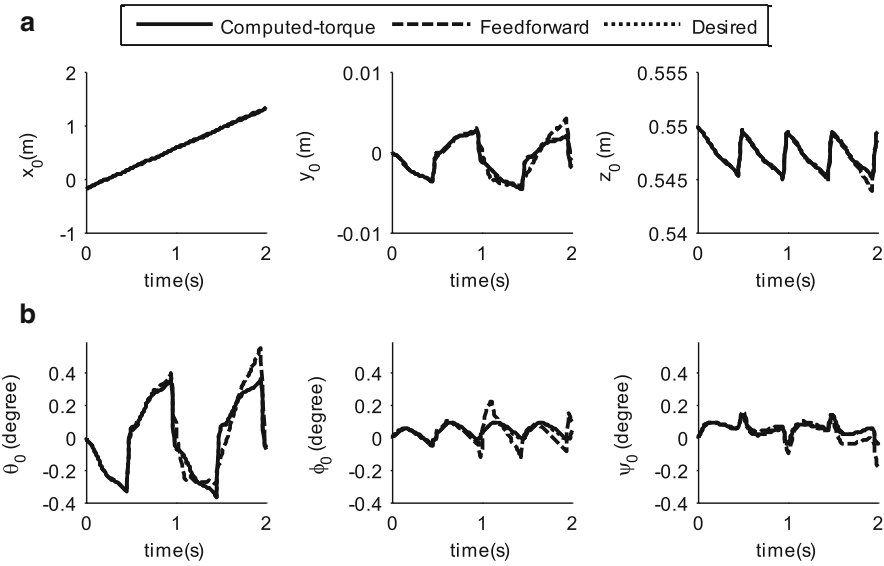


Fig. 9.11 Simulated motion of the trunk of the hexapod under control. (**a**) Center-of-mass. (**b**) Euler angles

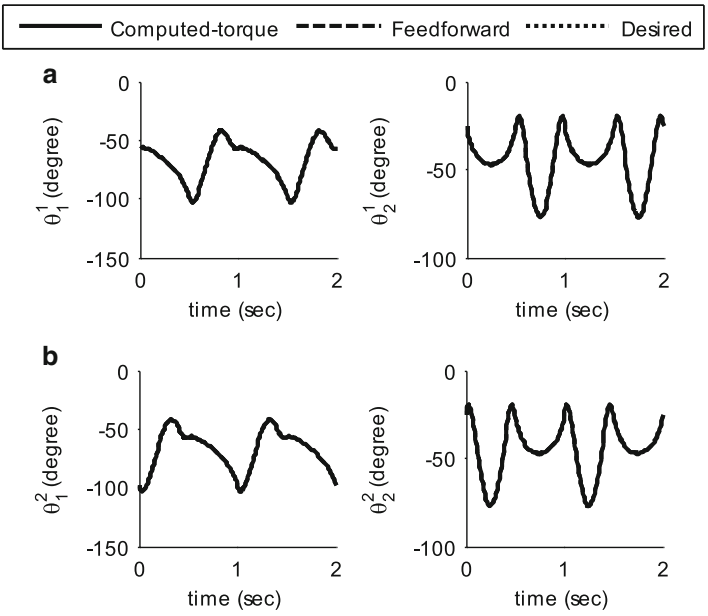


Fig. 9.12 Simulated joint motions of the hexapod under control. (**a**) Support legs (Module 1). (**b**) Swing legs (Module 2)

9.5 Summary

Two model-based control schemes, namely, the computed-torque control and the feedforward control, have been presented for the controlled motion study of several robotic systems. Simulations of the biped, quadruped and hexapod were performed. The simulation study showed that the desired trajectory was attained using both the control schemes for longer simulation time, and this was not achievable if the control schemes are not implemented. Feedforward control scheme, however, showed better tracking of the desired trajectories than computed-torque for the spatial biped and quadruped, whereas both controllers showed good tracking ability in the case of planar biped and hexapod.

Chapter 10
Recursive Dynamics Simulator (ReDySim)

Based on the recursive algorithms presented in Chaps. 6 and 7, a Recursive Dynamics Simulator (ReDySim) is developed in MatLab (2009) platform for analyses of fixed- and floating-base tree-type, and closed-loop systems. Its usage is shown in this chapter.

10.1 How to Use ReDySim?

The ReDySim can be downloaded using the link provided on the webpage http://www.redysim.co.nr/book.html. The downloaded folder needs to be extracted before actually start working using ReDySim. The folder has directory tree as shown in Fig. 10.1. The *Dynamics of Tree-type Robotic Systems* forms the root directory whose immediate folders are *Help* and *Recursive Dynamic Simulator (ReDySim*). The *Help* folder contains *Instruction Manual* and definition of the *Denavit-Hartenberg (DH) Parameters*, whereas *Recursive Dynamic Simulator (ReDySim)* contains the program modules. The user is required to have MATLAB 2009 or any higher version before start using *ReDySim*.

Note that the folder *Recursive Dynamic Simulator (ReDySim)* has two program modules, namely, *Fixed-base Systems* and *Floating-base systems*. The folder *Fixed-base systems* has further two modules *Closed-loop Systems* and *Tree-type Systems*. The modules *Closed-loop Systems*, *Tree-type Systems* and *Floating-base Systems* have directories named *Forward Dynamics* and *Inverse Dynamics*, which contain several MATLAB functions and protected files (*pcodes*), and folders containing files to be inputted for the problems solved in this book. Use of the three modules in ReDySim is illustrated in the following sections.

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7_10,

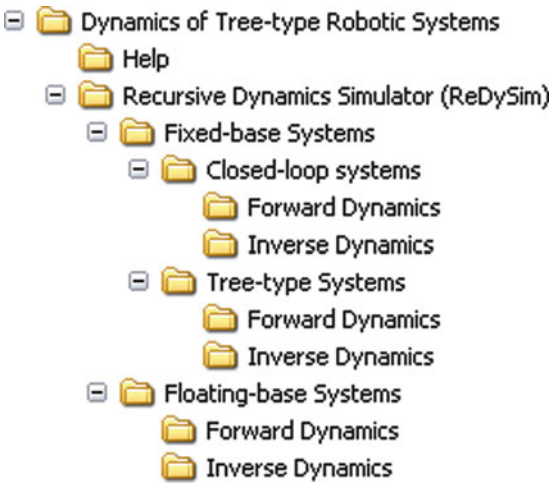


Fig. 10.1 Directory tree

10.2 Fixed-Base Systems

Analysis of fixed-base systems involves either inverse or forward dynamics. The program module for the fixed-based system is generic and can solve any arbitrary tree-type and closed-loop systems. The user inputs for both the inverse and forward dynamics are discussed in the following subsections.

10.2.1 Inverse Dynamics

Inverse dynamics attempts to find the joint torques and forces for a given set of joint motions to the system under study. In order to perform inverse dynamics, the following input parameters are required:

Model Parameters

1. Type of system (*type*), i.e., open-loop or closed-loop.
2. Degree-of-freedom of the system.
3. Number of modules (s).
4. Vector containing number of links in each module (n_s).
5. Vector containing number of joint variables associated with each joint (n_j).
6. Parent of each link (β).
7. Constant Denavit-Hartenberg (DH) parameters for revolute joints (a, α and b).
8. Vector $\mathbf{d}_k$ measured from origin O_k to the Center-of-Mass (COM) C_k of the kth link.
9. Mass of each link (m_k).
10. Inertia tensor of each link about COM and represented in body-fixed frame ($\mathbf{I}_k^C$).
11. Vector of gravitational acceleration ($\mathbf{g}$) in the inertial frame.

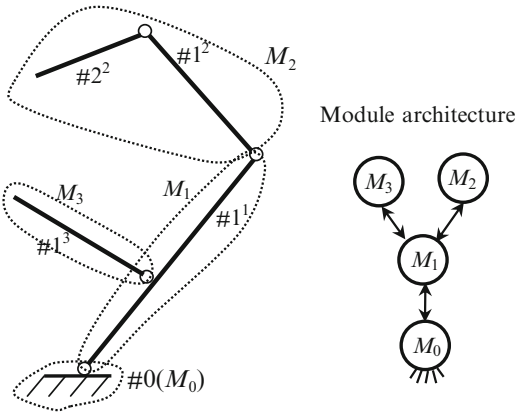


Fig. 10.2 A tree-type robotic gripper and it modularization

12. Actuated joint in the system (a_j).
13. Time period (Tp) and step size (dt).

The above values are entered in the function file named *inputs.m* and are common for both tree-type and closed-loop systems.

10.2.1.1 Tree-Type Systems

In this subsection the inputs required for the inverse dynamics of the tree-type robotic gripper of Chap. 6 is discussed. It is worth noting that the efficient implementation of the proposed recursive algorithms in ReDySim requires assignment of an integer index for each of the link in a tree-type system. As discussed in Sect. 6.1.1, an integer index for the link k^i of the ith module is obtained as

$$\text{integer index for } k^i = k + \sum_{h=1}^{i-1} \eta^h \tag{10.1}$$

where η^h represents the total number of links in the hth module. The robotic gripper, as shown in Fig. 10.2, has three modules where modules M_1 and M_3 have one link each (i.e., $\eta^1 = \eta^3 = 1$), and module M_2 has two links (i.e., $\eta^2 = 2$). Integer index corresponding to links 1^1, 1^2, 2^3 and 1^3 can then be assigned from Eq. (10.1) as

$$\begin{aligned} 1^1 &= 1 + 0 = 1 \\ 1^2 &= 1 + \eta^1 = 1 + 1 = 2 \\ 2^2 &= 2 + \eta^1 = 2 + 1 = 3 \\ 1^3 &= 1 + (\eta^1 + \eta^2) = 1 + (1 + 2) = 4 \end{aligned} \tag{10.2}$$

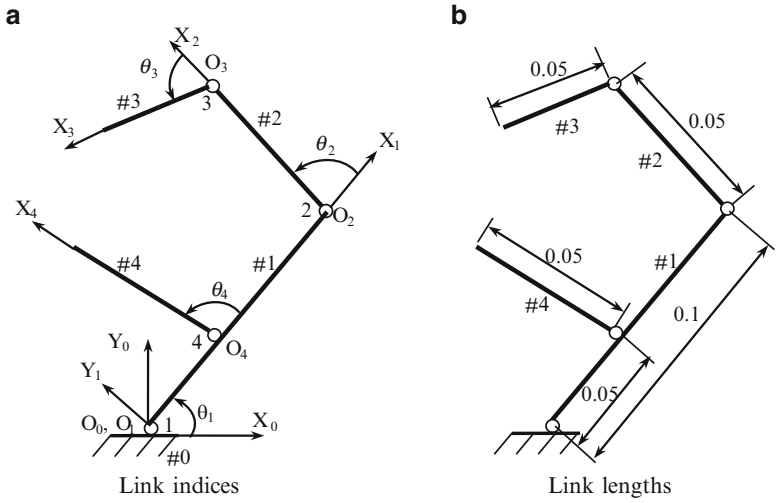


Fig. 10.3 A tree-type robotic gripper. (**a**) Link indices. (**b**) Link lengths

The resulting integer indexes are shown in Fig. 10.3. The link lengths are indicated in Fig. 10.3. It is worth noting that Eq. (10.1) ensures that each child link has as an index higher than its parent. A typical input file, i.e., *inputs.m*, for the robotic gripper is shown in Table 10.1.

Next, the user defined trajectory can be entered in the file *trajectory.m*. The cycloidal trajectory of Eq. (6.14) is used by default as this ensures zero velocities and accelerations in the beginning and at the end. The function file *trajactory.m* is shown in Table 10.2. It is worth noting that the cycloidal trajectory for each joint requires initial and final joint angles as inputs. The joint torques are then calculated by calling protected function *runinv.p*. The joint torques are stored in data file *tor.dat* and can be plotted by using the file *plot_tor.m* which are shown in Fig. 6.2. The numerical results of the joint trajectories are stored in *statevar.dat* and the same can be plotted using the file *plot_motion.m*.

Note that if a user wants to use *ReDySim* without having gone through Chaps. 4, 5, 6, and 7, i.e., the concept of kinematic modules and the resulting modular framework, number of modules can be assumed to be equal to the number of links. Therefore, the number of links in each module can be provided as one, i.e., `n_s=[1, 1, 1, 1]`, in the item 4 above. One must ensure that the index of child link is always higher than its parent. The results, i.e., torque required at each joint, are however same. One can also visualize input joint trajectory by using file *animate.m*. The file *animate.m* is not generic and need to be modified for animating different systems. For the benefit of users *animate.m* file has been provided for all the problems solved in this book.

Table 10.1 The file *inputs.m* for the fixed-base robotic gripper

```
function [type dof sn_s n_l n_j bt al alp a b dx dy dz m g Icxx Icyy
Iczz Icxy Icyz Iczx aj Tp step]=inputs()

  % SYSTEM:GRIPPER

  %1: Type of system
  type=0; %0 for open-loop and 1 for Closed-loop

  %2: Degree-of-freedom of the system
  dof=4;

  %3: Number of modules excluding fixed base(s)
  s=3;

  %4: Number of links in each module
  n_s=[1, 2, 1];
  %Total number of links
  n_l=sum(n_s);

  %5: Parent of each link (bt)
  bt=[0, 1, 2, 1];

  %6: Joint variables associated with each joint in s modules
  n_j=[1, 1, 1, 1];

  % Link lengths
  al=[0.10, 0.05, 0.05, 0.05];

  %7: Constant DH parameters, i.e., (alp, a, b)
  alp=[0, 0, 0, 0];
  a=[0, 0.10, 0.05, 0.05];
  b=[0; 0; 0; 0];

  %8: Vector di=[dxi, dyi, dzi]^T  from the origin(Ok) to Center-
  %of-mass(Ck)
  dx=[0.1/2, 0.05/2, 0.05/2, 0.05/2];
  dy=[0,     0,      0,      0];
  dz=[0,     0,      0,      0];

  %9: Mass of each link (m)
  m=[0.4, 0.2, 0.2, 0.2];

  %10: Inertia tensor (Ic) of the kth link about Center-Of-Mass (COM) in
  %kth frame, which is rigidly attached to the link
  Icxx=zeros(n,1);Icyy=zeros(n,1);Iczz=zeros(n,1);%Initialization
  Icxy=zeros(n,1);Icyz=zeros(n,1);Iczx=zeros(n,1);%Initialization
```

(continued)

Table 10.1 (continued)

```
Icxx(1)=0;Icyy(1)=(1/12)*m(1)*0.1*0.1;
Iczz(1)=(1/12)*m(1)*0.1*0.1;
Icxx(2)=0; Icyy(2)=(1/12)*m(2)*0.05*0.05;
Iczz(2)=(1/12)*m(2)*0.05*0.05;
Icxx(3)=0; Icyy(3)=(1/12)*m(3)*0.05*0.05;
Iczz(3)=(1/12)*m(3)*0.05*0.05;
Icxx(4)=0; Icyy(4)=(1/12)*m(4)*0.05*0.05;
Iczz(4)=(1/12)*m(4)*0.05*0.05;

%11: Gravitational acceleration (g) in the inertial frame
g=[0 ; -9.81; 0];

%12: Actuated joints of the equivalent open-tree
aj=[1, 1, 1, 1]; %enter 1 for actuated joints and 0 otherwise

%13: Time period (Tp) and step size (dt)
Tp=1;ts=0.01;
```

Table 10.2 The file *trajectory.m* for robotic gripper

```
function [th_ind,dth_ind,ddth_ind]=trajectory(time, dof, Tp)

% Cycloidal Trajectory (Default)
%Initial(thi) and final(thf)joint variables in radians
thi=[0, 0, 0, pi/2];
t2r=pi/180;%Degree to radian conversion factor
thf=[pi/3, 80*t2r, 80*t2r, pi/2+pi/6];

for i=1:n
  th_ind(i)=thi(i)+((thf(i)-thi(i))/Tp)*(time-
(Tp/(2*pi))*sin((2*pi/Tp)*time));
  dth_ind(i)=((thf(i)-thi(i))/Tp)*(1-cos((2*pi/Tp)*time));
  ddth_ind(i)=(2*pi*(thf(i)-thi(i))/(Tp*Tp))*sin((2*pi/Tp)*time);
end
```

10.2.1.2 Closed-Loop Systems

In contrast to an open-loop tree-type system, some of the joint variables in closed-loop systems are dependent. Hence the joint trajectories of all the joints cannot be entered independently; rather, one requires relationships between the independent and dependent joint variables. Moreover, one also requires the information of the Jacobian matrix resulting out of the loop-closure equations. In order to solve the inverse dynamics of a closed-loop system first trajectories of the independent joints are calculated using the function *trajectory.m* whereas the dependent joint trajectories and Jacobian matrix are calculated in the function file *inv_kine.m*. The *inv_kine.m* file is specific to a given system and the user is required to modify it

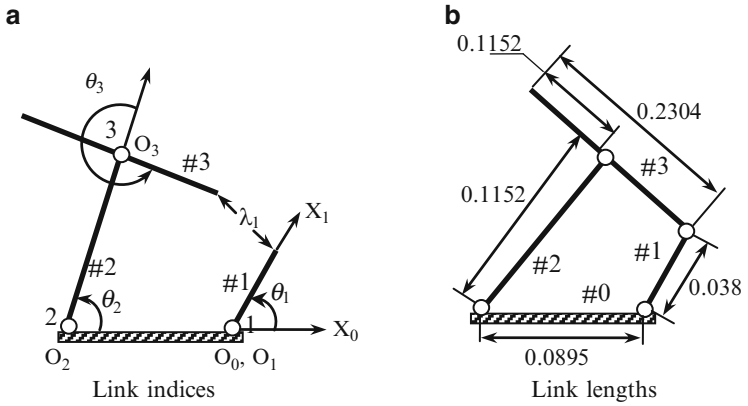


Fig. 10.4 A four-bar mechanism. (**a**) Link indices. (**b**) Link lengths

depending on the type of system to be analyzed. Figure 10.4 shows the integer indices, link lengths and joint variables of the four-bar mechanism presented in Chap. 8. The files *inputs.m*, *trajectory.m* and *inv_kine.m* for the four-bar mechanism are shown in Tables 10.3, 10.4, and 10.5.

Next, the joint torques and Lagrange multipliers at the cut opened joint are calculated by calling protected function *runinv.p*. They are stored in data file *tor.dat* and can be plotted by using function file *plot_tor.m*. They are shown in Fig. 8.5.

10.2.2 Forward Dynamics

Forward dynamics problem attempts to find the joint motions from the knowledge of external joint torques and forces. This enables simulation studies, which provide configuration of a system at hand. In order to perform forward dynamics, input parameters are provided in the file *inputs.m*. These input parameters are nothing but the model parameters, i.e., the first twelve items of Table 10.1, which are explained in Sect. 10.2.1. In addition to these input parameters, following parameters are required for the purpose of integration:

1. Initial conditions $\mathbf{y}_0(\equiv [\boldsymbol{\theta}^T \dot{\boldsymbol{\theta}}^T E_{act}]^T)$, where $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}$ and E_{act} are initial joint positions, joint rates, and actuator energy (0 by default).
2. Initial time (t_i) and final time (t_f) of simulation, and step size (*step*).
3. Relative tolerance (r_{tol}) and absolute tolerance (a_{tol}), and the type of integrator. Note that one may use either adaptive solver *ode45* (for non-stiff problem) or *ode15s* (for stiff problem) or fixed step solver *ode5* by specifying the index 0, 1 or 2, respectively.

These parameters are entered in the function file named *initials.m*.

Table 10.3 The file *inputs.m* for the four-bar mechanism

```
function [type dof s n_s n_l n_j bt al alp a b dx dy dz m g Icxx
Icyy Iczz Icxy Icyz Iczx aj Tp step]=inputs()

  % SYSTEM:GRIPPER

  %1: Type of system
  type=1; %0 for open-loop and 1 for Closed-loop

  %2: Degree-of-freedom of the system
  dof=1;

  %3: Number of modules excluding fixed base(s)
  s=2;

  %4: Number of links in each module
  n_s=[1, 2];
  %Total number of links
  n_l=sum(n_s);

  %5: Parent of each link (bt)
  bt=[0, 0, 2];

  %6: Joint variables associated with each joint in s modules
  n_j=[1, 1, 1];

  % Link lengths
  a1=0.038; %crank
  a2=0.1152; %connecting link
  a3=0.1152;%output link
  a4=0.0895;%fixed base
  al=[a1, a2, a3, a4];

  %7: Constant DH parameters, i.e., (alp, a, b)
  alp=[0, 0, 0];
  a=[0, 0, a2];
  b=[0, 0, 0];

  %8: Vector di=[dxi, dyi, dzi]^T from the origin(Ok) to Center-
  %of-mass(Ck)
  dx=[a1/2, a2/2,   a3/2];
  dy=[ 0,     0,      0];
  dz=[ 0,     0,      0];

  %9: Mass of each link (m)
  m=[1.5, 3, 5];
```

(continued)

Table 10.3 (continued)

```
   %10: Inertia tensor (Ic) of the kth link about Center-Of-Mass (COM) in
   %kth frame, which is rigidly attached to the link
   Icxx=zeros(n,1);Icyy=zeros(n,1);Iczz=zeros(n,1); % Initialization
   Icxy=zeros(n,1);Icyz=zeros(n,1);Iczx=zeros(n,1); % Initialization
Icxx(1)=(1/12)*m(1)*.01*.01; Icyy(1)=(1/12)*m(1)*al(1)*al(1);
Iczz(1)=(1/12)*m(1)*al(1)*al(1);
Icxx(2)=(1/12)*m(2)*.01*.01; Icyy(2)=(1/12)*m(2)*al(2)*al(2);
Iczz(2)=(1/12)*m(2)*al(2)*al(2);
Icxx(3)=(1/12)*m(3)*.01*.01; Icyy(3)=(1/12)*m(3)*al(3)*al(3);
Iczz(3)=(1/12)*m(3)*al(3)*al(3);

   %11: Gravitational acceleration (g) in the inertial frame
   g=[0 ; -9.81; 0];

   %12: Actuated joints of the equivalent open-tree
   aj=[1, 0, 0]; %enter 1 for actuated joints and 0 otherwise

   %13: Time period (Tp) and step size (dt)
   Tp=4/3;ts=0.01;
```

Table 10.4 The file *trajectory.m* for the four-bar mechanism

```
function [th_ind,dth_ind,ddth_ind]=trajectory(time, dof, Tp)

%Constant angular velocity
omega=45;
dth_ind(1:dof)=omega*2*pi/60;
ddth_ind(1:dof)=0;
th_ind(1:dof)=dth_ind(1:dof)*time;
```

10.2.2.1 Tree-Type Systems

For the tree-type gripper shown in Fig. 10.3, the input parameters are provided in Table 10.1 whereas the initial conditions are provided in the file *initials.m* as shown in Table 10.6.

The joint torques, which are input for the simulation study, can be entered in the function file *torque.m*. Their default values are zeros. There is also a commented subroutine for PD control in *torque.m* file where a user may test control performance of the system for the different gain values. Moreover, one may also write his/her own control algorithm in function *torque.m* as it has both position and velocity feedback. The function file *torque.m* is shown in Table 10.7. Finally, simulation is performed by running protected function file *runfor.p*. The output joint motions are stored in data file *statevar.dat* whereas the time history is stored in *timevar.dat*. The joint motions can be plotted by using the function *plot_motion.m*. The simulated motion is shown in Fig. 6.3. Moreover, the total energy can be calculated by running the file

Table 10.5 The function *inv_kine.m* for the four-bar mechanism

```
function [th dth ddth J]=inv_kine(th_ind, dth_ind, ddth_ind)

% Four-bar mechanism
[n dof type alp a b bt dx dy dz m g Icxx Icyy Iczz Icxy Icyz
Iczx aj al]=inputs();

a1(1)=al(1); a2=al(2);a31=al(3);a12=al(4);
th_f=pi; % angle with fixed link

%Independent joint motions th1, dth1 and ddth1
th(1,1)=th_ind;
dth(1)=dth_ind;
ddth(1)=ddth_ind;

%Computation of the dependent joint angles th2 and th3
alp1=th(1);
p1=a1*cos(alp1)-a12*cos(thf1);
q1=a1*sin(alp1)-a12*sin(thf1);
x1=(q1/a1);
y1=(p1/a1);
z11=(p1*p1+q1*q1+a31*a31-a2*a2)/(2*a31*a1);
alp3=2*atan2(x1-sqrt(x1*x1+y1*y1-z11*z11),y1+z11);
z12=(p1*p1+q1*q1+a2*a2-a31*a31)/(2*a2*a1);
alp2=2*atan2(x1+sqrt(x1*x1+y1*y1-z12*z12),y1+z12);
th(2,1)=alp2;
th(3,1)=(2*pi+alp3)-th(2);

%Jacobian J: I*dthh+C*dth=tau+J'*lamda and J*dth=0
J11=[a1*sin(alp1)
     -a1*cos(alp1) ];
J121= [-a2*sin(alp2)-a31*sin(alp3)
       a2*cos(alp2)+a31*cos(alp3) ];
J122= [-a31*sin(alp3)
       a31*cos(alp3)];
J12=[J121 J122];
J=[J11 J12];

%Computation of the dependent joint velocities dth2 and dth3
c1=J12\J11;
dthd=-c1*dth(1);
dth=[dth(1)
     dthd];

%Computation of the dependent joint acceleration ddth2 and ddth3
dalp1=dth(1);
dalp2=dth(2);
dalp3=dth(2)+dth(3);
```

(continued)

Table 10.5 (continued)

```
dJ11=[a1*cos(alp1)*dalp1
      a1*sin(alp1)*dalp1];
dJ121= [-a2*cos(alp2)*dalp2-a31*cos(alp3)*dalp3
        -a2*sin(alp2)*dalp2-a31*sin(alp3)*dalp3 ];
dJ122= [-a31*cos(alp3)*dalp3
        -a31*sin(alp3)*dalp3];

dJ12=[dJ121 dJ122];
c2=-dJ11*dth(1)-dJ12*dthd-J11*ddth(1);
ddthd=J12\c2;
ddth=[ddth(1)
      ddthd];

%Indepedent and depedent joint motions
th=th';dth=dth';ddth=ddth';
```

Table 10.6 The file *initials.m* for the robotic gripper

```
function[y0,len_sum,ti,tf,incr,rtol,atol,step,int_type]=initials()

  %SYSTEM:GRIPPER
  %1:Initial Conditions:State variables qand dq,and actuator
energy E_act
  th=[0,0,0]
  dth=[0,0,0,0];
  E_act=0;
  y0=[th,dth,E_act];

  %2:Simulation time and steps
  ti=0;%Initial time
  tf=3;%Final time
  step=0.01;%Sampling time for adaptive solver and step size
  for fixed step solver

  %3:Integration tolerances
  rtol=1e-3;    %relative tolerance in integration
  atol=1e-6;    %absolute tolerances in integration
  int_type=0;    %0 for ode45,1 for ode15s,2 for ode5
```

energy.p and plotted using *plot_en.m*. This can be used for the purpose of validating the simulation results. The system can also be animated using the file *animate.m*.

10.2.2.2 Closed-Loop Systems

In the case of closed-loop systems the Jacobian matrix and its time derivative are also inputted in addition to model parameters and initial conditions. For the four-bar

Table 10.7 The file *torque.m* for the robotic gripper

```
function [tau_d]=torque(time,dof,tf,th,dth)

%Driving Torque
% 1 Free-fall under gravity (Default)
tau_d=zeros(dof,1);%For free fall under gravity

% %2 Controlled simulation: PD control
% %Desired trajectories
% [thd,dthd,ddthd]=trajectory(time,dof,tf);

% kp=49; kd=14;
% tau_d=kp*(thd(1:n)'-th(1:n))+kd*(dthd(1:n)'-dth(1:n));
```

Table 10.8 The file *initials.m* for the four-bar mechanism

```
function [y0, ti, tf, incr, rtol, atol, step, int_type]=initials()

  %1: Initial Conditions: State variables q and dq, and actuator
energy E_act
  th=[0  0.9844  4.3144];
  dth=[4.7124  1.4045 -0.0000];
  E_act=0;
  y0=[th, dth, E_act];

  %2: Simulation time and steps
  ti=0; % Initial time
  tf=1.33; % Final time
  step=0.01; %Sampling time for adaptive solver and step size
  for fixed step solver

  %3: Integration tolerances
  rtol=1e-5;    %relative tolerance in integration
  atol=1e-7;    %absolute tolerances in integration
  int_type=0;    %0 for ode45, 1 for ode15s, 2 for ode5
```

mechanism shown in Fig. 10.4 the model parameters are provided in Table 10.3, and the initial conditions and the Jacobian matrix are provided in the file *initials.m* and *jacobian.m* in ReDySim as shown in Tables 10.8 and 10.9.

Similar to tree-type gripper, the torques can be entered in the function file *torque.m* and the simulation is carried out by running function *runfor.p*. Once again the output joint motions are stored in data file *statevar.dat* where the time history is stored in *timevar.dat*. The joint motions can be plotted by using function file *plot_motion.m*, whereas the total energy can be plotted using files *energy.p* and *plot_en.m*. The four-bar mechanism can be animated using the file *animate.m*.

Table 10.9 The file *jacobian.m* for the four-bar mechanism

```
function [J, dJ]=jacobian(th,dth)

[n dof type alp a b bt dx dy dz m g Icxx Icyy Iczz Icxy Icyz
Iczx aj al]=inputs();

%Link length
a1(1)=al(1); a2=al(2);a31=al(3);

% Jacobian
alp1=th(1);
alp2=th(2);
alp3=th(2)+th(3)-2*pi;

J11=[a1*sin(alp1)
     -a1*cos(alp1)
     ];
J121= [-a2*sin(alp2)-a31*sin(alp3)
       a2*cos(alp2)+a31*cos(alp3)
       ];
J122= [-a31*sin(alp3)
       a31*cos(alp3)
       ];
J12=[J121 J122];
J=[J11 J12];

%Derivative of the Jacobian
dalp1=dth(1);
dalp2=dth(2);
dalp3=dth(2)+dth(3);

dJ11=[a1*cos(alp1)*dalp1
       a1*sin(alp1)*dalp1
       ];
dJ121= [-a2*cos(alp2)*dalp2-a31*cos(alp3)*dalp3
        -a2*sin(alp2)*dalp2-a31*sin(alp3)*dalp3
         ];
dJ122= [-a31*cos(alp3)*dalp3
        -a31*sin(alp3)*dalp3
         ];

dJ12=[dJ121 dJ122];
dJ=[dJ11 dJ12];
```

10.3 Floating-Base Systems

In the cases of floating-base systems, the base is assumed to be floating and the interaction of the link with environment is modeled as a contact problem. The contact points on a link may vary depending on the shape of the link. In contrast to the modules of fixed-base systems, the same for floating-base systems are specific to examples of planar and spatial biped, quadruped and hexapod discussed in Chap. 7. This is due to the complexity involved in contact modeling. However, any system can be solved if each link is assumed to be a slender rod. The user inputs for inverse and forward dynamics for floating-base tree-type systems are discussed in the following subsections.

10.3.1 Inverse Dynamics

Inverse dynamics analysis requires input parameters similar to the model parameters i.e., items 2 to 11 of Table 10.1. In addition to the input parameters, the parameters for the ground model discussed in Appendix D are also required. They are listed below:

Ground Parameters

1. Spring stiffness (k), over-damping factor (odf), and maximum velocity of the foot tip (v_{zmax}) to represent ground model in vertical direction.
2. Co-efficient of the friction (μ) and damping co-efficient (c_h) to represent the ground model in horizontal direction.
3. A vector containing information of gravitational acceleration ($\mathbf{g}$).

Figure 10.5 shows the equivalent link indices, joint variables and link lengths of the floating base biped discussed in Sect. 7.2.1. A typical input file for the biped is shown in Table 10.10.

As discussed in Chap. 7, inverse dynamics of a floating-base system requires solution of the base acceleration. This calls for additional initial parameters for integrating the base accelerations. They are as follows:

1. Initial conditions $\mathbf{y}_0 \equiv [\mathbf{q}_0^T \ \dot{\mathbf{q}}_0^T \ E_{act} \ E_{gr}]^T$, where $\mathbf{q}_0, \dot{\mathbf{q}}_0, E_{act}$ and E_{gr} are initial position and velocity associated with the floating-base, actuator energy, and ground energy, respectively.
2. Initial time (t_i) and final time (t_f) of simulation and step size ($step$).
3. Relative tolerance (r_{tol}) and absolute tolerance (a_{tol})

The initial conditions for the floating-base planar biped are entered into the file *initials.m*, as shown in Table 10.11. The trajectory of a biped is entered in the file *trajectory.m* as discussed in Appendix B. The joint torques are then obtained by running the file *runinv.p*. The joint torques are stored in the result file *tor.dat* whereas the data of joint trajectories are stored in *statevar.dat*. The output torques, can be seen using the filr *plot_tor.m*, whereas the trajectories can be animated using *animate.m*.

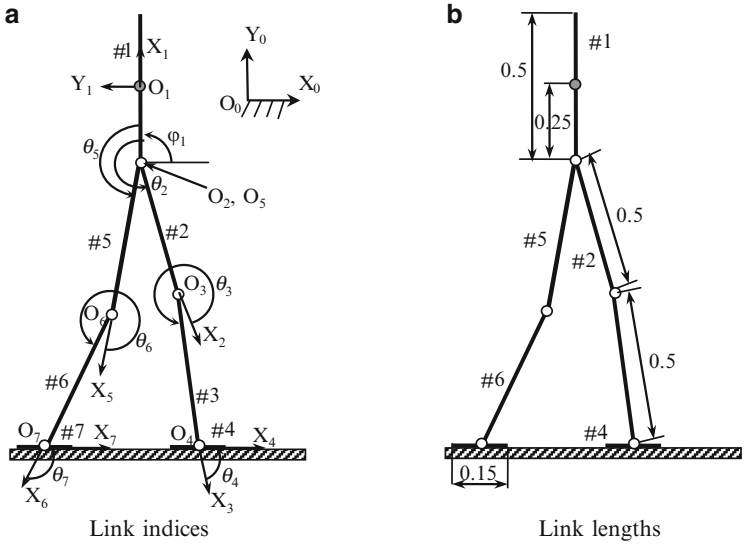


Fig. 10.5 A floating-base planar biped. (**a**) Link indices. (**b**) Link lengths

Table 10.10 The file *inputs.m* for the floating-base planar biped

```
function[dof s n_s n_l n_j bt al alp ab dx dy dz m g Icxx Icyy
Iczz Icxy Icyz Iczx g_prop]=inputs()

  %SYSTEM: Planar Biped
  %1: Degree-of-freedom of the system
  dof=12;

  %2: Number modules including floating base(s)
  s=3;

  %3: Number of links in each module
  n_s=[1, 3, 3]
  %Total number of links
  n_l=sum(n_s);

  %4: Parent module of each link (bt)
  bt=[0, 1, 2, 3, 1, 5, 6];

  %5: Joint variables associated with each joint in s modules
  n=[6, 1, 1, 1, 1, 1, 1];

  %Link lengths
  al=[0.5, 0.5, 0.5, 0.15, 0.05, 0.5, 0.15,];
```

(continued)

Table 10.10 (continued)

```
%6: Constant DH parameters, i.e., (alp, a, b)
alp=[0, 0, 0, 0, 0, 0, 0];
a=[0, 0.25, 0.5, 0.5, 0.25, 0.5, 0.5];
b=[0, 0, 0, 0, 0, 0, 0];

%7: Vector di=[dxi, dyi, dzi]^T form the origin(Ok) to Center-of-
%mass(Ck)
dx=[0,  0.5/2, 0.5/2, 0,  0.5/2, 0.5/2, 0];
dy=[0,  0,     0,     0,  0,     0,     0];
dz=[0,  0,     0,     0,  0,     0,     0];

%8: Mass of each link(m)
m=[5, 1, 1, 0.2, 1, 1, 0.2];

%9: Inertia tensor (Ic) of the kth link about Center-Of-Mass
(COM) in
%kth frame, which is rigidly attached to the link
Icxx=zeros(n,1);Icyy=zeros(n,1);Iczz=zeros(n,1);%Initialization
Icxy=zeros(n,1);Icyz=zeros(n,1);Iczx=zeros(n,1);%Initialization
Icxx(1)=(1/12)*0.1*0.1; Icyy(1)=(1/12)*m(1)*(0.5*0.5+0.1*0.1);
Iczz(1)=(1/12)*m(1)*0.5*0.5;
Icxx(2)=0; Icyy(2)=(1/12)*m(2)*0.5*0.5;
Iczz(2)=(1/12)*m(2)*0.5*0.5;
Icxx(3)=0; Icyy(3)=(1/12)*m(3)*0.5*0.5;
Iczz(3)=(1/12)*m(3)*0.5*0.5;
Icxx(4)=0; Icyy(4)=(1/12)*m(4)*0.15*0.15;
Iczz(4)=(1/12)*m(4)*0.15*0.15;
Icxx(5)=0; Icyy(5)=(1/12)*m(5)*0.5*0.5;
Iczz(5)=(1/12)*m(5)*0.5*0.5;
Icxx(6)=0; Icyy(6)=(1/12)*m(6)*0.5*0.5;
Iczz(6)=(1/12)*m(6)*0.5*0.5;
Icxx(7)=0; Icyy(7)=(1/12)*m(7)*0.15*0.15;
Iczz(7)=(1/12)*m(7)*0.15*0.15;

%10: Gravitational acceleration(g) in the inertial frame
g=[0 ; -9.81; 0];

%11: Ground parameters
%Parameters for vertical reaction
k=2000; odf=10;
cv=odf*2*sqrt(K*sum(m));
vz_max=4.5;
%Parameters for horizontal reaction
mu=.7;ch=1000;
gr=[2, 1, 3];
g_prop=[K, Cv, vz_max, mu, ch, gr];
```

Table 10.11 The file *initials.m* for the floating base planar biped

```
function [y0, ti, tf, incr, rtol, atol, step,
int_type]=initials()

  %1: Initial Conditions: State variables q0 and dq0, actuator
and ground energy
  s=0;
  [th dth ddth Xh Yh Zh dXh dYh dZh]=trajectory(s);
  %Base motions
  q0=[Xh, Yh, Zh, (0/180)*pi, (0/180)*pi, (0/180)*pi];
  dq0=[dXh, dYh, dZh, 0, 0, 0];
  E_act=0; E_gr=0;

  yo=[q0, dq0, E_act, E_gr];

  %2: Simulation time and steps
  ti=0; % Initial time
  tf=3; % Final time
  step=0.01; %Sampling time for adaptive solver and step size
for fixed step solver

  %3: Integration tolerances
  rtol=1e-4;    %relative tolerance in integration
  atol=1e-6;    %absolute tolerances in integration
  int_type=0;    %0 for ode45, 1 for ode15s, 2 for ode5
```

10.3.2 Forward Dynamics

In order to perform the forward dynamics, input parameters are provided in the file *inputs.m*. These input parameters are the same as discussed in the case of inverse dynamics of floating-base system, as explained in Sect. 10.3.1. In addition to the input parameters, following parameters are also required in the file *initials.m* for the purpose of integration:

1. Initial conditions $\mathbf{y}_0 \equiv [\mathbf{q}^T \ \dot{\mathbf{q}}^T \ E_{act} \ E_{gr}]^T$, where $\mathbf{q}, \dot{\mathbf{q}}, E_{act}$ and E_{gr} are the initial values of generalized coordinates, their derivatives, actuator energy, and ground energy, respectively.
2. Initial time (t_i) and final time (t_f) of simulation and step size (*step*).
3. Relative tolerance (r_{tol}) and absolute tolerance (a_{tol})

A typical example of the file *initials.m* is shown in Table 10.12. The joint torques can be entered in the function file *torque.m*. Next, simulation is performed by running function file *runfor.p*. The generalized motions are stored in the result file *statevar.dat*, whereas the time history is stored in *timevar.dat*. Generalized motion can be plotted using the function *plot_motion.m*.

Table 10.12 The file *initials.m* for planar biped

```
function [y0, ti, tf, incr, rtol, atol, step,
int_type]=initials()

  %1: Initial Conditions: State variables q and dq, and actuator
energy
  s=0;
  [th dth ddth Xh Yh Zh dXh dYh dZh]=trajectory(s);
  %Base motions
  q0=[Xh, Yh, Zh, (0/180)*pi, (0/180)*pi, (0/180)*pi];
  dq0=[dXh, dYh, dZh, 0, 0, 0];
  E_act=0; E_gr=0;

  Y0=[q0, th, dq0, dth, E_act, E_gr];

  %2: Simulation time and steps
  ti=0; % Initial time
  tf=3; % Final time
  step=0.01; % Step size

  %3: Integration tolerances
  rtol=1e-4;    %relative tolerance in integration
  atol=1e-6;    %absolute tolerances in integration
  int_type=0;    %0 for ode45, 1 for ode15s
```

10.4 Summary

The detailed use of Recursive Dynamic Simulator (ReDySim), a MATLAB based computer algorithm for the dynamic analyses of robotic and multibody systems, has been presented for the problems illustrated in this book. The ReDySim contains program module, each for fixed-base system and floating-base system. The input parameters required for inverse and forward dynamics of the above systems were discussed and corresponding MATLAB files were illustrated.

Appendices

A Computational Complexity

In this Appendix, computational complexity of the various matrix–vector operations required in dynamic algorithms provided in Chaps. 6 and 7 are shown. For that, vectors and matrices associated with the kth link, e.g., vector $\mathbf{t}_k$ or matrix $\mathbf{M}_k$, are represented in the frame attached to link k and referred to as the kth frame. However for brevity as well as simplicity of writing, the frame of reference will not explicitly be mentioned. If a vector or matrix is represented in the frame other than kth frame, say the jth frame, it will then be expressed as $[\mathbf{r}_k]_j$ or $[\mathbf{M}_k]_j$.

A.1 Elementary Computations

Computations of the terms $\mathbf{e}_k\dot{\theta}_k, \mathbf{e}_k^T\mathbf{r}_k,$ and $(\mathbf{e}_k \times \mathbf{r}_k)$ are shown in this section. For this, the 3-dimensional vector $\mathbf{e}_k$ denoting the unit vector along the axis of rotation or translation of a joint is represented in the body-fixed frame k as $\mathbf{e}_k \equiv \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. Hence, for a scalar joint rate $\dot{\theta}_k$ and the 3-dimensional vector $\mathbf{r}_k \equiv \begin{bmatrix} r_{k1} & r_{k2} & r_{k3} \end{bmatrix}^T$, the terms $\mathbf{e}_k\dot{\theta}_k, \mathbf{e}_k^T\mathbf{r}_k,$ and $(\mathbf{e}_k \times \mathbf{r}_k)$ are calculated as

$$\mathbf{e}_k\dot{\theta}_k = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_k \end{bmatrix} (0\text{M}0\text{A}),\ \mathbf{e}_k^T\mathbf{r}_k = r_{k3}(0\text{M}0\text{A}),\ \mathbf{e}_k \times \mathbf{r}_k = \begin{bmatrix} -r_{k2} \\ r_{k1} \\ 0 \end{bmatrix} (0\text{M}0\text{A}) \tag{A.1}$$

The numbers within the parenthesis, i.e., (0M0A), denote the numbers of Multiplications/Divisions (M) and Additions/Subtractions (A) required to compute the expressions.

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7,

A.2 A Vector in a Different Frame

A rotation matrix $\mathbf{Q}_k$ representing the rotation between kth and jth frame, where j is the parent of k, can be considered as two successive rotations about X- and Z-axes (Eq. 3.4), i.e.,

$$\mathbf{Q}_k = \mathbf{Q}_{\alpha_k}\mathbf{Q}_{\theta_k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha_k & -S\alpha_k \\ 0 & S\alpha_k & C\alpha_k \end{bmatrix} \begin{bmatrix} C\theta_k & -S\theta_k & 0 \\ S\theta_k & C\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

A vector $\mathbf{r}_k$ can then be represented in the jth frame as

$$\begin{aligned} [\mathbf{r}_k]_j &= \mathbf{Q}_k \mathbf{r}_k \\ &= \mathbf{Q}_{\alpha_k} \mathbf{r}'_k \text{ where } \mathbf{r}'_k = \mathbf{Q}_{\theta_k} \mathbf{r}_k \end{aligned} \tag{A.3}$$

where $\mathbf{r}'_k$ and $\mathbf{Q}_{\alpha_k}\mathbf{r}'_k$ are computed as follows

$$\begin{aligned} &1)\ \mathbf{r}'_k \equiv \begin{bmatrix} r'_{k1} \\ r'_{k2} \\ r'_{k3} \end{bmatrix} = \begin{bmatrix} r_{k1}C\theta_k - r_{k2}S\theta_k \\ r_{k1}S\theta_k + r_{k2}C\theta_k \\ r_{k3} \end{bmatrix} (4M2A); \\ &2)\ \mathbf{Q}_{\alpha_k}\mathbf{r}'_k = \begin{bmatrix} r'_{k1} \\ r'_{k2}C\alpha_k - r'_{k3}S\alpha_k \\ r'_{k2}S\alpha_k + r'_{k3}C\alpha_k \end{bmatrix} (4M2A) \end{aligned} \tag{A.4}$$

So, the transformation $\mathbf{Q}_k\mathbf{r}_k$ requires computational count of 8M4A. Similarly, one may also represent vector $\mathbf{r}_j$ in kth frame as $\mathbf{Q}_k^T\mathbf{r}_j$, i.e.,

$$\begin{aligned} [\mathbf{r}_j]_k &= \mathbf{Q}_k^T \mathbf{r}_j \\ &= \mathbf{Q}_{\theta_k}^T \mathbf{r}'_j \text{ where } \mathbf{r}'_j = \mathbf{Q}_{\alpha_k}^T \mathbf{r}_j \end{aligned} \tag{A.5}$$

where $\mathbf{r}'_j$ and $\mathbf{Q}_{\theta_k}^T\mathbf{r}'_j$ are computed as follows

$$\begin{aligned} &1)\ \mathbf{r}'_j \equiv \begin{bmatrix} r'_{j1} \\ r'_{j2} \\ r'_{j3} \end{bmatrix} = \begin{bmatrix} r_{j1} \\ r_{j2}C\alpha_k + r_{j3}S\alpha_k \\ -r_{j2}S\alpha_k + r_{j3}C\alpha_k \end{bmatrix} (4M2A); \\ &2)\ \mathbf{Q}_k^T\mathbf{r}_j = \begin{bmatrix} r'_{j1}C\theta_k + r'_{j2}S\theta_k \\ -r'_{j1}S\theta_k + r'_{j2}C\theta_k \\ r'_{j3} \end{bmatrix} (4M2A) \end{aligned} \tag{A.6}$$

In the case of multiple-DOF universal or spherical joints, α is typically taken as 90°, i.e., joint axes intersecting orthogonally. Hence, $\mathbf{r}'$ does not require any computation.

A.3 Matrix in a Different Frame

A 3×3 matrix $\mathbf{F}_k$ can be represented in the jth frame as

$$\begin{aligned}[\mathbf{F}_k]_j &= \mathbf{Q}_k \mathbf{F}_k \mathbf{Q}_k^T \\ &= \mathbf{Q}_{\alpha_k} \mathbf{F'}_k \mathbf{Q}_{\alpha_k}^T \text{ where } \mathbf{F'}_k = \mathbf{Q}_{\theta_k} \mathbf{F}_k \mathbf{Q}_{\theta_k}^T\end{aligned} \tag{A.7}$$

Note that in Eq. (A.7), the 3×3 rotation matrix $\mathbf{Q}_k$ denotes the orientation of the kth frame with respect to jth where j is parent of k. Moreover, $\mathbf{Q}_{\alpha_k}$ and $\mathbf{Q}_{\theta_k}$ have the same representations as given in Eq. (A.2). Equation (A.7) is computed in the following two steps:

Step 1: Computation of $\mathbf{F'}_k = \mathbf{Q}_{\theta_k} \mathbf{F}_k \mathbf{Q}_{\theta_k}^T$

$$\mathbf{F}'_k = \begin{bmatrix} f_{11} - d_1 & f_{12} + d_2 & f_{13} C\theta_k - f_{23} S\theta_k \\ f_{21} + d_2 & f_{22} + d_1 & f_{13} S\theta_k + f_{23} C\theta_k \\ f_{31} C\theta_k - f_{32} S\theta_k & f_{31} S\theta_k + f_{32} C\theta_k & f_{33} \end{bmatrix} (8M8A)$$

where

$$\begin{aligned} &d_1 = b_1 t_1 + b_2 t_2, d_2 = b_1 t_2 - b_2 t_1, b_1 = (f_{11} - f_{22}), b_2 = (f_{12} + f_{21})(4M4A) \\ &t_1 = S\theta_k S\theta_k, t_2 = C\theta_k S\theta_k (2M0A) \end{aligned} \tag{A.8}$$

In Eq. (A.8), the term f_{rs} is the (r, s)th element of matrix $\mathbf{F}_k$.

Step 2: Computation of $[\mathbf{F}_k]_j = \mathbf{Q}_{\alpha_k} \mathbf{F'}_k \mathbf{Q}_{\alpha_k}^T$

$$[\mathbf{F}_k]_j = \begin{bmatrix} f'_{11} & f'_{12} C\alpha_k - f'_{13} S\alpha_k & f'_{12} S\alpha_k + f'_{13} C\alpha_k \\ f'_{21} C\alpha_k - f'_{31} S\alpha_k & f'_{22} - d'_1 & f'_{23} + d'_2 \\ f'_{21} S\alpha_k + f'_{31} C\alpha_k & f'_{32} + d'_2 & f'_{33} + d'_1 \end{bmatrix} (8M8A)$$

where

$$\begin{aligned} d'_1 &= b'_1 t'_1 + b'_2 t'_2, d'_2 = b'_1 t'_2 - b'_2 t'_1, b'_1 = (f'_{22} - f'_{33}), b'_2 \\ &= (f'_{23} + f'_{32})(4M4A) \\ t'_1 &= S\alpha_k S\alpha_k, t'_2 = C\alpha_k S\alpha_k \ (0M0A) \text{ (off-line)} \end{aligned} \tag{A.9}$$

In Eq. (A.9), the term f'_{rs} is the (r, s)th element of matrix $\mathbf{F}'_k$. The terms t'_1 and t'_2 in Eq. (A.9) may be computed offline as α_k is the constant DH parameter. Computation of Eqs. (A.8) and (A.9) need computational counts of 14M12A and 12M12A, respectively, and the total count of 26M24A is required to calculate $[\mathbf{F}_k]_j$. In the case of multiple-DOF universal or spherical joints, the angle between the intersecting revolute joint axes are 90°, so, Eq. (A.9) does not require any computation as it is simply

$$[\mathbf{F}_k]_j = \begin{bmatrix} f'_{11} & -f'_{13} & f'_{12} \\ -f'_{31} & f'_{33} & -f'_{32} \\ f'_{21} & -f'_{23} & f'_{22} \end{bmatrix}; (0M0A) \tag{A.10}$$

If $\mathbf{F}_k$ is a symmetric matrix, then $\mathbf{F}'_k$ and $[\mathbf{F}_k]_j$ are obtained as shown below:

$$\mathbf{F}'_k = \begin{bmatrix} f_{11} - d_1 & & sym \\ d_2 & f_{22} + d_1 & \\ f_{31}C\theta_k - f_{32}S\theta_k & f_{31}S\theta_k + f_{32}C\theta_k & f_{33} \end{bmatrix} (4M4A)$$

where

$$\begin{aligned} d_1 = b_1 t_1 + f_{21} t_3, d_2 = b_1 t_2 - f_{21} t_4, b_1 = (f_{11} - f_{22})(4M3A) \\ t_1 = S\theta_k S\theta_k, t_2 = C\theta_k S\theta_k, t_3 = 2t_2, t_4 = 2t_1 - 1; (4M1A) \end{aligned} \tag{A.11}$$

$$[\mathbf{F}_k]_j = \begin{bmatrix} f'_{11} & & sym \\ f'_{21}C\alpha_k - f'_{31}S\alpha_k & f'_{22} - d_1 & \\ f'_{21}S\alpha_k + f'_{31}C\alpha_k & d_2 & f'_{33} + d_1 \end{bmatrix} (4M4A)$$

where

$$\begin{aligned} d_1 = b'_1 t'_1 + f'_{32} t'_3, d_2 = b'_1 t'_2 - f'_{32} t'_4, b'_1 = (f'_{22} - f'_{33})(4M3A) \\ t'_1 = S\alpha_k S\alpha_k, t'_2 = C\alpha_k S\alpha_k, t'_3 = 2t'_2, t'_4 = 2t'_1 - 1; (0M0A) \ \textbf{(off-line)} \end{aligned} \tag{A.12}$$

Equations (A.11) and (A.12) require computational counts of 12M8A and 8M7A, respectively. As a result, the total computational count for the calculation of $[\mathbf{F}_k]_j$ is 20M15A. Moreover, if θ is constant then the computational count is 16M14A. In the case of multiple-DOF joints with $\alpha_k = 90$, Eq. (A.12) is simply

$$[\mathbf{F}_k]_j = \begin{bmatrix} f'_{11} & & sym \\ -f'_{31} & f'_{33} & \\ f'_{21} & -f'_{32} & f'_{22} \end{bmatrix} (0M0A) \tag{A.13}$$

A.4 Spatial Transformations

The 6×6 matrix $\mathbf{A}_{k,j}$ (where j is the parent of k) in Eq. (4.3) represents the twist-propagation matrix whereas $\mathbf{w}_k$ in Eq. (5.6) is the 6-dimesional wrench vector. Matrix $\mathbf{A}_{k,j}$ and vector $\mathbf{w}_k$ have the following representations:

$$\mathbf{A}_{k,j} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{a}_{k,j} \times \mathbf{1} & \mathbf{1} \end{bmatrix} \text{ and } \mathbf{w}_j \equiv \begin{bmatrix} \mathbf{n}_k \\ \mathbf{f}_k \end{bmatrix} \tag{A.14}$$

In Eq. (A.14), $\mathbf{a}_{k,j} \times \mathbf{1}$ is the cross-product tensor associated with the vector $\mathbf{a}_{k,j} \equiv [-a_k \;\; b_k S\alpha_k \;\; -b_k C\alpha_k]^T$, where a_k and b_k are the DH parameters, which are the link length and the joint offset, respectively. The tensor $\mathbf{a}_{k,j} \times \mathbf{1}$, when operates on any 3-dimensional Cartesian vector, $\mathbf{x}$, results in the cross-product vector $\mathbf{a}_{k,j} \times \mathbf{x}$. The tensor $\mathbf{a}_{k,j} \times \mathbf{1}$ has the following representation:

$$\mathbf{a}_{k,j} \times \mathbf{1} = \begin{bmatrix} 0 & b_k C\alpha_k & b_k S\alpha_k \\ -b_k C\alpha_k & 0 & a_k \\ -b_k S\alpha_k & -a_k & 0 \end{bmatrix} \tag{A.15}$$

If matrix $\mathbf{A}_{k,j}$ and vector $\mathbf{w}_k$ are available in jth and kth frame, respectively, the transformation $[\mathbf{A}_{k,j}^T \mathbf{w}_k]_j$ in the jth frame may be calculated as

$$[\mathbf{A}_{k,j}^T \mathbf{w}_k]_j = \begin{bmatrix} \mathbf{1} & (\mathbf{a}_{k,j} \times \mathbf{1})^T \\ \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_k \mathbf{n}_k \\ \mathbf{Q}_k \mathbf{f}_k \end{bmatrix} \tag{A.16}$$

The above equation is written as

$$[\mathbf{A}_{k,j}^T \mathbf{w}_k]_j = \begin{bmatrix} \mathbf{Q}_k & (\mathbf{a}_{k,j} \times \mathbf{1})^T \mathbf{Q}_k \\ \mathrm{O} & \mathbf{Q}_k \end{bmatrix} \begin{bmatrix} \mathbf{n}_k \\ \mathbf{f}_k \end{bmatrix} \tag{A.17}$$

Now, Eq. (A.17) can also be rewritten as

$$[\mathbf{A}_{k,j}^T \mathbf{w}_k]_j = \begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times \mathbf{1})^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times \mathbf{1})^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix} \begin{bmatrix} \mathbf{n}_k \\ \mathbf{f}_k \end{bmatrix} \tag{A.18}$$

In Eq. (A.18), $(\mathbf{a}_k \times \mathbf{1})$ and $(\mathbf{b}_k \times \mathbf{1})$ are cross-product tensors associated with the vectors $\mathbf{a}_k \equiv [-a_k \;\; 0 \;\; 0]^T$ and $\mathbf{b}_k \equiv [0 \;\; 0 \;\; -b_k]^T$. Moreover, the matrices

$$\begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times \mathbf{1})^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times \mathbf{1})^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix}$$

may be interpreted as planar screw transformation (Featherstone 1987) about X and Z axes, respectively. Now Eq. (A.18) is computed in two steps as

$$\begin{aligned} 1)\ \mathbf{w}'_k &= \begin{bmatrix} \mathbf{n}'_k \\ \mathbf{f}'_k \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times \mathbf{1})^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix} \begin{bmatrix} \mathbf{n}_k \\ \mathbf{f}_k \end{bmatrix} \\ 2)\ [\mathbf{A}^T_{k,j} \mathbf{w}_k]_j &= \begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times \mathbf{1})^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \begin{bmatrix} \mathbf{n}'_k \\ \mathbf{f}'_k \end{bmatrix} \end{aligned} \tag{A.19}$$

These two steps are explained below:

Step 1: Computation of $\mathbf{w}'_k$

$$\mathbf{w}'_k = \begin{bmatrix} \mathbf{Q}_{\theta_k} \mathbf{n}_k + (\mathbf{b}_k \times \mathbf{1})^T \mathbf{Q}_{\theta_k} \mathbf{f}_k \\ \mathbf{Q}_{\theta_k} \mathbf{f}_k \end{bmatrix} \tag{A.20}$$

Equation (A.20) is computed next as

$$\mathbf{Q}_{\theta_k} \mathbf{n}_k + (\mathbf{b}_k \times \mathbf{1})^T \mathbf{Q}_{\theta_k} \mathbf{f}_k = \begin{bmatrix} n_{k1} C\theta_k - n_{k2} S\theta_k - b_k h_2 \\ n_{k1} S\theta_k + n_{k2} C\theta_k + b_k h_1 \\ \omega_{k3} \end{bmatrix} (6M4A)$$

$$\mathbf{Q}_{\theta_k} \mathbf{f}_k = \begin{bmatrix} h_1 \\ h_2 \\ f_{k3} \end{bmatrix} (0M0A)$$

$$\text{where } h_1 = f_{k1} C\theta_k - f_{k2} S\theta_k, h_2 = f_{k1} S\theta_k + f_{k2} C\theta_k (4M2A). \tag{A.21}$$

Step 2: Computation of $[\mathbf{A}^T_{k,j} \mathbf{w}_k]_j$

$$[\mathbf{A}^T_{k,j} \mathbf{w}_k]_j = \begin{bmatrix} \mathbf{Q}_{\alpha_k} \mathbf{n}'_k + (\mathbf{a}_k \times \mathbf{1})^T \mathbf{Q}_{\alpha_k} \mathbf{f}'_k \\ \mathbf{Q}_{\alpha_k} \mathbf{f}'_k \end{bmatrix} \tag{A.22}$$

Eq. (A.22) is computed as

$$\mathbf{Q}_{\alpha_k} \mathbf{n}'_k + (\mathbf{a}_k \times \mathbf{1})^T \mathbf{Q}_{\alpha_k} \mathbf{f}'_k = \begin{bmatrix} n'_{k1} \\ n'_{k2} C\alpha_k - n'_{k3} S\alpha_k - a_k h'_2 \\ n'_{k2} S\alpha_k + n'_{k3} C\alpha_k + a_k h'_1 \end{bmatrix} (6M4A)$$

$$\mathbf{Q}_{\alpha_k} \mathbf{f}'_k = \begin{bmatrix} f'_{k1} \\ h'_1 \\ h'_2 \end{bmatrix} (0M0A) \tag{A.23}$$

where $h'_1 = f'_{k2} C\alpha_k - f'_{k3} S\alpha_k, h'_2 = f'_{k2} S\alpha_k + f'_{k3} C\alpha_k (4M2A)$.

Steps 1 and 2 require computational count of 10M6A each and hence a total computational count of 20M12A is required to find $\mathbf{A}_{j,k}^T \mathbf{w}_j$ in the kth frame. Similarly, it may also be shown that the computational count to find out $[\mathbf{A}_{j,k}\mathbf{t}_k]_j$ is 20M12A, where $\mathbf{t}_k$ is the 6-dimensional twist vector as defined in Eq. (4.2).

A.5 Special Computations

The linear acceleration ($\ddot{\mathbf{o}}_k$) of the kth link in terms of its parent link, j, is obtained as

$$\ddot{\mathbf{o}}_k = \mathbf{Q}_k^T[\ddot{\mathbf{o}}_j + \dot{\boldsymbol{\omega}}_j \times \mathbf{a}_{j,k} + \dot{\boldsymbol{\omega}}_j \times (\boldsymbol{\omega}_j \times \mathbf{a}_{j,k})] \tag{A.24}$$

where $\boldsymbol{\omega}_j$ and $\dot{\boldsymbol{\omega}}_j$ are the 3-dimensional vectors of angular velocity and angular acceleration, respectively, of the jth link. Equation (A.24) may also be rewritten as

$$\ddot{\mathbf{o}}_k = \mathbf{Q}_k^T \left[\ddot{\mathbf{o}}_j + \boldsymbol{\varpi}_j \mathbf{a}_{j,k}\right] \text{where } \boldsymbol{\varpi}_j = (\dot{\boldsymbol{\omega}}_j \times \mathbf{1}) + (\boldsymbol{\omega}_j \times \mathbf{1})(\boldsymbol{\omega}_j \times \mathbf{1}) \tag{A.25}$$

In Eq. (A.25), $\boldsymbol{\omega}_j \times \mathbf{1}$ and $\dot{\boldsymbol{\omega}}_j \times \mathbf{1}$ are the cross-product tensors associated with $\boldsymbol{\omega}_j$ and $\dot{\boldsymbol{\omega}}_j$, respectively. The 3×3 matrix $\boldsymbol{\varpi}$ is computed efficiently as follows:

$$\boldsymbol{\varpi} = \begin{bmatrix} -\omega_{33} - \omega_{22} & -\dot{\omega}_3 + \omega_{12} & \dot{\omega}_2 + \omega_{13} \\ \dot{\omega}_3 + \omega_{12} & -\omega_{33} - \omega_{11} & -\dot{\omega}_1 + \omega_{23} \\ -\dot{\omega}_2 + \omega_{13} & \dot{\omega}_1 + \omega_{23} & -\omega_{22} - \omega_{11} \end{bmatrix} (0M9A) \tag{A.26}$$

where $\omega_{13} = \omega_1\omega_3, \omega_{23} = \omega_2\omega_3, \omega_{13} = \omega_1\omega_3, \omega_{11} = \omega_1\omega_1, \omega_{22} = \omega_2\omega_2, \omega_{33} = \omega_3\omega_3$ (6M0A). Moreover, ω_s and $\dot{\omega}_s$, for $s = 1, \ldots, 3$, are the sth element of the vectors, $\boldsymbol{\omega}_j$ and $\dot{\boldsymbol{\omega}}_j$, respectively. Hence, $\boldsymbol{\varpi}$ requires a computational count of 6M9A, whereas computation of $\ddot{\mathbf{o}}_k$ requires a computational count of 17M13A.

Next, the Euler equations of motion (as obtained in Eq. 5.4) for the kth link is given as

$$\mathbf{n}_k = \mathbf{I}_k \dot{\boldsymbol{\omega}}_k + \boldsymbol{\omega}_k \times \mathbf{I}_k \boldsymbol{\omega}_k + m_k \mathbf{d}_k \times \ddot{\mathbf{o}}_k \tag{A.27}$$

In Eq. (A.27), $\mathbf{I}_k \dot{\boldsymbol{\omega}}_k + \boldsymbol{\omega}_k \times \mathbf{I}_k \boldsymbol{\omega}_k$ is efficiently computed as

$$\begin{aligned} \mathbf{n}' &= \mathbf{I}_k \dot{\boldsymbol{\omega}}_k + \boldsymbol{\omega}_k \times \mathbf{I}_k \boldsymbol{\omega}_k \\ &= \begin{bmatrix} \dot{\omega}_1 i_{11} - \varpi_{31} i_{12} + \varpi_{21} i_{13} + i_{23}(\omega_{22} - \omega_{33}) - p_{23}\omega_{23} \\ \dot{\omega}_2 i_{22} - \varpi_{21} i_{23} + \varpi_{32} i_{12} + i_{13}(\omega_{33} - \omega_{11}) - p_{31}\omega_{13} \\ \dot{\omega}_3 i_{33} - \varpi_{23} i_{13} + \varpi_{13} i_{23} + i_{12}(\omega_{11} - \omega_{22}) - p_{12}\omega_{12} \end{bmatrix} (15M15A) \end{aligned} \tag{A.28}$$

where $p_{23} = (i_{22} - i_{33})$, $p_{31} = (i_{33} - i_{11})$, and $p_{12} = (i_{11} - i_{22})$, which are constant terms and may be calculated offline. Moreover, i_{rs} and ϖ_{rs} are the (r,s)th elements of the matrices $\mathbf{I}_k$ and $\boldsymbol{\varpi}$, respectively. Note that, matrix $\boldsymbol{\varpi}$ is already computed

while obtaining the linear acceleration, as in Eq. (A.26). Hence, the computation of $\mathbf{n}_k$ requires a total computational count of 21M21A.

A.6 Mass Matrix of a Composite Body

The mass matrix of a composite body is expressed as

$$\tilde{\mathbf{M}}_j = \mathbf{M}_j + \mathbf{A}_{k,j}^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,j} \tag{A.29}$$

where $\mathbf{M}_j$, $\tilde{\mathbf{M}}_j$ and $\mathbf{A}_{k,j}$ have the following representations:

$$\mathbf{M}_j \equiv \begin{bmatrix} \mathbf{I}_j & m_j \mathbf{d}_j \times \mathbf{1} \\ -m_j \mathbf{d}_j \times \mathbf{1} & m_j \mathbf{1} \end{bmatrix}, \tilde{\mathbf{M}}_j \equiv \begin{bmatrix} \tilde{\mathbf{I}}_j & (\tilde{\boldsymbol{\rho}}_j \times 1)^T \\ \tilde{\boldsymbol{\rho}}_j \times 1 & \tilde{m}_j \mathbf{1} \end{bmatrix}, \mathbf{A}_{k,j} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \tilde{\mathbf{a}}_{k,j} & \mathbf{1} \end{bmatrix} \tag{A.30}$$

in which $\mathbf{d}_j \times \mathbf{1}$ and $\tilde{\boldsymbol{\rho}}_j \times 1$ are the cross-product tensors associated with vectors $\mathbf{d}_k$ and $\tilde{\boldsymbol{\rho}}_j$, respectively. In Eq. (A.29), evaluating $\mathbf{A}_{k,j}^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,j}$ in jth frame of reference is computationally very expensive step as $\tilde{\mathbf{M}}_k$ is available in the kth frame of reference. The transformation, $\mathbf{A}_{k,j}^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,j}$ in the jth frame is obtained as

$$\begin{aligned} &[\mathbf{A}_{k,j}^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,j}]_j \\ &= \begin{bmatrix} \mathbf{1} & (\mathbf{a}_{k,j} \times 1)^T \\ \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_k \tilde{\mathbf{I}}_k \mathbf{Q}_k^T & \mathbf{Q}_k (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1})^T \mathbf{Q}_k^T \\ \mathbf{Q}_k (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) \mathbf{Q}_k^T & \mathbf{Q}_k \tilde{m}_k \mathbf{1} \mathbf{Q}_k^T \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{a}_{k,j} \times 1 & \mathbf{1} \end{bmatrix} \end{aligned} \tag{A.31}$$

Equation (A.31) may also be rewritten as

$$\begin{aligned} &[\mathbf{A}_{k,j}^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,j}]_j \\ &= \begin{bmatrix} \mathbf{Q}_k & (\mathbf{a}_{k,j} \times 1)^T \mathbf{Q}_k \\ \mathbf{O} & \mathbf{Q}_k \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{I}}_k & (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1})^T \\ (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) & \tilde{m}_k \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_k^T & \mathbf{O} \\ \mathbf{Q}_k^T (\mathbf{a}_{k,j} \times 1) & \mathbf{Q}_k^T \end{bmatrix} \end{aligned} \tag{A.32}$$

The above equation is now expressed in terms of the planar screw transformations about X and Z axes as

$$\begin{aligned} [\mathbf{A}_{k,j}^T \tilde{\mathbf{M}}_k \mathbf{A}_{k,j}]_j &= \begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times 1)^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times 1)^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix} \\ &\begin{bmatrix} \tilde{\mathbf{I}}_k & (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1})^T \\ (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) & \tilde{m}_k \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\theta_k}^T & \mathbf{O} \\ \mathbf{Q}_{\theta_k}^T (\mathbf{b}_k \times 1) & \mathbf{Q}_{\theta_k}^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\alpha_k}^T & \mathbf{O} \\ \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) & \mathbf{Q}_{\alpha_k}^T \end{bmatrix} \end{aligned} \tag{A.33}$$

Equation (A.33) is then obtained in two steps as shown below:

$$
\begin{aligned}
1)\ \tilde{\mathbf{M}}'_k &= \begin{bmatrix} \tilde{\mathbf{I}}'_k & (\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1})^T \\ (\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1}) & \tilde{m}'_k \mathbf{1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times 1)^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{I}}_k & (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1})^T \\ (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) & \tilde{m}_k \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^T_{\theta_k} & \mathbf{O} \\ \mathbf{Q}^T_{\theta_k} (\mathbf{b}_k \times 1) & \mathbf{Q}^T_{\theta_k} \end{bmatrix} \\
2)\ &[\mathbf{A}^T_{k,j} \tilde{\mathbf{M}}_k \mathbf{A}_{k,j}]_j \\
&= \begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times 1)^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{I}}'_k & (\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1})^T \\ (\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1}) & \tilde{m}'_k 1 \end{bmatrix} \begin{bmatrix} \mathbf{Q}^T_{\alpha_k} & \mathbf{O} \\ \mathbf{Q}^T_{\alpha_k} (\mathbf{a}_k \times 1) & \mathbf{Q}^T_{\alpha_k} \end{bmatrix}
\end{aligned}
\tag{A.34}
$$

These above steps are expanded below:

Step 1: Computation of $\tilde{\mathbf{M}}'_k$

$$
\begin{aligned}
\tilde{\mathbf{M}}'_k &= \begin{bmatrix} \tilde{\mathbf{I}}'_k & (\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1})^T \\ (\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1}) & \tilde{m}'_k \mathbf{1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{Q}_{\theta_k} \tilde{\mathbf{I}}_k \mathbf{Q}^T_{\theta_k} + \mathbf{Q}_{\theta_k} (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1})^T \mathbf{Q}^T_{\theta_k} (\mathbf{b}_k \times 1) + (\mathbf{b}_k \times 1)^T \mathbf{Q}_{\theta_k} (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) \mathbf{Q}^T_{\theta_k} + \mathbf{Q}_{\theta_k} \tilde{m}_k \mathbf{1} \mathbf{Q}^T_{\theta_k} (\mathbf{b}_k \times 1) & sym \\ \mathbf{Q}_{\theta_k} (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) \mathbf{Q}^T_{\theta_k} + \mathbf{Q}_{\theta_k} \tilde{m}_k \mathbf{1} \mathbf{Q}^T_{\theta_k} (\mathbf{b}_k \times 1) & \mathbf{Q}_{\theta_k} \tilde{m}_k \mathbf{1} \mathbf{Q}^T_{\theta_k} \end{bmatrix}
\end{aligned}
\tag{A.35}
$$

The transformation $\mathbf{Q}_{\theta_k} \tilde{\mathbf{I}}_k \mathbf{Q}^T_{\theta_k}$ is obtained by using Eq. (A.11), whereas $\mathbf{Q}_{\theta_k} \tilde{m}_k \mathbf{1} \mathbf{Q}^T_{\theta_k} = \tilde{m}_k \mathbf{Q}_{\theta_k} \mathbf{Q}^T_{\theta_k} = \tilde{m}_k \mathbf{1}$ as $\mathbf{Q}_{\theta_k} \mathbf{Q}^T_{\theta_k} = \mathbf{1}$. Computation of $\mathbf{Q}_{\theta_k} \tilde{\mathbf{I}}_k \mathbf{Q}^T_{\theta_k}$ and $\mathbf{Q}_{\theta_k} \tilde{m}_k \mathbf{1} \mathbf{Q}^T_{\theta_k}$ require, counts of 12M8A and 0M0A, respectively. Moreover, $\mathbf{Q}_{\theta_k} (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) \mathbf{Q}^T_{\theta_k}$ is obtained efficiently as follows:

$$
\mathbf{Q}_{\theta_k} (\tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) \mathbf{Q}^T_{\theta_k} = (\mathbf{Q}_{\theta_k} \tilde{\boldsymbol{\rho}}_k \times \mathbf{1}) (4M2A) \tag{A.36}
$$

The block elements of Eq. (A.35) are obtained as

$$
\begin{aligned}
\tilde{\mathbf{I}}'_k &= \begin{bmatrix} i_{11} + c_2 & i_{12} & i_{13} + \rho_1 b_k \\ & i_{22} + c_2 & i_{23} + \rho_2 b_k \\ sym & & i_{33} \end{bmatrix} (2M4A); \\
(\tilde{\boldsymbol{\rho}}'_k \times \mathbf{1}) &= \begin{bmatrix} 0 & c_1 & \rho_2 \\ -c_1 & 0 & -\rho_1 \\ -\rho_2 & \rho_1 & 0 \end{bmatrix} (0M0A)
\end{aligned}
\tag{A.37}
$$

where $c_1 = c_3 - \rho_3, c_2 = (c_1 + \rho_3) b_k (1M2A)$; and $c_3 = \tilde{m}_k b_k (0M0A)$ (off-line).

In Eq. (A.37), i_{rs} and ρ_s are the $(r,s)^{\text{th}}$ and s^{th} elements of the matrix $\mathbf{Q}_{\theta_k}\tilde{\mathbf{I}}_k\mathbf{Q}_{\theta_k}^T$ and vector $\mathbf{Q}_{\theta_k}\tilde{\boldsymbol{\rho}}_k$, respectively. The total computational count for step 1 is 19M16A.

Step 2: Computation of $[\mathbf{A}_{k,j}^T\tilde{\mathbf{M}}_k\mathbf{A}_{k,j}]_j$

$$[\mathbf{A}_{k,j}^T\tilde{\mathbf{M}}_k\mathbf{A}_{k,j}]_j = \begin{bmatrix} \mathbf{Q}_{\alpha_k}\tilde{\mathbf{I}'}_k\mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})^T\mathbf{Q}_{\alpha_k}^T(\mathbf{a}_k\times 1) + (\mathbf{a}_k\times 1)^T\mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})\mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k}\tilde{m}'_k\mathbf{1}\mathbf{Q}_{\alpha_k}^T(\mathbf{a}_k\times 1) & sym \\ \mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})\mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k}\tilde{m}'_k\mathbf{1}\mathbf{Q}_{\alpha_k}^T(\mathbf{a}_k\times 1) & \mathbf{Q}_{\alpha_k}\tilde{m}'_k\mathbf{1}\mathbf{Q}_{\alpha_k}^T \end{bmatrix} \tag{A.38}$$

In Eq. (A.38), $\mathbf{Q}_{\alpha_k}\tilde{\mathbf{I}'}_k\mathbf{Q}_{\alpha_k}^T$ is obtained using Eq. (A.12) and has computational count of 8M7A, whereas $\mathbf{Q}_{\alpha_k}\tilde{m}'_k\mathbf{1}\mathbf{Q}_{\alpha_k}^T = \tilde{m}'_k\mathbf{1}(0M0A)$ requires no computations. Moreover $\mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})\mathbf{Q}_{\alpha_k}^T$ is obtained as

$$\mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})\mathbf{Q}_{\alpha_k}^T = (\mathbf{Q}_{\alpha_k}\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})(4M2A) \tag{A.39}$$

Now, the block elements of Eq. (A.38) are given by

$$\mathbf{Q}_{\alpha_k}\tilde{\mathbf{I}'}_k\mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})^T\mathbf{Q}_{\alpha_k}^T(\mathbf{a}_k\times 1) + (\mathbf{a}_k\times 1)^T\mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})\mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k}\tilde{m}'_k\mathbf{1}\mathbf{Q}_{\alpha_k}^T(\mathbf{a}_k\times 1) = \begin{bmatrix} i'_{11} & i'_{12}+\rho'_2a_k & i'_{13}+\rho'_3a_k \\ & i'_{22}+c'_2 & i'_{23} \\ sym & & i'_{33}+c'_2 \end{bmatrix}(2M4A)$$

$$\mathbf{Q}_{\alpha_k}(\tilde{\boldsymbol{\rho}'}_k\times\mathbf{1})\mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k}\tilde{m}'_k\mathbf{1}\mathbf{Q}_{\alpha_k}^T(\mathbf{a}_k\times 1) = \begin{bmatrix} 0 & -\rho'_3 & \rho'_2 \\ \rho'_3 & 0 & c'_1 \\ -\rho'_2 & -c'_1 & 0 \end{bmatrix}(0M0A) \tag{A.40}$$

where $c'_1 = c'_3 - \rho'_1, c'_2 = (c'_1 - \rho'_1)a_k(1M2A)$ and $c'_3 = \tilde{m}'_k a_k(0M0A)$ (off-line).

In Eq. (A.40), i'_{rs} and ρ'_s are the (r,s)th and sth elements of the matrix $\mathbf{Q}_{\alpha_k}\tilde{\mathbf{I}'}_k\mathbf{Q}_{\alpha_k}^T$ and vector $\mathbf{Q}_{\alpha_k}\tilde{\boldsymbol{\rho}'}_k$, respectively. Computational count for step 2 is obtained as 15M15A. Hence, the total count required for $[\mathbf{A}_{k,j}^T\tilde{\mathbf{M}}_k\mathbf{A}_{k,j}]_j$ is 34M31A. In the case of a prismatic joint it would have required a computational count of 31M30A instead. Finally, the computation of $\mathbf{M}_j + [\mathbf{A}_{k,j}^T\tilde{\mathbf{M}}_k\mathbf{A}_{k,j}]_j$ has a computational count of 0M9A for the summation. So the final computational count to obtain $\tilde{\mathbf{M}}_j$ is 34M40A.

A.7 Mass Matrix of an Articulated Body

The mass matrix of an articulated body is expressed as

$$\hat{\mathbf{M}}_j = \mathbf{M}_j + \mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}, \text{ where } \hat{\mathbf{M}}_{k,k} = \mathbf{\Phi}_k \hat{\mathbf{M}}_k \tag{A.41}$$

The matrices $\hat{\mathbf{M}}_j$, and $\hat{\mathbf{M}}_{k,k}$ in Eq. (A.41) have the following representations:

$$\hat{\mathbf{M}}_j \equiv \begin{bmatrix} \hat{\mathbf{I}}_j & \hat{\mathbf{F}}_j^T \\ \hat{\mathbf{F}}_j & \hat{\mathbf{G}}_j \end{bmatrix} \text{ and } \hat{\mathbf{M}}_{k,k} \equiv \begin{bmatrix} \hat{\mathbf{I}}_{kk} & \hat{\mathbf{F}}_{kk}^T \\ \hat{\mathbf{F}}_{kk} & \hat{\mathbf{G}}_{kk} \end{bmatrix} \tag{A.42}$$

In Eq. (A.41), evaluating $\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}$ in the jth frame of reference is computationally a very expensive as $\hat{\mathbf{M}}_{k,k}$ is available in the kth frame. The term $\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}$ in the jth frame is obtained as

$$[\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j = \begin{bmatrix} \mathbf{1} & (\mathbf{a}_{k,j} \times 1)^T \\ \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_k \hat{\mathbf{I}}_{kk} \mathbf{Q}_k^T & \mathbf{Q}_k \hat{\mathbf{F}}_k^T \mathbf{Q}_k^T \\ \mathbf{Q}_k \hat{\mathbf{F}}_{kk} \mathbf{Q}_k^T & \mathbf{Q}_k \hat{\mathbf{G}}_{kk} \mathbf{Q}_k^T \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{a}_{k,j} \times 1 & \mathbf{1} \end{bmatrix} \tag{A.43}$$

Similar to Eq. (A.31), the above equation may also be represented in terms of planar screw transformations as

$$\begin{aligned} &[\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j \\ &= \begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times 1)^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times 1)^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{I}}_{kk} & \hat{\mathbf{F}}_{kk}^T \\ \hat{\mathbf{F}}_{kk} & \hat{\mathbf{G}}_{kk} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{Q}_{\theta_k}^T & \mathbf{O} \\ \mathbf{Q}_{\theta_k}^T (\mathbf{b}_k \times 1) & \mathbf{Q}_{\theta_k}^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\alpha_k}^T & \mathbf{O} \\ \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) & \mathbf{Q}_{\alpha_k}^T \end{bmatrix} \end{aligned} \tag{A.44}$$

Equation (A.44) is then obtained in two steps as

$$\begin{aligned} &1)\ \hat{\mathbf{M}}'_{k,k} = \begin{bmatrix} \hat{\mathbf{I}}'_{kk} & \hat{\mathbf{F}}'^T_{kk} \\ \hat{\mathbf{F}}'_{kk} & \hat{\mathbf{G}}'_{kk} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{\theta_k} & (\mathbf{b}_k \times 1)^T \mathbf{Q}_{\theta_k} \\ \mathbf{O} & \mathbf{Q}_{\theta_k} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{I}}_{kk} & \hat{\mathbf{F}}_{kk}^T \\ \hat{\mathbf{F}}_{kk} & \hat{\mathbf{G}}_{kk} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\theta_k}^T & \mathbf{O} \\ \mathbf{Q}_{\theta_k}^T (\mathbf{b}_k \times 1) & \mathbf{Q}_{\theta_k}^T \end{bmatrix} \\ &2)\ [\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j = \begin{bmatrix} \mathbf{Q}_{\alpha_k} & (\mathbf{a}_k \times 1)^T \mathbf{Q}_{\alpha_k} \\ \mathbf{O} & \mathbf{Q}_{\alpha_k} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{I}}'_{kk} & \hat{\mathbf{F}}'^T_{kk} \\ \hat{\mathbf{F}}'_{kk} & \hat{\mathbf{G}}'_{kk} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{\alpha_k}^T & \mathbf{O} \\ \mathbf{Q}_{\alpha_k}^T (a_k \times 1) & \mathbf{Q}_{\alpha_k}^T \end{bmatrix} \end{aligned} \tag{A.45}$$

These above two steps are expanded below:

Step 1: Computation of $\hat{\mathbf{M}}'_{k,k}$

$$\hat{\mathbf{M}}'_{k,k} = \begin{bmatrix} \mathbf{Q}_{\theta_k}\hat{\mathbf{I}}_{kk}\mathbf{Q}_{\theta_k}^T + \mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}^T\mathbf{Q}_{\theta_k}^T(\mathbf{b}_k \times 1) + (\mathbf{b}_k \times 1)^T\mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}\mathbf{Q}_{\theta_k}^T + \mathbf{Q}_{\theta_k}\hat{\mathbf{G}}_{kk}\mathbf{Q}_{\theta_k}^T(\mathbf{b}_k \times 1) & sym \\ \mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}\mathbf{Q}_{\theta_k}^T + \mathbf{Q}_{\theta_k}\hat{\mathbf{G}}_{kk}\mathbf{Q}_{\theta_k}^T(\mathbf{b}_k \times 1) & \mathbf{Q}_{\theta_k}\hat{\mathbf{G}}_{kk}\mathbf{Q}_{\theta_k}^T \end{bmatrix} \tag{A.46}$$

In Eq. (A.46), $\hat{\mathbf{I}}_{kk}$ and $\hat{\mathbf{G}}_{kk}$ are the symmetric matrices. For the kth joint being revolute one, matrices $\hat{\mathbf{I}}_{kk}$ and $\hat{\mathbf{F}}_{kk}$ have the following forms:

$$\hat{\mathbf{I}}_{kk} = \begin{bmatrix} \times & \times & 0 \\ \times & \times & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \hat{\mathbf{F}}_{kk} = \begin{bmatrix} \times & \times & 0 \\ \times & \times & 0 \\ \times & \times & 0 \end{bmatrix} \tag{A.47}$$

where '×' denotes non-zero value of elements. In order to compute Eq. (A.46), first $\mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}\mathbf{Q}_{\theta_k}^T$ is obtained using Eq. (A.8), whereas $\mathbf{Q}_{\theta_k}\hat{\mathbf{I}}_{kk}\mathbf{Q}_{\theta_k}^T$ and $\mathbf{Q}_{\theta_k}\hat{\mathbf{G}}_{kk}\mathbf{Q}_{\theta_k}^T$ are obtained by using Eq. (A.11). Many computations in $\mathbf{Q}_{\theta_k}\hat{\mathbf{I}}_{kk}\mathbf{Q}_{\theta_k}^T$ and $\mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}\mathbf{Q}_{\theta_k}^T$ are simplified for $\hat{\mathbf{I}}_{kk}$ and $\hat{\mathbf{F}}_{kk}$ because they have many zero elements as indicated in Eq. (A.47). The transformations $\mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}\mathbf{Q}_{\theta_k}^T$, $\mathbf{Q}_{\theta_k}\hat{\mathbf{I}}_{kk}\mathbf{Q}_{\theta_k}^T$ and $\mathbf{Q}_{\theta_k}\hat{\mathbf{G}}_{kk}\mathbf{Q}_{\theta_k}^T$ have the following representations:

$$\mathbf{Q}_{\theta_k}\hat{\mathbf{I}}_{kk}\mathbf{Q}_{\theta_k}^T = \begin{bmatrix} i_{11} & i_{12} & 0 \\ i_{12} & i_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}(8M6A), \mathbf{Q}_{\theta_k}\hat{\mathbf{F}}_{kk}\mathbf{Q}_{\theta_k}^T = \begin{bmatrix} f_{11} & f_{12} & 0 \\ f_{21} & f_{22} & 0 \\ f_{31} & f_{32} & 0 \end{bmatrix}(8M10A),$$

$$\text{and } \mathbf{Q}_{\theta_k}\hat{\mathbf{G}}_{kk}\mathbf{Q}_{\theta_k}^T = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ & g_{22} & g_{23} \\ sym & & g_{33} \end{bmatrix}(8M7A) \tag{A.48}$$

Given Eq. (A.48), the block elements of Eq. (A.46) are obtained below:

$$\hat{\mathbf{I}}'_{kk} = \begin{bmatrix} i_{11} - (c_4 + f_{12})b_k & i_{12} + (c_2 - f_{22})b_k & 0 \\ & i_{22} + (c_3 + f_{12})b_k & 0 \\ sym & & 0 \end{bmatrix}; (3M6A)$$

$$\hat{\mathbf{F}}'_{kk} = \begin{bmatrix} c_2 & c_3 & 0 \\ c_4 & f_{22} + c_1 & 0 \\ f_{31} - b_k g_{23} & f_{32} + b_k g_{13} & 0 \end{bmatrix}; (2M3A)$$

$$\text{where } c_1 = b_k g_{12}, c_2 = f_{11} - c_1, c_3 = f_{12} + b_k g_{11}, c_4 = f_{21} - b_k g_{22}(3M3A) \tag{A.49}$$

The total computational count for step 1 is 32M35A. Next, step 2 is evaluated.

Step 2: Computation of $[\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j$

$$[\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j = \begin{bmatrix} \mathbf{Q}_{\alpha_k} \hat{\mathbf{I}}'_{kk} \mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}''_{kk} \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) + (\mathbf{a}_k \times 1)^T \mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) & sym \\ \mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) & \mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T \end{bmatrix} \tag{A.50}$$

Here, $\mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T$ is obtained using Eq. (A.9), whereas $\mathbf{Q}_{\alpha_k} \hat{\mathbf{I}}'_{kk} \mathbf{Q}_{\alpha_k}^T$ and $\mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T$ are obtained using Eq. (A.12). The representations and computational complexity of the transformations $\mathbf{Q}_{\alpha_k} \hat{\mathbf{I}}'_{kk} \mathbf{Q}_{\alpha_k}^T$, $\mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T$ and $\mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T$ are shown below:

$$\mathbf{Q}_{\alpha_k} \hat{\mathbf{I}}'_{kk} \mathbf{Q}_{\alpha_k}^T = \begin{bmatrix} i'_{11} & i'_{12} & i'_{13} \\ & i'_{22} & i'_{23} \\ sym & & i'_{33} \end{bmatrix} (4M1A), \mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T = \begin{bmatrix} f'_{11} & f'_{12} & f'_{13} \\ f'_{21} & f'_{22} & f'_{23} \\ f'_{31} & f'_{32} & f'_{33} \end{bmatrix} (10M6A),$$
$$\text{and } \mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T = \begin{bmatrix} g'_{11} & g'_{12} & g'_{13} \\ & g'_{22} & g'_{23} \\ sym & & g'_{33} \end{bmatrix} (8M7A) \tag{A.51}$$

Now, the block elements of Eq. (A.50) are obtained as

$$\mathbf{Q}_{\alpha_k} \hat{\mathbf{I}}'_{kk} \mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}''_{kk} \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) + (\mathbf{a}_k \times 1)^T \mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1)$$
$$= \begin{bmatrix} i'_{11} & i'_{12} - f'_{31} a_k & i'_{13} + f'_{21} a_k \\ & i'_{22} - (f'_{32} + c'_4) a_k & i'_{23} + (f'_{22} - c'_2) a_k \\ sym & & i'_{33} + (f'_{23} + c'_3) a_k \end{bmatrix}; (5M8A)$$
$$\mathbf{Q}_{\alpha_k} \hat{\mathbf{F}}'_{kk} \mathbf{Q}_{\alpha_k}^T + \mathbf{Q}_{\alpha_k} \hat{\mathbf{G}}'_{kk} \mathbf{Q}_{\alpha_k}^T (\mathbf{a}_k \times 1) \tag{A.52}$$
$$= \begin{bmatrix} f'_{11} & f'_{12} - g'_{13} a_k & f'_{13} + g'_{12} a_k \\ f'_{21} & f'_{22} - c'_1 & c'_3 \\ f'_{31} & c'_4 & c'_2 \end{bmatrix}; (2M3A)$$
$$\text{where } c'_1 = g'_{23} a_k, c'_2 = f'_{33} + c'_1, c'_3 = f'_{23} + g'_{22} a_k, c'_4 = f'_{32} - g'_{33} a_k; (3M3A)$$

The computational count for step 2 is obtained as 32M28A. Hence, the total computational count to compute $[\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j$ (i.e., step1 + step2) is 64M63A. The proposed computational count is little better than that obtained by McMillan and Orin (1995), who reported the count as 70M71A. If the kth joint is prismatic, the computational count will be 60M62A as θ_k is constant. Finally, the computational count of 0M15A is required for summation $\mathbf{M}_j + [\mathbf{A}_{k,j}^T \hat{\mathbf{M}}_{k,k} \mathbf{A}_{k,j}]_j$. So, the final count to obtain $\hat{\mathbf{M}}_j$ is 60M77A.

B Trajectory Generation for Legged Robots

The trajectories used for the dynamic analyses of legged robots are presented in this appendix.

B.1 Biped

The motion of a biped is generally designed such that the Zero-Moment-Point (Vukobratovic et al. 1989) remains within the convex hull of the supporting feet. Hence, the concept of Zero-Moment-Point (ZMP) is introduced first.

B.1.1 Zero-Moment-Point (ZMP)

Zero-Moment-Point (ZMP) is a point on the walking surface about which the horizontal components (i.e., X and Y) of the resultant moment generated by the active forces and moments acting on the links are zero. This is illustrated in Fig. B.1. In other words, it is a point about which the sum of the moments caused by the inertia and gravitational forces is equal to zero. Hence, with the assumption that ZMP remains within the convex hull of the supporting feet, a biped requires no external moment to attain instantaneous equilibrium.

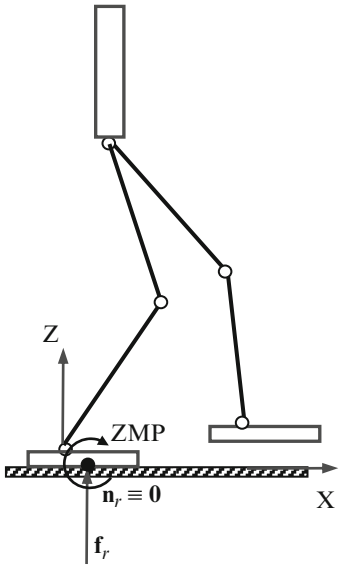


Fig. B.1 Zero-Moment-Point

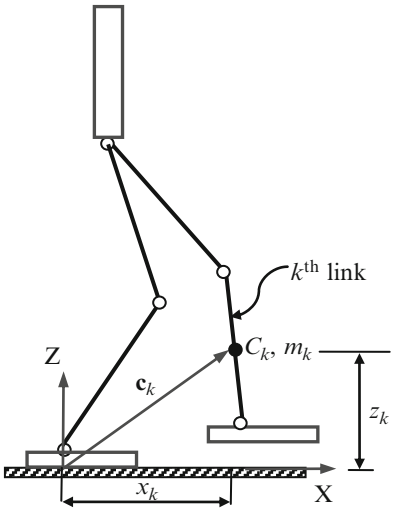


Fig. B.2 Parameters of biped

The ZMP $(x_{zmp}, y_{zmp}, 0)$ can be calculated as

$$x_{zmp} = \frac{\sum_{k=1}^{n} [m_k(\ddot{z}_k + g)x_k - m_k \ddot{x}_k z_k - i_{k,yy}\dot{\omega}_{k,y}]}{\sum_{k=1}^{n} m_k(\ddot{z}_k + g)}$$

$$y_{zmp} = \frac{\sum_{k=1}^{n} [m_k(\ddot{z}_k + g)y_k - m_k \ddot{y}_k z_k - i_{k,xx}\dot{\omega}_{k,x}]}{\sum_{k=1}^{n} m_k(\ddot{z}_k + g)} \tag{B.1}$$

where (x_k, y_k, z_k) are the Cartesian coordinates of the center-of-mass C_k of the kth link and m_k is the mass of the link as shown in Fig. B.2. Moreover, $i_{k,xx}$ and $i_{k,yy}$ are the mass moments of inertia about X and Y axes, $\dot{\omega}_{k,x}$ and $\dot{\omega}_{k,y}$ are the X and Y components of the vector of angular acceleration $\dot{\boldsymbol{\omega}}$, and $g = 9.81$.

B.1.2 Hip Trajectory: Inverted Pendulum Model (IPM)

Trajectory generation of the hip of a biped based on the equations of ZMP in Eq. (B.1) results into computationally expensive procedure as it requires first the evaluation of the expression in Eq. (B.1) followed by the solution of complex differential equations involving many dynamic parameters. Alternatively, Inverted-

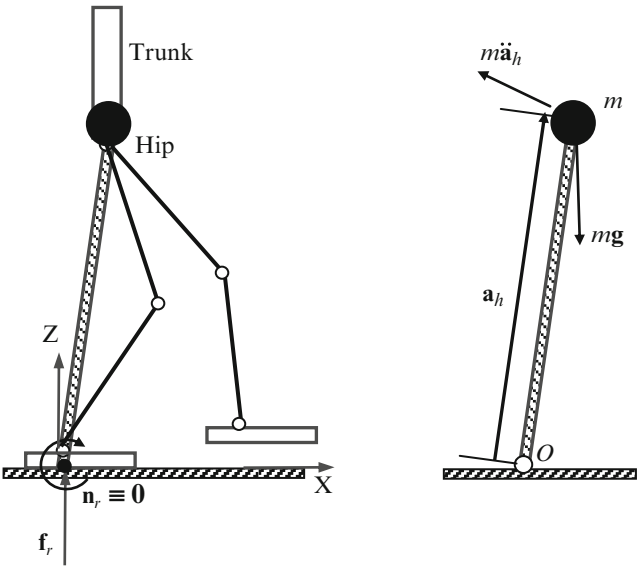


Fig. B.3 Inverted-Pendulum-Model

Pendulum-Model (IPM), shown by Kajita and Tani (1991), assumes that mass of the biped is concentrated at the hip, i.e., the floating base, as shown in Fig. B.3. As a result, the biped motion can be approximated by considering it as an inverted pendulum. For the IPM, the moment equation about the ZMP can be written as

$$\mathbf{a}_h \times (m\ddot{\mathbf{a}}_h) - \mathbf{a}_h \times (m\mathbf{g}) = \mathbf{0} \tag{B.2}$$

where $\mathbf{a}_h \equiv [x_h \ y_h \ z_h]^T$ represents a vector from the origin O to the point mass m and $\mathbf{g} \equiv [0 \ 0 \ g]^T$. Equation (B.2) may be rewritten in component form as

$$\begin{aligned} y_h\ddot{z} - z_h\ddot{y}_h + g_h y_h &= 0 \\ z_h\ddot{x}_h - x_h\ddot{z}_h - g_h z_h &= 0 \\ x_h\ddot{y}_h - y_h\ddot{x}_h &= 0 \end{aligned} \tag{B.3}$$

The IPM assumes that the trajectory is such that the height of the hip remains constant, i.e., $z_h =$ constant, throughout the motion cycle of the biped. Hence, the above equations get simplified as

$$\ddot{x}_h - \frac{g}{z_h} x_h = 0$$
$$\ddot{y}_h - \frac{g}{z_h} y_h = 0 \tag{B.4}$$

Solutions of the above equations are then obtained as

$$\text{For } 0 \leq t \leq T$$
$$x_h(t) = c_1 e^{wt} + c_2 e^{-wt}$$
$$y_h(t) = c_3 e^{wt} + c_4 e^{-wt} \tag{B.5}$$

If the biped moves in the sagittal plane, i.e., in the XZ-plane, then $y_h(t) = 0$. Given the initial positions $x_h(0)$ and $y_h(0)$ and the velocities $\dot{x}_h(0)$ and $\dot{y}_h(0)$, the coefficient c_1, c_2, c_3, and c_4, are obtained as follows

$$c_1 = \frac{1}{2}\left[x_h(0) + \frac{1}{w}\dot{x}_h(0)\right], c_2 = \frac{1}{2}\left[x_h(0) - \frac{1}{w}\dot{x}_h(0)\right]$$
$$c_3 = \frac{1}{2}\left[y_h(0) + \frac{1}{w}\dot{y}_h(0)\right], c_4 = \frac{1}{2}\left[y_h(0) - \frac{1}{w}\dot{y}_h(0)\right] \tag{B.6}$$

where $w = \sqrt{g/z_h}$. Substituting Eq. (B.6) into Eq. (B.5) the coordinates of the hip are obtained

$$\text{For } 0 \leq t \leq T$$
$$x_h(t) = x_h(0)\cosh(wt) + \left(\frac{\dot{x}_h(0)}{w}\right)\sinh(wt)$$
$$y_h(t) = y_h(0)\cosh(wt) + \left(\frac{\dot{y}_h(0)}{w}\right)\sinh(wt) \tag{B.7}$$

In order to obtain the periodic and symmetric biped pattern, the following repeatability conditions are used:

$$x_h(0) = -x_h(T); \dot{x}_h(0) = \dot{x}_h(T)$$
$$y_h(0) = y_h(T); \dot{y}_h(0) = -\dot{y}_h(T) \tag{B.8}$$

The above repeatability conditions help in obtaining the values of initial velocities $\dot{x}_h(0)$ and $\dot{y}_h(0)$, i.e.,

$$\dot{x}_h(0) = \frac{1 + e^{-wT}}{1 - e^{-wT}} w x_h(0) \text{ and } \dot{y}_h(0) = \frac{1 - e^{-wT}}{1 + e^{-wT}} w y_h(0) \tag{B.9}$$

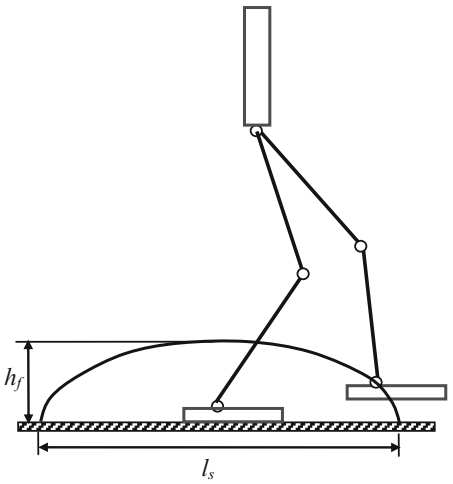


Fig. B.4 Parameters of ankle trajectories

Table B.1 Trajectory parameters for biped

	T	$x_h(0)$	$y_h(0)$	z_h	l_s	h_f
	(Sec)	(m)	(m)	(m)	(m)	(m)
Planar biped	1	−0.15	−	0.96	0.3	0.1
Spatial biped	0.5	−0.15	−0.08	0.92	0.3	0.1

B.1.3 Ankle Trajectories

Ankle trajectories are synthesized as cosine functions, i.e.,

$$x_a(t) = -l_s \cos\left(\frac{\pi}{T}t\right)$$
$$z_a(t) = \frac{h_f}{2}\left[1 - \cos\left(\frac{\pi}{T}t\right)\right] \tag{B.10}$$

In Eq. (B.10), l_s, and h_f are stride length and maximum foot height, respectively, as indicated in Fig. B.4.

For the planar and spatial bipeds discussed in Chap. 7, the parameters of hip and ankle trajectories are shown in Table B.1.

B.1.4 Joint Motions

The joint motions or variables associated with the leg of a biped are shown in Fig. B.5. Based on the hip and ankle trajectories calculated in the previous subsections, the joint motions can be obtained using inverse kinematics.

Using the geometry of a leg shown in Fig. B.6, the associated joint variables are obtained as follows:

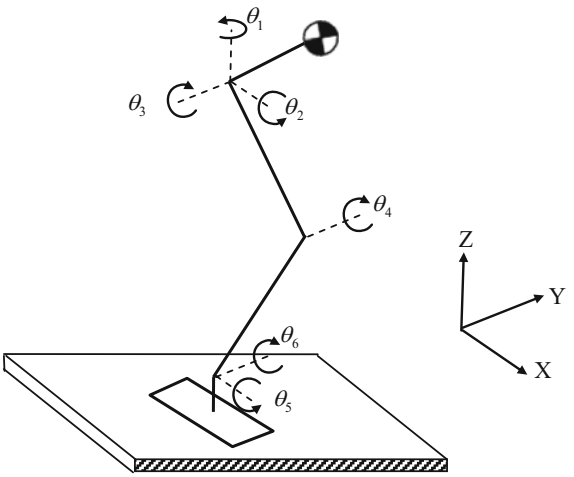


Fig. B.5 Joint motions of a biped

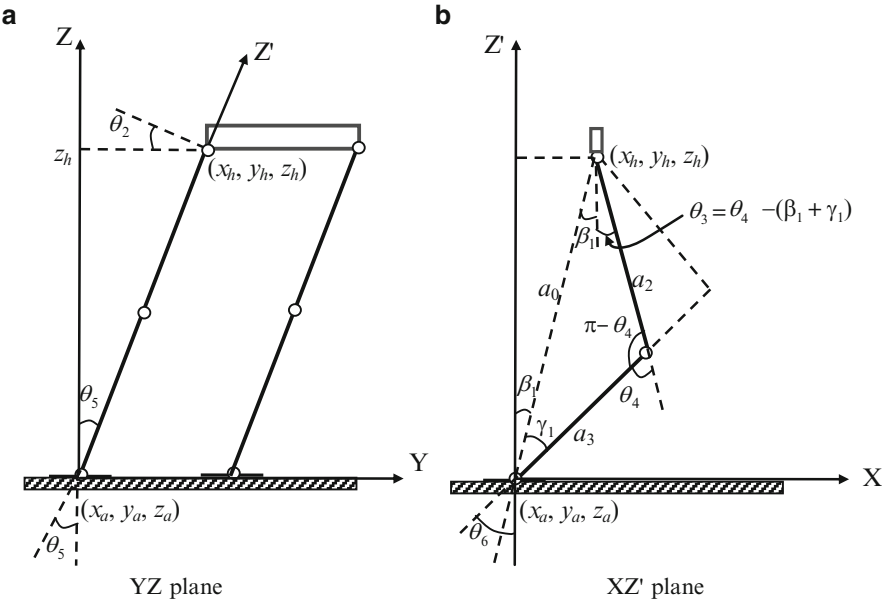


Fig. B.6 Geometry of a biped to determine the joint motions. (a) YZ plane. (b) XZ′ plane

$$\theta_2 = -\theta_5 = \tan^{-1}\left(\frac{y_h - y_a}{z_h - z_a}\right)$$

$$\theta_4 = \cos^{-1}\left(\frac{a_0^2 - a_2^2 - a_3^2}{2a_1a_1}\right) \text{where } a_0 = \sqrt{(x_h - x_a)^2 + (y_h - y_a)^2 + (z_h - z_a)^2}$$

$$\theta_3 = \theta_4 - (\beta_1 + \gamma_1), \text{where } \beta_1 = \sin^{-1}\left(\frac{x_h - x_a}{a_0}\right), \gamma_1 = \sin^{-1}\left(\frac{a_2 \sin\theta_4}{a_0}\right)$$

$$\theta_6 = \beta_1 + \gamma_1 \tag{B.11}$$

Table B.2 Trajectory parameter for legged robots

	T	$x_h(0)$	$y_h(0)$	z_h	l_s	h_f
	(Sec)	(m)	(m)	(m)	(m)	(m)
Quadruped	1	−0.15	−	0.50	0.3	0.09
Hexapod	0.5	−0.20	−	0.55	0.4	0.08

Moreover, angle θ_1, in XY plane, is assumed to be zero, i.e., $\theta_1 = 0$. Similarly, the joint motion may also be obtained for the other leg of the biped.

B.2 Quadruped and Hexapod

Motion of the trunk of the quadruped and hexapod used in Chap. 7 was also approximated by using the Inverted Pendulum Model discussed in Sect. B.1.2. The trajectory parameters for the quadruped and hexapod are shown in Table B.2.

C Energy Balance

Law of energy conservation states that energy cannot be created or destroyed, but, it can only be transformed from one form to another. Energy in a system takes different forms such as kinetic, potential, heat, etc., and the total energy remains constant. As a result, the principle of energy conservation can be used to validate the numerical results obtained during the dynamic simulation of legged robots reported in Chaps. 6, 7, and 9. For the systems studied in this thesis, energy was present in four forms. They are kinetic energy, potential energy, work done by the actuator, and the energy dissipation by the ground.

C.1 Kinetic Energy (KE) and Potential Energy (PE)

Kinetic energy of a system is a function of velocities and positions of the constituting bodies, whereas and potential is function of position only. If $\mathbf{c}_k, \dot{\mathbf{c}}_k,$ and $\boldsymbol{\omega}_k$ denote the 3-dimesional vectors of the position, linear velocity and angular velocity of the Center-of-Mass of the kth link, then the total kinetic and potential energy of the system are calculated as

$$KE = \frac{1}{2}\sum_{k=1}^{\eta}\left(m_k \dot{\mathbf{c}}_k^T \dot{\mathbf{c}}_k + \boldsymbol{\omega}_k^T \mathbf{I}_k \boldsymbol{\omega}_k\right)$$

$$PE = \sum_{k=1}^{\eta} m_k \mathbf{g}_k^T \mathbf{c}_k \tag{C.1}$$

where η is total number of links, m_k is mass of the kth link and $\mathbf{g}$ is the vector due to gravitational acceleration.

C.2 Work Done by Actuator and Energy Dissipation by Ground

Both, work done by the actuator and energy dissipation by the ground can be calculated by integrating the power delivered by the actuating torques and forces, and the power dissipated due to ground interaction, respectively. If τ_k, and $\dot{q}_k$ denote the actuating torque/force and the corresponding joint rate associated with the kth link then the energy input by the actuator may be calculated as

$$\text{Work done by actuator} = \int \left(\sum_{k=1}^{\eta} \tau_k \dot{\theta}_k\right) dt \tag{C.2}$$

Similarly, the energy dissipation by the ground is calculated as

$$\text{Energy dissipation by ground} = \int \left(\sum_{k=1}^{\eta} \mathbf{f}_k^T \mathbf{v}_k\right) dt \tag{C.3}$$

where $\mathbf{f}_k$ and $\mathbf{v}_k$ are the vectors of ground reactions and linear velocity of the contact point on the kth link.

C.3 Energy Balance

The sum of kinetic energy, potential energy, actuator work, and ground dissipation over a period of simulation time must remain constant. Hence, checking the energy balance validates the simulation results. The energy balance for the planar and spatial biped, quadruped and hexapod, simulated in Chaps. 7 and 8, are shown in Figs. C.1, C.2, C.3, and C.4. It is evident form Figs. C.1, C.2, C.3, and C.4 that the total energy remains constant throughout the simulation period. Moreover, the maximum error in the total energy is negligible, i.e., of the order of 10^{-2}–10^{-5}. This validates the simulation results.

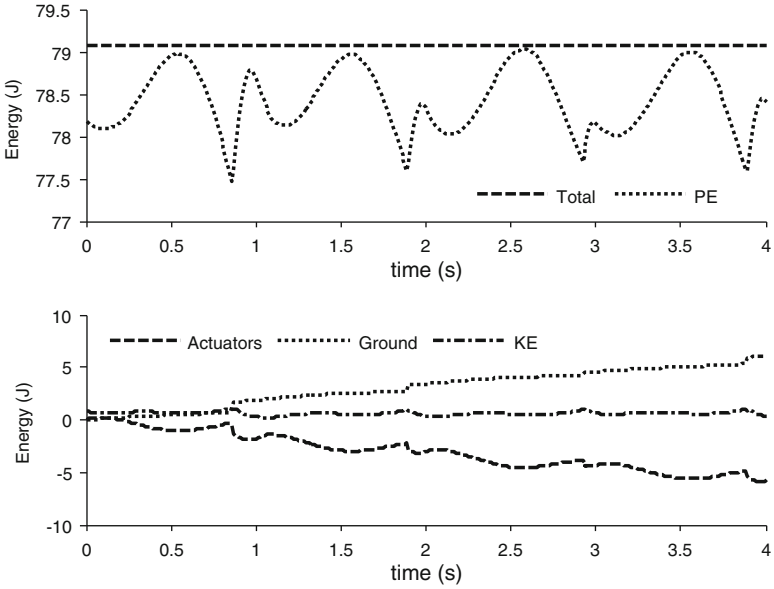


Fig. C.1 Energy balance for planar biped (Maximum error in total energy = 8.48E-04)

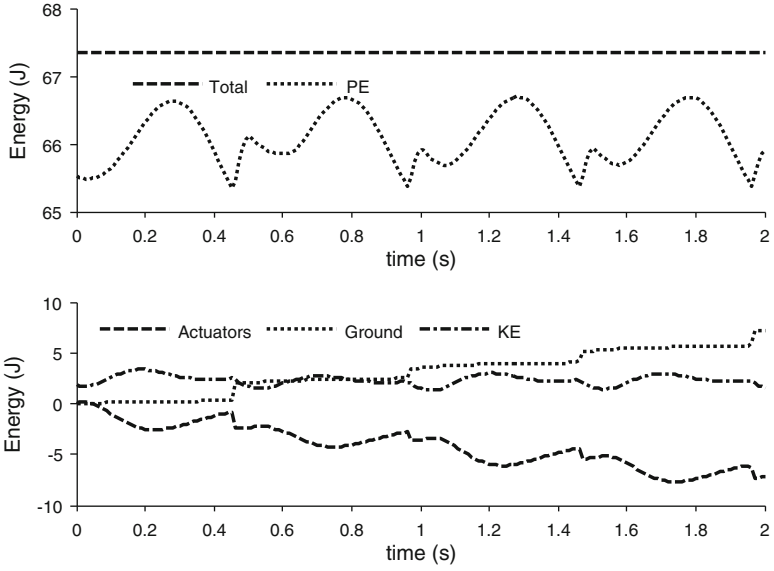


Fig. C.2 Energy balance for spatial biped (Maximum error in total energy = 3.6E-03)

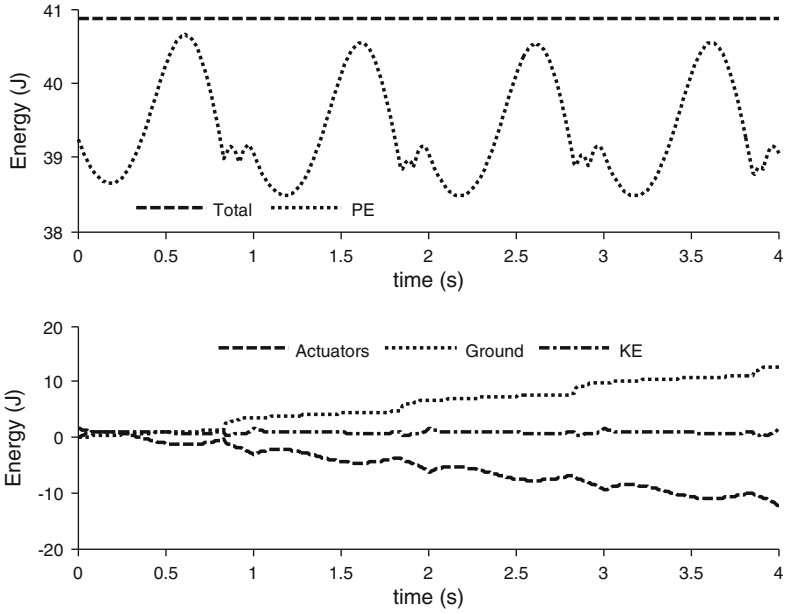


Fig. C.3 Energy balance for quadruped (Maximum error in total energy = 6.28E-05)

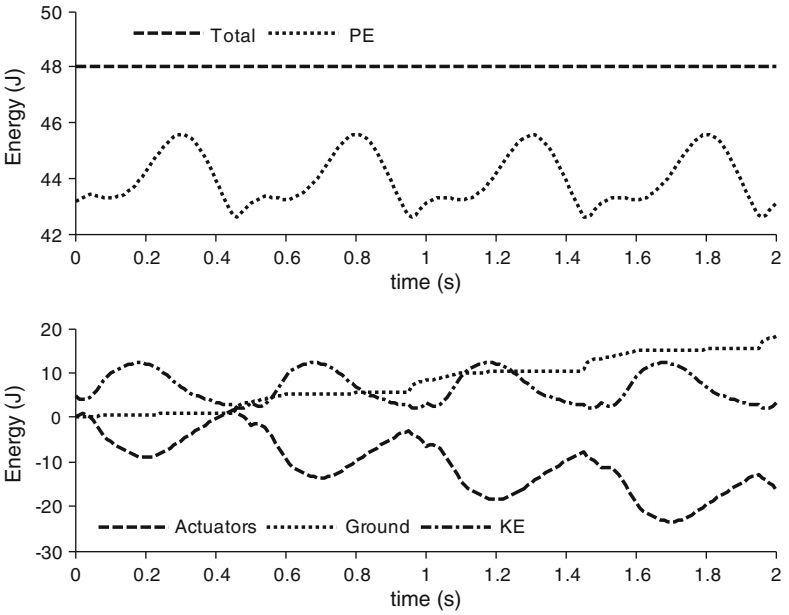


Fig. C.4 Energy balance for hexapod (Maximum error in total energy = 3.31E-02)

D Foot-Ground Interaction

As discussed in Chap. 7, methods of contact modeling can mainly be divided into analytical, impulse and penalty-based approaches. In this book, the penalty-based approach was used for contact modeling. In penalty-based approach the vertical reactions are approximated by using visco-elastic (spring-damper) model such that the foot does not bounce back while landing and the ground does not pull the leg while lift off. On the other hand, sliding of the foot is approximated by using Coulomb or viscous friction.

Various penalty based models were proposed in the literature in order to take into account the foot ground interactions. For example, Bogert et al. (1989) presented a penalty-base model, where the foot ground interaction was approximated vertically by linear spring-damper and horizontally by Coulomb friction. Later, Gerritsen et al. (1995) used nonlinear spring-damper model to simulate the heel-toe impact phase while running. Marhefka and Orin (1996) presented a contact simulation using the nonlinear damping model. Nigg and Herzog (1999), and Begg and Rahman (2000) showed typical ground reaction forces for walking and running for human foot having heel-toe shape. Mistri (2001) showed nonlinear visco-elastic model representing firm and dusty ground, which was later successfully used by Shah et al. (2006). Recently Yamane and Nakamura (2006), and Drumwright (2008) showed that a fast and stable simulation can be obtained by using the penalty-based approach.

D.1 Ground Models

The visco-elastic model used in Chap. 7 was inspired by the work of Mistri (2001) whereas the friction model was inspired by the work of Gerritsen et al. (1995). Two ground models, namely, firm and dusty ground, are introduced next.

D.1.1 Firm Ground Model

In the firm ground model, as shown in Fig. D.1, the vertical reaction is approximated by visco-elastic model, which is represented by

$$F_z = kz + c(\delta)\dot{z} \tag{D.1}$$

where k and c are spring stiffness and damping coefficient, respectively. In the above visco-elastic model spring is assumed to be linear, whereas the damper is chosen as nonlinear in a sense that the damping coefficient is a function of δ, the penetration of foot into ground. In the case of negative foot velocity, i.e., downward motion of the foot, the damping co-efficient offered by the firm ground model is assumed to remain constant as given below:

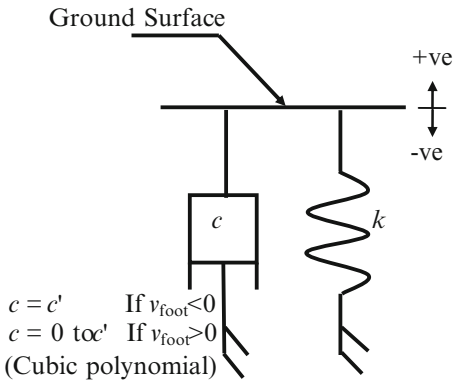


Fig. D.1 Firm ground model

$$k = k' \text{ and } c = c', \text{ if } v_{\text{foot}} < 0 \tag{D.2}$$

On the contrary, for positive foot velocity, the damping coefficient is a function of the penetration (δ), varying from zero to a maximum value at a specified penetration, and then remains constant, i.e.,

$$k = k' \text{ and } c = a_0 + a_1\delta + a_2\delta^2 + a_3\delta^3, \text{ if } v_{\text{foot}} > 0 \tag{D.3}$$

where the positive and negative foot velocities refer to upward and downward motion of the foot, and δ stands for foot penetration into the ground. Moreover, the cubic polynomial of Eq. (D.3) is solved to obtain damping coefficient c using initial conditions, (a) $c = \dot{c} = 0$ for $\delta = 0$, and (b) $c = c'$ and $\dot{c} = 0$ for $\delta = \delta_{\max}$, where c' is calculated assuming an over-damped behavior as

$$c' = 2\varsigma\sqrt{k'\sum_{i=0}^{n} m_i} \tag{D.4}$$

In Eq. (D.4), ς, k', and m_i are the over-damping factor, spring stiffness and mass of the ith link, respectively.

To avoid sliding/slipping, ground is represented by pseudo-Coulomb friction forces (Gerritsen et al. 1995) in the horizontal directions, i.e., X and Y directions represented by the horizontal plane. The pseudo-Coulomb friction force may be represented as

$$\begin{aligned} F_x &= -(\mu F_z)\frac{2}{\pi}\tan^{-1}\left(\frac{c_x\dot{x}}{\mu F_z}\frac{\pi}{2}\right) \\ F_y &= -(\mu F_z)\frac{2}{\pi}\tan^{-1}\left(\frac{c_y\dot{y}}{\mu F_z}\frac{\pi}{2}\right) \end{aligned} \tag{D.5}$$

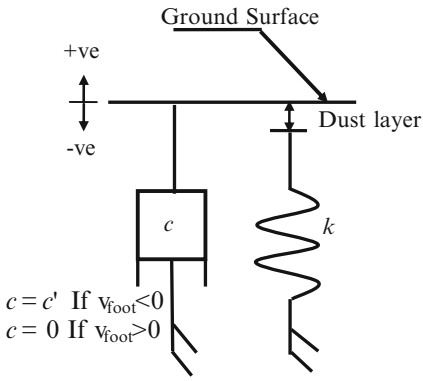


Fig. D.2 Dusty ground model

In Eq. (D.5), μ is coefficients of friction and c_x and c_y are the coefficients of damping. The above sliding model behaves as a Coulomb friction model if $\dot{x} >> \mu F_z/c_x$, and viscous friction model when $\dot{x} \leq \mu F_z/c_x$.

D.1.2 Dusty Ground Model

With the introduction of the restriction on damping coefficient, energy dissipation may not be sufficient and foot may bounce off the ground a little. In order to increase the dissipation, the dusty ground model shown in Fig. D.2 was also used. Initial sinking of the foot contributes only to energy dissipation, while the lowered spring ensures that the foot does not sink without stopping. When the foot moves upwards, damping is switched off to prevent ground pull. By changing the extent to which spring is lowered, different ground characteristics can be obtained. This model requires higher computational time for simulation. In the case of dusty ground model (Fig. D.2) values of k and c are assumed for different dust layer thickness of t_d, i.e.,

$$\begin{aligned} &k = 0 \text{ and } c = c', \text{ and } v_{\text{foot}} < 0 \text{ and } \delta > t_d \\ &k = k' \text{ and } c = c', \text{ and } v_{\text{foot}} < 0 \text{ and } \delta \leq t_d \\ &k = k' \text{ and } c = 0, \text{ if } v_{\text{foot}} > 0 \end{aligned} \tag{D.6}$$

D.2 Multi-point and Whole Body Contacts

The number of contact points to take into account for a foot depends on the contact area and shape of the foot. Quadruped and hexapod discussed in Chap. 7 have point

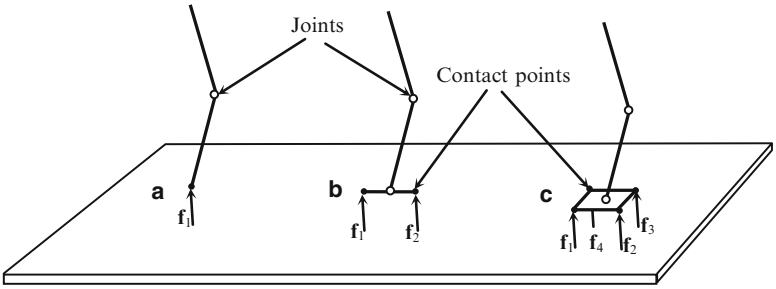


Fig. D.3 Different types of foot contact (**a**) Point contact (**b**) Line contact (**c**) Surface contact

contacts with the ground, as shown in Fig. D.3a, whereas feet of planar and spatial biped have line and surface contact, respectively, as shown in Fig. D.3b, c. As a result, one contact point per foot is required for analysis of quadruped and hexapod, on the other hand two and four contact points per foot are sufficient for analysis of planar and spatial biped, respectively.

The contact model presented above has also been extended for the modeling of whole body contact of the robot, i.e., for the links other than feet. The effect of the contact forces for the kth link having n_f contact points can then be taken into account in the dynamic model as shown in Sect. 5.1.4. In order to simulate the whole body contact, several free simulations of legged robots were performed under gravity, and these showed realistic motions.

References

Ahmadi, M., & Buehler, M. (1999). *The ARL Monopod II Running Robot: Control andEnergetic*. IEEE International conference on robotics and automation (pp. 1689–1694), Detroit, Michigan, USA.

Anderson, K. S. (1991). An order-N formulation for the motion simulation of general multi-rigid tree systems. *Journal of Computers and Structures, 46*(3), 547–559.

Anderson, K. S., & Duan, S. (2000). Highly parallelizable low order algorithm for the dynamics of complex multi rigid body systems. *Journal of Guidance Control and Dynamics, 23*(2), 355–364.

Angeles, J., & Lee, S. (1988). The formulation of dynamical equations of holonomic mechanical systems using a natural orthogonal complement. *ASME Journal of Applied Mechanics, 55*, 243–244.

Angeles, J., & Ma, O. (1988). Dynamic simulation of N-axis serial robotic manipulators using a natural orthogonal complement. *International Journal of Robotics Research, 7*(5), 32–47.

Angeles, J., Ma, O., & Rojas, A. (1989). An algorithm for the inverse dynamics of N-axis general manipulator using Kane's formulation of dynamical equations. *Computers and Mathematics with Applications, 17*(12), 1545–1561.

Armstrong, W. W. (1979). *Recursive solution to the equations of motion of an N-link manipulator*. World Congress on theory of machines and mechanisms (ASME) (vol. 2, pp. 1343–1346), Montreal, Canada.

Asada, H. (1984). Dynamic analysis and design of robot manipulators using inertia ellipsoids. *IEEE International Conference on Robotics and Automation, 1*, 94–102.

Ascher, U. M., Pai, D. K., & Cloutier, B. P. (1997). Forward dynamics, elimination methods, and formulation stiffness in robot simulation. *International Journal of Robotics Research, 16*(6), 749–758.

Automated Dynamic Analysis of Mechanical System (ADAMS). (2004). Version 2005.0.0, MSC. Software.

Bae, D. S., & Haug, E. J. (1987). A recursive formulation for constrained mechanical system dynamics: Part I, open loop systems. *Mechanics of Structure and Machine, 15*, 359–382.

Bae, D., Kuhl, J. G., & Haug, E. J. (1988). A recursive formation for constrained mechanical system dynamics: Part III, parallel processing implementation. *Mechanisms, Structures, and Machines, 16*, 249–269.

Balafoutis, C. A., & Patel, R. V. (1991). *Dynamic analysis of robot manipulators: A Cartesian tensor approach*. Boston: Kluwer Academic.

Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of SIGGRAPH*, Orlando, FL.

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7,

Baraff, D. (1996). Linear-time dynamics using Lagrange multipliers. In *Proceedings of ACM SIGGRAPH* (pp. 137–146), New Orleans.

Baumgarte, J. (1972). Stabilization of constraints and integrals of motion. *Computer methods in Applied Mechanics and Engineering, 1*, 1–16.

Begg, K. R., & Rahman, S. M. (2000). A method for the reconstruction of the ground reaction force-time characteristics during gait from force platform recording of simultaneous foot faults. *IEEE Transactions on Biomedical Engineering, 47*(4), 547–551.

Berkemeier, M. D. (1998). Modeling the dynamics of quadrupedal running. *International Journal of Robotics Research, 17*, 971–985.

Bhagat, R., Choudhury, S. B., & Saha, S. K. (2011). *Design and development of a 6-DOF parallel manipulator*. International conference on multuibody dynamics (pp. 15–24), Vijaywada, India.

Bhangale, P. P., Saha, S. K., & Agrawal, V. P. (2004). A dynamic model based robot arm selection criterion. *International Journal of Multibody System Dynamics, 12*(2), 95–115.

Bicchi, A. (2000). Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Transactions on Robotics and Automation, 16*(6), 652–662.

Blajer, W., Bestle, D., & Schiehlen, W. (1994). An orthogonal complement matrix formulation for constrained multibody systems. *ASME Journal of Mechanical Design, 116*, 423–428.

Bogert, A. J., Schamhardt, H. C., & Crowe, A. (1989). Simulation of quadrupedal locomotion using a rigid body model. *Journal of Biomechanics, 22*(1), 33–41.

Brandl, H., Johanni, R., & Otter, M. (1988). A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. In *Theory of robots* (pp. 95–100). Oxford: Pergamon Press.

Brown, B., & Zeglin, G., (1998). *The Bow Leg Hopping robot*. IEEE International conference on robotics and automation (pp. 781–786), Leuven, Belgium.

Buehler, M., Battaglia, R., Cocosco, A., Hawker, G., Sarkis, J., & Yamazaki, K. (1998). *SCOUT: A simple quadruped that walks, climbs and runs*. IEEE international conference on Robotics and Automation (pp. 1707–1712), Leuven, Belgium.

Buehler, M., Cocosco, A., Yamazaki, K., & Battaglia, R. (1999). *Stable open loop walking in quadruped robots with stick legs*. IEEE international conference on robotics and automation (pp. 2348–2353), Detroit, Michigan, USA.

Cameron, J. M., & Book, W. J. (1997). Modeling mechanisms with nonholonomic joints using the Boltzmann-Hammel equations. *International Journal of Robotics Research, 16*(1), 47–59.

Cham, J. G., Bailey, S. A., Clark, J. E., Full, R. J., & Cutkosky, M. R. (2002). Fast and robust: Hexapedal robots via shape deposition manufacturing. *International Journal of Robotics Research, 21*(10), 869–882.

Chaudhary, H., & Saha, S. K. (2007). Constraint wrench formulation for closed-loop systems using Two-level recursions. *ASME Journal of Mechanical Design, 129*, 1234–1242.

Chaudhary, H., & Saha, S. K. (2009). *Dynamics and balancing of multibody systems*. Berlin: Springer.

Collins, S. H., & Ruina, A. (2005). *A Bipedal walking robot with efficient sand human-like gait*. IEEE international conference on robotics and automation (pp. 1983–1988).

Coset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). *Principle of robot motion: Theory, algorithms, and implementations*. Cambridge, MA: MIT Press.

Craig, J. J. (2006). *Introduction to robotics, mechanics and control*. Delhi: Pearson Education.

Critchley, J. S., & Anderson, K. S. (2003). A generalized recursive coordinate reduction method for multibody system dynamics. *Multibody System Dynamics, 9*, 185–212.

Critchley, J. H., & Anderson, K. S. (2004). Parallel logarithmic order algorithm for general multibody system dynamics. *Journal of Multiscale Computational Engineering, 12*(1), 75–93.

Cyril, X. (1988). *Dynamics of flexible link manipulators*. Ph.D. dissertation, McGill University, Canada

Denavit, J., & Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics, 22*, 215–221.

Deo, N. (1974). *Graph theory with application in engineering and computer science*. Englewood Cliffs: Prentice-Hall.

Doi, T., Hodoshima, R., Hirose, S., Fukuda, Y., Okamoto, T., & Mori, J. (2005). *Development of a Quadruped walking robot to work on steep slopes, TITAN XI*. IEEE/RSJ international conference on intelligent robots and systems (pp. 2067–2072), Edmonton, Alberta, Canada.

Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge–Kutta formulae. *Journal Computational and Applied Mathematics, 6*, 19–26.

Drumwright, E. (2008). A fast and stable penalty method for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics, 14*(1), 231–240.

Duffy, J. (1978). Displacement analysis of the generalized RSSR mechanism. *Mechanism and Machine Theory, 13*, 533–541.

Eberhard, P., & Schiehlen, W. (2006). Computational dynamics of multibody systems: History, formalisms, and applications. *ASME Journal of Computational and Nonlinear Dynamics, 1*(1), 3–12.

Espenschied, K. S., Quinn, R. D., Beer, R. D., & Chiel, H. J. (1996). Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. *Robotics and Autonomous Systems, 18*(1–2), 59–64.

Fang, A. C., & Pollard, N. S. (2003). Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (SIGGRAPH), 22*(3), 417–426.

Featherstone, R. (1983). The calculation of robotic dynamics using articulated body inertias. *International Journal of Robotics Research, 2*, 13–30.

Featherstone, R. (1987). *Robot dynamics algorithms*. Boston: Kluwer Academic.

Featherstone, R. (1999). A divide-and-conquer articulated body algorithm for parallel O.Log.N calculation of rigid body dynamics. Part 1: Basic algorithm. *International Journal of Robotics Research, 18*(9), 867–875.

Featherstone, R. (2005). Efficient factorization of the joint-space inertia matrix for branched kinematic tree. *International Journal of Robotics Research, 24*(6), 487–500.

Featherstone, R., & Orin, D. (2000). Robot dynamics: Equations and algorithms. *IEEE International Conference on Robotics and Automation, 1*, 826–834.

Fijany, A., Sharf, I., & D'Eleuterio, G. M. T. (1995). Parallel O(Log N) algorithms for computation of manipulator forward dynamics. *IEEE Transactions on Robotics and Automation, 11*(3), 389–400.

Freeman, P. S., & Orin, D. E. (1991). Efficient dynamic simulation of a quadruped using a decoupled tree structured approach. *International Journal of Robotics Research, 10*, 619–627.

Fritzkowski, P., & Kaminski, H. (2008). Dynamics of a rope as a rigid multibody system. *Journal of Mechanics of Materials and Structures, 3*(6), 1059–1075.

Fritzkowski, P., & Kamiński, H. (2010). Dynamics of a rope modeled as a multi-body system with elastic joints. *Computational Mechanics, 46*(6), 901–909.

Fukuoka, Y., Kimura, H., & Cohen, A. H. (2003). Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *International Journal of Robotics Research, 22*, 187–202.

Furukawa, N., Namiki, A., Taku, S., & Ishikawa, M. (2006). Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system. *IEEE International Conference on Robotics and Automation, 2006*, 181–187.

Furusho, J., Sano, A., Sakaguchi, M., & Koizumi, E. (1995). *Realization of bounce gait in a Quadruped robot with articular-joint-type legs*. IEEE international conference on robotics and automation (pp. 697–702), Nagoya, Aichi, Japan.

Gambaryan, P. P. (1974). *How mammals run: Anatomical adaptations*. New York: Wiley.

Gatti-Bono, C., & Perkins, N. C. (2002). Physical and numerical modelling of the dynamic behavior of a fly line. *Journal of Sound and Vibration, 255*(3), 555–577.

Gerritsen, K. G. M., Van Den Bogert, A. J., & Nigg, B. M. (1995). Direct dynamics simulation of the impact phase in heel-toe running. *Journal of Biomechanics, 28*(6), 661–668.

Goldenberg, A. A., Benhabib, B., & Fenton, R. G. (1985). Complete generalized solution to the inverse kinematics of robots. *IEEE Journal Robotics and Automation, RA-1*(1), 14–20.

Greenwood, D. T. (1988). *Principles of dynamics*. New Delhi: Prentice-Hall.

Harada, K., Kajita, S., Kanehiro, F., Fujiwara, K., Kaneko, K., Yokoi, K., & Hirukawa, H. (2004). Real-time planning of humanoid robot's gait for force controlled manipulation. *IEEE International Conference on Robotics and Automation, 1*, 616–622.

Hasegawa, Y., Higashiura, M., & Fukuda, T. (2003). Simplified generation algorithm of regrasping motion – Performance comparison of online-searching approach with EP-based approach. *IEEE International Conference on Robotics and Automation, 2*, 1811–1816.

He, G., Tan, X., Zhang, X., & Lu, Z. (2008). Modeling, motion planning, and control of one-legged hopping robot actuated by two arms. *Mechanism and Machine Theory, 43*(1), 33–49.

Hemami, H., & Weimer, F. C. (1981). Modeling of nonholonomic dynamic systems with applications. *ASME Journal of Applied Mechanics, 48*(1), 177–182.

Hirai, K., Hirose, M., Haikawa, Y., & Takenaka, T. (1998). *The development of Honda humanoid robot*. IEEE international conference on robotic and automation (pp. 1321–1326), Leuven, Belgium.

Hollerbach, J. M. (1980). A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics, 10*(11), 730–736.

Hollerbach, J. M., & Gideon, S. (1983). Wrist-partitioned inverse kinematic accelerations and manipulator dynamics. *International Journal of Robotics Research, 4*, 61–76.

http://ijts-jrirc.blogspot.ca/2010/11/history-of-industrial-robotics.html. Retrieved on May 7, 2012

http://www.dipity.com/RoboticsResearch/History-of-Robotics/?mode=fs. Retrieved on May 7, 2012

Hu, W., Marhefka, D. W., & Orin, D. E. (2005). Hybrid kinematic and dynamic simulation of running machines. *IEEE Transactions on Robotics, 21*(3), 490–497.

Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Aral, H., Koyachi, N., & Tanie, K. (2001). Planning walking patterns for a biped robot. *IEEE Transactions on Robotics and Automation, 17*(3), 280–289.

Huston, R. L., & Passerello, C. E. (1974). On constraint equations-a new approach. *ASME Journal of Applied Mechanics, 41*, 1130–1131.

Hyon, S., Emura, T., & Mita, T. (2003). Dynamics-based control of a one-legged hopping robot. *Journal of Systems and Control Engineering, 217*(2), 83–98.

Jacobsen, S. C, Iversen, E. K., Knutti, D. F., Johnson, R. T., & Biggers, K. B. (1986). *Design of the Utah/MIT Dextrous Hand*. IEEE international conference on robotics & automation (pp. 1520–1532), San Francisco, California, USA.

Kagami, S., Kitagawa, T., Nishiwaki, K., Sugihara, T., Inaba, M., & Inoue, H. (2002). A fast dynamically equilibrated walking trajectory generation method of humanoid robot. *Autonomous Robots, 12*(1), 71–82.

Kahn, M. E., & Roth, B. (1971). The near minimum-time control of open-loop articulated kinematic chains. *ASME Journal of Dynamics Systems Measurement and Control, 91*, 164–172.

Kahtib, O., & Burdick, J. (1987). Optimization of dynamics in manipulator design: The operational space formulation. *International Journal of Robotics and Automation, 2*(2), 90–98.

Kajita, S., & Tani, K. (1991). *Study of dynamic biped locomotion on rugged Terrain*. IEEE international conference on robotic and automation (pp. 1405–1411), Sacramento, California, USA.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. *IEEE International Conference on Robotics and Automation, 2*, 1620–1626.

Kamman, J. W., & Huston, R. L. (1984). Constrained multibody system dynamics: An automated approach. *Computers and Structures, 18*(6), 999–1003.

Kane, T. R., & Levinson, D. A. (1983). The use of Kane's dynamical equations for robotics. *International Journal of Robotics Research, 2*(3), 3–21.

Kaneko, K., Harada, K., Kanehiro, F., Miyamori, G., & Akachi, K. (2008). *Humanoid robot HRP-3*. IEEE/RSJ international conference on intelligent robots and systems (pp. 2471–2478), Nice, France.

Kelly, R., Santibanez, V., & Loria, A. (2005). *Control of robot manipulators in joint space*. London: Springer.

Khalil, W., & Kleinfinger, J. (1986). A new geometric notation for open and closed-loop robots. *IEEE International Conference on Robotics and Automation, 3*, 1174–1179.

Khalil, W., Kleinfinger, J. F., & Gautier, M. (1986). *Reducing the computational burden of the dynamical models of robots*. IEEE international conference on robotics & automation (pp. 525–531), San Francisco, California, USA.

Khan, W. A., Krovi, V. N., Saha, S. K., & Angeles, J. (2005). Recursive kinematics and inverse dynamics for a planar 3R parallel manipulator. *Journal of Dynamic Systems, Measurement, and Control, 127*(4), 529–536.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research, 5*(1), 90–98.

Khatib, O. (1987). Unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation, RA-3*(1), 43–53.

Kim, S. S., & Vanderploeg, M. J. (1986). A general and efficient method for dynamic analysis of mechanical systems using velocity transformations. *ASME Journal of Mechanisms, Transmissions, and Automation in Design, 108*(6), 176–182.

Kimura, H., Akiyama, S., & Sakurama, K. (1999). Realization of dynamic walking and running of the quadruped using neural oscillator. *Autonomous Robots, 7*(3), 247–258.

Kimura, H., Fukuoka, Y., & Cohen, A. H. (2007). Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *International Journal of Robotics Research, 26*(5), 475–490.

Kurazume, R., Hirose, S., & Yoneda, K. (2001). Feedforward and feedback dynamic trot gait control for a quadruped walking vehicle. *IEEE International Conference on Robotics and Automation, 3*, 3172–3180.

Kurazume, R., Hasegawa, T., & Yoneda, K. (2003). The sway compensation trajectory for a biped robot. *IEEE International Conference on Robotics and Automation, 1*, 925–931.

Kuroki, Y., Fujita, M., Ishida, T., Nagasaka, K., & Yamaguchi, J. (2003). *A small biped entertainment robot exploring attractive applications*. IEEE international conference on robotics and automation (pp. 471 –476).

Kwon, O., & Park, J. H. (2009). Asymmetric trajectory generation and impedance control for running of biped robots. *Autonomous Robots, 26*(1), 47–78.

Lee, K., & Chirikjian, S. G. (2005). *A new perspective on O(N) mass-matrix Inversion for serial revolute manipulators*. IEEE conference on robotics and automation (pp. 4733–4737), Barcelona, Spain.

Lewis, F. L., Dawson, D. M., & Abdallah, C. T. (2004). *Robot manipulator control: Theory and practice*. New York: Marcel Dekker Inc.

Li, C., & Sankar, T. S. (1992). Fast inverse dynamics computation in real-time robot control. *Mechanism and Machine Theory, 27*(6), 741–750.

Lilly, K. W. (1993). *Efficient dynamic simulation of robotic mechanisms*. Boston: Kluwer Academic.

Lilly, K. W., & Orin, D. E. (1991). Alternate formulations for the manipulator inertia matrix. *International Journal of Robotics Research, 10*(1), 64–74.

Lloyd, J. (2005). *Fast implementation of Lemke's algorithm for rigid body contact simulation*. IEEE international conference on robotics and automation (pp. 4538–4543), Barcelona, Spain.

Luh, J. Y. S., Walker, M. W., & Paul, R. P. C. (1980). On-Line computational scheme for mechanical manipulators. *ASME Journal Of Dynamic Systems Measurement and Control, 102*, 69–76.

Ma, O., & Angeles, J. (1990). The concept of dynamic isotropy and its applications to inverse kinematics and trajectory planning. *IEEE International Conference on Robotics and Automation, 1*, 481–486.

Mani, N. K., Haug, E. J., & Atkinson, K. E. (1985). Application of singular value decomposition for analysis of mechanical system dynamics. *ASME Journal of Mechanisms, Transmissions, and Automation in Design, 107*(1), 82–87.

Marhefka, D. W., & Orin, D. E. (1996). Simulation of contact using a nonlinear damping model. *IEEE International Conference on Robotics and Automation, 2*, 1662–1668.

Marhefka, D. W., Orin, D. E., Schmiedeier, J. P., & Waldron, K. J. (2003). Intelligent control of quadruped gallops. *IEEE/ASME Transactions on Mechatronics, 8*(4), 446–456.

Mason, M., & Salisbury, J. K. (1985). *Robot hands and the mechanics of manipulation*. Cambridge, MA: MIT Press.

Matlab. (2009). *Version 7.4 Release 2009a*, MathWorks Inc.

McGeer, T. (1990). Passive dynamic walking. *International Journal of Robotics Research, 9*(2), 62–82.

McMillan, S., & Orin, D. E. (1995). Efficient computation of articulated-body inertias using successive axial screws. *IEEE Transactions on Robotics and Automation, 11*(2), 606–611.

McMillan, S., & Orin, D. E. (1998). *Forward dynamics of Multilegged vehicles*. IEEE international conference on robotics and automation(pp. 464–470).

McMillan, S., Orin, D. E., & McGhee, R. B. (1995). Efficient dynamic simulation of an underwater vehicle with a robotic manipulator. *IEEE Transactions on Systems, Man, and Cybernetics, 25*(8), 1194–1205.

McPhee, J. J. (1996). On the use of linear graph theory in multibody system dynamics. *Nonlinear Dynamics, 9*, 73–90.

Mirtich, B., & Canny, J. (1995). *Impulse-based simulation of rigid bodies*. Symposium on interactive 3D graphics (pp. 181–188), Monterey, CA.

Mistri, B. D. (2001). *Simulation of Quadruped Gallop*. M.Tech. thesis. Bombay: Mechanical Department, Indian Institute of Technology.

Miura, H., & Shimoyama, I. (1984). Dynamic walk of a biped. *International Journal of Robotics Research, 3*(2), 60–74.

Miura, H., Shimoyama, I., Mitsuishi, M., & Kimura, H. (1985). Dynamical walk of quadruped robot (collie-1). In *International symposium on robotics research* (pp. 317–324). Cambridge: MIT Press.

Mohan, A., & Saha, S. K. (2007). A recursive, numerically stable, and efficient algorithm for serial robots. *Multibody System Dynamics, 17*(4), 291–319.

Morisawa, M., Kajita, S., Kaneko, K., Harada, K., Kanehiro, F., Fujiwara, K., & Hirukawa, H. (2005). *Pattern generation of biped walking constrained on parametric surface*. IEEE international conference on robotics and automation (pp. 2405–2410), Barcelona, Spain.

Muybridge, E. (1957). *Animals in motion*. New York: Dover Publications.

Nelson, G. M., & Quinn, R. D. (1998). *Posture control of a cockroach-like robot*. IEEE international conference on robotics and automation (pp. 157–162), Leuven, Belgium.

Nigg, B. M., & Herzog, W. (1999). *Biomechanics of the musculo-skeletal system*. Chichester: Wiley.

Nikravesh, P. E. (1988). *Computer-aided analysis of mechanical systems*. Englewood Cliffs: Prentice-Hall.

Nikravesh, P. E., & Gim, G. (1993). Systematic construction of the equations of motion for multibody systems containing closed kinematic loops. *ASME Journal of Mechanical Design, 115*, 143–149.

Ono, K., Takahashi, R., & Shimada, T. (2001). Self-excited walking of a biped mechanism. *International Journal of Robotics Research, 20*(12), 953–966.

Ottaviano, E., Ceccarelli, M., & Tavolieri, C. (2004). *Kinematic and dynamic analyses of a pantograph-leg for a biped walking machine*. International conference on climbing and walking robots CLAWAR, Madrid.

Ouezdou, F. B., Bruneau, O., & Guinot, J. C. (1998). Dynamic analysis tool for legged robots. *Multibody System Dynamics, 2*(4), 369–391.

Park, J. H., & Kim, K. D. (1998). *Biped robot walking using gravity-compensated inverted pendulum mode and computed torque control*. IEEE international conference on robotics and automation (pp.3528–3533), Leuven, Belgium.

Park, F. C., Bobrow, J. E., & Ploen, S. R. (1995). A Lie Group formulation of robot dynamics. *International Journal of Robotics Research, 14*(6), 606–618.

Perrin, B., Chevallereau, C., & Verdier, C. (1997). *Calculation of the direct dynamic model of walking robots: Comparison between two methods*. IEEE international conference on robotics and automation (pp. 1088–1093), Raleigh, North Carolina, USA.

Pfeiffer, F. (1996). Grasping with hydraulic fingers – An example of mechatronics. *IEEE/ASME Transactions on Mechatronics, 1*(2), 158–167.

Pons, J. L., Ceres, R., & Pfeiffer, F. (1999). Multifingered dextrous robotics hand design and control: A review. *Robotica, 17*(6), 661–674.

Poulakakis, I., Papadopoulos, E., & Buehler, M. (2006). On the stability of the passive dynamics of quadrupedal running with a bounding gait. *International Journal of Robotics Research, 25*(7), 669–687.

Raibert, M. H. (1984). Hopping in legged system- modeling and simulation for the two-Dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics, 14*(3), 451–463.

Raibert, M. H. (1986). *Legged robot that balance*. Cambridge, MA: MIT Press.

Raibert, M. H. (1990). Trotting, pacing and bounding by a quadruped robot. *Journal of Biomechanics, 23*(1), 79–98.

Raibert, M., Tzafestas, S., & Tzafestas, C. (1993). Comparative simulation study of three control techniques applied to a biped robot. *IEEE International Conference on Systems Man and Cybernetics, 1*, 494–502.

RecurDyn. (2009). *Multibody Simulation Software*. FunctionBay Inc.

Ridderström, C., Ingvast, J., Hardarson, F., Gudmundsson, M., Hellgren, M., Wikander, J., Wadden, T., & Rehbinder, H. (2000). *The basic design of the quadruped robot Warp1*. International conference on climbing and walking robots, Madrid, Spain.

Ringrose, R. (1997). Self-stabilizing running. *IEEE International Conference on Robotics and Automation, 1*, 487–493.

Roberson, R. E., & Schwertassek, R. (1988). *Dynamics of multibody systems*. Berlin: Springer.

Rodriguez, G. (1987). Kalman filtering, smoothing, and recursive robot Arm forward and inverse dynamics. *IEEE Journal of Robotics and Automation, 3*(6), 624–639.

Rodriguez, G., Jain, A., & Kreutz-Delgado, K. (1991). A spatial operator algebra for manipulator modeling and control. *International Journal of Robotics Research, 10*(4), 371–381.

Rodriguez, G., Jain, A., & Kreutz-Delgado, K. (1992). Spatial operator algebra for multibody system dynamics. *Journal of the Astronautical Sciences, 40*(1), 27–50.

Rosenthal, D. E. (1990). An order n formulation for robotic systems. *Journal of Astronautical Sciences, 38*(4), 511–529.

Saha, S. K. (1997). A decomposition of the manipulator inertia matrix. *IEEE Transactions on Robotics and Automation, 13*(2), 301–304.

Saha, S. K. (1999a). Analytical expression for the inverted inertia matrix of serial robots. *International Journal of Robotic Research, 18*(1), 116–124.

Saha, S. K. (1999b). Dynamics of serial multibody systems using the decoupled natural orthogonal complement matrices. *ASME Journal of Applied Mechanics, 66*, 986–996.

Saha, S. K. (2003). Simulation of industrial manipulators based on the UDU^T decomposition of inertia matrix. *International Journal of Multibody System Dynamics, 9*(1), 63–85.

Saha, S. K. (2008). *Introduction to robotics*. New Delhi: Tata Mcgraw Hill.

Saha, S. K., & Angeles, J. (1991). Dynamics of nonholonomic mechanical systems using a natural orthogonal complement. *ASME J of Applied Mechanics, 58*, 238–243.

Saha, S. K., & Schiehlen, W. O. (2001). Recursive kinematics and dynamics for closed loop multibody systems. *International Journal of Mechanics of Structures and Machines, 29*(2), 143–175.

Saha, S. K., Shirinzadeh, B., & Gl, A. (2006). *Dynamic model simplification of serial manipulators*. Mexico: ISRA.

Sakagami, Y., Watanabe,R., Aoyama, C., Matsunaga, S., Higaki, N., & Fujimura, K. (2002). *The intelligent ASIMO: System overview and integration*. IEEE international conference on intelligent robots and systems (pp. 2478–2483), Lausanne, Switzerland.

Salisbury, J. K., & Craig, J. J. (1982). Articulated hands: Force and kinematic issues. *International Journal Robotics Research, 1*(1), 4–17.

Saranli, U., Buehler, M., & Koditschek, D. E. (2001). Rhex – A simple and highly mobile hexapod robot. *International Journal of Robotics Research, 20*(7), 616–631.

Saranli, U., Rizzi, A. A., & Koditschek, D. E. (2004). Model-based dynamic self-righting maneuvers for a hexapedal robot. *International Journal of Robotics Research, 23*(9), 903–918.

Schiehlen, W. (1990). *Multibody systems handbook*. Berlin: Springer.

Schiehlen, W. (1997). Multibody system dynamics: Roots and perspectives. *Multibody System Dynamics, 1*, 49–188.

Shabana, A. A. (2001). *Computational dynamics*. New York: Wiley.

Shah, S. V. (2011). *Modular framework for dynamics modeling and analysis of tree-type robotic system*. Ph.D. thesis. Delhi: Mechanical Engineering Department, IIT.

Shah, S. V., Mistri, B. D., & Issac, K. K. (2006). *Evaluation of foot ground interaction model using monopod forward hopping*. International conference on recent trends in automation and its adaptation to industries, Nagpur, India

Shah, S. V., Saha, S. K., & Dutt, J. K. (2009). *Denavit-Hartenberg parameters of Euler-Angle-Joints for order (N) recursive forward dynamics*. ASME International conference on multibody systems, nonlinear dynamics and control, USA

Shah, S. V., Saha, S. K., & Dutt, J. K. (2012a). Modular framework for dynamics of tree-type legged robots. *Mechanism and Machine Theory, Elsevier, 49*, 234–255.

Shah, S. V., Saha, S. K., & Dutt, J. K. (2012b). Denavit-Hartenberg (DH) Parametrization of Euler Angles. *ASME J of Nonlinear and Computational Dynamics, 7*(2).

Shih, C. L., Gruver, W. A., & Lee, T. T. (1993). Inverse kinematics and inverse dynamics for control of a biped walking machine. *Journal of Robotic Systems, 10*(4), 531–555.

Shuster, M. D. (1993). A survey of attitude representations. *Journal of the Astronautical Sciences, 41*(4), 439–517.

Shuster, M. D., & Oh, S. D. (1981). Three-axis attitude determination from vector observation. *Journal of Guidance, Control and Dynamics, 4*(1), 70–77.

Singla, P., Mortari, D., & Junkins, J. L. (2004). *How to avoid singularity for Euler Angle Set?*. AAS space flight mechanics conference, Hawaii.

Stejskal, V., & Valasek, M. (1996). *Kinematics and dynamics of machinery*. New York: Marcel Dekkar Inc.

Stelzle, W., Kecskeméthy, A., & Hiller, M. (1995). A comparative study of recursive methods. *Archive of Applied Mechanics, 66*, 9–19.

Stewart, G. W. (1973). *Introduction to matrix computations*. Orlando: Academy Press.

Stewart, D., & Trinkle, J. (2000). *An implicit time-stepping scheme for rigid body dynamics with coulomb friction*. IEEE international conference on robotics and automation (pp. 162–169), San Francisco, California, USA.

Stokes, A., & Brockett, R. (1996). Dynamics of kinematic chains. *International Journal of Robotics Research, 15*, 393–405.

Strang, G. (1998). *Linear algebra and its applications*. Orlando: Harcourt, Brace, Jovanovich, Publisher.

Sugihara, T., Nakamura, Y., & Inoue, H. (2002). Realtime Humanoid motion generation through ZMP manipulation based on inverted pendulum control. *IEEE International Conference on Robotics and Automation, 2*, 1404–1409.

Uicker, Jr., J.J. (1965). *On the dynamic analysis of spatial linkages using 4x4 matrices*. Ph.D. thesis. Evanston: Northwestern University.

Vereshchagin, A. F. (1975). Gauss principle of least constraint for modeling the dynamics of automatic manipulators using a digital computer. *Soviet Physics – Doklady, 20*(1), 33–34.

Vukobratovic, M., Borovac, B., Surla, D., & Stokic, D. (1989). *Biped locomotion: Dynamics, stability, control and application*. Berlin: Springer.

Vukobratovic, M., Potkonjak, V., Babkovic, K., & Borovac, B. (2007). Simulation model of general human and humanoid motion. *Multibody System Dynamics, 17*(1), 71–96.

Walker, M. W., & Orin, D. E. (1982). Efficient dynamic computer simulation of robotic mechanisms. *ASME Journal of Dynamic Systems, Measurement and Control, 104*, 205–211.

Wehage, R. A., & Haug, E. J. (1982). Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *ASME Journal of Mechanical Design, 104*, 247–255.

Wisse, M., Schwab, A. L., van der Linde, R. Q., & van der Helm, F. C. T. (2005). How to keep from falling forward: Elementary swing leg action for passive dynamic walkers. *IEEE Transactions on Robotics, 21*(3), 393–401.

Wittenburg, J. (2008). *Dynamics of multibody systems*. Berlin: Sprnger.

Wong, C. W., & Yasui, K. (2006). Falling chains. *American Journal of Physics, 74*(6), 490–496.

Yamaguchi, J., Soga, E., Inoue, S., & Takanishi, A. (1999). *Development of a bipedal humanoid robot – Control method of whole body cooperative dynamic biped walking*. IEEE international conference on robotics and automation (pp. 368–374), Detroit, Michigan, USA.

Yamane, K., & Nakamura, Y. (1999). Dynamics computation of structure-varying kinematic chains for motion synthesis of humanoid. *IEEE International Conference on Robotics and Automobiles, 1*, 714–721.

Yamane, K., & Nakamura, Y. (2002). Efficient parallel dynamics computation of human figures. *IEEE International Conference on Robotics and Automation, 1*, 530–537.

Yamane, K., & Nakamura, Y. (2006). *Stable penalty-based model of frictional contacts*. IEEE international conference on robotics and automation (pp. 1904–1909), Orlando, Florida, USA.

Yen, J., & Petzold, L. R. (1998). An efficient Newton-type iteration for the numerical solution of highly oscillatory constrained multibody dynamic systems. *SIAM Journal on Scientific Computing, 19*(5), 1513–1534.

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *International Journal of Robotics Research, 4*(2), 3–9.

Yu, Q., & Chen, I. M. (2000). A direct violation correction method in numerical simulation of constrained multibody systems. *Computational Mechanics, 26*(1), 52–57.

Index

S.V. Shah et al., *Dynamics of Tree-Type Robotic Systems*, Intelligent Systems, Control and Automation: Science and Engineering 62, DOI 10.1007/978-94-007-5006-7,