

# Planar Multibody Dynamics

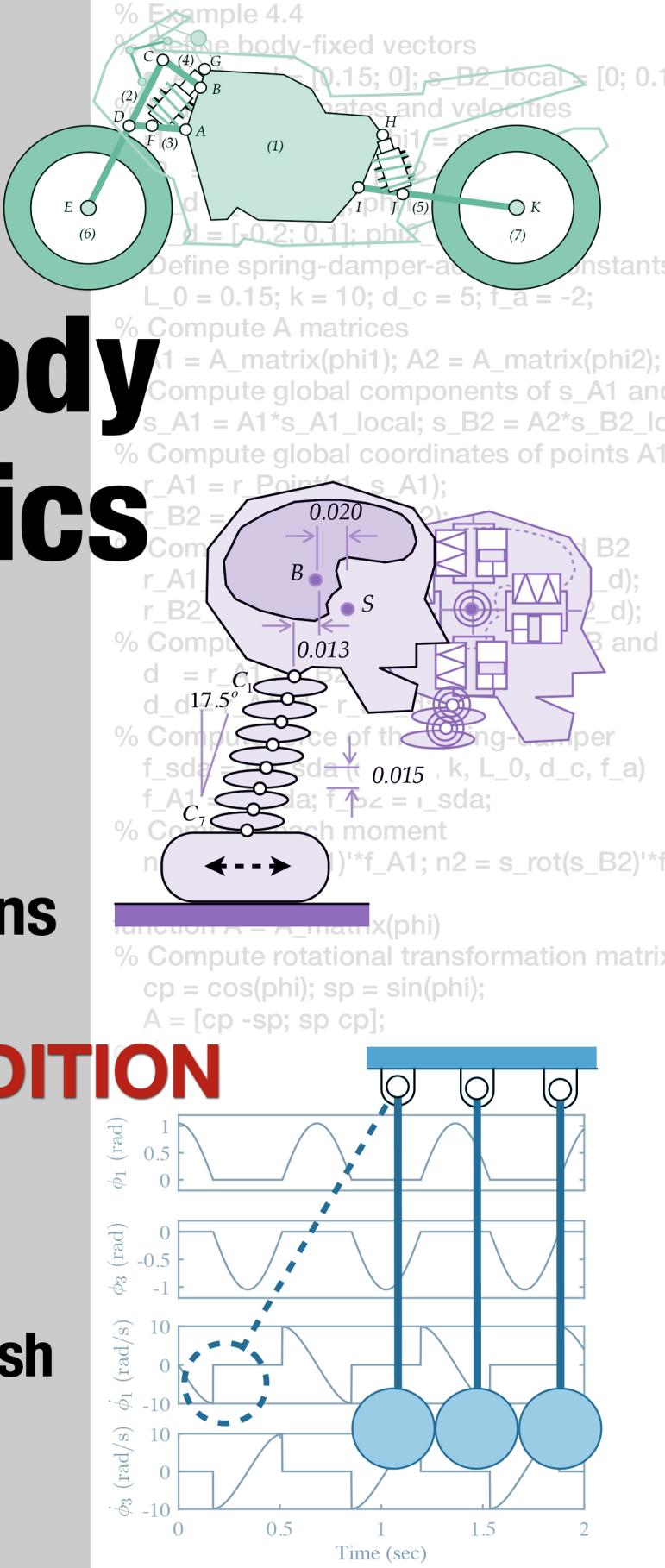
Formulation,  
Programming  
with MATLAB,  
and Applications

**SECOND EDITION**

Parviz E. Nikravesh



CRC Press  
Taylor & Francis Group



# Planar Multibody Dynamics

# Planar Multibody Dynamics

Formulation, Programming with MATLAB<sup>®</sup>,  
and Applications

Second Edition

Parviz E. Nikravesh



CRC Press

Taylor & Francis Group

Boca Raton London New York

---

CRC Press is an imprint of the  
Taylor & Francis Group, an **informa** business

MATLAB® is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® software.

CRC Press  
Taylor & Francis Group  
6000 Broken Sound Parkway NW, Suite 300  
Boca Raton, FL 33487-2742

© 2019 by Taylor & Francis Group, LLC  
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works

Printed on acid-free paper

International Standard Book Number-13: 978-1-138-09612-7 (Hardback)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access [www.copyright.com](http://www.copyright.com) (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

**Trademark Notice:** Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

---

#### Library of Congress Cataloging-in-Publication Data

---

Names: Nikravesh, Parviz E., 1946- author.  
Title: Planar multibody dynamics : formulation, programming with MATLAB®, and applications / Parviz Nikravesh.  
Description: Second edition. | Boca Raton : Taylor & Francis, CRC Press, 2018. | Includes bibliographical references.  
Identifiers: LCCN 2018017483 | ISBN 9781138096127 (hardback : alk. paper) | ISBN 9781315105437 (e-book)  
Subjects: LCSH: Machinery, Kinematics of. | Machinery, Dynamics of.  
Classification: LCC TJ175 .N527 2018 | DDC 621.8/11—dc23  
LC record available at <https://lccn.loc.gov/2018017483>

---

Visit the Taylor & Francis Web site at  
<http://www.taylorandfrancis.com>

and the CRC Press Web site at  
<http://www.crcpress.com>

*This book is dedicated to the memory of*

*Dr. Moris Nikravesh*

*“The Medicine Man”*

---

# *Contents*

---

Preface.....	xiii
Acknowledgments .....	xvii
Author .....	xix
<b>1. Introduction .....</b>	<b>1</b>
1.1 Multibody Mechanical Systems.....	1
1.2 Types of Analyses .....	2
1.3 Methods of Formulation .....	2
1.4 Computer Programs .....	9
<b>2. Preliminaries .....</b>	<b>13</b>
2.1 Reference Axes .....	13
2.2 Scalars and Vectors .....	13
2.2.1 Arrays .....	20
2.3 Matrices .....	21
2.3.1 Matrix Operations.....	23
2.3.2 Linear Algebraic Equations.....	26
2.4 Vector, Array, and Matrix Differentiation .....	27
2.4.1 Time Derivatives .....	27
2.4.2 Partial Derivatives .....	28
2.5 Equations and Expressions.....	31
2.5.1 Compact and Expanded Forms .....	31
2.6 Problems .....	32
<b>3. Fundamentals of Planar Kinematics .....</b>	<b>37</b>
3.1 A Particle .....	37
3.1.1 Kinematics of a Particle.....	37
3.2 Kinematics of a Rigid Body .....	38
3.2.1 Coordinates of a Body .....	39
3.2.2 Velocity of a Body .....	43
3.2.3 Acceleration of a Body.....	44
3.3 Definitions.....	46
3.3.1 Array of Coordinates.....	46
3.3.2 Degrees of Freedom.....	48
3.3.3 Constraint Equations.....	49
3.3.4 Kinematic Joints .....	51
3.4 Problems .....	53
<b>4. Fundamentals of Planar Dynamics .....</b>	<b>57</b>
4.1 Newton's Laws of Motion .....	57
4.2 Particle Dynamics .....	58
4.2.1 Dynamics of a System of Particles.....	58
4.3 Rigid body Dynamics.....	61
4.3.1 Moment of a Force and Torque .....	62

4.3.2	Centroidal Equations of Motion .....	63
4.3.3	Noncentroidal Equations of Motion.....	67
4.4	Multibody Dynamics.....	69
4.4.1	Applied Forces.....	69
4.4.2	Reaction Forces.....	75
4.5	Friction Force .....	76
4.5.1	Wheel and Tire.....	80
4.5.2	Motor and Driver .....	82
4.6	Work and Energy .....	83
4.7	Problems .....	84
	References .....	88
<b>5.</b>	<b>Vector Kinematics.....</b>	<b>89</b>
5.1	Use of Vectors .....	89
5.1.1	Unit Vectors .....	90
5.1.2	Types of Vectors .....	92
5.2	Open-Chain Systems .....	94
5.3	Closed-Chain Systems.....	97
5.3.1	Slider-Crank Mechanism.....	97
5.3.2	Four-Bar Mechanism.....	101
5.3.3	Six-Bar Quick-Return Mechanism.....	105
5.3.4	Six-Bar Dwell Mechanism .....	107
5.3.5	Complete Kinematic Analysis.....	108
5.4	Problems .....	108
<b>6.</b>	<b>Free-Body Diagram .....</b>	<b>113</b>
6.1	FBD Examples.....	113
6.1.1	Two-Body System (Unconstrained).....	113
6.1.2	Two-Body System (Constrained) .....	116
6.1.3	Sliding Pendulum .....	118
6.1.4	Slider-Crank Mechanism.....	120
6.1.5	Four-Bar Mechanism.....	122
6.2	Equations of Motion .....	126
6.3	Force Analysis .....	126
6.3.1	Slider-Crank Mechanism.....	127
6.3.2	Four-Bar Mechanism.....	128
6.3.3	Generalization of Force Analysis.....	130
6.4	Problems .....	130
<b>7.</b>	<b>Body-Coordinate Formulation.....</b>	<b>135</b>
7.1	General Procedure .....	135
7.2	Kinematic Joints .....	137
7.2.1	Revolute (Pin) Joint .....	139
7.2.2	Translational (Sliding) Joint.....	141
7.2.3	Revolute–Revolute Joint .....	142
7.2.4	Revolute–Translational Joint.....	144
7.2.5	Rigid Joint .....	145
7.2.6	Simple Constraints.....	146

7.2.7	Circular Disc.....	146
7.2.8	Driver Constraints .....	147
7.2.9	System Jacobian.....	148
7.3	Unconstrained Equations of Motion .....	149
7.4	Constrained Equations of Motion .....	152
7.4.1	Reaction Forces and Lagrange Multipliers.....	153
7.5	Total Energy.....	160
7.6	Problems.....	161
<b>8.</b>	<b>Body-Coordinate Simulation Program .....</b>	<b>169</b>
8.1	Application Examples.....	170
8.1.1	Double A-Arm Suspension.....	170
8.1.2	MacPherson Suspension .....	173
8.1.3	Cart.....	177
8.1.4	Conveyor Belt and Friction.....	180
8.1.5	Rod Impacting Ground .....	182
8.2	Problems.....	183
<b>9.</b>	<b>Joint-Coordinate Formulation.....</b>	<b>185</b>
9.1	Definitions.....	185
9.1.1	Joint Coordinate and Joint Reference Point.....	186
9.1.2	Recursive Kinematics .....	188
9.2	Open-Chain Systems.....	190
9.2.1	Absolute Angle.....	195
9.2.2	Equations of Motion .....	197
9.3	Closed-Chain Systems.....	203
9.3.1	Cut-Joint Constraints .....	203
9.3.2	Equations of Motion .....	206
9.3.3	Jacobian Matrix .....	213
9.3.4	Initial Conditions .....	217
9.3.5	Reaction Forces.....	218
9.3.6	Driver Constraint .....	220
9.4	A MATLAB Program.....	221
9.5	Problems.....	222
<b>10.</b>	<b>Point-Coordinate Formulation.....</b>	<b>229</b>
10.1	Classical Method .....	229
10.2	Primary and Stationary Points.....	231
10.3	Constraints.....	233
10.3.1	Length Constraint.....	235
10.3.2	Angle Constraints .....	235
10.3.3	Simple Constraints.....	236
10.4	Secondary Points.....	238
10.5	Equations of Motion .....	241
10.6	Force and Torque Distribution .....	242
10.7	Mass Distribution.....	246
10.8	Mass Condensation.....	248
10.8.1	Two Primary Points .....	248

10.8.2 Three Primary Points .....	253
10.9 Force and Mass Addition.....	254
10.10 Problems.....	255
<b>11. Contact and Impact.....</b>	<b>261</b>
11.1 Piecewise Analysis.....	261
11.1.1 Momentum .....	261
11.1.2 Impact of Particles .....	262
11.1.3 Unconstrained Bodies .....	267
11.1.4 Constrained Bodies .....	271
11.1.5 Impact with Friction.....	273
11.2 Continuous Analysis .....	274
11.2.1 A Body Contacting a Rigid Surface.....	274
11.2.2 Two-Body Contact.....	277
11.3 Problems.....	281
References .....	283
<b>12. Kinematics and Inverse Dynamics.....</b>	<b>285</b>
12.1 Kinematic Analysis.....	285
12.1.1 Solution Procedures.....	288
12.1.2 Nonlinear Algebraic Equations .....	290
12.1.3 A Program for Four-Bar Kinematics.....	294
12.2 Inverse Dynamic Analysis.....	297
12.2.1 A Program for Four-Bar Inverse Dynamics.....	302
12.2.2 Application in Robotics.....	305
12.3 Problems.....	307
Reference .....	309
<b>13. Forward Dynamics.....</b>	<b>311</b>
13.1 Unconstrained Formulation .....	311
13.1.1 Initial Value Problems .....	312
13.1.2 Runge–Kutta Algorithm .....	313
13.1.3 Integration Time-Step Size .....	315
13.1.4 General Procedure .....	318
13.2 Constrained Formulation.....	319
13.2.1 Initial Conditions .....	321
13.2.2 General Integration Procedure .....	321
13.3 Constraint Violation.....	323
13.3.1 Constraint Violation Stabilization Method .....	323
13.3.2 Coordinate Partitioning Method .....	326
13.3.3 Penalty Method .....	328
13.3.4 Joint-Coordinate Method.....	331
13.3.5 Momentum Method .....	331
13.4 Contact and Impact.....	333
13.5 Adding or Deleting Constraints .....	337
13.6 Combined Analyses.....	340
13.7 Problems.....	341
References .....	348

<b>14. Complementary Analyses.....</b>	349
14.1 Static Analysis .....	349
14.2 Static Equilibrium .....	351
14.3 Initial Condition Correction.....	353
14.4 Redundant Constraints .....	355
14.5 Friction.....	356
14.6 Deformable Body .....	360
14.7 Problems.....	361
<b>15. Application Examples.....</b>	365
15.1 Film-Strip Advancer .....	365
15.2 Web-Cutter Mechanism.....	366
15.3 Six-Bar Quick-Return Mechanism.....	368
15.4 Six-Bar Dwell Mechanism .....	368
15.5 Windshield Wiper Mechanism .....	369
15.6 Double A-Arm Suspension .....	371
15.7 MacPherson Strut Suspension.....	372
15.8 Half-Car.....	374
15.9 Mountain Bike .....	374
15.10 Motorcycle.....	377
15.11 Dump Truck.....	377
15.12 Creeping Robot.....	381
15.13 Sled Test and Belted Dummy .....	382
15.14 Head and Neck.....	385
15.15 Elliptical Exercise Machine.....	386
15.16 Swing .....	388
15.17 Satellite Panels .....	389
<b>Appendix A: L-U Factorization.....</b>	393
<b>Appendix B: Dynamic Analysis Program: Body Coordinates (DAP_BC) .....</b>	399
<b>Appendix C: Dynamic Analysis Program: Joint Coordinates (DAP_JC).....</b>	401
<b>Index.....</b>	403

---

# *Preface*

---

The basic premise of this textbook is to introduce fundamental theories, computational methods, and program development for analyzing simple to complex planar mechanical systems. Such a combination of theory, computational methods, and programming did not exist in any mechanical engineering curricula four decades ago, but since then it has become a standard course in most programs. In the late 1970s, different institutions began to offer graduate-level courses on these combined subjects. Several textbooks and monograms were published soon after, all at the graduate level. Eventually, selected chapters from these books were offered as technical elective courses at the senior undergraduate level, and shortly after, some of these topics were introduced at the junior level, either replacing or complementing some of the more traditional courses.

The first edition of this textbook that was published in 2008 was the first book written specifically for undergraduate students and practicing engineers. In the process of writing the first edition, I intended for the book not to become a simpler version of a graduate-level research monogram. The first edition covered three methods for formulating the kinematic and dynamic equations of motion, namely, the point-coordinate, body-coordinate, and joint-coordinate methods. The methodologies for deriving the equations of motion were all based on Newton's laws of motion, where as much as possible the use of graduate-level principles of mechanics was avoided. The methods of analyses that were described in the first edition included kinematics, inverse dynamics, and forward dynamics, in addition to other complementary procedures. Due to its popularity and ease of use, MATLAB® was chosen as the programming language for the book.

---

## New Features

This edition may appear as a completely new textbook to some readers due to numerous changes and new materials. These changes were made based on the feedback I received from colleagues and students, and also from my own experience. Every time I taught a course on planar multibody dynamics, I noticed that most undergraduate students needed a review of the methods of analyses from their earlier course on kinematics and dynamics of mechanisms. Therefore, in this edition, an overview of the classical vectorial method of kinematic analysis, which most mechanical engineering students should be familiar with by their junior year, and a review of the free-body diagram approach for constructing Newton's equations of motion have been added. A review of these classical methods should clarify to students that the more modern computational multibody approaches are not very different than the classical methods, except for being applied in a more systematic process.

Other added topics to the second edition are modeling impact between bodies of a multibody system and a thorough discussion on modeling friction. The discussion of impact covers two common approaches: the piecewise and continuous methods. In the piecewise method, the conservation of momenta and momentum-impulse concepts are discussed. The methodology of modeling friction is extended to represent interacting forces between

a wheel–tire model and the ground. These new topics should provide a wide range of application examples for modeling and simulation.

---

## **Computer Programs**

Most chapters contain examples that are solved with MATLAB. All of the programs from the first edition have completely been revised. Since a reader may not be a skilled programmer, the examples and exercises in the earlier chapters provide a tutorial that begins with basic commands before introducing the reader to more advanced programming techniques. All of the programs can be downloaded from the following website: [www.crcpress.com](http://www.crcpress.com).

Two user manuals that accompany a general-purpose program based on the body-coordinate formulation and a semi-general-purpose program based on the joint-coordinate formulation can also be downloaded. These programs can be used to model and analyze a variety of multibody systems based on the description provided by the user. A special-purpose program for kinematic and inverse dynamic analyses of four-bar mechanisms is also included in this edition. Most of these programs contain an animation routine to display a stick drawing of the simulated system in motion. Because the listed programs may contain programming, logical, or typesetting errors, the posted programs will be revised as the errors are found. Additional complimentary programs and other relevant materials may also be posted on the website as they become available. The reader is encouraged to visit the website on a regular basis.

---

## **Organization**

Almost all of the topics from the first edition have been kept in this edition but slightly rearranged. With the exception of the first and last chapters, the other chapters can be categorized into four groups: fundamentals, fundamental formulations, multibody formulations, and analyses.

Chapter 1 provides a brief introduction to multibody dynamics and different forms of describing the equations of motion for a system.

## **Fundamentals**

Chapter 2 describes the notation and reviews the fundamentals of vector and matrix algebra.

Chapter 3 begins with a brief review of the fundamentals of particle and rigid-body kinematics before discussing commonly used types of kinematic joints, constraints, degrees of freedom, and other related topics.

Chapter 4 provides a review of planar dynamics beginning with a particle, a system of particles, a rigid body, and then a multibody system. Formulations for computing applied forces, such as springs and dampers, and representing reaction forces caused by kinematic

joints are discussed. A discussion on modeling friction between rigid bodies, and computing kinetic and potential energies are new additions to this chapter.

Understanding these fundamentals is essential in learning the remainder of the textbook.

## Fundamental Formulations

Chapter 5 (new to this edition) reviews the classical method of vector kinematics and the vector-loop method that can be found in most textbooks on mechanisms.

Chapter 6 (new to this edition) provides an overview of the classical free-body diagram technique for constructing the dynamic equations of motion.

## Multibody Formulations

Chapter 7 presents the method of body coordinates to formulate multibody equations of motion. This formulation may be considered the simplest and, at the same time, the most powerful method for computational multibody dynamics. This formulation is a systematic extension of the free-body diagram method in Chapter 6, which is suitable for computational procedures.

Chapter 8 introduces the reader to a general-purpose program based on the body-coordinate formulation of Chapter 7. Several examples that have been modeled and analyzed by this program are presented in this chapter. The program and its user manual are available for download.

Chapter 9 describes a method known as the joint-coordinate formulation. This method is a systematic process that transforms the equations of motion from the body coordinates to a minimal or a much smaller set of equations. This formulation provides computational efficiency while preserving all the advantages of the body-coordinate formulation. A semi-general-purpose program accompanies this chapter that can be downloaded. The program requires some programming by the user to describe a multibody system for analysis.

Chapter 10 presents a method that describes a body or a multibody system as a collection of interconnected particles, and therefore, it bypasses the use of rotational coordinates. The resulting equations of motion do not introduce any approximation in describing the dynamics of a system.

Chapter 11 (new to this edition) discusses two well-known methods for modeling the impact or contact in multibody dynamics. In the piecewise analysis method, the concepts of momentum and impulse are reviewed. In the continuous analysis method, several contact models are studied.

## Analyses

Chapter 12 provides algorithms for kinematic and inverse dynamic analyses. Numerical methods for solving linear and nonlinear algebraic equations are reviewed. A special-purpose program for these two types of analyses of four-bar mechanisms is also presented.

Chapter 13 begins with a brief discussion on numerical methods for solving ordinary differential equations. Algorithms for forward dynamic analysis of unconstrained and constrained equations of motion are presented. The issue of constraint violations in solving constrained equations of motion is discussed in detail. Finally, solution techniques for impact analysis with the piecewise method, and adding or deleting constraints are reviewed.

Chapter 14 provides a review of several complementary methods in solving the equations of motion for static and static equilibrium analyses, correcting initial conditions prior to a forward dynamic analysis, and a method to improve computational efficiency in problems that involve friction.

Chapter 15 provides several application examples of planar multibody systems. Some of the simple examples have been used throughout the book as exercises, or as end-of-chapter problems. Other more complex examples are provided as simulation projects.

An instructor, or a reader, may choose to cover the book in its entirety or to consider only a selected number of chapters. Four possible options for covering the topics are suggested in the following. In all these options, Chapters 1–6 and 12–15 must be considered—the options only apply to the multibody formulation chapters:

- Option A: Chapters 7 and 8, and Chapter 11 as a secondary option
- Option B: Chapters 7–9
- Option C: Chapter 10
- Option D: All chapters

As an undergraduate course, consider Option A, B, or C. Option D is recommended for a graduate-level course.

MATLAB® is a registered trademark of The MathWorks, Inc. For product information, please contact:

The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098 USA  
Tel: 508-647-7000  
Fax: 508-647-7001  
E-mail: [info@mathworks.com](mailto:info@mathworks.com)  
Web: [www.mathworks.com](http://www.mathworks.com)

---

## *Acknowledgments*

---

Since my first book on the subject of multibody dynamics was published in 1988, I have received many compliments and acknowledgments from colleagues and readers, many of whom I have not had the privilege of ever meeting. I am humbled by their kind words, and because of their encouraging remarks, I decided to write the first edition of this textbook, and now the second edition.

Over the years, I have received many useful suggestions and ideas from students who have taken my course. It is their appreciation that makes writing a textbook worthwhile. I am grateful to all of them.

Finally, I extend my special gratitude to my colleague Dr. Mohammad A. Poursina for his valuable suggestions, comments, and “cheerleading” during the course of this project. Without his enthusiastic comments, some of the newly added topics would not have been included in this edition.

**Parviz E. Nikravesh**  
*Tucson, Arizona*

---

## ***Author***

---

**Parviz E. Nikravesh** has been a researcher and an educator for more than 40 years. He is currently a professor in the department of aerospace and mechanical engineering at the University of Arizona, Tucson, Arizona. He is the author of a large number of journal publications in theoretical and computational dynamics. His first book, titled *Computer-Aided Analysis of Mechanical Systems*, has been translated from English to several other languages and is considered to be the first textbook on the subject of multibody dynamics.

Professor Nikravesh is a member of the American Society of Mechanical Engineers and the Society of Automotive Engineers. He has served on the editorial board of the journal *Multibody System Dynamics* since its conception. He has received many awards for his contributions to the field of computational dynamics including an honorary doctorate degree.

# 1

---

## Introduction

---

The major goal of the engineering profession is to design and manufacture marketable products of high quality. Today's industries are utilizing computers in every phase of the design and manufacture of their products. The process of design and manufacture, beginning with an idea and ending with a final product, is a closed-loop process. The design process requires a thorough understanding and ability to analyze the product. Computer-aided analysis allows an engineer to simulate and predict the behavior of a product. Based on the analysis results, the product design can be optimized prior to actual production.

To simulate the behavior of a product, we must know the individual components that make up the product. A product may contain mechanical, electrical, or other components. If the mechanical components are allowed to move relative to one another, the product is called a multibody system. The interconnection between various components, or bodies, can be through kinematic joints, springs, dampers, impact, or other elements. Bodies of a mechanical system are generally deformable. But in most cases, they can be assumed nondeformable (rigid) due to their negligible deformation. The behavior (e.g., the motion) of a multibody system can be analyzed by using pencil and paper (classical methods) only if the system is extremely simple and simplifying assumptions are made. Even for simple systems, it may not be feasible to perform an analysis without a computer. This is definitely true for realistically complex multibody systems. Therefore, it is the objective of this textbook to present computational analysis techniques that can be applied systematically to systems composed of nondeformable bodies undergoing large planar motion regardless of their complexity.

---

### 1.1 Multibody Mechanical Systems

Multibody mechanical systems range from the very simple to the very complex. A pendulum and a slider–crank mechanism are examples of simple systems, whereas the suspension and steering systems of an automobile, a walking robotic device, and an exercise machine are examples of more complex systems. The bodies of some multibody systems can only move in parallel planes. Such systems are called *planar* or two-dimensional. Systems whose bodies do not move in parallel planes are called *spatial* or three-dimensional, such as a robotic device that is capable of operating in all three directions. The motion of some spatial systems, if projected onto a plane, could be approximated as being planar.

Most common components in multibody systems are bodies (also referred to as links), kinematic joints, and compliance elements. Examples of kinematic joints are pins, sliders, gears, and cam-followers. Typical compliance elements are springs and dampers. Kinematic joints and compliance elements provide the connectivity between the bodies of a system. A multibody system may contain nonmechanical components such as an electronic controller. Analyzing the dynamics of such a system must also include an analytical

model of the controller. The bodies of most multibody systems can be considered as non-deformable. However, in some applications, deformation of a link may not be negligible and should be considered in the analysis.

Studying a multibody system involves two fundamental steps: *modeling* and *analysis*. Modeling or *formulation* is the process of constructing the necessary equations that, if solved, would reveal the behavior of a system. In this textbook, we will consider several methods of formulation, each having its own advantages and disadvantages. Depending on the application of a multibody system, different types of analyses could be considered.

---

## 1.2 Types of Analyses

There are two different aspects to the study of a mechanical system: *analysis* and *design*. When a mechanical system is acted upon by a given excitation, for example, an external force, the system exhibits a certain response. The process that allows an engineer to study the response of an already existing system to a known excitation is called *analysis*. This requires a complete knowledge of the physical characteristics of the mechanical system, such as material composition, shape, and arrangement of parts. Conversely, the process of determining the physical characteristics that are necessary for a mechanical system to perform a prescribed task is called *design* or *synthesis*. The design process requires the application of scientific techniques along with the engineering judgment. The scientific techniques in the design process, such as *analysis* and *optimization*, are merely tools to be used by the engineer. Although these methodologies can be applied in a systematic manner, the overall design process hinges on the judgment of the designer. Because the scientific aspect of the design process requires analysis techniques as tools, it is important to learn about the methods of analysis prior to design.

The branch of analysis that studies motion, time, and force is called *mechanics*. It consists of two parts—*statics* and *dynamics*. In statics, we analyze stationary systems—systems in which time is not a factor. Dynamics, on the other hand, deals with systems that are nonstationary—systems that change their positions with respect to time. Dynamics is divided into two disciplines—*kinematics* and *kinetics*. Kinematics is the study of motion regardless of forces that produce the motion. More explicitly, kinematics is the study of displacement, velocity, and acceleration. Kinetics, on the other hand, is the study of motion and its relationship with the forces that produce that motion. It is, however, very common to refer to *kinetic analysis* as *dynamic analysis*, because kinetic analysis must be based on the knowledge of the kinematics of a system as well. Therefore, in this textbook, we will use the term *dynamic* instead of *kinetic*. We will discuss several computational methods of analyses—*kinematic analysis*, *inverse dynamic analysis*, *forward dynamic analysis*, *static analysis*, and *static equilibrium analysis*.

---

## 1.3 Methods of Formulation

Classical methods of analysis in mechanics have relied upon graphical and often quite complex techniques. These techniques are mostly based on geometrical interpretations

of the system under consideration. Furthermore, these techniques have been developed for hand derivation and solution of the equations. Some of these techniques can be implemented in computer programs. However, more modern solution techniques can take full advantage of the capabilities of computational methods. In this section, we provide an overview of some of the formulation methods that are discussed in this textbook through several simple examples. In this overview, we do not discuss the details of each method or how a set of equations has been derived. Our objective is to realize that there is more than one way to formulate a problem for a particular form of analysis.

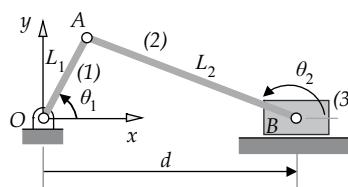
As the first example, we consider the slider–crank mechanism shown in Figure 1.1. The lengths of the crank and connecting rod are  $L_1 = 0.2\text{m}$  and  $L_2 = 0.4\text{m}$ , respectively. The crank, link (1), rotates with a constant angular velocity of  $\omega_1 = 1.5\text{ rad/s}$  in the counter-clockwise direction. Let us assume that our objective is to determine the position, velocity, and acceleration of the connecting rod, link (2), and the slider at the configuration where the crank makes a  $30^\circ$  angle with the horizontal axis.

For our analysis, we define the angles of links (1) and (2) with respect to the horizontal axis as  $\theta_1$  and  $\theta_2$ , respectively. The position of the slider with respect to point O is defined as  $d$ . Since we are not interested in the forces that cause or are the result of this motion, the process is purely a *kinematic* analysis. In the following discussion on different forms of solution, we present each method in a general form without explaining the details of the implementation.

The first method of kinematic analysis that we consider is the classical graphical technique. For position analysis, the triangle  $OAB$ , depicted in Figure 1.2a, is drawn as accurately as possible since the lengths  $OA$  and  $AB$  and the angle of the crank are given. This process reveals that there are two solutions for the given data—the constructed triangle could be either  $OAB$  or  $OAB'$ . Although both solutions are feasible, based on the original diagram, we choose point  $B$  to represent the position of the slider. From the constructed triangle, by measurements, we determine  $d = 0.53\text{ m}$  and  $\theta_1 = 165^\circ$ . This completes the graphical position analysis for this mechanism in the specified configuration.

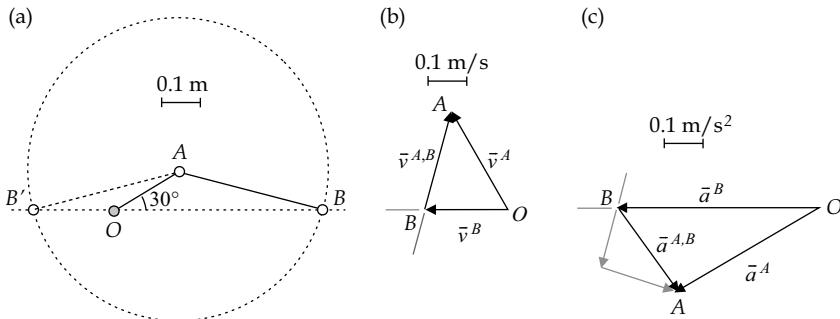
To perform a velocity analysis, for the given orientation of the slider–crank, a *velocity vector polygon* is constructed as shown in Figure 1.2b. This polygon is constructed based on the results from the position analysis (orientation of the links) and the given angular velocity of the crank as  $\omega_1 = 1.5\text{ rad/s}$  counterclockwise. The polygon shows the velocity of point  $A$ ,  $\vec{v}^A$ ; the velocity of the slider,  $\vec{v}^B$ ; and the velocity of point  $A$  relative to point  $B$ ,  $\vec{v}^{A,B}$ . We measure the magnitude of  $\vec{v}^B$  to be  $0.21\text{ m/s}$  with a direction to the left. Based on the measured magnitude of  $\vec{v}^{A,B}$  and the length of link  $AB$ , we determine the angular velocity of the connecting rod to be  $\omega_2 = 0.6\text{ rad/s}$  clockwise.

Similar to the velocity analysis, an acceleration polygon can be drawn for acceleration analysis as shown in Figure 1.2c. This polygon is constructed based on the results of the position and velocity analyses, and the given angular acceleration of the crank,  $\alpha_1 = 0$



**FIGURE 1.1**

A slider–crank mechanism.



**FIGURE 1.2**  
Graphical methods for (a) position, (b) velocity, and (c) acceleration analyses.

(constant angular velocity). The polygon shows the acceleration of point  $A$ ,  $\bar{a}^A$ ; the acceleration of the slider,  $\bar{a}^B$ ; and the acceleration of point  $A$  relative to point  $B$ ,  $\bar{a}^{A,B}$ . Direct measurements reveal the magnitude of  $\bar{a}^B$  to be  $0.51 \text{ m/s}^2$  with a direction to the left. Based on the measured magnitude of the so-called tangential component of  $\bar{a}^{A,B}$  and the length of link  $AB$ , we determine the angular acceleration of the connecting rod to be  $\alpha_2 = 1.4 \text{ rad/s CCW}$ .

The graphical process provides a visual understanding of the kinematics of a system. However, the process is not accurate, and it could become impractical if we need to repeat the process for many different configurations. The accuracy of the results from a position analysis depends on how accurately we draw the lines and circles, and on the accuracy of the measurements for the lengths and angles. Obviously, the measurement errors from the position analysis will be included in the measurement errors from the corresponding velocity polygon and will further be magnified in the results from the acceleration polygon. Therefore, the overall results from a graphical analysis cannot be very accurate.

A classical analytical formulation for kinematics of planar mechanisms is known as the vector-loop method. For the slider–crank mechanism of Figure 1.1, this method requires constructing algebraic relationships between the defining variables  $\theta_1$ ,  $\theta_2$ , and  $d$ . A vector loop for the closed triangle  $OABO$  yields the following equations:

$$\begin{aligned} 0.2 \cos \theta_1 - 0.4 \cos \theta_2 - d &= 0 \\ 0.2 \sin \theta_1 - 0.4 \sin \theta_2 &= 0 \end{aligned} \tag{1.1}$$

For the crank angle  $\theta_1 = 30^\circ$ , these two equations can be solved for the two unknown variables,  $\theta_2$  and  $d$ . The method of solution for such nonlinear algebraic equations will be discussed later in this textbook. At this point we are only interested in the concept and not in the details. Such a solution provides  $\theta_2 = 165.5247^\circ$  and  $d = 5.6057 \text{ m}$ .

For velocity analysis, the time derivative of Eq. (1.1) provides the velocity equations as follows:

$$\begin{aligned} -0.2 \sin \theta_1 \omega_1 + 0.4 \sin \theta_2 \omega_2 - \dot{d} &= 0 \\ 0.2 \cos \theta_1 \omega_1 - 0.4 \cos \theta_2 \omega_2 &= 0 \end{aligned} \tag{1.2}$$

Knowing the values of  $\theta_1$ ,  $\theta_2$ , and  $\omega_1 = 1.5$ , the velocity equations can be solved for  $\omega_2$  and the linear velocity of the slider,  $\dot{d}$ . The solution to these equations yields  $\dot{\theta}_2 = -0.6708 \text{ rad/s}$  and  $\dot{d} = -2.1707 \text{ m/s}$ . The negative value of  $\dot{d}$  indicates that the slider is moving to the left.

For acceleration analysis, the time derivative of Eq. (1.1) provides the acceleration equations as follows:

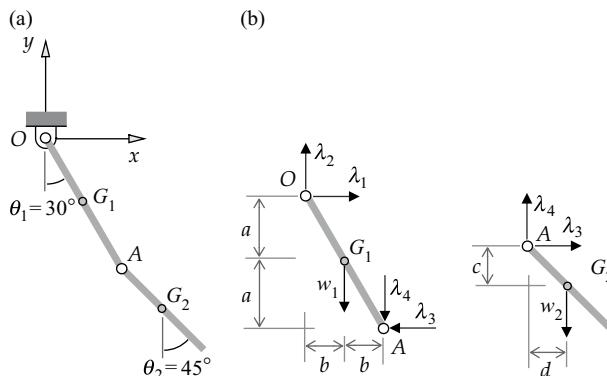
$$\begin{aligned} -0.2 \sin \theta_1 \alpha_1 - 0.2 \cos \theta_1 \omega_1^2 + 0.4 \sin \theta_2 \alpha_2 + 0.4 \cos \theta_2 \omega_2^2 - \ddot{d} &= 0 \\ 0.2 \cos \theta_1 \alpha_1 - 0.2 \sin \theta_1 \omega_1^2 - 0.4 \cos \theta_2 \alpha_2 + 0.4 \sin \theta_2 \omega_2^2 &= 0 \end{aligned} \quad (1.3)$$

Knowing the values of  $\theta_1$ ,  $\theta_2$ ,  $\omega_1$ ,  $\omega_2$ , and  $\alpha_1 = 0$ , the acceleration equations are solved to determine the two unknowns as  $\alpha_2 = 0.4648 \text{ rad/s}^2$  and  $\ddot{d} = -0.5175 \text{ m/s}^2$ .

This classical method of kinematic analysis of planar mechanical systems is reviewed in detail in Chapter 5.

Although, in the vector-loop method, it is possible to solve the position, velocity, and acceleration equations of simple systems by using pencil and paper, the equations are more suitable for development into a computer program. The computed values, obviously, are much more accurate than those obtained graphically. Furthermore, the computational process could easily be repeated for different values of the crank angle. In this textbook, in addition to the vector-loop method, several other methods of analytical formulation for the kinematics of multibody systems are also discussed.

So far, our discussion has concentrated on kinematic analysis of multibody systems. To discuss the methods of formulation for dynamic analysis of multibody systems, we consider a simpler example than a slider–crank mechanism. The example is a double pendulum shown in Figure 1.3a. The pendulum is composed of two slender rods:  $OA$ , link (1), that is pinned to the ground at  $O$ , and  $AB$ , link (2), that is pinned to link (1) at  $A$ . The link lengths are  $L_1 = 2.0 \text{ m}$  and  $L_2 = 1.5 \text{ m}$ . The mass and moment of inertia for each link are given as  $m_1 = 3.0 \text{ kg}$ ,  $J_1 = 1.0 \text{ kg m}^2$ , and  $m_2 = 1.6 \text{ kg}$ ,  $J_2 = 0.8 \text{ kg m}^2$ . Gravity is the only force that acts on the system ( $g = 9.81 \text{ m/s}^2$ ). The mass centers,  $G_1$  and  $G_2$ , are at the geometric center of the links. It is assumed that in the configuration shown in Figure 1.3a, the links are oriented by angles  $\theta_1 = 30^\circ$  and  $\theta_2 = 45^\circ$  from the vertical axis, and that the links have



**FIGURE 1.3**  
(a) A double pendulum and (b) its FBD representation.

initial angular velocities  $\dot{\theta}_1 = 0.5 \text{ rad/s}$  and  $\dot{\theta}_2 = -0.4 \text{ rad/s}$ . We construct the equations of motion for this system in several forms for comparison.

The first method of formulating the equations of motion for the double pendulum is based on the classical free-body diagram (FBD) of the system, as shown in Figure 1.3b. The weights of the links are shown as  $w_1$  and  $w_2$ . The  $x$  and  $y$  components of reaction forces at the pin joints are shown as  $\lambda_1, \dots, \lambda_4$ . Some of the distances that are needed in the equations of motion are computed as follows:

$$a = \frac{L_1}{2} \cos \theta_1 = 0.87 \text{ m}$$

$$b = \frac{L_1}{2} \sin \theta_1 = 0.50 \text{ m}$$

$$c = \frac{L_2}{2} \cos \theta_2 = 0.53 \text{ m}$$

$$d = \frac{L_2}{2} \sin \theta_2 = 0.53 \text{ m}$$

Considering the two mass centers having coordinates  $x_1, y_1$  and  $x_2, y_2$ , based on the FBDs of the two links, the following six equations can be constructed:

$$\begin{aligned} m_1 \ddot{x}_1 &= \lambda_1 - \lambda_3 \\ m_1 \ddot{y}_1 &= -w_1 + \lambda_2 - \lambda_4 \\ J_1 \ddot{\theta}_1 &= -a\lambda_1 - b\lambda_2 - a\lambda_3 - b\lambda_4 \\ m_2 \ddot{x}_2 &= \lambda_3 \\ m_2 \ddot{y}_2 &= -w_2 + \lambda_4 \\ J_2 \ddot{\theta}_2 &= -c\lambda_3 - d\lambda_4 \end{aligned} \tag{1.4}$$

The two links having constant lengths yield four more equations as follows:

$$\begin{aligned} \ddot{x}_1 - a\ddot{\theta}_1 &= -b\dot{\theta}_1^2 \\ \ddot{y}_1 - b\ddot{\theta}_1 &= a\dot{\theta}_1^2 \\ -\ddot{x}_1 - a\ddot{\theta}_1 + \ddot{x}_2 - c\ddot{\theta}_2 &= -b\dot{\theta}_1^2 - d\dot{\theta}_2^2 \\ -\ddot{y}_1 - b\ddot{\theta}_1 + \ddot{y}_2 - d\ddot{\theta}_2 &= a\dot{\theta}_1^2 + c\dot{\theta}_2^2 \end{aligned} \tag{1.5}$$

In these ten equations, since the positions and velocities are known, there are ten unknowns: six accelerations and four components of reaction forces. The ten equations can be solved for the ten unknowns yielding the following results for the linear accelerations of the mass centers ( $\text{m/s}^2$ ), the rotational accelerations ( $\text{rad/s}^2$ ) of the links, and the components of reaction forces (N):

$$\ddot{x}_1 = -2.16, \ddot{y}_1 = -0.96, \ddot{\theta}_1 = -2.35, \ddot{x}_2 = -5.75, \ddot{y}_2 = -3.17, \ddot{\theta}_2 = -2.52 \quad (1.6)$$

$$\lambda_1 = -15.68, \lambda_2 = 37.17, \lambda_3 = -9.19, \lambda_4 = 10.62$$

The use of FBD for constructing the equations of motion for dynamic analysis of planar mechanical systems is reviewed in detail in Chapter 6. The process is extended for systematic generation of the equations of motion in Chapter 7. A general-purpose MATLAB® program based on this formulation is discussed in Chapter 8.

In the formulation of Eqs. (1.4) and (1.5), we constructed ten equations to determine the accelerations of the two links. As a by-product, we also determined the reaction forces that act between the links at the pin joints. However, since this system has only two degrees of freedom, it is also possible to construct only two equations representing the equations of motion for this double pendulum:

$$\begin{aligned} & \left( J_1 + \left( \frac{1}{4}m_1 + m_2 \right) L_1^2 \right) \ddot{\theta}_1 + \frac{1}{2} m_2 L_1 L_2 \cos(\theta_2 - \theta_1) \ddot{\theta}_2 \\ &= \frac{1}{2} m_2 L_1 L_2 \sin(\theta_2 - \theta_1) \dot{\theta}_2^2 - \frac{1}{2} (w_1 + 2w_2) L_1 \sin \theta_1 \\ & \frac{1}{2} m_2 L_1 L_2 \cos(\theta_2 - \theta_1) \ddot{\theta}_1 + \left( J_2 + \frac{1}{4} m_2 L_2^2 \right) \ddot{\theta}_2 \\ &= -\frac{1}{2} m_2 L_1 L_2 \sin(\theta_2 - \theta_1) \dot{\theta}_1^2 - \frac{1}{2} w_2 L_2 \sin \theta_2 \end{aligned} \quad (1.7)$$

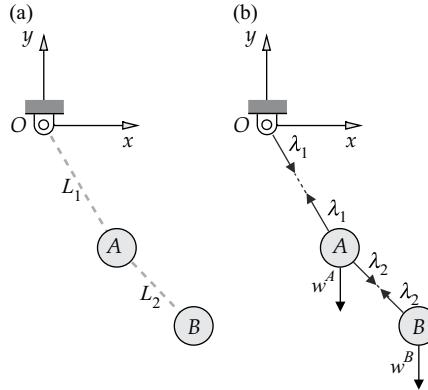
Substituting all the known quantities in these equations, including the initial conditions, the two equations can be solved for the rotational accelerations as follows:

$$\ddot{\theta}_1 = -2.35 \text{ rad/s}^2, \ddot{\theta}_2 = -2.52 \text{ rad/s}^2 \quad (1.8)$$

Knowing the rotational accelerations of the links, we can determine the linear accelerations of the mass centers, if needed. Furthermore, we note that the reaction forces do not appear in Eq. (1.7). If we are interested in knowing the values of these forces, they can be computed after the accelerations have been determined.

The method of formulation that resulted in Eq. (1.7) is discussed in detail in Chapter 9. A MATLAB program based on this formulation is provided for download, which is briefly discussed at the end of the chapter.

In another formulation, the double pendulum is viewed as a system of two particles *A* and *B* shown in Figure 1.4a, where the distances between particle *A* and point *O*, and particles *A* and *B* are known constants. This representation of the double pendulum requires

**FIGURE 1.4**

(a) The double pendulum represented as a system of two particles and (b) the FBD representation of the two particles.

the masses, moment of inertias, and applied forces (weights, in this example) to be properly distributed to the particles. The FBD representation of the two particles is shown in Figure 1.4b, where  $\lambda_1$  and  $\lambda_2$  are the reaction forces acting on the particles along the axes of the links. The FBD provides the following set of equations:

$$\begin{aligned}
 \frac{1}{3}(m_1 + m_2)\ddot{x}^A + \frac{1}{6}m_2\ddot{x}^B &= -\sin\theta_1\lambda_1 + \sin\theta_2\lambda_2 \\
 \frac{1}{3}(m_1 + m_2)\ddot{y}^A + \frac{1}{6}m_2\ddot{y}^B &= -\frac{1}{2}(w_1 + w_2) + \cos\theta_1\lambda_1 - \cos\theta_2\lambda_2 \\
 \frac{1}{6}m_2\ddot{x}^A + \frac{1}{3}m_2\ddot{x}^B &= -\sin\theta_2\lambda_2 \\
 \frac{1}{6}m_2\ddot{y}^A + \frac{1}{3}m_2\ddot{y}^B &= -\frac{1}{2}w_2 + \cos\theta_2\lambda_2 \\
 x^A\ddot{x}^A + y^A\ddot{y}^A &= -(\dot{x}^A)^2 - (\dot{y}^A)^2 \\
 (x^B - x^A)(\dot{x}^B - \dot{x}^A) + (y^B - y^A)(\dot{y}^B - \dot{y}^A) &= -(\dot{x}^B - \dot{x}^A)^2 - (\dot{y}^B - \dot{y}^A)^2
 \end{aligned} \tag{1.9}$$

For these equations, the coordinates and velocities are determined as follows:

$$x^A = L_1 \sin\theta_1 = 1.0, y^A = -L_1 \cos\theta_1 = -1.73 \text{ m}$$

$$x^B = x^A + L_2 \sin\theta_2 = 2.06, y^B = y^A - L_2 \cos\theta_2 = -3.93 \text{ m}$$

$$\dot{x}^A = L_1 \cos\theta_1 \dot{\theta}_1 = 0.87, \dot{y}^A = L_1 \sin\theta_1 \dot{\theta}_1 = 0.50 \text{ m/s}$$

$$\dot{x}^B = \dot{x}^A + L_2 \cos\theta_2 \dot{\theta}_2 = 0.44, \dot{y}^B = \dot{y}^A + L_2 \sin\theta_2 \dot{\theta}_2 = 0.08 \text{ m/s}$$

Substituting these values in Eq. (1.9) and then solving the equations for the unknown accelerations and reaction forces yields

$$\ddot{x}^A = -4.32, \ddot{y}^A = -1.92, \ddot{x}^B = -7.17, \ddot{y}^B = -4.42 \text{ m/s}^2$$

$$\lambda_1 = -13.52, \lambda_2 = -4.69 \text{ N}$$

The acceleration of the mass centers of the links can be determined as follows:

$$\ddot{x}_1 = \frac{1}{2} \ddot{x}^A = -2.16, \ddot{y}_1 = \frac{1}{2} \ddot{y}^A = -0.96 \text{ m/s}^2$$

$$\ddot{x}_2 = \frac{1}{2} (\ddot{x}^A + \ddot{x}^B) = -5.75, \ddot{y}_2 = \frac{1}{2} (\ddot{y}^A + \ddot{y}^B) = -3.17 \text{ m/s}^2$$

These values are exactly the same as those obtained from the method given in Eq. (1.6).

The method of formulation that resulted in Eq. (1.9) is discussed in detail in Chapter 10.

The slider–crank and double pendulum examples demonstrated that a multibody system could be formulated in different ways. If the principles of mechanics are followed and no approximations are considered, all of the formulations should lead to the same results. A problem that is formulated analytically and developed into a computer program can be solved repeatedly, accurately, and efficiently. The usefulness of a computer program becomes even more apparent when the mechanical system under consideration contains numerous interconnected components. Regardless of the complexity of the equations of motion, we should be able to find a numerical solution describing the response.

## 1.4 Computer Programs

Analytical methods for formulating the equations of motion and numerical methods for solving them are the basis for developing computer programs to determine the response of a multibody system. This requires systematic techniques for formulating and generating the kinematic and dynamic equations of motion, numerical methods for solving them, and preferably a front-end graphical interface for communicating the input and the output to the user. Such a computer program can be developed either as a *special-purpose* or as a *general-purpose* program.

A special-purpose program is a computer code that deals with only one particular multibody system, for example, a slider–crank mechanism. The equations of motion for that system are derived *a priori* and coded in the program. As an input to the program, the user provides the data on the physical characteristics of the system. The numerical algorithms for solving the equations of motion can be fine-tuned for that particular form of equations to gain computational efficiency. The main disadvantage of a special-purpose program is its lack of flexibility for analyzing any other mechanical system.

In contrast to a special-purpose program, a general-purpose program can analyze different multibody systems without requiring any changes to the program. For example, the motion of a slider–crank mechanism under a given set of applied loads, the response of an automobile and its suspension system driven over a rough terrain, and the opening of the panels of a satellite antenna can all be simulated with the same general-purpose program. The input data to such a program must completely describe the mechanical system under consideration. The input must contain all the necessary information, for example, number of bodies, connectivity between the bodies, joint types, applied forces, and geometric and inertial characteristics. Based on the given data, the program generates the equations of motion for the requested type of analysis and solves the equations numerically. A general-purpose program may not be computationally as efficient as a special-purpose program, but it provides flexibility in its use.

The formulations and methodologies that are discussed in this textbook can be used to develop either special- or general-purpose programs. The choice of the programming platform is up to us. We can develop programs in Fortran, C, C++, or any other languages. We can use high-performance languages that integrate programming, computation, and visualization such as Mathematica, Maple, or MATLAB.

In this textbook we exclusively use MATLAB due to its popularity and ease of use. It is assumed that the reader has some fundamental knowledge of MATLAB but may not be an expert programmer. Although the MATLAB examples in the book start with some fundamental commands and exercises, the burden of learning MATLAB is upon the reader—the main objective of this textbook is to learn analytical and computational methods of kinematics and dynamics.

Wherever possible, a brief explanation is provided for a fundamental command or an operation that is used for the first time in this textbook. Because each MATLAB statement in a program refers to an already discussed formulation, it should be easy for the inexperienced programmer to figure out what a command means and how it operates. If the use of a command or a function is not clear, the reader should consult the *help* command in MATLAB for further explanation.

Most chapters of this textbook are accompanied with MATLAB programs that should be downloaded from the publisher’s website. Some of the developed programs that refer to certain exercises or examples are short, but some programs are long, such as those discussed in Chapters 8 and 9. A MATLAB program is saved in one or several *M-files*. The M-files that are used in more than one program, possibly in different chapters, have been duplicated and placed in those chapter folders. To conveniently execute a program from a chapter, it is highly recommended to make the MATLAB folder of that chapter the *Current Directory* of MATLAB. For instance, Example 13.8 is accompanied by a program named *Example\_13\_8*, as shown in the following:

### Example 13.8

...

```
MATLAB/Chapter 13/Example_13_8, Ex_13_8
```

```
...
c = [0.4330; -0.2500; pi/3; 0.6928; -0.4000; pi/3];
...
```

To execute this program, make the subfolder Chapter\_13 that resides in the folder MATLAB your current directory. When you execute Example\_13\_8, the program will use another new M-file named Ex\_13\_8, which is also discussed with the example. A program may use other previously developed M-files that reside in the subfolder, but their names will not appear in the reference.

# 2

---

## Preliminaries

---

Geometry of motion is at the heart of both the kinematics and the dynamics of mechanical systems. Vector analysis is the time-honored tool for describing geometry; hence, vector algebra forms the mathematical foundation for kinematics and dynamics. However, vector algebra, in its *geometric form* of presentation, is not well suited to computer implementation.

In this chapter, a systematic matrix formulation of vector algebra, referred to as *algebraic vector* representation, is discussed for use throughout the book. This form of vector representation, in contrast to the more traditional geometric vector representation, is easier to use for formula manipulation and computer implementation. Some elementary properties of vector and matrix algebra are reviewed in this chapter without proof. It is assumed that the reader has the fundamental knowledge of vector and matrix algebra. Some of the fundamental formulas involving vectors are shown in both the classical vector and the algebraic representations in order to assist readers in relating the two representations. In all other chapters, only the algebraic representation is used.

The compact notation for performing matrix algebra will be used extensively throughout this textbook. The notation allows a simple transition from the formula derivation to MATLAB® programming. The reader is highly encouraged to become comfortable with the notation—this leads to taking full advantage of MATLAB’s capabilities. Although we use the compact notation in most examples and discussions, in certain cases we revert to the expanded notation in order to clarify certain points. Since one emphasis of this textbook is on the computational methods of kinematics and dynamics, a reader who is not familiar with programming with MATLAB is encouraged to follow the programming exercises in this and the following chapters closely.

---

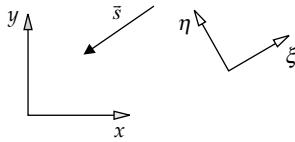
### 2.1 Reference Axes

To specify the configuration of a vector, such as vector  $\vec{s}$  in Figure 2.1, we need to define one or more *Cartesian reference axes*, also called *reference frames*. Two typical reference frames, the  $x$ - $y$  and  $\xi$ - $\eta$  frames, are shown in Figure 2.1. We normally consider the  $x$ - $y$  reference frame to be a nonmoving frame and the  $\xi$ - $\eta$  reference frame to be a moving frame. If  $\vec{s}$  is a moving vector with respect to the  $x$ - $y$  frame, then the  $\xi$ - $\eta$  frame could be defined as being attached to, and therefore moving with, vector  $\vec{s}$ .

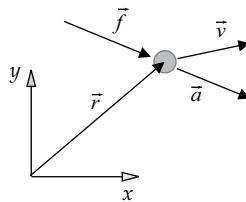
---

### 2.2 Scalars and Vectors

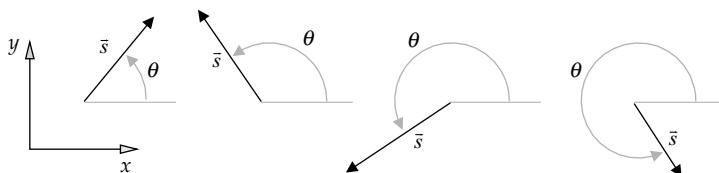
A quantity possessing only a magnitude is called a *scalar*. Length, mass, and speed are examples of scalar quantities. In this book, scalars are shown in *lightface italic* characters such as  $L$ ,  $m$ ,  $J$ , and  $v$ .



**FIGURE 2.1**  
Two Cartesian reference frames and a vector.



**FIGURE 2.2**  
Typical vectors.



**FIGURE 2.3**  
Angle of a vector.

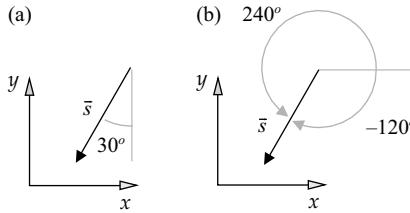
A vector is represented by a *lowercase* character either in *boldface* (algebraic representation) such as  $\mathbf{r}$  or in *lightface italic* with an overscore arrow (geometric representation) such as  $\bar{r}$ . Examples of vectors in their geometric forms are shown in Figure 2.2: a position vector  $\vec{r}$ , a velocity vector  $\vec{v}$ , an acceleration vector  $\vec{a}$ , and a force vector  $\vec{f}$ . All of these vectors possess an axis (or a line of action) with a direction and a magnitude.

The magnitude of a typical vector  $\bar{s}$  is denoted by the scalar  $s$ .\* The direction of a vector is determined by the angle it makes with respect to a known reference axis. The reference axis may be a stationary or a moving axis; it may be an axis of a reference frame or another vector. Unless stated otherwise, in this text we define the angle of a vector as the angle that it makes with the  $x$ -axis of a nonmoving reference frame. We adopt the *convention* to measure the angle in a *counterclockwise* (CCW) direction, known as the positive direction, between the  $x$ -axis and the axis of the vector as shown in Figure 2.3 for four cases. If the angle is measured in a *clockwise* (CW) direction, it must be assigned a negative sign.

### Example 2.1

Vector  $\bar{s}$  makes a  $30^\circ$  angle with the  $y$ -axis as shown in Figure 2.4a. Determine the angle of this vector according to our convention.

\* Traditionally, the magnitude of a vector, such as  $\bar{s}$ , is described as  $|\bar{s}|$  or  $\|\bar{s}\|$ .

**FIGURE 2.4**

Angle of a vector can be described with respect to different axes.

### Solution

The angle of this vector according to our convention is  $\theta = 240^\circ$  or  $\theta = -120^\circ$  as shown in Figure 2.4b. The preference is to use  $\theta = 240^\circ$ .

A typical vector  $\vec{s}$  can be resolved into its *Cartesian components*  $s_{(x)}$  and  $s_{(y)}$  along the  $x$  and  $y$  axes. If the angle of  $\vec{s}$  with respect to the  $x$ -axis is  $\theta$ , the components of the vector are computed as

$$\begin{aligned} s_{(x)} &= s \cos \theta \\ s_{(y)} &= s \sin \theta \end{aligned} \tag{2.1}$$

The *algebraic* representation of a vector describes the vector in terms of its Cartesian components. For vector  $\vec{s}$  with Cartesian components described by Eq. (2.1), the algebraic representation is

$$\mathbf{s} = \begin{Bmatrix} s_{(x)} \\ s_{(y)} \end{Bmatrix} = \begin{Bmatrix} s \cos \theta \\ s \sin \theta \end{Bmatrix} \tag{2.2}$$

By definition, the algebraic representation of a vector is a *column* array. The *transpose* operation, denoted by a right *apostrophe*, transforms an algebraic *column* vector to a *row* vector (or vice versa) as\*†

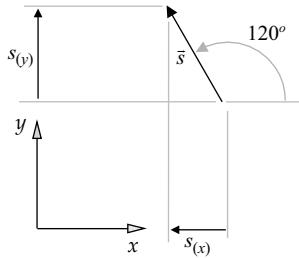
$$\mathbf{s}' = \begin{Bmatrix} s_{(x)} & s_{(y)} \end{Bmatrix}$$

### Example 2.2

A typical vector with its  $x$  and  $y$  components is shown in Figure 2.5. The angle of the vector is  $\theta = 120^\circ$  and its magnitude is 0.2m. For this vector, since  $90^\circ < \theta < 180^\circ$ , i.e.,  $\cos \theta < 0$  and  $\sin \theta < 0$ ,  $s_{(x)}$  is in the direction of negative  $x$ -axis and  $s_{(y)}$  is in the direction of positive  $y$ -axis. Therefore, the  $x$ -component of this vector is negative and its  $y$ -component is positive. This shows that due to our convention for measuring angles, regardless of the value of  $\theta$ , the components of a vector always end up with correct signs (directions).

\* We have adopted the *apostrophe* notation to denote *transpose* in order to be consistent with MATLAB instead of using the traditional notation of a right superscript  $T$ .

† In MATLAB, there are two ways to transpose an array or a matrix. If matrix (array)  $X$  is followed by an apostrophe, such as  $X'$ , it is called *conjugate transpose*. If it is followed by a period and an apostrophe, such as  $X.'$ , it is called a *nonconjugate transpose*. Because in this textbook we deal with matrices (arrays) that do not have imaginary parts,  $X'$  and  $X.'$  yield the same result. Therefore, we will use the simpler of the two (i.e.,  $X'$ ).



**FIGURE 2.5**  
Components of vector  $\bar{s}$ .

Because a geometric vector may be resolved into its components in any Cartesian reference frame and not necessarily in the familiar  $x$ - $y$  frame, to generalize the following observations, in this section we denote the components with subscripts 1 and 2, instead of  $x$  and  $y$ . Assume that two vectors  $\vec{a}$  and  $\vec{b}$  are described in algebraic form as

$$\mathbf{a} = \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix}, \mathbf{b} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix}$$

These are two-dimensional vectors, or two-vectors (or two-arrays), because each has only two components.

Multiplication of a vector  $\vec{a}$  by a scalar  $\alpha$  is defined as a vector in the same direction as  $\vec{a}$  that has a magnitude  $\alpha a$ . This operation in algebraic form occurs component by component:

$$\alpha \mathbf{a} = \begin{Bmatrix} \alpha a_1 \\ \alpha a_2 \end{Bmatrix} \quad (2.3)$$

The negative of a vector is obtained by multiplying the vector by  $-1$ ; it changes the direction of the vector.

The vector sum of two vectors  $\vec{a}$  and  $\vec{b}$  is written as (the geometric form appears on the left and the algebraic form on the right)

$$\vec{a} + \vec{b} = \vec{c} \quad \mathbf{a} + \mathbf{b} = \mathbf{c} \Rightarrow \begin{Bmatrix} a_1 + b_1 \\ a_2 + b_2 \end{Bmatrix} = \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} \quad (2.4)$$

It is also true that  $\mathbf{a} = \mathbf{b}$  if  $a_1 = b_1$  and  $a_2 = b_2$ .

A vector with a unit magnitude is called a *unit vector*, and it is denoted as  $\vec{u}$ . If the angle of a unit vector with the  $x$ -axis is  $\theta$ , the components of the unit vector are

$$\mathbf{u} = \begin{Bmatrix} \cos \theta \\ \sin \theta \end{Bmatrix} \quad (2.5)$$

If a unit vector is defined along the axis of vector  $\vec{a}$ , the unit vector is written as  $\vec{u}_{(a)}$ . Any vector, such as  $\vec{a}$ , can be described as the product of its magnitude and a unit vector defined along its axis as

$$\vec{a} = a\vec{u}_{(a)} \quad \mathbf{a} = a\mathbf{u}_{(a)} \Rightarrow \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} au_1 \\ au_2 \end{Bmatrix} \quad (2.6)$$

In situations when it is clear that vector  $\vec{u}$  is along the axis of vector  $\vec{a}$ , for notational simplification we may not use the subscript  $(a)$ ; we simply write  $\mathbf{a} = a\mathbf{u}$ . Unit vectors along the  $x$  and  $y$  axes are denoted as  $\vec{u}_{(x)}$  and  $\vec{u}_{(y)}$ , respectively. These vectors have the following algebraic representations:

$$\mathbf{u}_{(x)} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}, \quad \mathbf{u}_{(y)} = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \quad (2.7)$$

Vectors having the same or parallel axes are called *parallel vectors*. *Collinear vectors* have the same direction and the same axis. A *zero* or *null vector* has zero magnitude and therefore no specified direction. A zero vector, denoted by  $\vec{0}$  or  $\mathbf{0}$ , has its every component equal to zero.

The *scalar* or *dot product* of two vectors  $\vec{a}$  and  $\vec{b}$  is defined as the product of the magnitudes of the two vectors and the cosine of the angle between them, that is,  $\theta_{a,b}$ . The angle could be measured either from  $\vec{a}$  to  $\vec{b}$  or from  $\vec{b}$  to  $\vec{a}$ . In algebraic form, the scalar product of two vectors is determined in component form. The geometric and algebraic presentations are

$$\vec{a} \cdot \vec{b} = ab \cos \theta_{a,b} \quad \mathbf{a}'\mathbf{b} = a_1 b_1 + a_2 b_2 \quad (2.8)$$

If the two vectors are nonzero (i.e., if  $a \neq 0$  and  $b \neq 0$ ), then their scalar product is zero only if  $\cos \theta = 0$ . Two nonzero vectors are thus said to be *orthogonal* (perpendicular) if their scalar product is zero:

$$\vec{a} \cdot \vec{b} = 0 \quad \mathbf{a}'\mathbf{b} = 0 \quad (2.9)$$

We note that the order of the two vectors in a scalar product operation could be reversed:

$$\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a} \quad \mathbf{a}'\mathbf{b} = \mathbf{b}'\mathbf{a} \quad (2.10)$$

It should be obvious that the scalar product of a vector by itself yields the square of the magnitude of the vector:

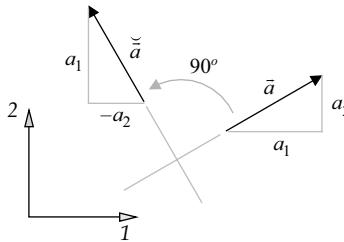
$$\vec{a} \cdot \vec{a} = a^2 \quad \mathbf{a}'\mathbf{a} = a_1^2 + a_2^2 = a^2 \quad (2.11)$$

Vector  $\vec{a}$  (or  $\mathbf{a}$ ) rotated  $90^\circ$  in the positive sense (CCW) is denoted by  $\check{\vec{a}}$  (or  $\check{\mathbf{a}}$ ) as shown in Figure 2.6. The components of  $\check{\vec{a}}$  in terms of the components of  $\vec{a}$  are\*

$$\check{\mathbf{a}} = \begin{Bmatrix} -a_2 \\ a_1 \end{Bmatrix} \quad (2.12)$$

---

\* The semicircle overscore is called a *breve*. We could read  $\check{x}$  as x-rotated.

**FIGURE 2.6**

Rotating a vector  $90^\circ$  in the positive direction.

The following function M-file rotates a vector  $90^\circ$  positively. This function will be used in several programs to follow in this and other chapters.

```
function s_r = s_rot(s)
    s_r = [-s(2); s(1)];
    a = [2; 3];
    a_rotated = s_rot(a);
```

```
a =
2
3
a_rotated =
-3
2
```

It should be obvious that

$$\check{a} \cdot \bar{a} = 0 \quad \check{a}' \mathbf{a} = 0 \quad (2.13)$$

The *vector or cross product* of two vectors  $\bar{a}$  and  $\bar{b}$  is defined as vector  $\bar{c}$ :

$$\bar{a} \times \bar{b} = \bar{c} \quad (2.14)$$

where  $\bar{c}$  is perpendicular to the plane of  $\bar{a}$  and  $\bar{b}$ . In planar kinematics and dynamics, since  $\bar{a}$  and  $\bar{b}$  are in the  $x-y$  plane, vector  $\bar{c}$  will be directed out of (or into) the plane. The magnitude of vector  $\bar{c}$  can be computed as

$$c = ab \sin \theta_{a,b} \quad c = \check{\mathbf{a}}' \mathbf{b} \quad (2.15)$$

where  $\theta_{a,b}$  is the angle between  $\bar{a}$  and  $\bar{b}$  measured by rotating  $\bar{a}$  toward  $\bar{b}$  in a positive sense. If the computed value of  $c$  is positive, vector  $\bar{c}$  is coming out of the plane; if the value is negative, the vector is going into the plane.

### Proof 2.1

The following is the proof for  $c = ab \sin \theta_{a,b} = \check{\mathbf{a}}' \mathbf{b}$ . We denote the angles of  $\bar{a}$  and  $\bar{b}$  with respect to the  $x$ -axis as  $\theta_a$  and  $\theta_b$ . Therefore, the angle between the two vectors is  $\theta_{a,b} = \theta_b - \theta_a$ . We can then write as

$$\begin{aligned}
c &= ab \sin \theta_{a,b} = ab \sin(\theta_b - \theta_a) = ab(\sin \theta_b \cos \theta_a - \cos \theta_b \sin \theta_a) \\
&= (b \sin \theta_b)(a \cos \theta_a) + (b \cos \theta_b)(-a \sin \theta_a) \\
&= (b_2)(a_1) + (b_1)(-a_2) = \left\{ \begin{array}{cc} -a_2 & a_1 \end{array} \right\} \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} = \bar{\mathbf{a}}' \mathbf{b}
\end{aligned}$$

Since reversal of the order of the two vectors in Eq. (2.15) yields a vector in the opposite direction, it is clear that

$$\vec{a} \times \vec{b} = -\vec{b} \times \vec{a} \quad \bar{\mathbf{a}}' \mathbf{b} = -\bar{\mathbf{b}}' \mathbf{a} \quad (2.16)$$

For two vectors  $\vec{a}$  and  $\vec{b}$  to be parallel, we must have

$$\vec{a} \times \vec{b} = 0 \quad \bar{\mathbf{a}}' \mathbf{b} = 0 \quad (2.17)$$

This equation states that the condition for vector  $\vec{a}$  to be parallel to vector  $\vec{b}$  is equivalent to the condition for vector  $\vec{a}$  to be perpendicular to vector  $\vec{b}$ .

### Example 2.3

The components of two planar vectors  $\vec{a}$  and  $\vec{b}$  are given as  $\mathbf{a} = \{2 \ 3\}'$  and  $\mathbf{b} = \{4 \ 5\}'$ . Compute  $c = \bar{\mathbf{a}}' \mathbf{b}$  and  $d = \mathbf{b}' \mathbf{a}$ .

#### MATLAB/Chapter 2/Example \_ 2 \_ 3

```
a = [2; 3]; b = [4; 5];
c = s_rot(a)' * b
d = s_rot(b)' * a
```

```
c =
-2
d =
2
```

If two vectors,  $\vec{a}$  and  $\vec{b}$ , are both rotated, then their scalar product can be expressed as

$$\check{\vec{a}} \cdot \check{\vec{b}} = \vec{a} \cdot \vec{b} \quad \bar{\mathbf{a}}' \check{\mathbf{b}} = \mathbf{a}' \mathbf{b} \quad (2.18)$$

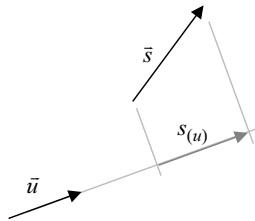
We can also show that the rotation of the sum of two (or more) vectors is distributive:

$$\check{\vec{a}} + \check{\vec{b}} = \check{\vec{a}} + \check{\vec{b}} \quad \mathbf{a} \check{+} \mathbf{b} = \bar{\mathbf{a}} + \check{\mathbf{b}} \quad (2.19)$$

where  $\check{+}$  denotes adding the vectors then rotating the sum.

Projection of a vector, such as  $\vec{s}$  onto an axis defined by the unit vector  $\vec{u}$ , as shown in Figure 2.7, is obtained from the scalar product of the two vectors:

$$s_{(u)} = \mathbf{u}' \mathbf{s} \quad (2.20)$$



**FIGURE 2.7**  
Projection of a vector onto an axis.

### 2.2.1 Arrays

A set of elements or functions stacked and arranged in *curly brackets* is called an *array*. For example, for vector  $\mathbf{a}$ , its two Cartesian components,  $a_{(x)}$  and  $a_{(y)}$ , and its angle,  $\theta$ , can be arranged in a three-array as

$$\left\{ \begin{array}{c} a_{(x)} \\ a_{(y)} \\ \theta \end{array} \right\}$$

An array can have a dimension of one or greater. An algebraic vector as given by Eq. (2.2) is a special case of a two-array containing only the Cartesian components of a vector. By our definition, an array is a *column stack* of elements. In compact form, arrays are represented by *lowercase boldface letters*. Sometimes, for convenience, we may show an array as a *row stack* with a *transpose superscript*. For example, the  $n$ -array  $\mathbf{q}$  can be expressed as

$$\mathbf{q} = \left\{ \begin{array}{cccc} q_1 & q_2 & \cdots & q_n \end{array} \right\}'$$

Stacking up arrays or algebraic vectors forms a new array. For example, if  $\mathbf{s}_i$ ,  $i = 1, 2, \dots, n$ , are two-vectors, then

$$\mathbf{s} = \left\{ \begin{array}{c} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_n \end{array} \right\} \quad (2.21)$$

is a  $2n$ -array. Note that if we need to write Eq. (2.21) as the transpose of a row array, we must express it as

$$\mathbf{s} = \left\{ \begin{array}{cccc} \mathbf{s}'_1 & \mathbf{s}'_2 & \cdots & \mathbf{s}'_n \end{array} \right\}'$$

```
a = [2; 3]; b = [4; 5]; c = [6; 7];
q = [a; b; c] .....
```

$Q = [a \ b \ c]$  .....

```
q =
2
3
4
5
6
7
Q =
2   4   6
3   5   7
```

For two arrays **a** and **b** with dimensions  $n$ , the array summation is defined as

$$\mathbf{c} = \mathbf{a} + \mathbf{b} \quad (2.22)$$

The scalar product of these two arrays is defined as

$$\mathbf{a}'\mathbf{b} = \mathbf{b}'\mathbf{a} = a_1b_1 + a_2b_2 + \cdots + a_nb_n \quad (2.23)$$

These two arrays are said to be orthogonal if

$$\mathbf{a}'\mathbf{b} = 0 \quad (2.24)$$

We define the square root of the sum of squares of the elements of an array (or a vector) **a** as the *norm* of **a**\*:

$$\text{norm}(\mathbf{a}) = \sqrt{\mathbf{a}'\mathbf{a}} \quad (2.25)$$

```
b = [4; 3]; mag_b = norm(b) .....
```

```
mag_b =
5
```

## 2.3 Matrices

Compact matrix notation often allows one to concentrate on the form of a system of equations and what it means, rather than on the minute details of its construction. Matrix manipulation also allows for the organized development and simplification of systems of equations. In this section, several useful definitions and identities are stated that are used extensively throughout this text.

---

\* There is a more general definition for the norm of an array or a matrix. However, in this textbook, we use the special case of the definition for an array as given in Eq. (2.25).

A matrix with  $m$  rows and  $n$  columns is said to be of dimension  $m \times n$  and is denoted by an uppercase boldface letter; it is written in the form

$$\mathbf{A} \equiv [\mathbf{a}_{ij}] \equiv \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

where a typical element  $a_{ij}$  is located at the intersection of the  $i$ th row and  $j$ th column. The *transpose* of a matrix is formed by interchanging rows and columns and is designated by the right *apostrophe*. Thus, if  $a_{ij}$  is the  $ij$ th element of matrix  $\mathbf{A}$ ,  $a_{ij}$  is the  $ij$ th element of its transpose  $\mathbf{A}'$ .

$\mathbf{A} = [2 \ 3 \ 4; \ 5 \ 6 \ 7]$	$\mathbf{A} =$ 2    3    4 5    6    7
$\mathbf{a}_{23} = \mathbf{A}(2, 3)$	$\mathbf{a}_{23} =$ 7
$\mathbf{C} = \mathbf{A}'$	$\mathbf{C} =$ 2    5 3    6 4    7
$\mathbf{c}_{32} = \mathbf{C}(3, 2)$	$\mathbf{c}_{32} =$ 7

There is more than one way to construct a matrix in MATLAB. For example:

$\mathbf{A} = [2 \ 3 \ 4; \ 5 \ 6 \ 7]$	$\mathbf{A} =$ 2    3    4 5    6    7
$\mathbf{B} = [2 \ 3 \ 4$ $\quad \quad \quad 5 \ 6 \ 7]$	$\mathbf{B} =$ 2    3    4 5    6    7

Assume that  $\mathbf{b}_i$  for  $i = 1, \dots, 4$  are two-vectors and matrix  $\mathbf{B}$  is defined as

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \\ \mathbf{b}_3 & \mathbf{b}_4 \end{bmatrix} \quad (2.26)$$

The transpose of this matrix must be presented as

$$\mathbf{B}' = \begin{bmatrix} \mathbf{b}'_1 & \mathbf{b}'_3 \\ \mathbf{b}'_2 & \mathbf{b}'_4 \end{bmatrix} \quad (2.27)$$

A matrix with only one column is called a *column* matrix, which is the same as an array. A matrix with only one row is called a *row* matrix. Therefore, the transpose of a column matrix is a row matrix and vice versa.

A *square* matrix has an equal number of rows and columns. A *diagonal* matrix is a square matrix with  $a_{ij} = 0$  for  $i \neq j$  and at least one nonzero diagonal term. An  $n \times n$  diagonal matrix may be denoted by

$$\mathbf{A} \equiv \text{diag} \begin{bmatrix} a_{11} & a_{22} & \cdots & a_{nn} \end{bmatrix} \quad (2.28)$$

<code>m = [2; 3; 4]</code>	<code>M =</code>
	2
	3
	4
<code>M = diag(m)</code>	<code>M =</code>
	2 0 0
	0 3 0
	0 0 4

If square matrices  $\mathbf{B}_i$ ,  $i = 1, \dots, k$ , are arranged along the diagonal of a matrix  $\mathbf{D}$ , it is called a *block diagonal* matrix and may be denoted as

$$\mathbf{D} = \text{blkdiag} \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_k \end{bmatrix} \quad (2.29)$$

An  $n \times n$  *unit* or *identity* matrix, denoted by  $\mathbf{I}$ , is a diagonal matrix with  $a_{ii} = 1$ ,  $i = 1, \dots, n$ . A *null* or *zero* matrix, designated by  $\mathbf{0}$ , has  $a_{ij} = 0$  for all  $i$  and  $j$ .

<code>I22 = eye(2) % or eye(2,2)</code>	<code>I22 =</code>
	1 0
	0 1
<code>z23 = zeros(2,3)</code>	<code>z23 =</code>
	0 0 0
	0 0 0
<code>z22 = zeros(2) % or zeros(2,2)</code>	<code>z22 =</code>
	0 0
	0 0

If  $a_{ij} = a_{ji}$  for all  $i$  and  $j$ , the matrix  $\mathbf{A}$  is called *symmetric*, that is,  $\mathbf{A} = \mathbf{A}'$ .

### 2.3.1 Matrix Operations

If two matrices  $\mathbf{A}$  and  $\mathbf{B}$  are of the same dimension, they are said to be *equal* matrices if  $a_{ij} = b_{ij}$  for all  $i$  and  $j$ . The sum or the difference of two *equidimensional* matrices  $\mathbf{A}$  and  $\mathbf{B}$  is a matrix with the same dimension, defined as

$$\mathbf{C} = \mathbf{A} \pm \mathbf{B} \quad (2.30)$$

where  $c_{ij} = a_{ij} \pm b_{ij}$  for all  $i$  and  $j$ .

Multiplication of a matrix by a scalar is defined as

$$\mathbf{C} = \alpha \mathbf{A} \quad (2.31)$$

where  $c_{ij} = \alpha a_{ij}$  for all  $i$  and  $j$ .

Assume  $\mathbf{A}$  is an  $m \times p$  matrix and  $\mathbf{B}$  is an  $p \times n$  matrix, written in the form

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{a}'_1 \\ \mathbf{a}'_2 \\ \vdots \\ \mathbf{a}'_m \end{bmatrix}, \mathbf{B} \equiv \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \end{bmatrix} \quad (2.32)$$

where  $\mathbf{a}'_i$ ,  $i, \dots, m$ , are row  $p$ -arrays and  $\mathbf{b}_i$ ,  $i, \dots, n$ , are column  $p$ -arrays. Then the matrix product of  $\mathbf{A}$  and  $\mathbf{B}$  is defined as the  $m \times n$  matrix

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} \mathbf{a}'_1 \mathbf{b}_1 & \mathbf{a}'_1 \mathbf{b}_2 & \cdots & \mathbf{a}'_1 \mathbf{b}_n \\ \mathbf{a}'_2 \mathbf{b}_1 & \mathbf{a}'_2 \mathbf{b}_2 & \cdots & \mathbf{a}'_2 \mathbf{b}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}'_m \mathbf{b}_1 & \mathbf{a}'_m \mathbf{b}_2 & \cdots & \mathbf{a}'_m \mathbf{b}_n \end{bmatrix} \quad (2.33)$$

where  $c_{ij} = \mathbf{a}'_i \mathbf{b}_j$ . It is important to note that the product of two matrices is defined only if the number of columns in the first matrix equals the number of rows in the second matrix. It is clear from the definition that, in general,

$$\mathbf{AB} \neq \mathbf{BA} \quad (2.34)$$

In fact, the products  $\mathbf{AB}$  and  $\mathbf{BA}$  are defined only if both  $\mathbf{A}$  and  $\mathbf{B}$  are square and of equal dimensions.

#### Example 2.4

##### MATLAB/Chapter 2/Example \_ 2 \_ 4

```
A = [1 2 3; 4 5 6];
B = [7 8 9; -1 -2 -3];
C = A + B      .....
D = A*B'      .....
```

```
C =
8   10   12
3   3    3
D =
50  -14
122 -32
```

The following identities are valid, assuming that the matrices have proper dimensions:

$$(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC} \quad (2.35)$$

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}) = \mathbf{ABC} \quad (2.36)$$

$$(\mathbf{A} + \mathbf{B})' = \mathbf{A}' + \mathbf{B}' \quad (2.37)$$

$$(\mathbf{AB})' = \mathbf{B}'\mathbf{A}' \quad (2.38)$$

Consider an  $m \times P$  matrix  $\mathbf{A}$ . If linear combinations of the rows of  $\mathbf{A}$  are nonzero, that is, if

$$\alpha'\mathbf{A} \neq \mathbf{0} \quad (2.39)$$

for all  $\alpha' = \{\alpha_1 \quad \alpha_2 \quad \alpha_m\} \neq \mathbf{0}$ , the rows of  $\mathbf{A}$  are said to be *linearly independent*. Otherwise, if

$$\alpha'\mathbf{A} = \mathbf{0} \quad (2.40)$$

for at least one  $\alpha \neq \mathbf{0}$ , then the rows of  $\mathbf{A}$  are said to be *linearly dependent*, and at least one row can be written as a linear combination of the other rows.

### Example 2.5

Consider the following matrices and array:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The only solution to the set of three algebraic equations  $\alpha'\mathbf{A} = \mathbf{0}$  is  $\alpha' = \{0 \quad 0\}$ . Therefore, the rows of matrix  $\mathbf{A}$  are linearly independent. For matrix  $\mathbf{B}$ ,  $\alpha' = \{3 \quad -1\}$  results in  $\alpha'\mathbf{B} = \mathbf{0}$ , stating that the rows of  $\mathbf{B}$  are not independent. Similarly,  $\alpha' = \{-1 \quad 2 \quad -1\}$  results in  $\alpha'\mathbf{C} = \mathbf{0}$ , and therefore, the rows of  $\mathbf{C}$  are not independent.

The *row rank (column rank)* of a matrix  $\mathbf{A}$  is defined as the largest number of linearly independent rows (columns) in the matrix. The row and column ranks of any matrix are equal. Each of them can thus be called the *rank* of the matrix. A square matrix with linearly independent rows (columns) is said to have *full rank*. When a square matrix does not have full rank, it is called *singular*. For a nonsingular matrix, there is an *inverse*, denoted by  $\mathbf{A}^{-1}$ , such that

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \quad (2.41)$$

$C = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9];$	
$\text{rank}(C)$	.....
$\text{rank}(C')$	.....

$A = [1 \ 2; \ 3 \ 4]; \ A_{\text{inv}} = \text{inv}(A) \quad \dots$	$A_{\text{inv}} =$
	-2      1.0
	1.5    -0.5

The following identities are valid:

$$(\mathbf{A}^{-1})' = (\mathbf{A}')^{-1} \quad (2.42)$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \quad (2.43)$$

A special nonsingular matrix that arises often in kinematics and dynamics is called an *orthonormal*\* matrix, with the property that

$$\mathbf{A}' = \mathbf{A}^{-1} \quad (2.44)$$

It is obvious that for an orthonormal matrix

$$\mathbf{A}'\mathbf{A} = \mathbf{I} \quad (2.45)$$

Because computing the inverse of a nonsingular matrix is generally time consuming, it is important to know when a matrix is orthonormal. In this special case, the inverse is trivially constructed by transposing the matrix.

### 2.3.2 Linear Algebraic Equations

Linear algebraic equations are expressed in a general form as

$$\mathbf{A}\mathbf{x} = \mathbf{c} \quad (2.46)$$

where  $\mathbf{A}$  is the matrix of known coefficients,  $\mathbf{x}$  is the array of unknowns, and  $\mathbf{c}$  is the array of known quantities. If  $\mathbf{A}$  is a square and nonsingular matrix, one obvious method of solution is to invert the matrix and express Eq. (2.46) as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{c} \quad (2.47)$$

Although matrix inversion provides us with the solution, computing the inverse of the coefficient matrix is computationally inefficient, especially for large-scale problems. Because our main objective is to find the unknowns and not necessarily the inverse of the coefficient matrix, instead of inverting the coefficient matrix, in MATLAB we can solve for the unknowns using the *backslash* operator (also known as *matrix left division*). The use of backslash operator is highly recommended over the inverse operation.

#### Example 2.6

##### MATLAB/Chapter 2/Example \_ 2 \_ 6

```
D = [-1 -1.64 2.14
      0 2.51 -0.51
      1 0 0];
rhs = [0 0 2*pi]';
ans_1 = inv(D)*rhs .....  

ans_2 = D\rhs .....
```

```
ans_1 =
6.2832
0.7066
3.4776
ans_2 =
6.2832
0.7066
3.4776
```

---

\* In some textbooks, the term *orthogonal* is used for *orthonormal*.

## 2.4 Vector, Array, and Matrix Differentiation

In the kinematics and dynamics of mechanical systems, vectors representing the position of points or bodies, or quantities describing the geometry of the dynamics of the motion, are often functions of time or some other variables. In analyzing these quantities, time derivatives or partial derivatives with respect to some variables are needed. In this section, these derivatives are defined and the notation used in the text is explained.

### 2.4.1 Time Derivatives

In analyzing velocities and accelerations, time derivatives of vectors and arrays that describe the position of points and bodies or the geometry of motion must often be calculated. The following identities are expressed for arrays, but they are also applicable to vectors. Consider an array

$$\mathbf{a} \equiv \mathbf{a}(t) = \begin{Bmatrix} a_1(t) \\ \vdots \\ a_n(t) \end{Bmatrix} \quad (2.48)$$

where  $t$  is the independent variable (or parameter) *time*. The time derivative of  $\mathbf{a}$  is denoted by

$$\dot{\mathbf{a}} \equiv \frac{d}{dt} \mathbf{a} = \begin{Bmatrix} \frac{d}{dt} a_1(t) \\ \vdots \\ \frac{d}{dt} a_n(t) \end{Bmatrix} \quad (2.49)$$

The derivative of the sum of two arrays gives

$$\frac{d}{dt} (\mathbf{a} + \mathbf{b}) = \dot{\mathbf{a}} + \dot{\mathbf{b}} \quad (2.50)$$

which is completely analogous to the ordinary differentiation rule that the derivative of the sum is the sum of the derivatives. If  $\alpha = \alpha(t)$  is a scalar function of time, the following vector/array forms of the product rule of differentiation can also be verified:

$$\frac{d}{dt} (\alpha \mathbf{a}) = \dot{\alpha} \mathbf{a} + \alpha \dot{\mathbf{a}} \quad (2.51)$$

$$\frac{d}{dt} (\mathbf{a}' \mathbf{b}) = \dot{\mathbf{a}}' \mathbf{b} + \mathbf{a}' \dot{\mathbf{b}} \quad (2.52)$$

$$\frac{d}{dt} \ddot{\mathbf{a}} = \ddot{\mathbf{a}} = \ddot{\mathbf{a}} \quad (2.53)$$

In Eq. (2.53),  $\mathbf{a}$  is assumed to be a two-vector.

The second time derivative of a vector/array  $\mathbf{a} \equiv \mathbf{a}(t)$  is denoted as

$$\ddot{\mathbf{a}} \equiv \frac{d}{dt} \dot{\mathbf{a}} = \frac{d}{dt} \left( \frac{d}{dt} \mathbf{a} \right) \quad (2.54)$$

Thus, the differentiation is performed on every component of the array.

The time derivative of a vector that is described in terms of its components in a nonmoving  $x$ - $y$  frame can be obtained, similar to Eq. (2.49) for an array, as the time derivative of its components:

$$\dot{\mathbf{a}} \equiv \frac{d}{dt} \mathbf{a} = \left\{ \begin{array}{l} \frac{d}{dt} a_x(t) \\ \frac{d}{dt} a_y(t) \end{array} \right\} \quad (2.55)$$

But if the vector is expressed in terms of its magnitude and a unit vector, as in Eq. (2.6), its time derivative can be determined as

$$\frac{d}{dt} \mathbf{a} = \left( \frac{d}{dt} a \right) \mathbf{u}_{(a)} + a \frac{d}{dt} \mathbf{u}_{(a)} \quad \dot{\mathbf{a}} = \dot{a} \mathbf{u}_{(a)} + a \dot{\mathbf{u}}_{(a)} \quad (2.56)$$

where we have denoted the *time derivative of the magnitude* of vector  $\mathbf{a}$  as  $\dot{a}$ . Note that for a vector, the *time derivative of the magnitude*  $\dot{a}$  and the *magnitude of the time derivative*  $|\dot{\mathbf{a}}|$  may or may not be the same.

The second time derivative of Eq. (2.56) is expressed as

$$\ddot{\mathbf{a}} = \ddot{a} \mathbf{u}_{(a)} + a \ddot{\mathbf{u}}_{(a)} + 2\dot{a} \dot{\mathbf{u}}_{(a)} \quad (2.57)$$

where we have denoted the second time derivative of the magnitude of the vector as  $\ddot{a}$ .

Just as in the differentiation of a vector or an array function, the derivative of a matrix whose components depend on  $t$  may be defined. The derivative of  $\mathbf{A}(t) = [a_{ij}(t)]$  is defined as

$$\dot{\mathbf{A}} \equiv \frac{d}{dt} \mathbf{A} = \left[ \frac{d}{dt} a_{ij}(t) \right] \quad (2.58)$$

With this definition, and assuming that  $\mathbf{B} = \mathbf{B}(t)$ ,  $\alpha = \alpha(t)$ , and  $\mathbf{a} = \mathbf{a}(t)$ , it can be verified that

$$\frac{d}{dt} (\mathbf{A} + \mathbf{B}) = \dot{\mathbf{A}} + \dot{\mathbf{B}} \quad (2.59)$$

$$\frac{d}{dt} (\mathbf{AB}) = \dot{\mathbf{A}}\mathbf{B} + \mathbf{A}\dot{\mathbf{B}} \quad (2.60)$$

$$\frac{d}{dt} (\alpha \mathbf{B}) = \dot{\alpha} \mathbf{B} + \alpha \dot{\mathbf{B}} \quad (2.61)$$

$$\frac{d}{dt} (\mathbf{A}\mathbf{a}) = \dot{\mathbf{A}}\mathbf{a} + \mathbf{A}\dot{\mathbf{a}} \quad (2.62)$$

## 2.4.2 Partial Derivatives

Assume that  $\mathbf{q}$  is an  $n$ -array, where every element of  $\mathbf{q}$  is a variable and  $\Phi$  is a function of  $\mathbf{q}$ , that is,  $\Phi = \Phi(\mathbf{q})$ . The partial derivative of  $\Phi$  with respect to  $\mathbf{q}$  is defined as

$$\frac{\partial \Phi}{\partial \mathbf{q}} \equiv \begin{bmatrix} \frac{\partial \Phi}{\partial q_1} & \frac{\partial \Phi}{\partial q_2} & \dots & \frac{\partial \Phi}{\partial q_n} \end{bmatrix} \quad (2.63)$$

This expression indicates that the partial derivative of a single function with respect to an  $n$ -array is a row matrix with  $n$  columns.

### Example 2.7

Array  $\mathbf{q}$  designates four variables as  $\mathbf{q} = \{x_1 \ x_2 \ x_3 \ x_4\}'$ . Determine the partial derivative of a scalar function  $\Phi$  with respect to  $\mathbf{q}$  where  $\Phi = -x_1 + 3x_2x_4^2$ .

#### Solution

Partial derivatives of the function with respect to the individual variables are as follows:  $\partial\Phi/\partial x_1 = -1$ ,  $\partial\Phi/\partial x_2 = 3x_4^2$ ,  $\partial\Phi/\partial x_3 = 0$ , and  $\partial\Phi/\partial x_4 = 6x_2x_4$ . Therefore,

$$\frac{\partial \Phi}{\partial \mathbf{q}} = \begin{bmatrix} -1 & 3x_4^2 & 0 & 6x_2x_4 \end{bmatrix}$$

If  $\Phi \equiv \Phi(\mathbf{q}) \equiv \{\Phi_1(\mathbf{q}) \ \Phi_2(\mathbf{q}) \ \dots \ \Phi_m(\mathbf{q})\}'$  is an  $m$ -array of differentiable functions of  $\mathbf{q}$ , the partial derivative of  $\Phi$  with respect to  $\mathbf{q}$  is defined as the following  $m \times n$  matrix:

$$\frac{\partial \Phi}{\partial \mathbf{q}} \equiv \begin{bmatrix} \frac{\partial \Phi_1}{\partial q_1} & \frac{\partial \Phi_1}{\partial q_2} & \dots & \frac{\partial \Phi_1}{\partial q_n} \\ \frac{\partial \Phi_2}{\partial q_1} & \frac{\partial \Phi_2}{\partial q_2} & \dots & \frac{\partial \Phi_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Phi_m}{\partial q_1} & \frac{\partial \Phi_m}{\partial q_2} & \dots & \frac{\partial \Phi_m}{\partial q_n} \end{bmatrix} \quad (2.64)$$

### Example 2.8

Array  $\mathbf{q}$  designates six variables as  $\mathbf{q} = \{x_1 \ y_1 \ x_2 \ y_2 \ x_3 \ y_3\}'$ . Determine the partial derivative of two functions  $\Phi = \{\Phi_1 \ \Phi_2\}'$  with respect to  $\mathbf{q}$  where

$$\Phi_1 = x_1 + 3y_1 - x_2 + 2x_3 - y_3$$

$$\Phi_2 = x_1y_1 + y_2 + 2y_3 - 4$$

#### Solution

The partial derivative of  $\Phi$  with respect to  $\mathbf{q}$  is the following  $2 \times 6$  matrix:

$$\frac{\partial \Phi}{\partial \mathbf{q}} = \begin{bmatrix} 1 & 3 & -1 & 0 & 2 & -1 \\ y_1 & x_1 & 0 & 1 & 0 & 2 \end{bmatrix}$$

Note that the rows of this matrix follow the order of the functions and the columns follow the order of the variables in  $\mathbf{q}$ .

The partial derivative of the scalar product of two  $n$ -array functions  $\mathbf{a}(\mathbf{q})$  and  $\mathbf{b}(\mathbf{q})$  is a row matrix given as

$$\frac{\partial(\mathbf{a}'\mathbf{b})}{\partial \mathbf{q}} = \mathbf{b}' \frac{\partial \mathbf{a}}{\partial \mathbf{q}} + \mathbf{a}' \frac{\partial \mathbf{b}}{\partial \mathbf{q}} \quad (2.65)$$

where the dimension of the resultant row matrix is the same as the dimension of array  $\mathbf{q}$ .

### Example 2.9

Arrays  $\mathbf{a}$  and  $\mathbf{b}$  are functions of a single variable  $x$ . Determine the partial derivative of  $\mathbf{a}'\mathbf{b}$  with respect to  $x$  if

$$\mathbf{a} = \begin{Bmatrix} 2x \\ -1 \\ x \end{Bmatrix} \quad \mathbf{b} = \begin{Bmatrix} -3 \\ x \\ -x \end{Bmatrix}$$

#### Solution

The derivatives of  $\mathbf{a}$  and  $\mathbf{b}$  with respect to  $x$  are

$$\frac{\partial \mathbf{a}}{\partial x} = \begin{Bmatrix} 2 \\ 0 \\ 1 \end{Bmatrix} \quad \frac{\partial \mathbf{b}}{\partial x} = \begin{Bmatrix} 0 \\ 1 \\ -1 \end{Bmatrix}$$

Using Eq. (2.65), it is found that

$$\frac{\partial(\mathbf{a}'\mathbf{b})}{\partial x} = \{-3 \ x \ -x\} \begin{Bmatrix} 2 \\ 0 \\ 1 \end{Bmatrix} + \{2x \ -1 \ x\} \begin{Bmatrix} 0 \\ 1 \\ -1 \end{Bmatrix} = [-2x - 7]$$

where the result is a  $1 \times 1$  matrix. To verify Eq. (2.65), this result can also be obtained directly by determining the scalar product  $\mathbf{a}'\mathbf{b} = -x^2 - 7x$ , and then taking its partial derivative with respect to  $x$ , resulting in the same answer as  $-2x - 7$ .

### Example 2.10

Array  $\mathbf{q}$  contains two variables as  $\mathbf{q} = [x \ y]'$ . Arrays  $\mathbf{a}$  and  $\mathbf{b}$  are functions of  $\mathbf{q}$ :

$$\mathbf{a} = \begin{Bmatrix} x - y \\ x + y \\ y - 1 \end{Bmatrix} \quad \mathbf{b} = \begin{Bmatrix} -x + y \\ x + 2 \\ -x - y \end{Bmatrix}$$

Determine the partial derivative of  $\mathbf{a}'\mathbf{b}$  with respect to  $\mathbf{q}$ .

#### Solution

The derivatives of  $\mathbf{a}$  and  $\mathbf{b}$  with respect to  $\mathbf{q}$  are

$$\frac{\partial \mathbf{a}}{\partial \mathbf{q}} = \begin{Bmatrix} 1 & -1 \\ 1 & 1 \\ 0 & 1 \end{Bmatrix} \quad \frac{\partial \mathbf{b}}{\partial \mathbf{q}} = \begin{Bmatrix} -1 & 1 \\ 1 & 0 \\ -1 & -1 \end{Bmatrix}$$

We can now compute the following terms:

$$\mathbf{b}' \frac{\partial \mathbf{a}}{\partial \mathbf{q}} = \left\{ \begin{array}{ccc} -x+y & x+2 & -x-y \end{array} \right\} \left[ \begin{array}{cc} 1 & -1 \\ 1 & 1 \\ 0 & 1 \end{array} \right] = \left[ \begin{array}{cc} y+2 & x-2y+2 \end{array} \right]$$

$$\mathbf{a}' \frac{\partial \mathbf{b}}{\partial \mathbf{q}} = \left\{ \begin{array}{ccc} x-y & x+y & y-1 \end{array} \right\} \left[ \begin{array}{cc} -1 & 1 \\ 1 & 0 \\ -1 & -1 \end{array} \right] = \left[ \begin{array}{cc} y+1 & x-2y+1 \end{array} \right]$$

Using Eq. (2.65), it is found that

$$\frac{\partial(\mathbf{a}'\mathbf{b})}{\partial \mathbf{q}} = \left[ \begin{array}{cc} 2y+3 & 2x-4y+3 \end{array} \right]$$

This result can also be verified by computing the scalar product  $\mathbf{a}'\mathbf{b}$  first and then taking its partial derivative with respect to  $x$  and  $y$ .

## 2.5 Equations and Expressions

In this text, we use the terms *equations* and *expressions* to denote different forms of identities. The term *equation* or *a set of equations* will refer to one or more analytical identities that need analytical or numerical solution to determine the corresponding unknown(s). For example, the following set of two algebraic equations needs to be solved simultaneously to determine the unknowns  $x^B$  and  $y^B$ :

$$\begin{aligned} (x^B - 2)^2 + (y^B - 3)^2 - 9 &= 0 \\ (x^B - 4)^2 + (y^B - 1)^2 - 4 &= 0 \end{aligned} \tag{2.66}$$

whereas, in the two identities

$$\begin{aligned} x^A &= L_1 \cos \theta \\ y^A &= L_1 \sin \theta \end{aligned} \tag{2.67}$$

if  $L_1$  and  $\theta$  are known quantities, a simple substitution yields the values for the unknowns  $x^A$  and  $y^A$ . The identities that require solutions of the equations, such as Eq. (2.66), are referred to as *equations*, whereas the identities that require only substitutions, such as Eq. (2.67), are referred to as *expressions*.

### 2.5.1 Compact and Expanded Forms

An equation or expression may appear in compact form, in matrix form, or in expanded form. As an example, consider the following equation:

$$\mathbf{r}^P = \mathbf{r} + \mathbf{A}\mathbf{s}_{(\xi-\eta)}^P \tag{2.68}$$

This is the *compact form* of presenting an equation, where  $\mathbf{r}^P$ ,  $\mathbf{r}$ , and  $\mathbf{s}_{(\xi-\eta)}^P$  are two-vectors, and  $\mathbf{A}$  is a  $2 \times 2$  matrix. In matrix form, these entities are expressed as

$$\mathbf{r}^P \equiv \begin{Bmatrix} x^P \\ y^P \end{Bmatrix}, \mathbf{r} \equiv \begin{Bmatrix} x \\ y \end{Bmatrix}, \mathbf{s}_{(\xi-\eta)}^P \equiv \begin{Bmatrix} \xi^P \\ \eta^P \end{Bmatrix}, \mathbf{A} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

Hence, Eq. (2.68) can be presented in *matrix form* as

$$\begin{Bmatrix} x^P \\ y^P \end{Bmatrix} = \begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{Bmatrix} \xi^P \\ \eta^P \end{Bmatrix} \quad (2.69)$$

The *expanded form* of Eq. (2.68) or (2.69) is given as

$$\begin{aligned} x^P &= x + \xi^P \cos\phi - \eta^P \sin\phi \\ y^P &= y + \xi^P \sin\phi + \eta^P \cos\phi \end{aligned} \quad (2.70)$$

All these representations express the same relationship. The compact form provides an overall view of a relationship between several entities; the matrix form shows the details within an equation in an organized vector/matrix form; and the expanded form also provides the details without the use of vector/matrix brackets. In the following chapters, we will use all the three representations, but mostly the compact form. An equation number may refer to more than one form of representing an equation.

## 2.6 Problems

Numerical calculations for the following problems should be performed with MATLAB.

2.1 Verify that for any unit vector  $\vec{u}$ , the Cartesian components are those expressed by Eq. (2.5). Assume an arbitrary value for the angle in the first, second, third, or fourth quadrant. Repeat the problem for angles in other quadrants.

2.2 Vectors  $\vec{a}$  and  $\vec{b}$ , and a constant  $\alpha$  have been defined as

$$\mathbf{a} = \begin{Bmatrix} 1.5 \\ 0.4 \end{Bmatrix}, \mathbf{b} = \begin{Bmatrix} -0.6 \\ 1.2 \end{Bmatrix}, \alpha = -0.3$$

Determine the following vectors or scalars:

- a.  $\mathbf{c} = \alpha \mathbf{b}$
- b.  $\mathbf{d} = \mathbf{a} + \alpha \mathbf{b}$
- c.  $\mathbf{u}_{(a)}$  and  $\mathbf{u}_{(b)}$
- d.  $\mathbf{a}'\mathbf{b}$
- e.  $\mathbf{b}'\mathbf{a}$

- f. The angle between  $\mathbf{a}$  and  $\mathbf{b}$
  - g. Magnitudes of  $\mathbf{a}$  and  $\mathbf{b}$
  - h.  $\bar{\mathbf{a}}$  and  $\check{\mathbf{b}}$
  - i. The angle between  $\bar{\mathbf{a}}$  and  $\mathbf{a}$
  - j.  $\bar{\mathbf{a}}' \mathbf{b}$
  - k.  $\check{\mathbf{b}}' \mathbf{a}$
- 2.3 Based on the scalar product expressions of Eq. (2.8), prove that  $ab \cos \theta_{a,b} = a_1 b_1 + a_2 b_2$ .
- 2.4 Show that if vector  $\vec{a}$  is rotated by  $90^\circ$ , the components of  $\check{\vec{a}}$  are those expressed by Eq. (2.12) regardless of the angle of  $\vec{a}$  being in any of the four quadrants.
- 2.5 Vector  $\vec{a}$  is rotated  $90^\circ$  in a negative sense (CW direction) to obtain vector  $\hat{\vec{a}}$ . What are the components of  $\hat{\vec{a}}$ ?
- 2.6 Show that for any arbitrary angle  $\phi$ , matrix  $\mathbf{A} = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix}$  is orthonormal.
- 2.7 Consider a unit vector  $\vec{u}$  and its corresponding rotated vector  $\check{\vec{u}}$ . Analytically show that matrix  $\mathbf{A} = [\mathbf{u} \ \check{\mathbf{u}}]$  is orthonormal.
- 2.8 Arrays  $\mathbf{a}$  and  $\mathbf{b}$  are defined as

$$\mathbf{a} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}, \mathbf{b} = \begin{Bmatrix} 4 \\ 5 \\ 6 \end{Bmatrix}$$

Construct the following entities:

- a.  $\mathbf{c} = \begin{Bmatrix} \mathbf{a} \\ \mathbf{b} \end{Bmatrix}$
- b.  $\mathbf{d} = \{\mathbf{a}' \ \mathbf{b}'\}$
- c.  $\mathbf{E} = [\mathbf{a} \ \mathbf{b}]$

- 2.9 The following matrices and arrays are defined as follows:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ -1 & 3 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -1 & 0 & 2 & 1 \\ 1 & 2 & -1 & 3 \\ 3 & 1 & 0 & -2 \end{bmatrix}, \mathbf{c} = \begin{Bmatrix} 2 \\ 1 \\ -1 \end{Bmatrix}, \mathbf{d} = \begin{Bmatrix} 2 \\ 0 \\ -1 \\ 1 \end{Bmatrix}$$

Determine:

- (a)  $\mathbf{Ac}$ ; (b)  $\mathbf{Bd}$ ; (c)  $\mathbf{AB}$ ; (d)  $\mathbf{BA}$ ; (e)  $\mathbf{B}'\mathbf{A}$ ; (f)  $\mathbf{c}'\mathbf{d}$ ; (g)  $\mathbf{c}'\mathbf{B}$ .

- 2.10 Show that Eq. (2.18) is valid.

- 2.11 Use the commands `rank( )` and `inv( )` to determine whether  $\mathbf{A}$  and  $\mathbf{B}$  are singular:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ -1 & 3 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 2 & 4 & -2 \\ -1 & -3 & 2 \\ 0 & -2 & 2 \end{bmatrix}$$

2.12 Given  $\mathbf{a} = \begin{Bmatrix} 4 & 2 \end{Bmatrix}'$  and  $\mathbf{b} = \begin{Bmatrix} 2 & 3 \end{Bmatrix}'$ :

- Compute a unit vector,  $\mathbf{u}_{(a)}$ , in the direction of  $\mathbf{a}$ .
- Project  $\mathbf{b}$  onto  $\mathbf{u}_{(a)}$  to obtain  $\mathbf{b}_{(a)}$ .
- Compute  $\mathbf{c} = \mathbf{b} - \mathbf{b}_{(a)}$ .
- Show that  $\mathbf{c}$  is perpendicular to  $\mathbf{a}$ .
- Show that  $\bar{\mathbf{a}}' \mathbf{c} = \bar{\mathbf{a}}' \mathbf{b}$  (i.e.,  $\mathbf{a} \times \mathbf{c} = \mathbf{a} \times \mathbf{b}$ ).

2.13 Try the following:

- Construct a  $3 \times 3$  diagonal matrix where every diagonal element is equal to "5."
- Try the statement `diag([5 5 5])`.
- Try the statement `5*eye(3)`.

2.14 Arrays  $\Phi$  and  $\mathbf{q}$  are defined as follows:

$$\Phi = \begin{Bmatrix} 2x - 3xy + y^2 - xz + yz^2 - 4xyz \\ -x^2 + xy^2 - 2y + 5yz - xz^2 \end{Bmatrix}, \mathbf{q} = \begin{Bmatrix} x & y & z \end{Bmatrix}^T$$

where  $x$ ,  $y$ , and  $z$  are functions of time.

- Construct  $\frac{\partial \Phi}{\partial \mathbf{q}}$  by taking the partial derivative of the function.
- Determine  $\dot{\Phi}$  by taking the time derivative of the function.
- Show that  $\dot{\Phi} = \frac{\partial \Phi}{\partial \mathbf{q}} \dot{\mathbf{q}}$ .

2.15 If vector  $\vec{a}$  has a constant length, describe the components of  $\dot{\vec{a}}$  in terms of the components of  $\vec{a}$ . Then, show that  $\bar{\mathbf{a}}' \mathbf{a} = 0$ .

2.16 Vectors  $\vec{a}$  and  $\vec{b}$  are defined as  $\mathbf{a} = \mathbf{A}_1 \boldsymbol{\alpha}$  and  $\mathbf{b} = \mathbf{A}_2 \boldsymbol{\beta}$ , where

$$\mathbf{A}_i = \begin{bmatrix} \cos \phi_i & -\sin \phi_i \\ \sin \phi_i & \cos \phi_i \end{bmatrix} \quad i = 1, 2 \quad \text{and} \quad \boldsymbol{\alpha} = \begin{Bmatrix} 1.2 \\ -0.5 \end{Bmatrix}, \boldsymbol{\beta} = \begin{Bmatrix} -0.3 \\ 0.8 \end{Bmatrix}$$

Vector  $\vec{d}$  is defined as  $\mathbf{d} = \begin{Bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{Bmatrix}$ . The variables  $x_1, y_1, \phi_1, x_2, y_2$ , and  $\phi_2$  are functions of time. At a given time, the variables and their first time derivatives have the following values:

$$x_1 = 6.2, y_1 = 1.0, \phi_1 = \pi/6, x_2 = -1.9, y_2 = 2.3, \phi_2 = \pi/4$$

$$\dot{x}_1 = 0.3, \dot{y}_1 = -0.2, \dot{\phi}_1 = 1.2, \dot{x}_2 = -0.6, \dot{y}_2 = 0.8, \dot{\phi}_2 = -0.5$$

Use Eq. (2.52) to evaluate  $\dot{\Phi}$  for the following cases:

- $\Phi = \mathbf{a}' \mathbf{b}$
- $\Phi = \bar{\mathbf{a}}' \mathbf{d}$

2.17 For the vectors and the data given in Problem 2.16, an array  $\mathbf{q}$  is defined as

$$\mathbf{q} = \begin{Bmatrix} x_1 & y_1 & \phi_1 & x_2 & y_2 & \phi_2 \end{Bmatrix}'$$

Use Eq. (2.65) to evaluate  $\partial\Phi/\partial\mathbf{q}$  for the following cases:

- a.  $\Phi = \mathbf{a}'\mathbf{b}$
- b.  $\Phi = \bar{\mathbf{a}}'\mathbf{d}$

2.18 For vector  $\mathbf{a} = \{1.2 \quad -0.4\}',$  determine its angle by drawing the vector on paper and measuring its angle,  $\phi_a.$  Reverse the direction of the vector and measure its angle,  $\phi_{-a}.$  Use MATLAB functions asin, acos, atan, and atan2 to calculate the following angles:

- a. Calculate the angle  $\phi_a$
- b. Calculate the angle  $\phi_{-a}$
- c. Use the components of vector  $\mathbf{a}$  in the function atan2.

Which command gives you the correct answer every time?

2.19 Determine the angle between vectors  $\vec{a}$  and  $\vec{b},$  described positively from  $\vec{a}$  to  $\vec{b}.$  Consider three cases:

Case 1:  $\mathbf{a} = \{2.0 \quad 1.0\}', \mathbf{b} = \{2.0 \quad 3.0\}'$

Case 2:  $\mathbf{a} = \{-2.0 \quad 1.0\}', \mathbf{b} = \{2.0 \quad 3.0\}'$

Case 3:  $\mathbf{a} = \{-2.0 \quad 1.0\}', \mathbf{b} = \{2.0 \quad -3.0\}'$

For each case, obtain the angle with two different approaches: functions asin, acos, atan, and atan2 to calculate the following angles:

- a. Use atan2 to calculate the angles  $\phi_a$  and  $\phi_b,$  then determine  $\phi_{ba} = \phi_b - \phi_a.$
- b. Calculate the scalar and vector products of the two vectors by using Eqs. (2.8) and (2.15). Then use atan2 to calculate  $\phi_{ba}.$
- c. Use the components of vector  $\mathbf{a}$  in the function atan2.

Which command gives you the correct answer every time?

# 3

---

## Fundamentals of Planar Kinematics

---

This chapter presents a summary of some fundamental concepts of *planar* kinematics. We first review the kinematics of a single particle, and then we consider the fundamental formulas for the kinematics of a single planar rigid body. Definitions for arrays of coordinates, constraint equations, degrees of freedom, and kinematic joints are also discussed.

The kinematic definitions and fundamental formulas for kinematics of particles and rigid bodies that are discussed in this chapter will be used extensively throughout this textbook. Therefore, it is important to understand these simple but fundamental formulas at this point. Exercises with MATLAB® are provided to carry out most of the computations, even though most of the examples are very simple and the calculations may be performed on paper. It is our intent to use MATLAB as much as possible and be prepared to solve more complex and realistic problems whenever needed.

---

### 3.1 A Particle

The simplest body arising in the study of motion is a *particle*, or a *point mass*. Analysis of the behavior of a point mass can lead to the kinematic and dynamic analysis of a body, and eventually to the analysis of a system of bodies. Therefore, it is essential to study the kinematics of particles first. In this textbook, we use the terms *particles* and *points* interchangeably.

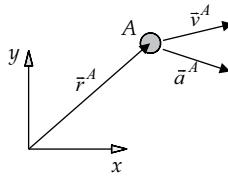
#### 3.1.1 Kinematics of a Particle

The most fundamental step in planar kinematics is to describe the position of a particle or a point in a plane. A plane is described by a nonmoving Cartesian reference frame  $x-y$ . The position of a typical particle  $A$  in the plane is described by vector  $\vec{r}^A$  as shown in Figure 3.1. The components of the position vector  $\vec{r}^A$ , or the  $x-y$  coordinates of particle  $A$ , represent the algebraic vector of coordinates for this particle. This vector and its components can be described in any of the following forms:

$$\mathbf{r}^A = \begin{Bmatrix} r_{(x)}^A \\ r_{(y)}^A \end{Bmatrix} = \begin{Bmatrix} x^A \\ y^A \end{Bmatrix} \quad (3.1)$$

The velocity of particle  $A$  is described by the time derivative of its position vector or of its coordinates as

$$\mathbf{v}^A = \dot{\mathbf{r}}^A = \begin{Bmatrix} \dot{x}^A \\ \dot{y}^A \end{Bmatrix} \quad (3.2)$$

**FIGURE 3.1**

Position, velocity, and acceleration vectors for a particle.

The acceleration of  $A$  is described by the second time derivative of the position vector, or the first time derivative of the velocity vector, as follows:

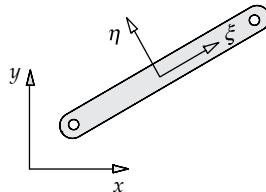
$$\mathbf{a}^A = \dot{\mathbf{v}}^A = \ddot{\mathbf{r}}^A = \begin{Bmatrix} \ddot{x}^A \\ \ddot{y}^A \end{Bmatrix} \quad (3.3)$$

Typical velocity and acceleration vectors,  $\bar{v}^A$  and  $\bar{a}^A$ , for particle  $A$  are shown in Figure 3.1.

### 3.2 Kinematics of a Rigid Body

A *rigid body* is defined as a system of particles for which distances between particles remain unchanged. If any of these particles is positioned by a vector fixed to the body, the vector never changes its position relative to the body, even when the body is in motion. In reality, all solid materials change shape to some extent when forces are applied to them. Nevertheless, if movement associated with the changes in shape is small compared with the overall movement of the body, then the concept of rigidity is applicable. For example, displacements due to elastic vibration of the connecting rod of an engine may be of no consequence in the description of engine dynamics as a whole, so in this case the rigid body assumption is clearly in order. On the other hand, if the problem is one of describing stress in the connecting rod due to vibration, then the deformation of the rod becomes of prime importance and cannot be neglected. In this text, all analyses are based on the rigid body assumption unless specified otherwise. Therefore, the term *body* refers to a *rigid body*.

To study the kinematics of a body, we need to define a nonmoving reference frame. In this text, we denote the *nonmoving* frame as  $x-y$ . This is also called a *global* or an *inertial* frame. We reserve  $\xi-\eta$  to denote a *moving* frame, also called a *local* or a *body-fixed* frame. Such two frames are shown in Figure 3.2. Although in a system we can only have one  $x-y$  frame, we can define as many  $\xi-\eta$  frames as the number of moving bodies in the system.

**FIGURE 3.2**

A nonmoving and a moving reference frames.

A free body in a plane can move along the  $x$  and  $y$  axes (*translational motion*), and it can also rotate about an axis perpendicular to the plane (*rotational motion*). Defining a set of coordinates to monitor the position and orientation of the body is essential in the kinematic analysis. The first and second time derivatives of these coordinates are used to define the velocity and acceleration of the body in the plane.

### 3.2.1 Coordinates of a Body

To specify the configuration of a planar body, we need to define three coordinates: position along the  $x$ -axis, position along the  $y$ -axis, and rotational orientation about an axis perpendicular to the  $x$ - $y$  plane. As shown in Figure 3.3a, we place a reference point  $O$  on the body. The position vector  $\bar{r}$  is used to locate point  $O$  from the origin of the global reference frame  $x$ - $y$ . The  $x$  and  $y$  components of this vector are used as the *translational coordinates* of the body:

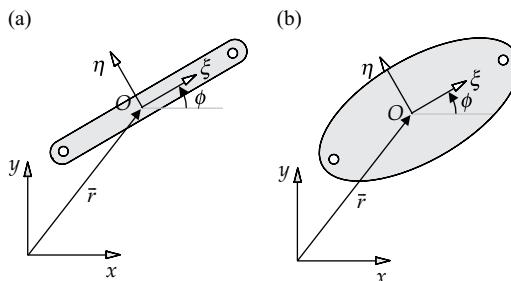
$$\mathbf{r} = \begin{Bmatrix} r_{(x)} \\ r_{(y)} \end{Bmatrix} = \begin{Bmatrix} x \\ y \end{Bmatrix}$$

The angle of *any* vector attached to the body can be used as the rotational coordinate of that body. However, we attach a body-fixed  $\xi$ - $\eta$  frame to the body with its origin at point  $O$ . The angle that the  $\xi$ -axis makes with the  $x$ -axis, measured in the positive sense (counterclockwise), is used as the *rotational coordinate* of the body and is denoted by  $\phi$ . The *array of coordinates* for the body is defined as

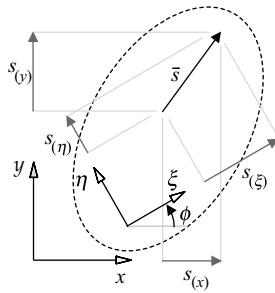
$$\mathbf{c} = \begin{Bmatrix} x \\ y \\ \phi \end{Bmatrix} = \begin{Bmatrix} x & y & \phi \end{Bmatrix}'$$

The position and orientation of a rigid body is completely described by array  $\mathbf{c}$ . The shape of a rigid body is not of any importance in most of the discussions we have in this textbook. Therefore, in some of the figures that deal with general discussions of bodies, the shape is represented as an oval, instead of the actual shape, as depicted in Figure 3.3b.

Assume that vector  $\bar{s}$  shown in Figure 3.4 is defined on a body. Although this vector rotates and translates with the body, it has a *fixed* length. The projection of this vector onto the  $\xi$ - $\eta$  frame results in two components represented as



**FIGURE 3.3**  
Coordinates of a body.

**FIGURE 3.4**

Components of a vector attached to a body.

$$\mathbf{s}_{(\xi-\eta)} = \mathbf{\bar{s}} = \begin{Bmatrix} s_{(\xi)} \\ s_{(\eta)} \end{Bmatrix}$$

These components will be referred to as the *local* or *body-fixed* components. The same vector can also be projected onto the global  $x$ - $y$  axes to yield

$$\mathbf{s}_{(x-y)} = \mathbf{s} = \begin{Bmatrix} s_{(x)} \\ s_{(y)} \end{Bmatrix}$$

For notational simplicity, when an algebraic vector contains body-fixed components, instead of showing the subscript  $(\xi-\eta)$ , we cross the vector with a line, such as  $\mathbf{\bar{s}}$ —we will read it as  $s$ -local.\* When a vector contains the  $x$ - $y$  components, we represent the vector without showing the subscript  $(x-y)$ .

Because  $\mathbf{s}$  and  $\mathbf{\bar{s}}$  represent the components of the same vector in two different reference frames, their components are related as (shown in both compact and expanded forms)

$$\begin{aligned} \mathbf{s} &= \mathbf{A}\mathbf{\bar{s}} & s_{(x)} &= s_{(\xi)} \cos\phi - s_{(\eta)} \sin\phi \\ & & s_{(y)} &= s_{(\xi)} \sin\phi + s_{(\eta)} \cos\phi \end{aligned} \tag{3.4}$$

where  $\mathbf{A}$  is a  $2 \times 2$  matrix as

$$\mathbf{A} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{3.5}$$

The columns of  $\mathbf{A}$  are unit vectors along the  $\xi$  and  $\eta$  axes expressed in the  $x$ - $y$  frame. Matrix  $\mathbf{A}$  is called the *rotational transformation matrix*, which is an *orthonormal* matrix, that is, its transpose is equal to its inverse. Because  $\mathbf{A}^{-1} = \mathbf{A}'$ , the inverse transformation of Eq. (3.4) is found easily as

---

\* In some literature and textbooks, the body-fixed components of a vector are denoted by an apostrophe or a prime, such as  $\mathbf{s}'$ . In this textbook, we have reserved the apostrophe for vector and matrix transpose to be consistent with the MATLAB notation.

$$\begin{aligned} \mathbf{s}' = \mathbf{A}' \mathbf{s} \quad & S_{(\xi)} = s_{(x)} \cos \phi + s_{(y)} \sin \phi \\ & S_{(\eta)} = -s_{(x)} \sin \phi + s_{(y)} \cos \phi \end{aligned} \quad (3.6)$$

**Proof 3.1**

To prove that the elements of matrix  $\mathbf{A}$  are those stated in Eq. (3.5), we assume that vector  $\vec{s}$  makes an angle  $\theta$  with the  $\xi$ -axis, and the  $\xi$ -axis makes an angle  $\phi$  with the  $x$ -axis. Therefore, the global components of  $\vec{s}$  can be expressed as (noting  $s_{(\xi)} = s \cos \theta, s_{(\eta)} = s \sin \theta$ )

$$s_{(x)} = s \cos(\phi + \theta) = s(\cos \phi \cos \theta - \sin \phi \sin \theta) = \cos \phi s_{(\xi)} - \sin \phi s_{(\eta)}$$

$$s_{(y)} = s \sin(\phi + \theta) = s(\sin \phi \cos \theta + \cos \phi \sin \theta) = \sin \phi s_{(\xi)} + \cos \phi s_{(\eta)}$$

$$\left\{ \begin{array}{c} s_{(x)} \\ s_{(y)} \end{array} \right\} = \left[ \begin{array}{cc} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{array} \right] \left\{ \begin{array}{c} s_{(\xi)} \\ s_{(\eta)} \end{array} \right\}$$

The following function M-file constructs the rotational transformation matrix  $\mathbf{A}$ . This function M-file will be used in many of the upcoming programs in this and other chapters.

```
function A = A_matrix(phi)
    cp = cos(phi); sp = sin(phi); A = [cp -sp; sp cp];
    theta = pi/6; s_local = [2; 3];
    A = A_matrix(theta) .....
    s = A*s_local .....
```

A =	0.8660	-0.5000
	0.5000	0.8660
s =	0.2321	
	3.5981	

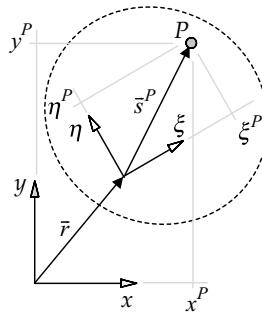
The magnitude of a vector that is defined on a body can be obtained from either its local or global components:

$$s^2 = \mathbf{s}' \mathbf{s} = \mathbf{s}' \mathbf{s}$$
 (3.7)

Vectors are also used to position body-attached points with respect to the origin of the body-fixed frame. As shown in Figure 3.5, vector  $\vec{s}^P$  is defined on a body to locate point  $P$  from its origin. The components of  $\vec{s}^P$  in the  $\xi-\eta$  and  $x-y$  frames are denoted as

$$\mathbf{s}'^P = \left\{ \begin{array}{c} s_{(\xi)}^P \\ s_{(\eta)}^P \end{array} \right\} = \left\{ \begin{array}{c} \xi^P \\ \eta^P \end{array} \right\}, \mathbf{s}^P = \left\{ \begin{array}{c} s_{(x)}^P \\ s_{(y)}^P \end{array} \right\}$$

The  $\xi-\eta$  and  $x-y$  components are related through the transformation matrix  $\mathbf{A}$  as



**FIGURE 3.5**  
A point attached to a body.

$$\mathbf{s}^P = \mathbf{A}\mathbf{s}'^P \quad (3.8)$$

The  $x$ - $y$  coordinates of point  $P$  are found from the vector relation

$$\mathbf{r}^P = \mathbf{r} + \mathbf{s}^P \quad (3.9)$$

or

$$\mathbf{r}^P = \mathbf{r} + \mathbf{A}\mathbf{s}'^P \quad \left\{ \begin{array}{c} x^P \\ y^P \end{array} \right\} = \left\{ \begin{array}{c} x \\ y \end{array} \right\} + \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \left\{ \begin{array}{c} \xi^P \\ \eta^P \end{array} \right\} \quad (3.10)$$

Equations (3.4) through (3.10) form some of the fundamental formulas in planar rigid body kinematics.

```
function r_P = r_Point(r, s_P)
    r_P = r + s_P;
```

### Example 3.1

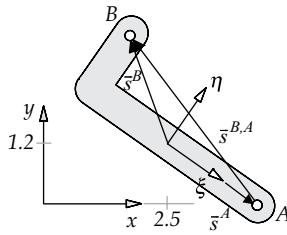
A body-fixed frame is defined on a link of a mechanism as shown in Figure 3.6. In a nonmoving frame, the link has translational and rotational coordinates:

$$\mathbf{r} = \begin{Bmatrix} 2.5 \\ 1.2 \end{Bmatrix} \text{ m}, \phi = 5.6723 \text{ rad (325 degrees)}$$

Points  $A$  and  $B$  on the link have local coordinates:

$$\mathbf{s}^A = \begin{Bmatrix} 2.18 \\ 0 \end{Bmatrix} \text{ m}, \mathbf{s}^B = \begin{Bmatrix} -1.8 \\ 1.3 \end{Bmatrix} \text{ m}$$

Find: (a) the global components of vectors  $\vec{s}^A$  and  $\vec{s}^B$ , (b) the global coordinates of points  $A$  and  $B$ , and (c) the global components of vector  $\vec{s}^{B,A} = \vec{s}^B - \vec{s}^A$ .

**FIGURE 3.6**

Points A and B are defined on a link.

**MATLAB/Chapter 3/Example \_ 3 \_ 1**

```
r = [2.5; 1.2]; phi = 5.6723;
s_A_local = [2.18; 0];
s_B_local = [-1.8; 1.3];
A = A_matrix(phi);
s_A = A*s_A_local .....  

s_B = A*s_B_local .....  

r_A = r + s_A % not using function r_Point .  

r_B = r_Point(r, s_B) % using func r_Point .  

% s_BA can be computed in different ways
s_BA_1 = A*(s_B_local - s_A_local) .....  

s_BA_2 = s_B - s_A .....  

s_BA_3 = r_B - r_A .....
```

```
s_A =
1.7857
-1.2504
s_B =
-0.7288
2.0973
r_A =
4.2857
-0.0504
r_B =
1.7712
3.2973
s_BA_1 =
-2.5145
3.3478
s_BA_2 =
-2.5145
3.3478
s_BA_3 =
-2.5145
3.3478
```

**3.2.2 Velocity of a Body**

The translational and rotational velocities of a body are described by the time derivative of the coordinates as  $\dot{\mathbf{r}} = \begin{Bmatrix} \dot{x} & \dot{y} \end{Bmatrix}'$  and  $\dot{\phi}_i$ , respectively. The *three-array of velocities* for the body is defined as  $\dot{\mathbf{c}} = \begin{Bmatrix} \dot{x} & \dot{y} & \dot{\phi} \end{Bmatrix}'$ .

The velocity equation for a vector fixed to the body is obtained from the time derivative of Eq. (3.4):

$$\dot{\mathbf{s}} = \dot{\mathbf{A}}\mathbf{s} \quad \begin{aligned} \dot{s}_{(x)} &= -\left(s_{(\xi)} \sin \phi + s_{(\eta)} \cos \phi\right) \dot{\phi} \\ \dot{s}_{(y)} &= \left(s_{(\xi)} \cos \phi - s_{(\eta)} \sin \phi\right) \dot{\phi} \end{aligned} \quad (3.11)$$

Note that  $\dot{\mathbf{s}} = \mathbf{0}$  since  $\mathbf{s}$  contains constant components. Equation (3.11) can also be written as

$$\begin{Bmatrix} \dot{s}_{(x)} \\ \dot{s}_{(y)} \end{Bmatrix} = \begin{bmatrix} -\sin\phi & -\cos\phi \\ \cos\phi & -\sin\phi \end{bmatrix} \begin{Bmatrix} s_{(\xi)} \\ s_{(\eta)} \end{Bmatrix} \dot{\phi} = \begin{Bmatrix} -s_{(y)} \\ s_{(x)} \end{Bmatrix} \dot{\phi} \Rightarrow \dot{\mathbf{s}} = \check{\mathbf{s}}\dot{\phi} \quad (3.12)$$

where the definition for vector  $\check{\mathbf{s}}$  is given in Eq. (2.12). Figure 3.7a illustrates how vector  $\dot{\check{\mathbf{s}}}$  is represented on an axis perpendicular to vector  $\check{\mathbf{s}}$ . In this figure, it is assumed that  $\dot{\phi}$  is positive (counterclockwise); otherwise,  $\dot{\check{\mathbf{s}}}$  would be in the opposite direction.

The first time derivative of Eq. (3.10) yields the velocity equations for point  $P$  as

$$\begin{aligned} \dot{\mathbf{r}}^P &= \dot{\mathbf{r}} + \check{\mathbf{s}}^P \dot{\phi} & \dot{x}^P &= \dot{x} - s_{(y)}^P \dot{\phi} \\ & & \dot{y}^P &= \dot{y} + s_{(x)}^P \dot{\phi} \end{aligned} \quad (3.13)$$

or

$$\begin{aligned} \dot{x}^P &= \dot{x} - (\xi^P \sin\phi + \eta^P \cos\phi) \dot{\phi} \\ \dot{y}^P &= \dot{y} + (\xi^P \cos\phi - \eta^P \sin\phi) \dot{\phi} \end{aligned} \quad (3.14)$$

```
function r_P_d = r_Point_d(r_d, s_P, phi_d)
r_P_d = r_d + s_rot(s_P)*phi_d;
```

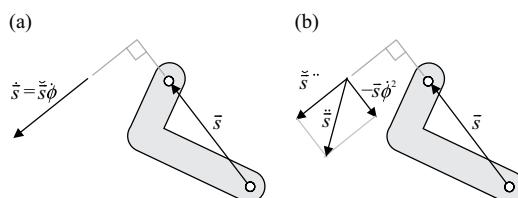
### 3.2.3 Acceleration of a Body

The translational and rotational accelerations of a body are described by the second time derivative of its coordinates as  $\ddot{\mathbf{r}} = \begin{Bmatrix} \ddot{x} & \ddot{y} \end{Bmatrix}'$  and  $\ddot{\phi}$ , respectively. Hence, the *three-array of accelerations* for the body is defined as  $\ddot{\mathbf{c}} = \begin{Bmatrix} \ddot{x} & \ddot{y} & \ddot{\phi} \end{Bmatrix}'$ .

The acceleration of a vector fixed to a body is obtained from the time derivative of Eq. (3.11) as

$$\begin{aligned} \ddot{\mathbf{s}} &= \ddot{\mathbf{A}}\mathbf{s} & \ddot{s}_{(x)} &= -\left(s_{(\xi)} \sin\phi + s_{(\eta)} \cos\phi\right) \ddot{\phi} - \left(s_{(\xi)} \cos\phi - s_{(\eta)} \sin\phi\right) \dot{\phi}^2 \\ & & \ddot{s}_{(y)} &= \left(s_{(\xi)} \cos\phi - s_{(\eta)} \sin\phi\right) \ddot{\phi} - \left(s_{(\xi)} \sin\phi + s_{(\eta)} \cos\phi\right) \dot{\phi}^2 \end{aligned} \quad (3.15)$$

or



**FIGURE 3.7**

(a) Velocity and (b) acceleration vectors corresponding to vector  $\mathbf{s}$  that has a constant magnitude.

$$\ddot{\mathbf{s}} = \ddot{\mathbf{s}}\ddot{\phi} - \mathbf{s}\dot{\phi}^2 \quad (3.16)$$

The two components of the acceleration vector are illustrated in Figure 3.7b where  $\ddot{\mathbf{s}}\ddot{\phi}$  is known as the *tangential* component and  $-\mathbf{s}\dot{\phi}^2$  as the *normal* component. In this illustration, it is assumed that  $\dot{\phi}$  is positive; otherwise, the tangential component would be in the opposite of the direction shown. Note that the normal component is always in the opposite direction of  $\ddot{\mathbf{s}}$  since  $\dot{\phi}^2$  is always positive.

The time derivative of Eq. (3.13) yields the acceleration equations for point  $P$  as

$$\begin{aligned}\ddot{\mathbf{r}}^P &= \ddot{\mathbf{r}} + \ddot{\mathbf{s}}^P\ddot{\phi} - \mathbf{s}^P\dot{\phi}^2 & \ddot{x}^P &= \ddot{x} - s_{(y)}^P\ddot{\phi} - s_{(x)}^P\dot{\phi}^2 \\ \ddot{y}^P &= \ddot{y} + s_{(x)}^P\ddot{\phi} - s_{(y)}^P\dot{\phi}^2\end{aligned} \quad (3.17)$$

or

$$\begin{aligned}\ddot{x}^P &= \ddot{x} - (\xi^P \sin \phi + \eta^P \cos \phi)\ddot{\phi} - (\xi^P \cos \phi - \eta^P \sin \phi)\dot{\phi}^2 \\ \ddot{y}^P &= \ddot{y} + (\xi^P \cos \phi - \eta^P \sin \phi)\ddot{\phi} - (\xi^P \sin \phi + \eta^P \cos \phi)\dot{\phi}^2\end{aligned} \quad (3.18)$$

```
function r_P_dd = r_Point_dd(r_dd, s_P, phi_d, phi_dd)
    r_P_dd = r_dd + s_rot(s_P)*phi_dd - s_P*phi_d^2;
```

### Example 3.2

As a continuation of Example 3.1, assume that the link has the following velocity and acceleration:

$$\dot{\mathbf{r}} = \begin{Bmatrix} 1 \\ -2 \end{Bmatrix} \text{ m/s}, \dot{\phi} = 1.0 \text{ rad/s}, \ddot{\mathbf{r}} = \begin{Bmatrix} -0.4 \\ 0.4 \end{Bmatrix} \text{ m/s}^2, \ddot{\phi} = 4.65 \text{ rad/s}^2$$

Compute the coordinates, velocity, and acceleration of points  $A$  and  $B$ .

#### MATLAB/Chapter 3/Example \_ 3 \_ 2

```
r = [2.5; 1.2]; phi = 5.6723;
s_A_local = [2.18; 0]; s_B_local = [-1.8; 1.3];
r_d = [1; -2]; phi_d = 1;
r_dd = [-0.4; 0.4]; phi_dd = 4.65;
A = A_matrix(phi);
s_A = A*s_A_local; s_B = A*s_B_local;
r_A_d = r_Point_d(r_d, s_A, phi_d) .....
r_B_d = r_Point_d(r_d, s_B, phi_d) .....
r_A_dd = r_Point_dd(r_dd, s_A, phi_d, phi_dd) .
r_B_dd = r_Point_dd(r_dd, s_B, phi_d, phi_dd) .
```

```
r_A_d =
2.2504
-0.2143
r_B_d =
-1.0973
-2.7288
r_A_dd =
3.6288
9.9541
r_B_dd =
-9.4239
-5.0862
```

### 3.3 Definitions

In a *planar* multibody system, all of the bodies move in a plane or in parallel planes. If some links undergo motion in three-dimensional space, it is called a *spatial* multibody system. The individual bodies that collectively form a multibody system may also be called *links*.

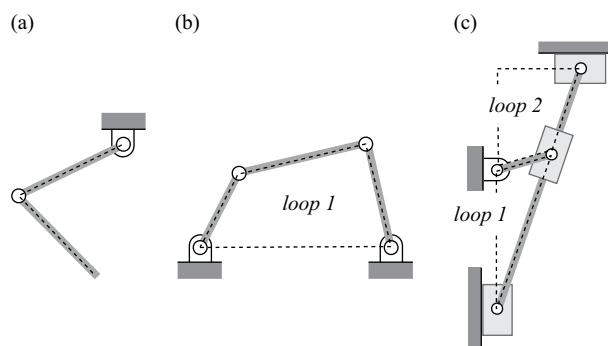
The combination of two links in contact constitutes a *kinematic pair*, or *joint*. An assemblage of interconnected links is called a *kinematic chain*. A system that is formed from a collection of links or bodies that are kinematically connected to one another, but for which it is not possible to move to successive links across kinematic joints and return to the starting link, is called an *open-chain* system. An example is the double pendulum shown in Figure 3.8a. A *closed-chain* system is formed where each link is connected to at least two other links of the system, and it is possible to transverse a *loop*. Figure 3.8b shows a four-bar linkage that is a closed-chain system.

A closed-chain system may contain one or more loops in its kinematic structure. If the number of loops in a closed-chain mechanism is one, then the mechanism is called a *single-loop* system; otherwise, it is called a *multi-loop* system. Figure 3.8b is an example of a single-loop system, and Figure 3.8c shows a double-loop system.

A mechanism is formed when at least one of the links of a closed-chain system is held fixed and its other links are allowed to move. The fixed link is called the *ground*, *frame*, or *chassis*. In this text, the terms *links* and *bodies* are used interchangeably. In general, a *mechanism* is a set of rigid elements that are arranged to produce a specified motion. This definition of mechanism includes classical linkages, as well as interconnected bodies that make up an aircraft landing gear, an engine, and many other mechanical systems.

#### 3.3.1 Array of Coordinates

Any set of parameters that can specify the position (configuration) of all the bodies in a system is called a set of *coordinates*. For systems in motion, these parameters vary with time; therefore, their first and second time derivatives could specify the *velocity* and *acceleration* of the bodies. Depending on the formulation used to perform an analysis, the defined set of coordinates could be different. For example, for the four-bar mechanism shown in Figure 3.9a, we can define the Cartesian coordinates of points A and B as a set of position coordinates. For this set, an *array of coordinates* is defined as



**FIGURE 3.8**

(a) An open-chain system, (b) a single-loop closed-chain system, and (c) a multi-loop closed-chain system.

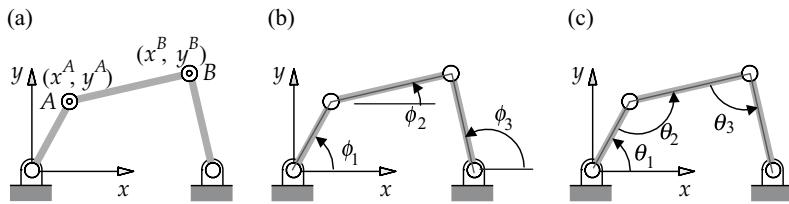


FIGURE 3.9

Different sets of coordinates for a four-bar mechanism.

$$\mathbf{q} = \begin{Bmatrix} x^A & y^A & x^B & y^B \end{Bmatrix}' \quad (3.19)$$

We can also perform kinematic analysis on the same mechanism by defining three angles as the position parameters, as shown in Figure 3.9b, forming an array of coordinates as

$$\mathbf{q} = \begin{Bmatrix} \phi_1 & \phi_2 & \phi_3 \end{Bmatrix}' \quad (3.20)$$

If the position parameters are measured with respect to the global reference frame, they are called *absolute* coordinates, such as the coordinates used in Eqs. (3.19) and (3.20). If the position parameters are defined between two moving bodies, they are called *relative* coordinates. Such relative coordinates shown in Figure 3.9c that yield an array of coordinates as

$$\mathbf{q} = \begin{Bmatrix} \theta_1 & \theta_2 & \theta_3 \end{Bmatrix}' \quad (3.21)$$

An array of coordinates may contain a mix of absolute and relative coordinates. For example, for the four-bar mechanism, the array of coordinates may be defined as

$$\mathbf{q} = \begin{Bmatrix} x^A & y^A & \theta_3 \end{Bmatrix}' \quad (3.22)$$

In the upcoming chapters, different formulations are presented that use different sets of coordinates. Therefore, the array of coordinates for one formulation may be denoted differently from the array of coordinates for another formulation. In general, when we are not emphasizing any particular formulation, the array of coordinates is denoted as  $\mathbf{q}$ .

The total number of coordinates that are defined for a system is denoted by  $n_v$  (number of variables). Since the first and second time derivatives of the coordinates are used to denote *velocities* and *accelerations*,  $n_v$  also represents the number of velocities and the number of accelerations for the system. For example, the arrays of velocities and accelerations for the coordinates described by Eq. (3.22) are expressed as

$$\dot{\mathbf{q}} = \begin{Bmatrix} \dot{x}^A & \dot{y}^A & \dot{\theta}_3 \end{Bmatrix}' \quad (3.23)$$

$$\ddot{\mathbf{q}} = \left\{ \begin{array}{ccc} \dot{x}^A & \dot{y}^A & \ddot{\phi}_3 \end{array} \right\}' \quad (3.24)$$

### 3.3.2 Degrees of Freedom

The minimum number of coordinates required to fully describe the configuration of a multibody system is called the number of *degrees of freedom* (DoFs). A free point or particle on a plane has two DoFs—it can move along any two perpendicular axes independently. A free body on the plane has three DoFs—it can translate along any two perpendicular axes, and it can also rotate about an axis perpendicular to the plane.

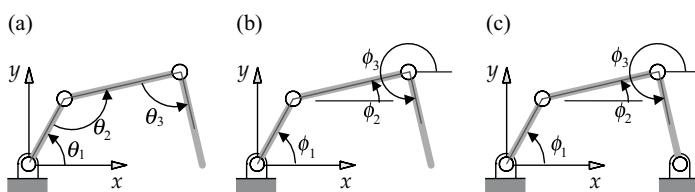
In order to determine the number of DoFs of a system, we must consider all the bodies and the joints that make the system. For example, consider the triple pendulum shown in Figure 3.10a, where no fewer than three angles,  $\theta_1$ ,  $\theta_2$ , and  $\theta_3$ , can uniquely determine its configuration. The value of any of the three angles can be varied independent of the other two. Therefore, the triple pendulum has three DoFs. For the same pendulum, as shown in Figure 3.10b, three independent angles can be defined with respect to the  $x$ -axis, denoted as  $\phi_1$ ,  $\phi_2$ , and  $\phi_3$ . We note that the number of DoFs of this pendulum, or any other multi-body system, does not depend on how we define the coordinates.

For the four-bar mechanism shown in Figure 3.10c, three angles,  $\phi_1$ ,  $\phi_2$ , and  $\phi_3$ , define the configuration of the system. However, these angles are not independent; there exist two algebraic equations relating the angles:

$$\begin{aligned} L_1 \cos \phi_1 + L_2 \cos \phi_2 + L_3 \cos \phi_3 - L_0 &= 0 \\ L_1 \sin \phi_1 + L_2 \sin \phi_2 + L_3 \sin \phi_3 &= 0 \end{aligned} \quad (3.25)$$

where  $L_1$ ,  $L_2$ , and  $L_3$  are the link lengths, and  $L_0$  is the distance between the two ground pin joints. The two equations can be solved for two of the angles as a function of the third; for example,  $\phi_2$  and  $\phi_3$  can be expressed as a function of  $\phi_1$ . Therefore,  $\phi_1$  is the only variable needed to describe the configuration of the system, and so there is only one DoF for the four-bar mechanism.

In a mechanical system, if  $n_{dof}$  is the number of DoFs, then at least  $n_v = n_{dof}$  coordinates are required to completely describe the configuration of the system. If the number of defined coordinates is greater than the number of DoFs of the system (that is,  $n_v > n_{dof}$ ), then any subset of these coordinates that are independent and are equal to the number of DoFs is called a set of *independent coordinates*. The remaining coordinates, which may be determined as a function of the independent coordinates, are called *dependent coordinates*. In the triple pendulum example, all the three coordinates are independent, whereas in the



**FIGURE 3.10**

(a) A triple pendulum with three DoFs and (b) a four-bar mechanism with one DoF.

four-bar linkage example, one of the three coordinates is independent and the other two are dependent.

### 3.3.3 Constraint Equations

The kinematic relationships between the defined coordinates of a system are called *constraint equations*. The maximum number of independent constraint equations that can be written for a system is equal to the number of its dependent coordinates. For example, for the four-bar mechanism of Figure 3.10b, the two algebraic equations of Eq. (3.25) are the constraint equations. In contrast, for the triple pendulum of Figure 3.10a, there are no constraint equations because the three defined coordinates are independent.

A constraint equation describing a condition on the array of coordinates,  $\mathbf{q}$ , is expressed in generic form as

$$\Phi \equiv \Phi(\mathbf{q}) = 0 \quad (3.26)$$

The constraints on the coordinates are also referred to as *position constraints*. Assuming that there are  $n_c$  position constraints between the coordinates of a system, an array of constraints is expressed as

$$\Phi \equiv \Phi(\mathbf{q}) = 0 \quad (3.27)$$

Position constraints are most likely nonlinear algebraic equations.

The first time derivative of the position constraints yields the *velocity constraints*. Velocity constraints provide relationships between the velocity variables that are defined for a system. The time derivative of Eq. (3.27) can be written as

$$\dot{\Phi} \equiv \mathbf{D}\dot{\mathbf{q}} = 0 \quad (3.28)$$

where  $\mathbf{D}$  is called the *Jacobian matrix*.<sup>\*</sup> Velocity constraints are algebraic equations linear in velocities.

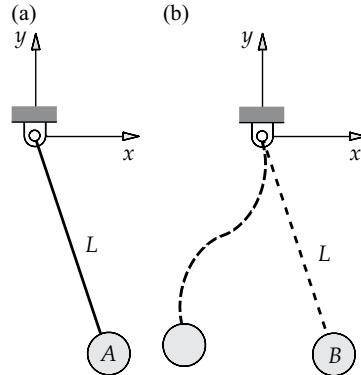
The time derivative of the velocity constraints yields the *acceleration constraints*. The time derivative of Eq. (3.28) is written as

$$\ddot{\Phi} \equiv \mathbf{D}\ddot{\mathbf{q}} + \dot{\mathbf{D}}\dot{\mathbf{q}} = 0 \quad (3.29)$$

Algebraic equality constraints in terms of the coordinates, and perhaps time,<sup>†</sup> are said to be *holonomic* constraints. In general, if constraint equations contain *inequalities* or relations between velocity components that are *not integrable* in closed form (meaning that one cannot find a position constraint whose time derivative is the given relation), they are said to be *non-holonomic* constraints. Consider, as an example, the pendulum shown in

<sup>\*</sup> Slightly different definitions of the Jacobian matrix may be found in literature. This term is most commonly referred to the matrix of partial derivative of the position constraints with respect to the array of coordinates that may or may not be the same as the coefficient matrix in the velocity constraints. This discrepancy arises when the velocities are not defined to be the time derivative of the coordinates.

<sup>†</sup> If time appears explicitly in a constraint, we will refer to it as a *driver* constraint. Driver constraints will be discussed in other chapters.



**FIGURE 3.11**  
Example of a system with (a) a holonomic or (b) a non-holonomic constraint.

Figure 3.11a. The rod that connects the mass  $A$  to the hinge on the ground is rigid. For the coordinates of  $A$ , we can write the holonomic constraint as follows:

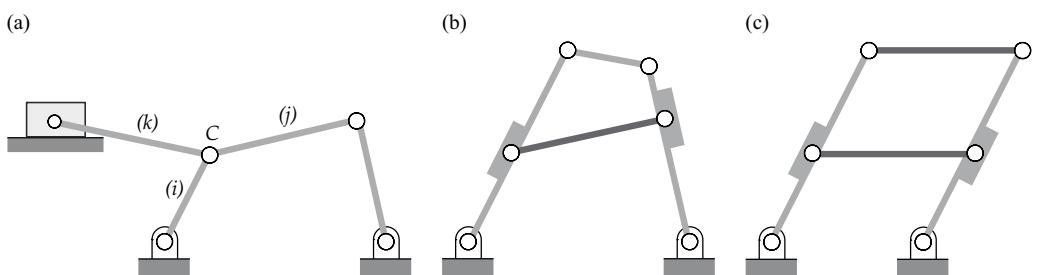
$$(x^A)^2 + (y^A)^2 - L^2 = 0$$

In the system of Figure 3.11b, the mass  $B$  is connected to the stationary hinge by a rope. For the coordinates of  $B$ , we can write the non-holonomic constraint as follows:

$$(x^B)^2 + (y^B)^2 - L^2 \leq 0$$

In this textbook, for brevity, the term *constraint* will refer to a holonomic constraint, unless specified otherwise.

Knowledge of the number of DoFs of a system can be useful when constraint equations are being formulated. Often, the pictorial description of a system can be misleading. Several joints may restrict the same DoFs, and may therefore be *equivalent* or *redundant*. As an example, consider the six-bar mechanism shown in Figure 3.12a where three links are connected to each other at point  $C$  by apparently one pin joint. However, due to the definition that a pin joint connects *two* bodies, we must consider two pin joints at  $C$ : one between links  $(i)$  and  $(j)$ , and the other between links  $(i)$  and  $(k)$ . Considering a third joint between



**FIGURE 3.12**  
(a) A six-bar mechanism with multiple pin joints at  $C$ , (b) a structure, and (c) a parallelogram six-bar mechanism.

links (*j*) and (*k*) would be redundant, and hence unnecessary. As another example, consider the system shown in Figure 3.12b. This system consists of four links, the ground, and six pin joints. This system is a structure with zero DoF, that is, none of the links is able to move. In contrast, the system shown in Figure 3.12c, which has the same number of links and joints as the system in Figure 3.12b, has one DoF. If this system is modeled for analysis by four moving bodies and six pin joints, the set of constraint equations will contain *redundant* equation(s). In general, for a system having  $n_c$  independent constraint equations and  $n_v$  coordinates, the number of DoFs is determined as

$$n_{dof} = n_v - n_c \quad (3.30)$$

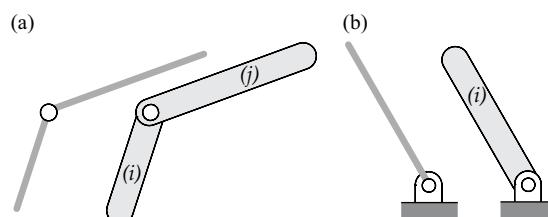
### 3.3.4 Kinematic Joints

Kinematic joints (or pairs) can be classified generally into two groups. If we consider joints with their three-dimensional geometry, the joints with surface contact are referred to as *lower pairs*, and those with line or point contact are referred to as *higher pairs*. The most commonly used joints in planar multibody systems are *revolute* (pin) joints and *translational* (sliding) joints, which are examples of lower pairs. *Point-followers* and *cam-followers* are examples of higher pair joints.

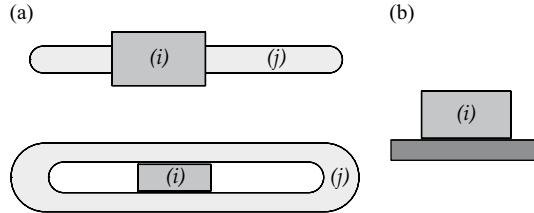
In planar representation, a revolute joint connects two bodies such that they have a common point. The schematic presentation of a revolute joint between two bodies is shown in Figure 3.13a. If one of the bodies is the ground, the connection will be depicted as shown in Figure 3.13b. A revolute joint eliminates *two* DoFs, that is, the joint allows only *one relative* DoF between the two bodies.

A translational joint connects two bodies in such a way that they translate relative to each other along a common axis. The axis of relative translational motion, or any axis parallel to that, is called the *line of translation*. A translational joint removes *two* DoF allowing only *one relative* DoF between the two bodies. This type of joint may appear in different forms. Two schematic representations of translational joints are shown in Figure 3.14a. If one of the bodies is the ground, the joint is depicted schematically as shown in Figure 3.14b.

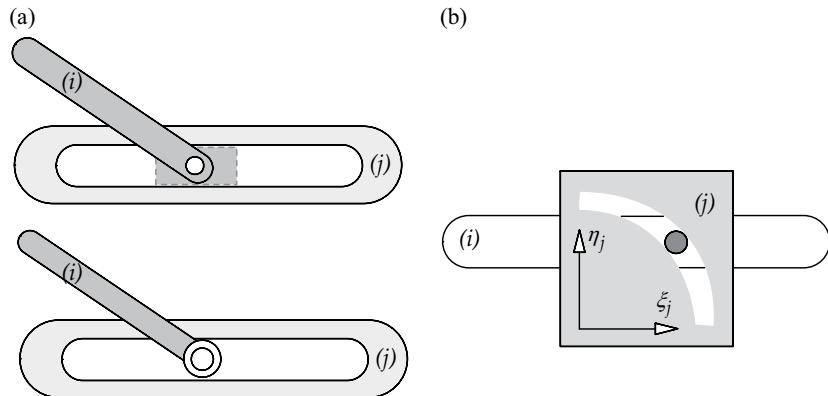
In planar systems, a joint that eliminates *two relative* DoFs is called a *full joint*, whereas a joint eliminating only *one relative* DoF is called a *half joint*. A pin-in-slot, which is a combination of a revolute and a translational joint, is one example of a half joint. This joint will be referred to as a *revolute-translational* joint, where two examples are shown in Figure 3.15a. Another similar joint is called a *point-follower* shown in Figure 3.15b. In this



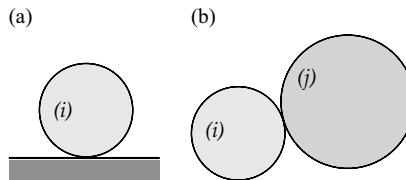
**FIGURE 3.13**  
Schematic representations of a revolute joint.



**FIGURE 3.14**  
Schematic representations of a translational joint.



**FIGURE 3.15**  
Examples of (a) revolute-translational joints and (b) a point-follower joint.



**FIGURE 3.16**  
Examples of (a) a disc in contact with the ground and (b) two discs in contact.

joint, the pin on one body slides in a nonlinear (curved) slot that is cut into the second body. If the slot is a straight line, then the point-follower joint becomes equivalent to a revolute-translational joint.

A circular disc in contact with a flat surface, such as the horizontal ground shown in Figure 3.16a, can be considered as a kinematic joint. The condition for the two bodies to remain in contact eliminates one relative DoF. If the disc is not allowed to slip as it rolls, that is, *no-slip* condition, then one more relative DoF is removed. In other words, contact and no-slip conditions eliminate two DOFs, whereas contact only (allowing slip)

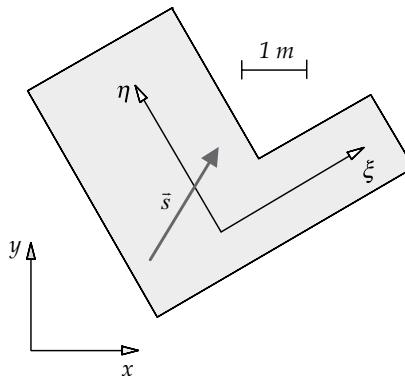
eliminates one DoF. Similar conditions can be defined between two rolling discs as shown in Figure 3.16b.

Kinematic joints or conditions represent constraints. Other types of kinematic constraints will be discussed in the examples and problems of the following chapters. Springs, dampers, and actuators are considered as force elements that are discussed in Chapter 4.

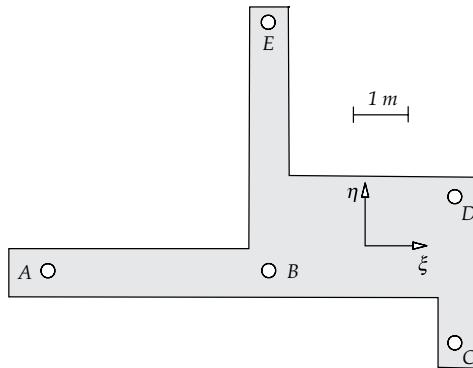
### 3.4 Problems

Solve problems 3.1–3.7 on paper—do not use MATLAB.

- 3.1 Vector  $\bar{s}$  is attached to a body as shown. Take direct measurements from the figure and determine:
- Components of the vector in the  $x$ - $y$  frame, i.e.,  $s$ .
  - Components of the vector in the  $\xi$ - $\eta$  frame, i.e.,  $s'$ .
  - Magnitude of the vector using first the  $x$ - $y$  components and then the  $\xi$ - $\eta$  components. Are the two computed magnitudes the same or different? Explain.
  - Measure the angle between the  $x$  and  $\xi$  axes. Use this angle to test the validity of Eq. (3.4).

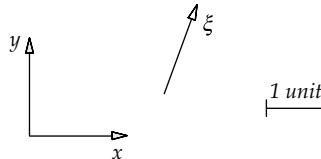


- 3.2 A link of a mechanism is shown. A body-fixed frame is defined on the link. Determine the local coordinates of the points  $A$ ,  $B$ ,  $C$ ,  $D$ , and  $E$ . Determine the local components of three unit vectors along the following axes:  $A$  to  $B$ ,  $B$  to  $E$ , and  $A$  to  $E$ .



3.3 The  $\xi$ -axis of a moving reference frame with respect to a nonmoving  $x$ - $y$  frame is shown.

- Draw the  $\eta$ -axis on the figure.
- Define unit vectors  $\vec{u}_{(\xi)}$  and  $\vec{u}_{(\eta)}$  along the  $\xi$  and  $\eta$  axes, respectively, and by measurement determine their components.
- Construct the  $A$  matrix using the unit vectors. Check for the orthogonality of the matrix.
- Measure the angle of the  $\xi$ -axis with respect to the  $x$ -axis and construct the  $A$  matrix using Eq. (3.5). Compare the result against that obtained in (c).



3.4 Point  $P_i$  has local and global coordinates  $s_i^P = \begin{bmatrix} 1.3 & -2.2 \end{bmatrix}'$  and  $r_i^P = \begin{bmatrix} -1.7 & 0.5 \end{bmatrix}'$ .

Find the translational coordinates of the body if  $\phi_i = 32^\circ$ . On a graph paper, show the position and orientation of the body-fixed frame with respect to the global frame and locate point  $P$ .

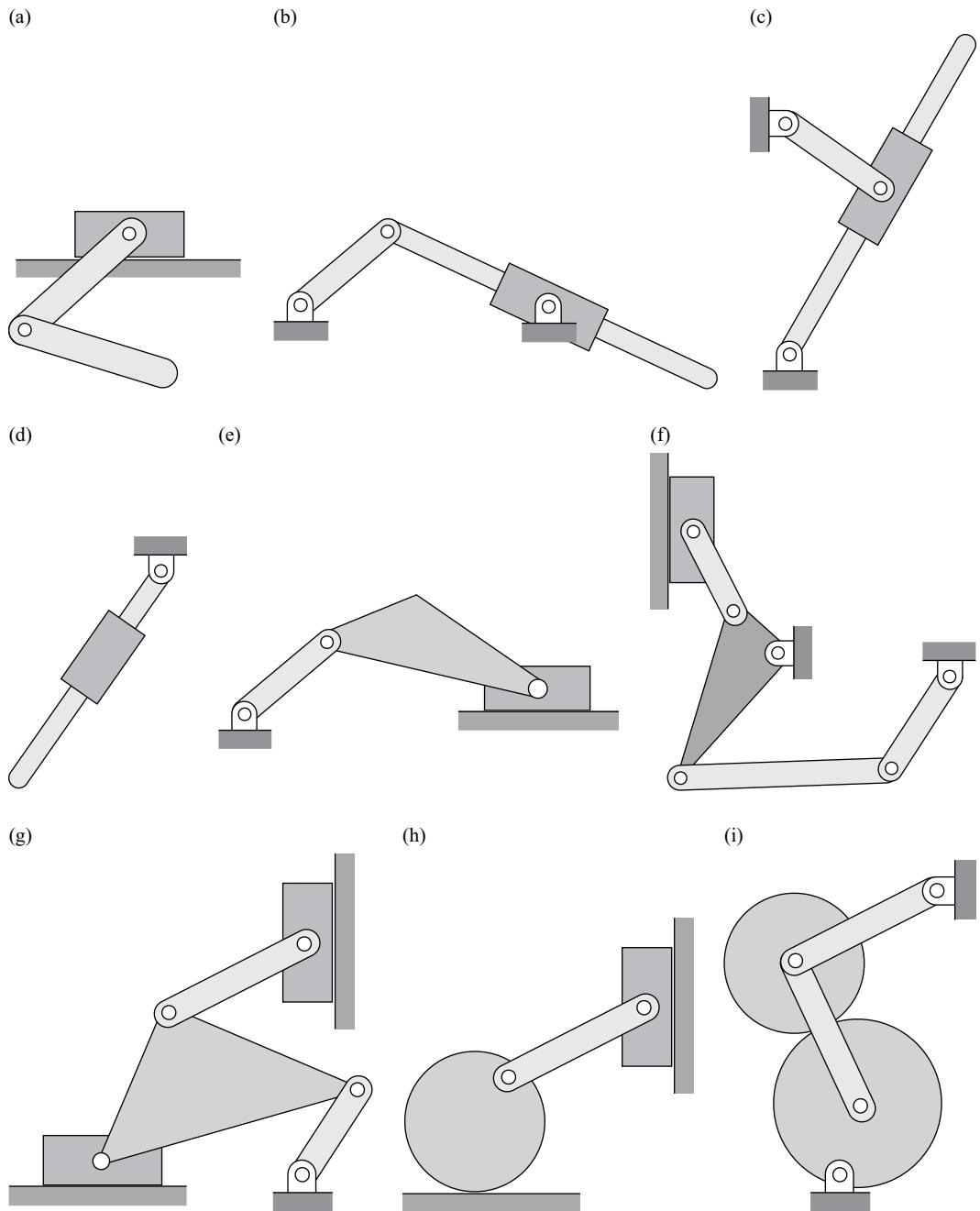
3.5 The coordinates of body ( $i$ ) are  $r_i = \begin{bmatrix} 3.2 & 2.8 \end{bmatrix}'$  and  $\phi_i = 80^\circ$ . Points  $A$  and  $B$  have local coordinates  $s_i^A = \begin{bmatrix} -1.1 & -0.4 \end{bmatrix}'$  and  $s_i^B = \begin{bmatrix} -1.9 & 2.3 \end{bmatrix}'$ . Point  $C$  has global coordinates  $r_i^C = \begin{bmatrix} 5.3 & 4.0 \end{bmatrix}'$ . Find the following:

- The global coordinates of  $A$ .
- The global components  $s_i^B$ .
- The local coordinates of  $C$ .

3.6 For each of the planar multibody systems shown, answer the following questions:

- Is the system open chain or closed chain?
- If the system contains any closed chains, identify the corresponding (independent) loop(s).
- Determine the number of DoFs of the system.

Note: For systems (h) and (i), consider two cases for the discs: slip and no-slip.



3.7 Consider the following position constraints:

$$\Phi_1 \equiv x_4 + 1.6 \cos \phi_4 - 0.3 \sin \phi_4 - x_1 + 0.75 \sin \phi_1 = 0$$

$$\Phi_2 \equiv y_4 + 1.6 \sin \phi_4 + 0.3 \cos \phi_4 - y_1 - 0.75 \cos \phi_1 = 0$$

where  $x_1, y_1, \phi_1, x_4, y_4$ , and  $\phi_4$  are variables.

- Construct the corresponding velocity constraints and express them in the form of Eq. (3.28), i.e., as a coefficient matrix times an array of velocities.
- Identify the Jacobian matrix and state the order for which you have arranged the columns of the matrix.
- Construct the acceleration constraints and identify the quadratic velocity terms.

**Note:** Use MATLAB to solve the following problems.

- 3.8 The angle between two reference frames  $x-y$  and  $\xi-\eta$ , according to our convention, is  $\phi_i = 328^\circ$ . The  $x-y$  components of vector  $\vec{a}$  are  $\mathbf{a} = \{-1.5 \quad 0.6\}'$ , and the  $\xi-\eta$  components of vector  $\vec{b}$  are  $\mathbf{b} = \{-0.7 \quad -0.4\}'$ . Compute  $\mathbf{a}'$  and  $\mathbf{b}$ . Make use of the function `A_matrix`.
- 3.9 Repeat Problem 3.8 for  $\phi_i = -32^\circ$  and compare the results. What do you conclude?
- 3.10 Assume that the body-fixed frame in Problem 3.2 is positioned in an  $x-y$  frame by the following coordinates:  $\mathbf{r} = \{2 \quad -3\}'$ ,  $\phi = \pi/6$  rad. Determine the global coordinates of all five points and the global components of the three unit vectors.
- 3.11 Point  $Q$  on a body has local coordinates  $\mathbf{s}^Q = \{1.2 \quad -0.5\}'$ . At a given time, the position, velocity, and acceleration of the body are as follows:  $\mathbf{r} = \{3 \quad 2\}'$ ,  $\phi = \pi/3$  rad,  $\dot{\mathbf{r}} = \{-0.4 \quad 1.1\}'$ ,  $\dot{\phi} = 0.2\pi$  rad/s,  $\ddot{\mathbf{r}} = \{-0.2 \quad -0.3\}'$ ,  $\ddot{\phi} = -0.1\pi$  rad/s<sup>2</sup>.
  - Determine the global position, velocity, and acceleration of point  $Q$ . Make use of functions `r_Point`, `r_Point_d`, and `r_Point_dd`.
  - On engineering paper, draw the  $x-y$  frame at a convenient position. Then draw the body-fixed frame and locate point  $Q$ . Measure the global position of the point and verify your computed results.
- 3.12 The global position of point  $B$  attached to a body is  $\mathbf{r}^B = \{5 \quad -2\}'$ . The origin of the body-fixed  $\xi-\eta$  frame is located by the position vector  $\mathbf{r} = \{4 \quad -1\}'$ . The rotational coordinate between the two reference frames is measured as  $\phi_i = 25^\circ$ . Determine the body-fixed components of vector  $\vec{s}^B$ .
- 3.13 Point  $A$  on body (1) has local coordinates  $\mathbf{s}_1^A = \{2 \quad -3\}'$ . Point  $B$  on body (2) has local coordinates  $\mathbf{s}_2^B = \{-1 \quad 2\}'$ . At a given time, the two bodies have the following coordinates:  $\mathbf{r}_1 = \{6 \quad 4\}'$ ,  $\phi_1 = 25^\circ$ ,  $\mathbf{r}_2 = \{-3 \quad 7\}'$ ,  $\phi_2 = -85^\circ$ . Determine the distance between points  $A$  and  $B$ .
- 3.14 The first time derivative of a fixed-length vector  $\vec{s}$  can be computed as  $\dot{\mathbf{s}} = \check{\mathbf{s}}\phi$ . The second time derivative of  $\vec{s}$  can be determined either as  $\ddot{\mathbf{s}} = \check{\mathbf{s}}\ddot{\phi} - \mathbf{s}\dot{\phi}^2$  or as  $\ddot{\mathbf{s}} = \check{\mathbf{s}}\ddot{\phi} + \dot{\mathbf{s}}\dot{\phi}$ . The components of  $\vec{s}$  are found to be  $\mathbf{s} = \{-1 \quad 2\}'$ . Numerically show that either formula for  $\ddot{\mathbf{s}}$  yields the same answer. Assume  $\phi = 2.5$  rad/s and  $\ddot{\phi} = -0.8$  rad/s<sup>2</sup>.

# 4

---

## *Fundamentals of Planar Dynamics*

---

This chapter reviews some of the fundamental concepts of planar dynamics. We first review the dynamics of a single particle based on Newton's laws of motion and then extend it to a system of particles. The dynamic equations of motion for a rigid body are derived that relate the mass and moment of inertia of a body to the forces and moments/torques that act on the body. The forces and torques could be either applied or reaction.

Reaction forces and torques are the result of kinematic joints, or constraints, that exist between the bodies of a multibody system. This chapter provides a review of the reaction forces and torques associated with commonly used planar kinematic joints. Several formulations for including friction in the equations of motion are also discussed.

Applied forces are the results of force elements such as springs and dampers. These elements could be either translational or torsional, having linear or nonlinear characteristics. The concepts of representing applied forces and friction are combined to provide a simple model to determine the interacting force between a tire and the ground for vehicle dynamic simulations. The chapter concludes with a representation of a motor modeled either as a constraint equation or as a force element.

---

### 4.1 Newton's Laws of Motion

Newton's laws of motion form the basis of dynamics. These laws are stated as follows:

- I. A particle remains at rest or continues to move in a straight line with constant speed if no force acts on it.
- II. The acceleration of a particle is proportional to and in the same direction of the force that acts on it.
- III. The forces of action and reaction between interacting particles are equal in magnitude, collinear, and opposite in direction.

Although these laws are for particles, they can be applied to rigid bodies if interpreted properly. Newton's first law is the basis of static or static equilibrium analysis, whereas the second law provides the basis of dynamic analysis for one or more interacting particles or bodies. Newton's third law can be applied to analyze the *action* and *reaction* forces between contacting particles or bodies.

Newton's second law provides precise meanings to the terms *mass* and *force*. Once a unit mass is chosen, a unit force is defined to be the force necessary to give one unit of mass an acceleration of one unit magnitude.

In the discussion of Newton's laws of motion, it is presumed that position, and hence acceleration, is measured in an *inertial* reference frame. Such a reference frame should

technically be defined as one fixed in the stars. However, for most engineering purposes, an adequate frame is an *earth-fixed* reference system. It is important to note that for applications concerning space dynamics, or even in long-range trajectories, the rotation of the earth has a significant effect on the precision with which points can be located by means of Newton's equations of motion. In such applications, an earth-fixed reference system may be inadequate. In this textbook, we use the nonmoving  $x$ - $y$  axes as the inertial frame.

## 4.2 Particle Dynamics

Newton's second law of motion for a particle relates the total force acting on the particle,  $\vec{f}$ ; the mass of the particle,  $m$ ; and the acceleration,  $\ddot{\vec{r}}$ , as

$$\begin{aligned} m\ddot{x} &= f_{(x)} \\ m\ddot{y} &= f_{(y)} \end{aligned} \Rightarrow \left[ \begin{array}{cc} m & 0 \\ 0 & m \end{array} \right] \left\{ \begin{array}{c} \ddot{x} \\ \ddot{y} \end{array} \right\} = \left\{ \begin{array}{c} f_{(x)} \\ f_{(y)} \end{array} \right\} \Rightarrow m\ddot{\vec{r}} = \mathbf{f} \quad (4.1)$$

where  $\vec{r}$  is the position vector for the particle in a nonmoving  $x$ - $y$  frame, and it is assumed that a consistent system of units is used. The condition for particle *equilibrium* (Newton's first law) may be deduced from Eq. (4.1); that is, a particle remains at rest (in static equilibrium), or in a state of *constant velocity*, if the total force that acts on the particle is zero. In a state of equilibrium,  $\ddot{\vec{r}} = \vec{0}$ , since  $\vec{f} = \vec{0}$ .

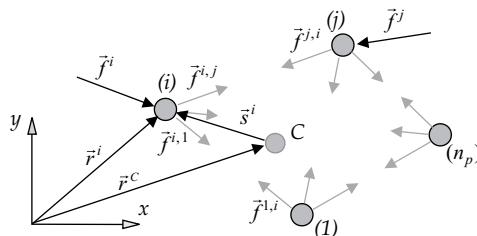
### 4.2.1 Dynamics of a System of Particles

The governing laws of dynamics of individual particles can be extended to systems of interacting particles. The equations of motion for such systems can be written simply as the collection of equations of motion for all the particles taken individually.

Consider the system of  $n_p$  particles shown in Figure 4.1 where a typical particle ( $i$ ) has a mass  $m^i$  and is located by a position vector  $\vec{r}^i$ . External forces may act on some of the particles; for example, an external force  $\vec{f}^i$  acts on particle ( $i$ ).

We define the total mass of the system as the sum of the individual masses:

$$m \equiv \sum_{i=1}^{n_p} m^i \quad (4.2)$$



**FIGURE 4.1**  
A system of  $n_p$  particles.

The sum of the external forces that act on the system is denoted as

$$\mathbf{f} = \sum_{i=1}^{n_p} \mathbf{f}^i \quad (4.3)$$

We locate the *center of mass* (or *centroid*) of the system of particles, as shown in Figure 4.1, by vector  $\bar{\mathbf{r}}^C$ . This vector is defined as

$$\mathbf{r}^C \equiv \frac{1}{m} \sum_{i=1}^{n_p} m^i \mathbf{r}^i \quad (4.4)$$

Then Newton's second law can describe the motion of the mass center of the system as

$$m \ddot{\mathbf{r}}^C = \mathbf{f} \quad (4.5)$$

This equation states that the resultant of all external forces on any system of mass equals the total mass of the system times the acceleration of the mass center; that is, the center of the mass moves as if it were a particle of mass  $m$  under the action of the force  $\vec{f}$ . This result will be the basis for deriving the equations of motion for a rigid body.

#### Proof 4.1

The following is a derivation of the equation of motion for the mass center of a system of particles:

In addition to external forces, internal forces also act between particles of a system. Assume that the internal forces that act on a typical particle ( $i$ ) are  $\mathbf{f}^{i,j}, j = 1, \dots, n_p$ , where  $\mathbf{f}^{i,i} = \mathbf{0}$ . The total force acting on particle ( $i$ ) is the summation of external and internal forces. Thus, for particle ( $i$ ), Eq. (4.1) becomes

$$m^i \ddot{\mathbf{r}}^i = \mathbf{f}^i + \sum_{j=1}^{n_p} \mathbf{f}^{i,j}, \quad i = 1, \dots, n_p \quad (a)$$

This system of equations describes the motion of the system of  $n_p$  particles.

Because the forces of interaction between particles in a system must satisfy Newton's law of action and reaction, the force on particle ( $i$ ) due to particle ( $j$ ) must be equal to the negative of the force on particle ( $j$ ) due to particle ( $i$ ):

$$\mathbf{f}^{i,j} = -\mathbf{f}^{j,i}, \quad i, j = 1, \dots, n_p \quad (b)$$

Note that because  $\mathbf{f}^{i,i} = \mathbf{0}$ , Eq. (b) also holds for  $i = j$ .

We write Eq. (a) for each particle in the system, and then the equations are summed to obtain

$$\sum_{i=1}^{n_p} m^i \ddot{\mathbf{r}}^i = \sum_{i=1}^{n_p} \mathbf{f}^i + \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \mathbf{f}^{i,j} \quad (c)$$

The double sum in Eq. (c) contains both  $\mathbf{f}^{i,i}$  and  $\mathbf{f}^{i,j}$ , and hence from Eq. (b), it is found that

$$\sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \mathbf{f}^{i,j} = 0 \quad (d)$$

Thus, Eq. (c) reduces to

$$\sum_{i=1}^{n_p} m^i \mathbf{r}^i = \sum_{i=1}^{n_p} \mathbf{f}^i \quad (e)$$

Substituting Eqs. (4.2), (4.3), and the second time derivative of Eq. (4.4) in Eq. (e) yields Eq. (4.5).

Assume that particle ( $i$ ) is positioned from the center of mass by vector  $\vec{s}^i$ ,  $i = 1, \dots, n_p$ , as shown in Figure 4.1. Therefore,

$$\mathbf{r}^i = \mathbf{r}^C + \mathbf{s}^i, \quad i = 1, \dots, n_p \quad (4.6)$$

One useful characteristic of the center of mass of a system of particles is obtained by substituting Eq. (4.6) into Eq. (4.4):

$$\sum_{i=1}^{n_p} m^i \mathbf{s}^i = \mathbf{0} \quad (4.7)$$

This is known as the equation of *first moments*.

### Example 4.1

For a system of three particles, we have the following data for the coordinates, masses, and applied forces:

$$\mathbf{r}^1 = \begin{Bmatrix} 2 \\ 1 \end{Bmatrix} \text{m}, \quad m^1 = 2 \text{ kg}, \quad \mathbf{f}^1 = \begin{Bmatrix} -6 \\ -2 \end{Bmatrix} \text{N}$$

$$\mathbf{r}^2 = \begin{Bmatrix} 1 \\ 4 \end{Bmatrix} \text{m}, \quad m^2 = 5 \text{ kg}, \quad \mathbf{f}^2 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \text{N}$$

$$\mathbf{r}^3 = \begin{Bmatrix} -3 \\ 2 \end{Bmatrix} \text{m}, \quad m^3 = 3 \text{ kg}, \quad \mathbf{f}^3 = \begin{Bmatrix} 5 \\ -1 \end{Bmatrix} \text{N}$$

Determine (a) the position of the mass center and (b) the acceleration of the mass center, and (c) numerically verify Eq. (4.7).

**Solution****MATLAB/Chapter 4/Example \_ 4 \_ 1**

First, we provide the data:

```
r1 = [2; 1]; r2 = [1; 4]; r3 = [-3; 2];
m1 = 2; m2 = 5; m3 = 3;
f1 = [-6; -2]; f2 = [0; 0]; f3 = [5; -1];
```

Next we determine the total mass and the position of the mass center as in Eq. (4.4):

```
m = m1 + m2 + m3;
rC = (m1*r1 + m2*r2 + m3*r3)/m . . . . .
```

$$\begin{aligned} rC &= \\ &\quad 0 \\ &\quad 2.8000 \end{aligned}$$

We compute the sum of forces acting on the system and then the acceleration of the mass center, that is, Eq. (4.5):

```
f = f1 + f2 + f3;
rC_dd = f/m . . . . .
```

$$\begin{aligned} rC_{dd} &= \\ &\quad -0.1000 \\ &\quad -0.3000 \end{aligned}$$

We can now determine the validity of Eq. (4.7):

```
s1 = r1 - rC; s2 = r2 - rC; s3 = r3 - rC;
sigma_m_s = m1*s1 + m2*s2 + m3*s3 . . . . .
```

$$\begin{aligned} \sigma_m_s &= \\ &\quad 1.0e-14 * \\ &\quad \quad 0 \\ &\quad \quad 0.1776 \end{aligned}$$

The computed value for  $\sigma_m_s$  is practically a zero vector—it is within the limits of the roundoff error.

### 4.3 Rigid body Dynamics

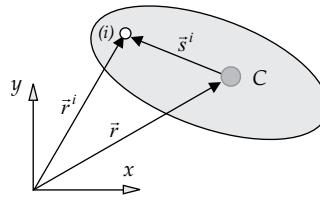
A rigid body is regarded as a collection of a large number of particles. In addition, from the definition of rigidity, the location of the particles in a body relative to one another remains unchanged.

The *center of mass* or the *centroid* of a body is defined in the same way as that of a system of particles, where the *summation* over all particles is replaced by an *integral* over the *area* of the planar body. If a particle is replaced by an infinitesimal mass  $dm$ , positioned at an arbitrary point ( $i$ ) on the area of the body, as shown in Figure 4.2, Eqs. (4.4) and (4.7) can be restated as

$$\mathbf{r} \equiv \frac{1}{m} \int_{(area)} \mathbf{r}^i dm \quad (4.8)$$

and

$$\int_{(area)} \mathbf{s}^i dm = \mathbf{0} \quad (4.9)$$



**FIGURE 4.2**  
A body is viewed as a collection of infinitesimal masses.

The first and second time derivatives of Eq. (4.9) yield two other useful identities as

$$\int_{(area)} \dot{s}^i dm = \mathbf{0}, \quad \int_{(area)} \ddot{s}^i dm = \mathbf{0} \quad (4.10)$$

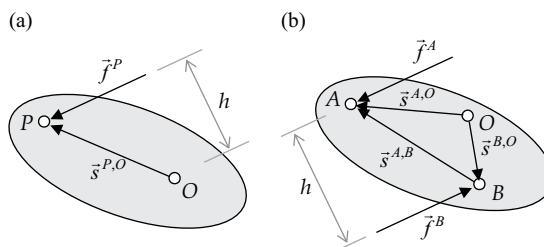
Since a rigid body can be considered as a collection of infinite number of particles, Newton's second law also describes how the mass center of a body accelerates under the application of a force. In addition to its tendency to move a body in the direction of its application, a force may also tend to rotate a body. The measure of this tendency is known as the *moment* of the force about a point or a given axis.

#### 4.3.1 Moment of a Force and Torque

Consider a force  $\vec{f}^P$  acting on a body at point  $P$  as shown in Figure 4.3a. The moment of this force about an axis perpendicular to the plane that passes through point  $O$  is computed from the cross product of vectors  $\vec{s}^{P,O}$  and  $\vec{f}^P$ . According to Eq. (2.15), the magnitude of this cross product can be obtained as

$$n = \vec{s}^{P,O} \cdot \vec{f}^P \quad (4.11)$$

The moment produced by two equal, opposite, and parallel forces is known as a *couple* or a *torque*. Couples have certain unique properties. The resultant force of a couple that acts on a body is zero. Figure 4.3b shows two equal and opposite forces,  $\vec{f}^A = -\vec{f}^B$ , acting on a body at points  $A$  and  $B$ . Vector  $\vec{s}^{A,B}$  joins point  $A$  to  $B$ . These two points are located by



**FIGURE 4.3**  
(a) A force acting on a body produces a moment about a typical point  $O$ , and (b) a couple resulting a torque.

position vectors  $\bar{s}^A$  and  $\bar{s}^B$  from an arbitrary point  $O$ . The combined moment of the two forces about  $O$  is

$$T = \bar{s}^{*A}\mathbf{f}^A + \bar{s}^{*B}\mathbf{f}^B = (\bar{s}^A - \bar{s}^B)' \mathbf{f}^A = \bar{s}^{*A,B}\mathbf{f}^A \quad (4.12)$$

The magnitude of the torque is  $T = hf$ , where  $h$  is the distance between the lines of action of the two forces; that is, the magnitude of the torque is independent of the position of point  $O$ .

#### 4.3.2 Centroidal Equations of Motion

The equation of motion for the mass center of a rigid body is directly obtained from Newton's second law as

$$m\ddot{\mathbf{r}} = \mathbf{f} \quad (4.13)$$

whereas  $m$  is the mass of the body,  $\ddot{\mathbf{r}}$  is the vector of acceleration of the *mass center*, and  $\mathbf{f}$  represents the vector sum of all external forces acting on the body.

In addition to the two translational degrees of freedom (DoFs) that a free body exhibits on a plane, a free body also exhibits one rotational DoF. The rotational equation of motion of a body is described as

$$J\ddot{\phi} = n \quad (4.14)$$

where  $J$  is the *mass moment of inertia* of the body about the mass center,\*  $\ddot{\phi}$  is the angular acceleration of the body, and  $n$  is the sum of all the torques and the moments of all forces with respect to the mass center that acts on the body.

Equations (4.13) and (4.14) can be appended and put in matrix form:

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} f_{(x)} \\ f_{(y)} \\ n \end{bmatrix} \Rightarrow \begin{bmatrix} m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & J \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{r}} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ n \end{bmatrix} \Rightarrow \mathbf{M}\ddot{\mathbf{c}} = \mathbf{h} \quad (4.15)$$

where  $\ddot{\mathbf{c}} \equiv \begin{bmatrix} \ddot{\mathbf{r}} \\ \ddot{\phi} \end{bmatrix}$  is the centroidal *array of accelerations*,  $\mathbf{h} \equiv \begin{bmatrix} \mathbf{f} \\ n \end{bmatrix}$  is the centroidal *array of forces*, and  $\mathbf{M} \equiv \begin{bmatrix} m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & J \end{bmatrix}$  is the *mass or inertia matrix*. Equation (4.15) is referred to as the *centroidal equations of motion*, because the reference point on the body is its mass center. In this text, the term *equations of motion* or *dynamic equilibrium equations* will be used to refer to the centroidal equations of motion unless specified otherwise.

---

\* Throughout this textbook, we use the term *moment of inertia* to refer to the mass moment of inertia about an axis perpendicular to the plane and going through the mass center, unless stated otherwise.

### Proof 4.2

The following is the derivation of the rotational equation of motion for a planar rigid body as described in Eq. (4.14):

We return to the idea that a rigid body can be viewed as a system of particles. We substitute Eq. (3.17) into Eq. (a) from Proof 4.1 to get

$$m^i \left( \ddot{\mathbf{r}} + \ddot{\mathbf{s}}^i \dot{\phi} - \mathbf{s}^i \dot{\phi}^2 \right) = \mathbf{f}^i + \sum_{j=1}^{n_p} \mathbf{f}^{i,j}, \quad i = 1, \dots, n_p \quad (\text{a})$$

We premultiply both sides of this equation by  $\ddot{\mathbf{s}}'^i$ :

$$m^i \ddot{\mathbf{s}}'^i \left( \ddot{\mathbf{r}} + \ddot{\mathbf{s}}^i \dot{\phi} - \mathbf{s}^i \dot{\phi}^2 \right) = \ddot{\mathbf{s}}'^i \left( \mathbf{f}^i + \sum_{j=1}^{n_p} \mathbf{f}^{i,j} \right), \quad i = 1, \dots, n_p \quad (\text{b})$$

The sum of all  $n_p$  equations in Eq. (b) yields

$$\sum_{i=1}^{n_p} \left( m^i \ddot{\mathbf{s}}'^i \left( \ddot{\mathbf{r}} + \ddot{\mathbf{s}}^i \dot{\phi} - \mathbf{s}^i \dot{\phi}^2 \right) \right) = \sum_{i=1}^{n_p} \left( \ddot{\mathbf{s}}'^i \mathbf{f}^i \right) + \sum_{i=1}^{n_p} \left( \ddot{\mathbf{s}}'^i \sum_{j=1}^{n_p} \mathbf{f}^{i,j} \right) \quad (\text{c})$$

We now examine the terms in Eq. (c) individually. The first term can be written as

$$\sum_{i=1}^{n_p} \left( m^i \ddot{\mathbf{s}}'^i \ddot{\mathbf{r}} \right) = \left( \sum_{i=1}^{n_p} \left( m^i \ddot{\mathbf{s}}'^i \right) \right) \ddot{\mathbf{r}} = 0 \quad (\text{d})$$

where we have used Eq. (4.7). Note that Eq. (4.7) is still valid if all the vectors are rotated in the same direction by  $90^\circ$  and/or written in transposed form.

The second term in Eq. (c) can be expressed as

$$\sum_{i=1}^{n_p} \left( m^i \ddot{\mathbf{s}}'^i \ddot{\mathbf{s}}^i \dot{\phi} \right) = \left( \sum_{i=1}^{n_p} m^i (s^i)^2 \right) \ddot{\phi} = J \ddot{\phi} \quad (\text{e})$$

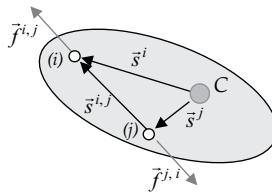
where we have defined the *moment of inertia* for the system as

$$J \equiv \sum_{i=1}^{n_p} m^i s^{i2} \quad (\text{f})$$

The third term in Eq. (c) becomes

$$\sum_{i=1}^{n_p} \left( m^i \ddot{\mathbf{s}}'^i \mathbf{s}^i \right) \dot{\phi}^2 \Rightarrow 0 \quad (\text{g})$$

since the scalar product of two perpendicular vectors is zero.

**FIGURE 4.4**

Reaction forces between two infinitesimal masses.

The first term on the right-hand side of Eq. (c) is the sum of the moments caused by all external forces (which also includes the sum of applied torques):

$$n \equiv \sum_{i=1}^{n_p} (\check{s}'^i \mathbf{f}^i) \quad (h)$$

The second term on the right-hand side of Eq. (c) can be expressed as

$$\begin{aligned} \sum_{i=1}^{n_p} \left( \check{s}'^i \sum_{j=1}^{n_p} \mathbf{f}^{i,j} \right) &= \dots + \check{s}'^i \mathbf{f}^{i,j} + \check{s}'^j \mathbf{f}^{j,i} + \dots = \dots + \check{s}'^i \mathbf{f}^{i,j} - \check{s}'^j \mathbf{f}^{i,j} + \dots \\ &= \dots + (\check{s}^i - \check{s}^j)' \mathbf{f}^{i,j} + \dots = \dots + \check{s}'^i \mathbf{f}^{i,j} + \dots = 0 \end{aligned} \quad (i)$$

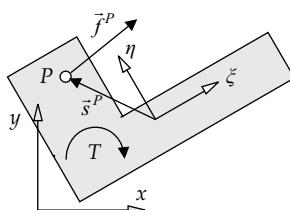
where we have noted that vectors  $\check{s}^{i,j}$  and  $\check{f}^{i,j}$  are collinear, as shown in Figure 4.4; therefore, the scalar product of  $\check{s}^{i,j}$  and  $\check{f}^{i,j}$  is zero.

The preceding process can be repeated with an integral over area replacing the summation to obtain the rotational equation of motion for a planar body as stated in Eq. (4.14). In the process, the *mass moment of inertia* is defined as

$$J \equiv \int_{\text{(area)}} (s^i)^2 dm \quad (j)$$

### Example 4.2

A body with a mass  $m = 0.2\text{ kg}$  is acted upon by gravity, a constant force, and a torque as shown in Figure 4.5. The torque has a magnitude of  $0.15\text{ N m}$  in CW direction. The local coordinates of the application point of the force and the  $x-y$  components of the force are

**FIGURE 4.5**

A force and a torque acting on a body.

$$\mathbf{s}^P = \begin{Bmatrix} -0.2 \\ 0.3 \end{Bmatrix} \text{m}, \quad \mathbf{f}^P = \begin{Bmatrix} 1.2 \\ 1.0 \end{Bmatrix} \text{N}$$

Determine the array of forces for the body if  $\phi = 30^\circ$  and  $\mathbf{r} = \begin{Bmatrix} 0.42 & 0.31 \end{Bmatrix}' \text{ m}$ .

**Solution**

#### MATLAB/Chapter 4/Example \_ 4 \_ 2

We first state the data:

```
r = [0.42; 0.31]; phi = pi/6;
s_P_local = [-0.2; 0.3];
m = 0.2;
f_P = [1.2; 1.0]; T = -0.15;
```

We determine the  $x$ - $y$  components of vector  $\bar{\mathbf{s}}^P$ , which is the moment arm of  $\bar{\mathbf{f}}^P$ , and then the moment caused by the force:

```
A = A_matrix(phi); s_P = A*s_P_local;
n = s_rot(s_P)'*f_P ..... n =
3.2862
```

Finally, we evaluate the weight and construct the array of force/moment for the body:

<pre>weight = m*9.81*[0; -1] ..... weight = 0</pre>	<pre>h = [f_P + weight; n + T] ..... h = 1.2000 -0.9620 3.1362</pre>
---	--

#### Example 4.3 (Example 3.1 cont.)

In addition to the coordinate data given in Example 3.1, assume that two external forces act on the link at points  $A$  and  $B$  as

$$\mathbf{f}^A = \begin{Bmatrix} 2 \\ -1 \end{Bmatrix} \text{N}, \quad \mathbf{f}^B = \begin{Bmatrix} -3 \\ 2 \end{Bmatrix} \text{N}$$

The mass and the moment of inertia for the link are provided as  $m = 2.5 \text{ kg}$  and  $J = 1.2 \text{ kg m}^2$ . Compute the linear and angular acceleration of the body.

**Solution**

#### MATLAB/Chapter 4/Example \_ 4 \_ 3

We first provide the data:

```
r = [2.5; 1.2]; phi = 5.6723;
s_A_local = [2.18; 0]; s_B_local = [-1.8; 1.3];
m = 2.5; J = 1.2;
f_A = [2; -1]; f_B = [-3; 2];
```

We compute the  $x$ - $y$  components of  $\vec{s}^A$  and  $\vec{s}^B$ :

```
A = A_matrix(phi);
s_A = A*s_A_local; s_B = A*s_B_local;
```

The sum of forces and the sum of the moments are computed next:

```
f = f_A + f_B;
n = s_rot(s_A)'*f_A + s_rot(s_B)'*f_B;
```

We then determine the accelerations:

$$r_{dd} = f/m \quad \dots \dots \dots$$

$$\phi_{dd} = n/J \quad \dots \dots \dots$$

$$\begin{aligned} r_{dd} &= \\ &-0.4000 \\ &0.4000 \\ \phi_{dd} &= \\ &4.6247 \end{aligned}$$

#### 4.3.3 Noncentroidal Equations of Motion

If the reference point on a body is not defined at its mass center, the corresponding equations of motion are called *noncentroidal*. For example, for the body shown in Figure 4.6, the mass center is at  $C$ , but the reference point is defined at  $O$ . The noncentroidal equations of motion for this body are expressed in matrix form as

$$\begin{bmatrix} m\mathbf{I} & m\vec{s}^{C,O} \\ m\vec{s}'^{C,O} & J^O \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{r}}^O \\ \ddot{\phi} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^O \\ n^O \end{Bmatrix} \Rightarrow {}^{(n-c)}\mathbf{M}^{(n-c)}\ddot{\mathbf{c}} = {}^{(n-c)}\mathbf{h} \quad (4.16)$$

where  $\vec{s}^{C,O}$  locates the mass center from point  $O$ , and  $J^O$  is the *moment of inertia* with respect to an axis passing through  $O$ , which is computed as

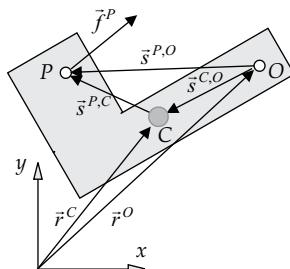
$$J^O = J + m(s^{C,O})^2 \quad (4.17)$$

In addition, a new array of forces is defined as

$$\mathbf{f}^O = \mathbf{f}^P + m\vec{s}'^{C,O}\dot{\phi}^2 \quad (4.18)$$

and  $n^O$  is the moment of the force with respect to point  $O$ , which is computed as

$$n^O = \vec{s}'^{C,O}\mathbf{f}^P \quad (4.19)$$



**FIGURE 4.6**

Vector descriptions for the centroidal and noncentroidal systems.

In the compact form of Eq. (4.16), the following arrays and matrix have been defined:

$$\begin{aligned} {}^{(n-c)}\ddot{\mathbf{c}} &\equiv \left\{ \begin{array}{c} \ddot{\mathbf{r}}^O \\ \ddot{\phi} \end{array} \right\}, \quad {}^{(n-c)}\mathbf{h} \equiv \left\{ \begin{array}{c} \mathbf{f}^O \\ n^O \end{array} \right\}, \quad {}^{(n-c)}\mathbf{M} \equiv \left[ \begin{array}{cc} m\mathbf{I} & m\ddot{\mathbf{s}}^{C,O} \\ m\ddot{\mathbf{s}}'^{C,O} & J^O \end{array} \right] \end{aligned}$$

### Proof 4.3

The centroidal equations of motion can be transformed to the noncentroidal form through a simple process. To simplify the process of transforming Eq. (a) to the noncentroidal equations of motion, we assume that a single force  $\ddot{\mathbf{f}}^P$  acts at point  $P$  on the body. This assumption does not make the proof any less general—other forces and moments can be added to the system. We first rewrite the centroidal equations of motion, from Eq. (4.15), as

$$\left[ \begin{array}{cc} m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & J^C \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^C \\ \ddot{\phi} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{f}^P \\ n^C = \ddot{\mathbf{s}}'^{P,C}\mathbf{f}^P \end{array} \right\} \quad (\text{a})$$

where we have emphasized that the body mass center, point  $C$ , is the reference point.

The acceleration of the new reference point  $O$  can be obtained from Eq. (3.17) as

$$\ddot{\mathbf{r}}^C = \ddot{\mathbf{r}}^O + \ddot{\mathbf{s}}^{C,O}\ddot{\phi} - \mathbf{s}^{C,O}\dot{\phi}^2 \quad (\text{b})$$

The rotational acceleration of the body does not depend on the location of the body's origin, or on how the local frame is attached to the body (note that we have not yet defined a body-fixed frame). Hence, a more complete transformation equation for both the translational and rotational accelerations can be written as

$$\left\{ \begin{array}{c} \ddot{\mathbf{r}}^C \\ \ddot{\phi} \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \ddot{\mathbf{s}}^{C,O} \\ 0 & 1 \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^O \\ \ddot{\phi} \end{array} \right\} - \left\{ \begin{array}{c} \mathbf{s}^{C,O}\dot{\phi}^2 \\ 0 \end{array} \right\} \quad (\text{c})$$

We substitute Eq. (c) into Eq. (a), then premultiply both sides of the equation by the transpose of the coefficient matrix in Eq. (c):

$$\begin{aligned} & \left[ \begin{array}{cc} \mathbf{I} & 0 \\ \ddot{\mathbf{s}}'^{C,O} & 1 \end{array} \right] \left[ \begin{array}{cc} m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & J^C \end{array} \right] \left[ \begin{array}{cc} \mathbf{I} & \ddot{\mathbf{s}}^{C,O} \\ 0 & 1 \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^O \\ \ddot{\phi} \end{array} \right\} - \left\{ \begin{array}{c} \mathbf{s}^{C,O}\dot{\phi}^2 \\ 0 \end{array} \right\} \\ &= \left[ \begin{array}{cc} \mathbf{I} & 0 \\ \ddot{\mathbf{s}}'^{C,O} & 1 \end{array} \right] \left\{ \begin{array}{c} \mathbf{f}^P \\ \ddot{\mathbf{s}}'^{P,C}\mathbf{f}^P \end{array} \right\} \end{aligned} \quad (\text{d})$$

This equation can be simplified to

$$\left[ \begin{array}{cc} m\mathbf{I} & m\ddot{\mathbf{s}}^{C,O} \\ m\ddot{\mathbf{s}}'^{C,O} & J^C + m(s^{C,O})^2 \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^O \\ \ddot{\phi} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{f}^P + m\mathbf{s}^{C,O}\dot{\phi}^2 \\ \ddot{\mathbf{s}}'^{P,O}\mathbf{f}^P \end{array} \right\} \quad (\text{e})$$

where the identities  $\bar{\mathbf{s}}'^{C,O} \mathbf{s}^{C,O} = 0$ ,  $(\mathbf{s}^{C,O})^2 = \bar{\mathbf{s}}'^{C,O} \bar{\mathbf{s}}^{C,O}$ , and  $\mathbf{s}^{P,O} = \mathbf{s}^{P,C} + \mathbf{s}^{C,O}$  have been used. This is the same noncentroidal equation of motion as described in Eq. (4.16).

---

## 4.4 Multibody Dynamics

The equations of motion for a mechanical system containing one or more bodies can be written in various forms. Several methods for constructing these equations are discussed in the upcoming chapters. For any of these methods, we should start the process of constructing the equations of motion by determining the forces and torques that act on each body of a system. A force (or a torque) that acts on a body may be categorized as an *applied force* or as a *reaction force*.

Applied forces are produced by *force elements*, such as springs and dampers. An applied force could be either a constant or a function of time. Furthermore, a nonconstant force could be a function of the position coordinates and/or a function of the velocities. Therefore, in general, if the positions and velocities of the bodies are known, applied forces between the bodies can be computed.

Reaction forces are the result of the existence of kinematic joints (or constraints) between bodies. These forces and torques are functions of the coordinates, velocities, accelerations, and applied forces. Therefore, reaction forces cannot be computed as easily as the applied forces.

### 4.4.1 Applied Forces

A variety of force elements may appear in multibody systems. In our formulations, we assume that the force elements are ideal components that do not impose any kinematic constraints on a system. Furthermore, we assume that force elements are massless, and therefore, they are not treated as bodies. If the mass of a force element is significant and should not be ignored, the mass could be included in the multibody equations of motion indirectly. In this section, we consider the formulation for some commonly used force elements.

One typical force that may act on a body is the gravitational force. As long as the mass of the body is known, the force of gravity can be computed. In this textbook, the gravitational field will be considered to be acting in the negative  $y$ -direction, that is,  $-\bar{\mathbf{u}}_{(y)}$ , unless indicated otherwise. This choice of direction for the gravity is totally arbitrary. If  $m_i$  is the mass of body ( $i$ ) and  $g$  is the gravitational constant, then the gravitational force is computed as

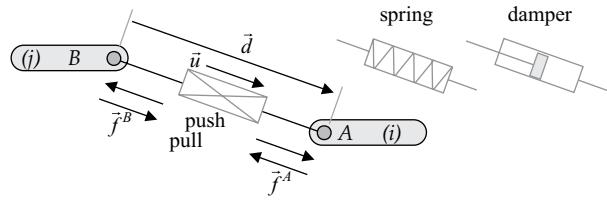
$${}^{(g)}\mathbf{f}_i = -w_i \mathbf{u}_{(y)} \quad (4.20)$$

where the *weight* of the body is denoted as

$$w_i = m_i g \quad (4.21)$$

The gravitational force acts directly on the mass center of a body, and therefore, there is no moment associated with it.

An actuator is a point-to-point force element that acts between points  $A$  and  $B$  on two different bodies, as shown in Figure 4.7. An actuator applies a *constant* or a *time-dependent*

**FIGURE 4.7**

A point-to-point actuator acting between two links.

pair of forces on points  $A$  and  $B$ , equal in magnitude with a common line of action but in opposite directions. The sign convention for the pair of forces can be defined as *positive* when the forces *pull* on the bodies and *negative* when the forces *push* on the bodies. If the actuator force is denoted by  ${}^{(a)}f$ , then  ${}^{(a)}f > 0$  constitutes a pull (tension) and  ${}^{(a)}f < 0$  constitutes a push (compression).

To determine the components of the pair of forces that act on the bodies, a unit vector on the line of action of the actuator must be defined. First, a vector  $\vec{d}$  connecting the two points is defined:

$$\mathbf{d} = \mathbf{r}^A - \mathbf{r}^B \quad (4.22)$$

The magnitude of this vector is computed as

$$L^2 = \mathbf{d}'\mathbf{d} \quad (4.23)$$

A unit vector  $\vec{u}$  is constructed as

$$\mathbf{u} = \frac{1}{L}\mathbf{d} \quad (4.24)$$

This unit vector has the same direction as the force that acts on  $A$ , in the case of a push, and opposite of that, in the case of a pull. Therefore,

$${}^{(a)}\mathbf{f}_i = -{}^{(a)}\mathbf{f}\mathbf{u}, \quad {}^{(a)}\mathbf{f}_j = {}^{(a)}\mathbf{f}\mathbf{u} \quad (4.25)$$

Note that since  ${}^{(a)}f$  could be either positive or negative, the direction of each force in Eq. (4.25) is automatically taken care of.

Point-to-point springs are the most commonly used force elements in mechanical systems. Assume that the force element in Figure 4.7 is a spring that is attached between the two bodies. The force of this spring is computed as

$${}^{(s)}\mathbf{f} = k(L - {}^0L) \quad (4.26)$$

where  $k$  is the *stiffness*,  $L$  is the deformed length, and  ${}^0L$  is the undeformed length of the spring. The deformed length of a spring is determined from Eq. (4.23).

The sign convention for a spring force is similar to that of the actuator force—positive in *tension* (pull) and negative in *compression* (push). The forces that the spring applies on the two bodies are

$${}^{(s)}\mathbf{f}_i = -{}^{(s)}\mathbf{f}\mathbf{u}, \quad {}^{(s)}\mathbf{f}_j = {}^{(s)}\mathbf{f}\mathbf{u} \quad (4.27)$$

This equation is valid in both tension and compression: if  $L > {}^0L$  (for tension),  ${}^{(s)}f$  is positive so the bodies are pulled toward each other; if  $L < {}^0L$  (for compression),  ${}^{(s)}f$  is negative so the bodies are pushed away from one another.

Another type of a point-to-point force element is a linear damper, where the damping force is computed as

$${}^{(d)}f = d_c \dot{L} \quad (4.28)$$

In this equation,  $d_c$  is the damping coefficient and  $\dot{L}$  is the time rate of change of the damper length. We can compute  $\dot{L}$  by taking the time derivative of Eq. (4.23):

$$2L\ddot{L} = 2\mathbf{d}'\dot{\mathbf{d}} \Rightarrow \dot{L} = \frac{1}{L}\mathbf{d}'\dot{\mathbf{d}} = \mathbf{u}'\dot{\mathbf{d}} \quad (4.29)$$

where  $\dot{\mathbf{d}}$  is computed as  $\dot{\mathbf{d}} = \dot{\mathbf{r}}^A - \dot{\mathbf{r}}^B$ . The sign convention for the damper force is defined as positive for  $\dot{L} > 0$  and negative for  $\dot{L} < 0$ . Since a damper opposes the motion between two bodies, when the bodies move away from each other ( $\dot{L} > 0$ ), the forces of the damper exhibit a pull, and when the bodies move toward each other ( $\dot{L} < 0$ ), the forces of the damper exhibit a push. Therefore, the forces acting on the bodies are

$${}^{(d)}\mathbf{f}_i = -{}^{(d)}f\mathbf{u}, \quad {}^{(d)}\mathbf{f}_j = {}^{(d)}f\mathbf{u} \quad (4.30)$$

These equations are valid for both pull and push cases.

A spring, a damper, and an actuator can act as a combined force element between points  $A$  and  $B$  of two links. If the point of application for the forces on each element is the same, we may combine the forces into a single force as

$${}^{(s,d,a)}f = {}^{(s)}f + {}^{(d)}f + {}^{(a)}f \quad (4.31)$$

Then the pair of force vectors is computed as

$${}^{(s,d,a)}\mathbf{f}_i = -{}^{(s,d,a)}f\mathbf{u}, \quad {}^{(s,d,a)}\mathbf{f}_j = {}^{(s,d,a)}f\mathbf{u} \quad (4.32)$$

Three MATLAB® functions are presented here:

`pp_s` : point-to-point spring

`pp_sd` : point-to-point combined spring-damper

`pp_sda` : point-to-point combined spring-damper-actuator

In addition to the element characteristics, all three functions need the  $x-y$  components of vector  $\vec{d}$ , and the combined functions also need the time derivative of this vector.

```
function f_s = pp_s(d, k, L0)
    L = norm(d); u = d/L;
    f = k*(L - L0);
    f_s = f*u;

function f_sd = pp_sd(d, d_d, k, L0, dc)
    L = norm(d); u = d/L; L_d = d'*d_d/L;
    f = k*(L - L0) + dc*L_d;
    f_sd = f*u;

function f_sda = ptp_sda(d, d_d, k, L0, dc, fa)
    L = norm(d); u = d/L; L_d = d'*d_d/L;
    f = k*(L - L0) + dc*L_d + fa;
    f_sda = f*u;
```

**Example 4.4**

A point-to-point combined spring–damper–actuator is connected between point  $A$  on link (1) and point  $B$  on link (2) as shown in Figure 4.8. The two points are positioned in their respective  $\xi$ – $\eta$  frame as

$$\mathbf{s}_1^A = \begin{Bmatrix} 0.15 \\ 0 \end{Bmatrix} \text{m}, \mathbf{s}_2^B = \begin{Bmatrix} 0 \\ 0.1 \end{Bmatrix} \text{m}$$

The actuator pushes on the bodies with a force of 2 N. The spring has an undeformed length  ${}^0L = 0.15$  m and a stiffness  $k = 10$  N/m. The damping coefficient is  $d_c = 5$  N s/m. In the shown configuration, the two bodies have the following coordinates and velocities:

$$\mathbf{c}_1 = \begin{Bmatrix} 0.10 \text{ m} \\ 0.05 \text{ m} \\ 0.785 \text{ rad} \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} 0.30 \text{ m} \\ -0.05 \text{ m} \\ 0.349 \text{ rad} \end{Bmatrix}$$

$$\dot{\mathbf{c}}_1 = \begin{Bmatrix} 0.1 \text{ m/s} \\ 0.2 \text{ m/s} \\ -0.25 \text{ rad/s} \end{Bmatrix}, \dot{\mathbf{c}}_2 = \begin{Bmatrix} -0.2 \text{ m/s} \\ 0.1 \text{ m/s} \\ 0.12 \text{ rad/s} \end{Bmatrix}$$

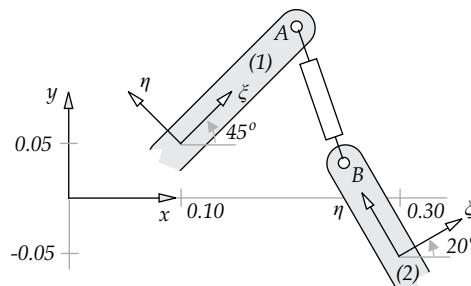
Determine the force and the moment that act on each link.

*Solution*

**MATLAB/Chapter 4/Example \_ 4 \_ 4**

We first state the data:

```
s_A1_local = [0.15; 0]; s_B2_local = [0; 0.1];
r1 = [ 0.1;  0.05]; phi1 = pi/4;
r2 = [ 0.3; -0.05]; phi2 = 20*pi/180;
r1_d = [ 0.1;  0.2]; phi1_d = -0.25;
r2_d = [-0.2;  0.1]; phi2_d = 0.12;
L0 = 0.15; k = 10; dc = 5; fa = -2;
```



**FIGURE 4.8**

A combined force element acting between two bodies.

We compute the coordinates of points  $A$  and  $B$ , and then vector  $\vec{d}$ :

```
A1 = A_matrix(phi1); A2 = A_matrix(phi2);
s_A1 = A1*s_A1_local; s_B2 = A2*s_B2_local;
r_A1 = r_Point(r1, s_A1); r_B2 = r_Point(r2, s_B2);
d     = r_A1 - r_B2;
```

The velocity of points  $A$  and  $B$ , and then vector  $\dot{\vec{d}}$  are determined next:

```
r_A1_d = r_Point_d(r1_d, s_A1, phi1_d);
r_B2_d = r_Point_d(r2_d, s_B2, phi2_d);
d_d    = r_A1_d - r_B2_d;
```

We compute the force of the spring-damper-actuator using function M-file pp\_sda.

Then we determine the forces that act at points  $A$  and  $B$ :

```
f_sda = pp_sda(d, d_d, k, L0, dc, fa);
f_A1 = -f_sda
f_B2 = f_sda
```

$f_A1 =$	$-1.2611$
$f_B2 =$	$2.3667$
$f_A1 =$	$1.2611$
$f_B2 =$	$-2.3667$

The associated moments are computed next:

```
n1 = s_rot(s_A1)'*f_A1
n2 = s_rot(s_B2)'*f_B2
```

$n1 =$	$0.3848$
$n2 =$	$-0.0376$

If we report the intermediate computational results within function pp\_sda, we obtain

$$L = 0.127 \text{ m}, \mathbf{u} = \begin{Bmatrix} -0.470 & 0.883 \end{Bmatrix}' \text{ m}, \dot{L} = 0.090 \text{ m/s}$$

$$^{(s)}f = -0.230 \text{ N}, ^{(d)}f = -0.452 \text{ N}, ^{(s,d,a)}f = -2.682 \text{ N}$$

Similar to the point-to-point force elements, we can also have *rotational* (or *torsional*) torque elements between two bodies. We assume that a rotational element, such as the one shown in Figure 4.9, acts about the axis of a pin joint. This element applies a pair of torques, equal in magnitude but opposite in direction, on the bodies. A rotational element could be an actuator, a spring, a damper, or a combination of them.

A *rotational actuator* is a motor that applies a pair of torques,  $^{(r-a)}T$  and  $-^{(r-a)}T$ , either constant or time varying, on the connecting bodies. We can adopt a sign convention that if the applied torque on a body is counterclockwise, we consider the torque positive; otherwise, it is negative.

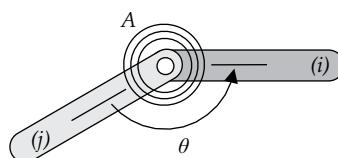


FIGURE 4.9

A rotational force element acting between two links.

The torque associated with a rotational spring between two bodies is calculated as

$${}^{(r-s)}T = {}^{(r)}k(\theta - {}^0\theta) \quad (4.33)$$

where  ${}^{(r)}k$  is the *rotational stiffness* of the spring, and  $\theta$  is a measure of the relative rotational displacement between the two bodies having a value of  ${}^0\theta$  in the undeformed state of the spring. This angle may be defined as the angle between two defined vectors on the bodies. We note that when  $\theta > {}^0\theta$ , the computed torque is positive, which indicates the spring being in *tension*, and when  $\theta < {}^0\theta$ , the spring is in *compression*.

Similarly, for a rotational damper between two bodies, the torque of the damper is computed as

$${}^{(r-d)}T = {}^{(r)}d_c \dot{\theta} \quad (4.34)$$

where  ${}^{(r)}d_c$  is the rotational damping coefficient and  $\dot{\theta}$  is the relative rotational velocity between the two bodies.

For a combined rotational spring–damper–actuator, the individual torques can be added to obtain a combined torque as

$${}^{(r-s,d,a)}T = {}^{(r-s)}T + {}^{(r-d)}T + {}^{(r-a)}T \quad (4.35)$$

This torque can then be applied to bodies  $(i)$  and  $(j)$  as

$$T_i = -{}^{(r-s,d,a)}T, T_j = {}^{(r-s,d,a)}T \quad (4.36)$$

A simple way to describe the angle for a rotational element is to define it as

$$\theta = \phi_i - \phi_j \quad (4.37)$$

Here, instead of defining new vectors on each body, we use the angle between the reference axes  $\xi_i$  and  $\xi_j$ . With this definition of angle, the undeformed angle  ${}^0\theta$  must be adjusted properly. This definition leads to computing  $\dot{\theta}$  as

$$\dot{\theta} = \dot{\phi}_i - \dot{\phi}_j \quad (4.38)$$

Three MATLAB functions are presented here:

`r_s` : rotational spring  
`r_sd` : rotational combined spring–damper  
`r_sda` : rotational combined spring–damper–actuator

```
function T_s = r_s(theta, k, theta0)
    T_s = k*(theta - theta0);

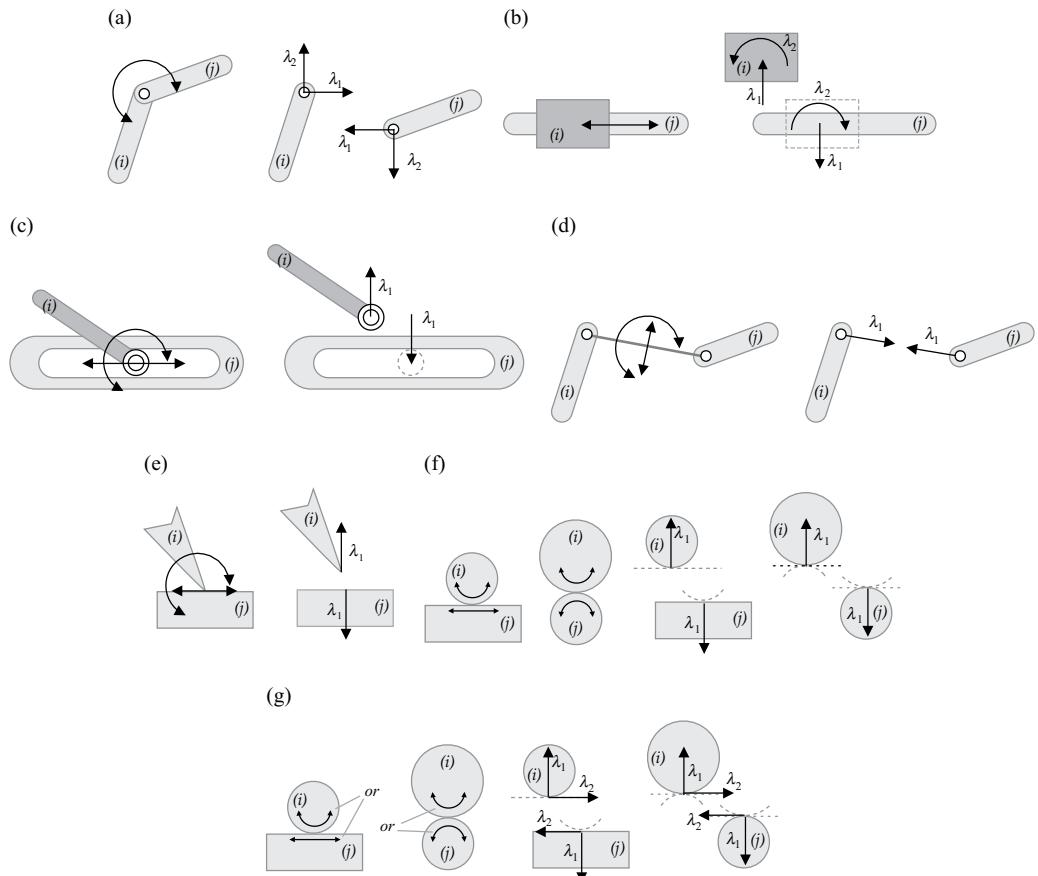
function T_sd = r_sd(theta, theta_d, k, theta0, dc)
    T_sd = k*(theta - theta0) + dc*theta_d;

function T_sda = r_sda(theta, theta_d, k, theta0, dc, Ta)
    T_sda = k*(theta - theta0) + dc*theta_d + Ta;
```

#### 4.4.2 Reaction Forces

When kinematic joints are present in a multibody system, we must consider their corresponding reaction forces and/or torques in the equations of motion. Newton's third law of motion between two particles can directly be correlated with the reaction forces due to kinematic joints between two bodies. Reaction forces (or torques) act between two bodies along (or about) the axis for which no *relative motion* is allowed. A kinematic joint removes one or more of the three relative DoFs between two planar bodies. The imposed restrictions on the relative motion expressed analytically are called *constraints*. In this section, we review the reaction forces/torques between two typical bodies (*i*) and (*j*) for several commonly used planar joints (or constraints).

Consider the two bodies connected by a pin joint shown in Figure 4.10a. A pin joint allows the bodies to rotate relative to one another, but it eliminates the relative translation between the attachment points along any two orthogonal axes. Since this joint removes two relative translational DoFs, the resultant reaction force on each body can be described with two orthogonal components  $\lambda_1$  and  $\lambda_2$ .



**FIGURE 4.10**

A variety of kinematic joints and their corresponding reaction forces and torques: (a) pin; (b) sliding; (c) pin-in-slot; (d) rod; (e) point contact; (f) roll with slip; (g) roll without slip.

The sliding joint shown in Figure 4.10b allows one relative translation between the two bodies along the axis of the joint. The joint eliminates any relative translation perpendicular to the joint axis and does not allow relative rotation between the bodies. This joint causes a reaction force,  $\lambda_1$ , perpendicular to the sliding axis and a reaction torque,  $\lambda_2$ .

Figure 4.10c shows a pin-in-slot joint that allows the two bodies to translate along the axis of the slot and rotate about an axis perpendicular to the plane. The joint eliminates only the relative translation of the pin perpendicular to the axis of the slot, resulting in a reaction force,  $\lambda_1$ , perpendicular to the axis of the slot.

The massless rod shown in Figure 4.10d does not allow its two end attachment points to translate along its axis. Therefore, it produces a reaction force,  $\lambda_1$ , along that axis.

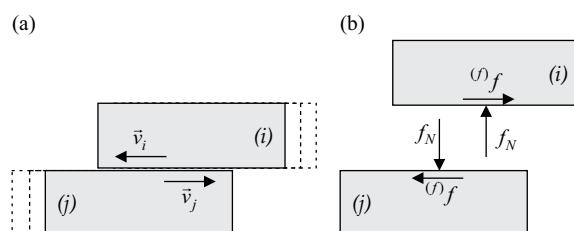
Figure 4.10e illustrates the pointed tip of body (i) in contact with the flat surface (line) of body (j). The tip is allowed to slide without losing contact, and the two bodies are allowed to rotate relative to each other. This point contact constraint results into a reaction force,  $\lambda_1$ , perpendicular to the line of contact.

A circular disc may roll against the flat surface (line) of another object or against another disc without losing contact, as shown in Figure 4.10f. If the two objects are allowed to slip (slide) relative to each other, then there is only a reaction force,  $\lambda_1$ , perpendicular to the line (or the tangent line) of contact. If no slippage is allowed between the two bodies, another DoF is removed. This condition is illustrated in Figure 4.10g, where a second component of reaction force,  $\lambda_2$ , must also be considered.

## 4.5 Friction Force

Friction force may be imposed between two contacting bodies to oppose their relative motion. It can be a function of the material of the contacting bodies, the shape and roughness of the contacting surfaces, relative velocity, normal force, lubrication, and other factors. Therefore, determining the force caused by all possible forms of friction in one formula is not possible.

In general, friction is known either as *Coulomb* or *viscous*, or as a combination of the two. Consider the two bodies being in contact as shown in Figure 4.11. Let us assume that each body applies a normal force  $f_N$  on the other causing a pair of friction forces,  $(^f)\vec{f}$ , opposing the relative motion. *Coulomb* friction, also known as the *dry* friction, may exist even if the two surfaces do not move relative to each other. If the relative motion between the two bodies is zero, the Coulomb friction is computed as



**FIGURE 4.11**

(a) Two bodies in contact; (b) the resultant normal and reaction forces.

$${}^{(sf)}f = \mu_s f_N \quad (4.39)$$

where  $\mu_s$  is the *coefficient of static friction*. When the relative motion is nonzero, the friction force is determined as

$${}^{(df)}f = \mu_d f_N \quad (4.40)$$

where  $\mu_d$  is the *coefficient of dynamic (or kinetic) friction* (normally  $\mu_d < \mu_s$ ). Assuming that the dynamic friction is not a function of the relative velocity, and the transition from static to dynamic friction is instantaneous, Eqs. (4.39) and (4.40) can be presented together graphically as shown in Figure 4.12a.

Viscous friction, also known as the *wet* or *lubricated* friction, is proportional to the relative velocity of the contacting surfaces. This type of friction can be described in its simplest form as

$${}^{(vf)}f = \mu_v v_{i,j} \quad (4.41)$$

where  $\mu_v$  is the *coefficient of viscous friction*, and  $v_{i,j}$  is the magnitude of the relative velocity  $\vec{v}_{i,j} = \vec{v}_i - \vec{v}_j$ , or *relative speed*. If the coefficient of viscous friction is not a constant and is a function of the normal force, then the viscous friction force may be determined as

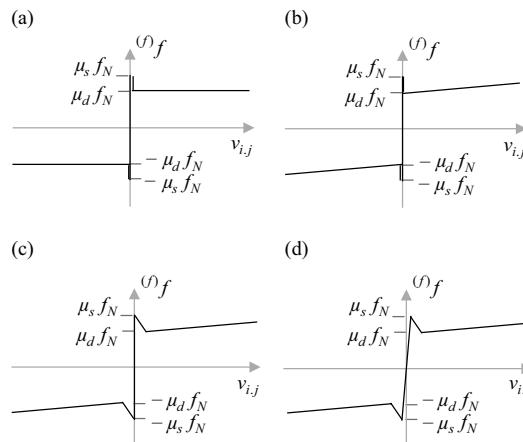
$${}^{(vf)}f = k_v v_{i,j} f_N \quad (4.42)$$

where  $k_v$  is another description of the coefficient of viscous friction.

When friction is assumed to be a combination of both Coulomb and viscous frictions, the combined models can be presented as shown in Figure 4.12b, whereas the relative velocity increases so does the friction force.

It has been found experimentally that the transition of friction force from zero to non-zero relative velocity is not instantaneous, but it takes place during a short period of time. This transition, known as the *Stribeck effect*, is depicted in Figure 4.12c.

Implementation of Coulomb and viscous frictions, including the Stribeck effect, in the equations of motion of a multibody system is not difficult as long as the relative velocity of



**FIGURE 4.12**

(a) Coulomb friction with static and dynamic coefficients; (b) addition of viscous friction; (c) Stribeck effect; (d) approximate representation of static friction.

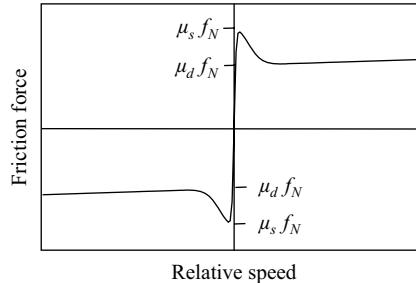
the contacting bodies is not zero. The difficulty arises when the relative velocity becomes zero (the *stiction* phase) during which the static friction force could be anywhere in the interval of  $-\mu_s f_N \leq {}^{(sf)}f \leq \mu_s f_N$ . To remedy this problem, the static friction force can be assumed to vary from  $-\mu_s f_N$  to  $\mu_s f_N$  during a short range of varied relative velocities in the vicinity of stiction, as shown in Figure 4.12d.

A variety of analytical formulas to closely represent the force–velocity characteristics of the function of Figure 4.12d can be found in literature. One example is the following function [1]:

$${}^{(f)}f = f_N \left( \mu_d + (\mu_s - \mu_d) e^{-\left(\frac{v_{i,j}}{v_s}\right)^p} \right) \tanh(k_t v_{i,j}) + {}^{(vf)}f \quad (4.43)$$

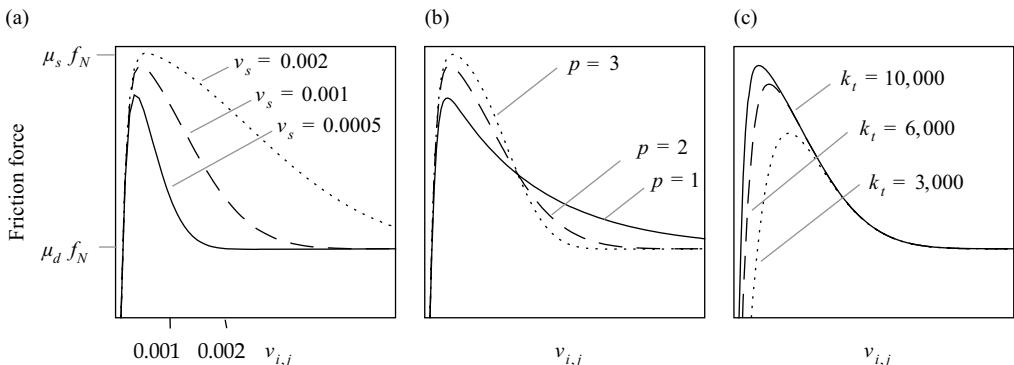
The general shape of this function is shown in Figure 4.13. The viscous friction in this formula can be defined as either Eq. (4.41) or Eq. (4.42).

In addition to the coefficients of friction and the relative speed, Eq. (4.43) contains other parameters. Each of these parameters can change the shape of the function. The coefficient of *sliding speed*,  $v_s$ , changes the shape of the decay in the Stribeck region as shown in Figure 4.14a, where a smaller value for this parameter results in a sharper drop from the



**FIGURE 4.13**

The friction force versus relative speed as described by Eq. (4.43).



**FIGURE 4.14**

Effects of three parameters on the friction force curve: (a) the coefficient of sliding speed  $v_s$ ; (b) the exponent  $p$ ; (c) the parameter  $k_t$ .

static to dynamic friction. The exponent  $p$  affects the drop from the static to dynamic friction in a different form as shown in Figure 4.14b. The parameter  $k_t$  adjusts the slope of the curve from zero relative speed to the maximum static friction, as depicted in Figure 4.14c.

The following is a function M-file based on the friction model of Eq. (4.43) without the inclusion of viscous friction.

```
function ff = Friction_A(mu_s, mu_d, v_s, p, k_t, v, fN)
% Friction force (Anderson et al. model) w/o viscous friction
ff = fN*(mu_d + (mu_s - mu_d)*exp(-(abs(v)/v_s)^p))* ...
tanh(k_t*v);
```

Typical values for the three parameters are  $v_s = 0.001 \text{ m/s}$ ,  $p = 2$ , and  $k_t = 10,000$ . A large value for  $k_t$  makes the slope of the curve in the static friction region almost vertical. We should also note that with this friction model, the maximum value of the static friction force does not reach the desired value  $\mu_s f_N$ . Therefore, we may consider a larger value for  $\mu_s$  to compensate for this shortcoming of the model.

Another example of an analytical formula that closely represents the force–velocity characteristics of the function of Figure 4.12d is as follows [2]:

$$^{(f)}f = f_N \left( \mu_d \tanh(4v_r) + (\mu_s - \mu_d) \frac{16v_r}{(v_r^2 + 3)^2} \right) + ^{(vf)}f \quad (4.44)$$

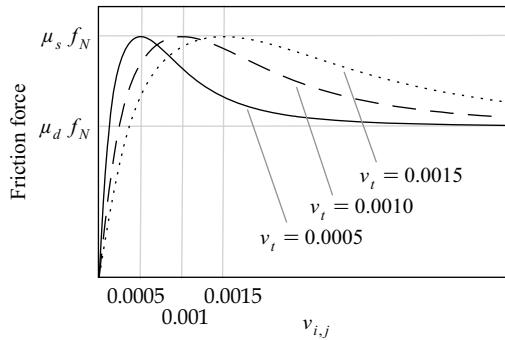
where  $v_r = \frac{v_{i,j}}{v_t}$  and  $v_t$  is the *transition velocity* indicating the velocity at which the friction force reaches its maximum value and the transition from the static to dynamic region begins. The model also suggests that the viscous friction should be represented as

$$^{(vf)}f = \mu_v v_{i,j} \tanh\left( 4 \frac{f_N}{f_{nt}} \right) \quad (4.45)$$

where the parameter  $f_{nt}$ , called the *transition force*, specifies the required minimum normal force for viscous friction to take place. However, the viscous friction in this formula can also be defined as either Eq. (4.41) or Eq. (4.42). Typical representations of the friction model of Eq. (4.44) for different values of the parameter  $v_t$  are shown in Figure 4.15. We note that the transition from the static to dynamic friction occurs at the specified transition velocity, at which the friction force has reached the limit of the static force  $\mu_s f_N$ .

The following is a function M-file based on the friction model of Eqs. (4.44) and (4.45):

```
function ff = Friction_B(mu_s, mu_d, v_s, v_t, fnt, v, fN)
% Friction force (Brown-McPhee model) w/o viscous friction
vr = v/v_t;
ff = fN*(mu_d*tanh(4*vr) + ... 
(mu_s - mu_d)*vr/(0.25*vr^2 + 0.75)^2) + ...
mu_v*v*tanh(4*fN/fnt);
```



**FIGURE 4.15**  
Typical friction force–speed curve for the model of Eq. (4.44).

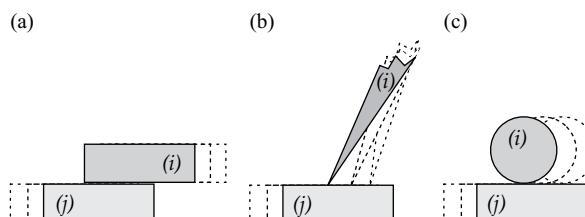
A friction model can be considered between bodies of different shapes, such as the examples shown in Figure 4.16. If the normal force  $f_N$  is the result of a contact represented as a penetration model (such models are discussed in Section 11.2), the friction force could be categorized as an applied force. However, if the contact is modeled as a constraint, for which a Lagrange multiplier represents the resulting contact force, the corresponding friction could be categorized as a reaction force.

#### 4.5.1 Wheel and Tire

One of the more interesting applications of multibody dynamics is to simulate the ride comfort or handling performance of vehicles. For such simulations, we need a reasonably accurate model of the wheel and tire. In a simple model, we may ignore the tire compliance completely and consider a wheel/tire as a rigid rolling disc. For such a model, we need to define an appropriate constraint to keep the disc in contact with the ground and to impose a no-slip condition (refer to Section 7.2.7). In a more realistic model, we need to consider the compliance of a tire in both radial and longitudinal directions.

The radial compliance can be represented as a spring–damper shown in Figure 4.17a. The force of this spring–damper can easily be determined based on the position and velocity of the wheel center relative to the ground in the radial direction ( $y$  and  $\dot{y}$ ), the undeformed radius ( $R$ ), and the radial stiffness and damping of the tire ( $k_{radial}$  and  $d_{radial}$ ):

$$f_N = k_{radial}(y - R) + d_{radial}\dot{y} \quad (4.46)$$



**FIGURE 4.16**  
Examples of contact with or without friction between different objects.

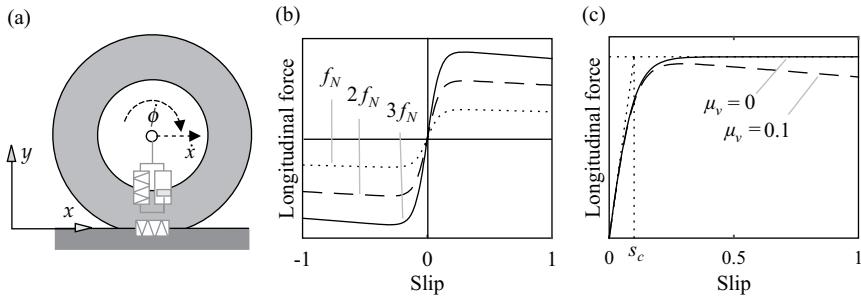


FIGURE 4.17

(a) Spring–damper elements representing radial and longitudinal tire compliance; (b) a simplified longitudinal force–slip characteristics; (c) visual description of a critical slip.

The longitudinal compliance can be described as a spring element in the longitudinal direction, as shown in Figure 4.17a. This element applies a force on the wheel at the contact patch tangent to the ground. The longitudinal force can be determined from a set of force–slip curves that are either obtained experimentally by tire manufacturers or computed (approximated) analytically. To define a longitudinal slip, or simply *slip* in planar motion, assume that  $v_x = |\dot{x}|$  is the magnitude of the forward velocity of the center of the wheel, and  $v_c = R_d |\dot{\phi}|$  is the magnitude of the circumferential velocity of the wheel, where  $R_d = y$  is the *effective* or deformed radius of the wheel and  $\dot{\phi}$  is the angular velocity of the wheel. If  $v_x > v_c$ , the wheel is braking, and if  $v_c > v_x$ , the wheel is in traction. These two velocities are used to define the slip as follows:

$$\text{Braking} : s = \frac{v_x - v_c}{v_x} > 0, \quad 0 < s \leq 1 \quad (4.47)$$

$$\text{Traction} : s = \frac{v_c - v_x}{v_c} < 0, \quad -1 \leq s < 0 \quad (4.48)$$

With this definition of slip, a set of force–slip curves can be obtained experimentally, similar to the curves shown in Figure 4.17b, where the traction and braking curves are symmetric with respect to the two axes. We note that as the radial force ( $f_N$ ) increases, so does the longitudinal force.

There are a variety of analytical methods available to represent the experimentally obtained longitudinal force of a tire. However, since the discussion of such methods is outside the scope of this textbook, a simple formula to approximate the force–slip curves is provided here as

$$f_L = f_N \left( \mu_d \tanh\left(\frac{s}{s_c}\right) - \mu_v s \right) \quad (4.49)$$

This formula, which is stated for braking ( $0 < s \leq 1$ ), has similarities to the friction formulas of Eqs. (4.43) and (4.44). In the formula,  $\mu_d$  and  $\mu_v$  are the dynamic and viscous friction coefficients, and  $s_c$  is known as the *critical slip*. A visual description of the critical slip, as shown in Figure 4.17c, indicates that the longitudinal force increases almost linearly for  $0 < s < s_c$  (due to longitudinal deformation), and then it remains unchanged for  $s_c < s < 1$ .

However, in the presence of viscous friction, the longitudinal force drops slightly as the slip increases. For traction, the signs for slip, the critical slip, and the longitudinal force must be adjusted accordingly. The longitudinal force  $f_L$  acts on a wheel at its contact point, tangent to the ground.

The dynamic coefficient of friction can be selected in the range of  $0.9 < \mu_d < 1.3$ . For most tires, the critical slip can be set to  $s_c = 0.1$ .

The force of a tire should be considered as an applied force since both the radial and longitudinal components are mainly due to the deformation of the tire.

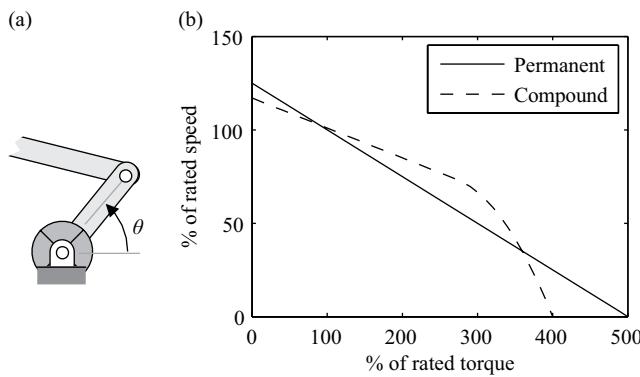
#### 4.5.2 Motor and Driver

A rotary motor can be included in a multibody model between two bodies about the axis of a pin joint. In most applications, one of the bodies is the ground as shown in Figure 4.18a. A rotary motor applies a pair of torques on the two bodies that are equal in magnitude and opposite in direction.

An idealistic representation of a motor with a known torque would be for the motor to generate a constant torque or a torque being a function of time. For example, the torque may follow a nonlinear function:

$$T = T(t)$$

In a more realistic representation of a motor, we should consider its torque–speed characteristics. For example, Figure 4.18b shows typical torque–speed curves for two DC motors [3]: the linear line is for a *permanent magnet* motor, and the nonlinear curve is for a *compound wound* motor. Either function shows that the torque varies with speed—as the speed increases, the torque drops, and vice versa. Whether the torque is assumed to be a constant or a function of time, or to follow known torque–speed characteristics, since the amount of torque can be determined at any given time and speed, we may consider such models in the category of *applied force/torque*.



**FIGURE 4.18**

(a) A motor rotating a link of a multibody system; (b) typical torque–speed characteristics for permanent magnet and compound wound DC motors.

In some applications, we may expect a specific motion from a motor without knowing the required torque. As an example, assume that we require the speed of the motor to remain constant at  $2\pi$  rad/s, as it rotates the link of a multibody system. This condition can be expressed in algebraic form on the angle that the link makes with the ground as

$$\theta = {}^0\theta + 2\pi t \quad (4.50)$$

where  ${}^0\theta$  is the initial angle of the link. The first and second time derivatives of this expression yield the desired angular speed  $\dot{\theta} = 2\pi$  rad/s and the angular acceleration  $\ddot{\theta} = 0$ .

A condition on the motion of a body, such as the one in Eq. (4.50), will be referred to as a *driver constraint*. A driver constraint, in general, is a simple equation that explicitly describes a coordinate (or possibly two) as a function of *time*. Because the torque of such a motor is unknown, we categorize this type of *driver* motor as a *reaction* force/torque.

---

## 4.6 Work and Energy

The work of a force  $\vec{f}$  during a displacement  $\vec{d}$  is defined as

$$U = \mathbf{f}' \mathbf{d} \quad (4.51)$$

This definition of work yields the definitions of kinetic and potential energies, which are briefly discussed in this section. Computing and monitoring the total energy can be useful in the analysis of some multibody systems.

The *kinetic energy* of a particle with a mass  $m^i$  and a velocity  $\dot{\vec{r}}^i$  is defined as

$$T^i = \frac{1}{2} m_i \dot{\vec{r}}'_i \cdot \dot{\vec{r}}_i \quad (4.52)$$

For a rigid body with a mass  $m_i$ , a moment of inertia  $J_i$ , and a velocity  $\dot{\vec{r}}_i$ , the *kinetic energy* is determined as

$$T_i = \frac{1}{2} (m_i \dot{\vec{r}}'_i \cdot \dot{\vec{r}}_i + J_i \dot{\phi}_i^2) \quad (4.53)$$

The *gravitational potential energy* of a body of mass  $m_i$  at a height  $h$  is defined as

$${}^{(g)}V = m_i g h \quad (4.54)$$

Since the height  $h$  is the distance from a reference plane, we can restate Eq. (4.54) as the scalar product of two vectors as

$${}^{(g)}V = \mathbf{w}'_i \mathbf{r}_i \quad (4.55)$$

where  $\mathbf{w}_i$  is the weight and  $\mathbf{r}_i$  is the position of the mass center from the origin of the  $x$ - $y$  frame. We can also express a similar equation for a *constant* force (constant magnitude and axis) that acts on point  $P$  of body ( $i$ ):

$${}^{(f)}V = \mathbf{f}' \mathbf{r}_i^P \quad (4.56)$$

For a linear point-to-point spring causing a pair of forces as described in Eq. (4.26), the stored *elastic potential* energy (or *strain energy*) is determined as

$${}^{(s)}V = \frac{1}{2}k(L - {}^0L)^2 \quad (4.57)$$

Similarly, for a linear torsional spring, as defined in Eq. (4.33), the stored strain energy is computed as

$${}^{(r-s)}V = \frac{1}{2}{}^{(r)}k(\theta - {}^0\theta)^2 \quad (4.58)$$

The total energy of a multibody system is the sum of the kinetic and potential energies of all its components. If the total energy of a system during motion remains unchanged, the system is referred to as being *conservative*. Monitoring the total energy of a conservative system during a computational simulation is one method of determining if there are any obvious errors in the constructed equations of motion.

---

## 4.7 Problems

- 4.1 Based on the definition of the mass center of a system of particles, derive the first moment equation given in Eq. (4.7).
- 4.2 The data for a system of four particles are given as follows:

$$\mathbf{r}^1 = \begin{Bmatrix} -1 \\ 3 \end{Bmatrix}, \mathbf{r}^2 = \begin{Bmatrix} 2 \\ -2 \end{Bmatrix}, \mathbf{r}^3 = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}, \mathbf{r}^4 = \begin{Bmatrix} -4 \\ -1 \end{Bmatrix}$$

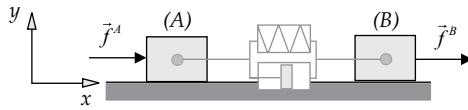
$$\dot{\mathbf{r}}^1 = \begin{Bmatrix} 2 \\ -0.5 \end{Bmatrix}, \dot{\mathbf{r}}^2 = \begin{Bmatrix} -1.5 \\ 1.2 \end{Bmatrix}, \dot{\mathbf{r}}^3 = \begin{Bmatrix} 1 \\ -1.2 \end{Bmatrix}, \dot{\mathbf{r}}^4 = \begin{Bmatrix} -2.1 \\ 0 \end{Bmatrix}$$

$$m^1 = 2, m^2 = 5, m^3 = 1.6, m^4 = 2.7$$

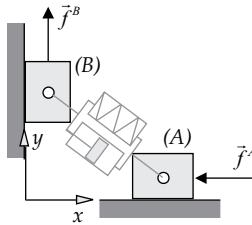
$$\mathbf{f}^1 = \begin{Bmatrix} 1 \\ -2 \end{Bmatrix}, \mathbf{f}^2 = \begin{Bmatrix} 3 \\ 0 \end{Bmatrix}, \mathbf{f}^3 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \mathbf{f}^4 = \begin{Bmatrix} -3 \\ 4 \end{Bmatrix}$$

Revise the MATLAB program in Example 4.1 to determine the following:

- a. Position of the mass center
  - b. Velocity of the mass center
  - c. Acceleration of the mass center
  - d. Position vector for each particle with respect to the mass center
- 4.3 Two particles that can only slide along the  $x$ -axis are connected by a spring–damper. External forces  $f^A = 35$  and  $f^B = 35$  N act on the particles as shown. Derive the equations of motion for the system. Consider the following data:  $m^A = 1$ ,  $m^B = 2$  kg,  $k = 800$  N/m,  ${}^0L = 0.2$  m,  $d_c = 15$  N s/m. In the given configuration, the coordinates and velocities are as follows:  $x^A = 0.15$ ,  $x^B = 0.33$  m;  $\dot{x}^A = 30$ ,  $\dot{x}^B = 40$  m/s.



- 4.4 Two particles,  $A$  and  $B$ , are connected by a spring–damper. Particle  $A$  slides along the  $x$ -axis and particle  $B$  slides along the  $y$ -axis. External forces  $f^A = 15$  and  $f^B = 5$  N act on the particles as shown. Derive the equations of motion for the system. Consider the following data:  $m^A = 10$ ,  $m^B = 20$  kg;  $k = 150$  N/m,  ${}^0L = 6$  m,  $d_c = 300$  N·s/m. In the given configuration, the coordinates and velocities are as follows:  $x^A = 4$ ,  $y^B = 3$  m;  $\dot{x}^A = 0.1$ ,  $\dot{y}^B = -0.2$  m/s.

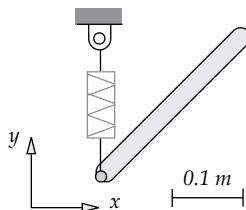


- 4.5 Two forces act on a body at points  $A$  and  $B$ . The position vector and the components of each force with respect to the body reference frame are given as

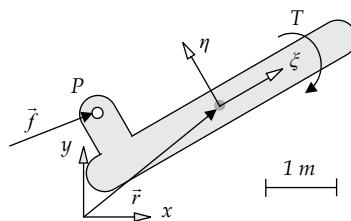
$$\mathbf{s}^A = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}, \mathbf{f}^A = \begin{Bmatrix} 3 \\ 1 \end{Bmatrix}, \mathbf{s}^B = \begin{Bmatrix} -3 \\ 2 \end{Bmatrix}, \mathbf{f}^B = \begin{Bmatrix} -2 \\ 1 \end{Bmatrix}$$

The rotational coordinate of the body is  $\phi = -35^\circ$ .

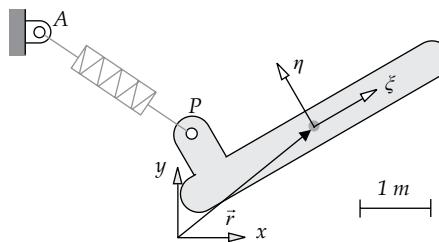
- Without transforming the position and force vectors to the global frame, compute the sum of the moments as the result of the two forces.
  - Transform the position and force vectors to the global frame, then compute the sum of the moments as the result of the two forces.
  - Compare the results from (a) and (b).
- 4.6 The rod shown has a length of 0.3 m and is attached to the ground by a spring. In the shown configuration, the deformed length of the spring is  $L = 0.2$  m. Assume that the mass center is at the rod's geometric center,  $m = 4$  kg,  $J = 3$  kg·m<sup>2</sup>,  $k = 40$  N/m, and  ${}^0L = 0.15$  m. Write the equations of motion for the rod. Take direct measurements from the figure for any needed dimensions.



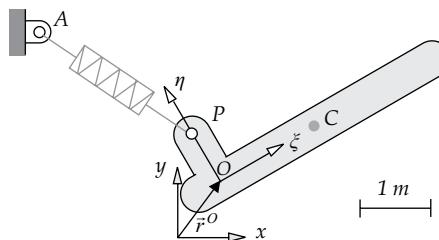
- 4.7 The single body shown has a mass of 5.0 kg and a moment of inertia of  $3.5 \text{ kg.m}^2$ . Gravity, a constant force at point  $P$ , and a torque act upon the body as shown. Point  $P$  has the local coordinates  $\mathbf{s}^P = \begin{Bmatrix} -1.6 & 0.8 \end{Bmatrix}' \text{ m}$ . The components of the force are  $\mathbf{f} = \begin{Bmatrix} 1.2 & 0.5 \end{Bmatrix}' \text{ N}$ , and the magnitude of the torque is 0.8 N m (clockwise). In the shown configuration, the body has the following coordinates:  $\mathbf{r} = \begin{Bmatrix} 2.0 & 1.6 \end{Bmatrix}' \text{ m}$ ,  $\phi = 30^\circ$ .
- Construct the equations of motion for this body.
  - Compute the accelerations.



- 4.8 Problem 4.7 is revised by removing the gravity, the external constant force, and the applied torque. Instead, a spring is considered between points  $P$  and  $A$  as shown. The coordinates of point  $A$  are  $\mathbf{r}^A = \begin{Bmatrix} -2.0 & 3.0 \end{Bmatrix}' \text{ m}$ . The spring has an undeformed length of  ${}^0L = 1.4 \text{ m}$  and a stiffness of  $6.0 \text{ N/m}$ .
- Construct the equations of motion.
  - Compute the accelerations.



- 4.9 The body-fixed frame in the system of Problem 4.8 is relocated to point  $O$  as shown, where  $\mathbf{r}^O = \begin{Bmatrix} 0.6144 & 0.8 \end{Bmatrix}' \text{ m}$ . The coordinates of the mass center and point  $P$  in this new frame are  $\mathbf{s}^{C,O} = \begin{Bmatrix} 1.6 & 0 \end{Bmatrix}' \text{ m}$  and  $\mathbf{s}^{P,O} = \begin{Bmatrix} 0 & 0.8 \end{Bmatrix}' \text{ m}$ . Assume all the velocities to be zeros.



- a. Construct the noncentroidal equations of motion.  
 b. Compute the accelerations.
- 4.10 Revise Problem 4.8 by including a damper with the spring element. Assume a damping coefficient  $d_c = 2.0 \text{ N}\cdot\text{s}/\text{m}$  and velocities  $\dot{\mathbf{r}} = \begin{Bmatrix} 0.1 & -0.3 \end{Bmatrix}' \text{ m/s}$ ,  $\dot{\phi} = 0.5 \text{ rad/s}$  counterclockwise for the body.
- a. Construct the equations of motion.  
 b. Compute the accelerations.
- 4.11 The single body shown has a mass of  $5.0 \text{ kg}$  and a moment of inertia of  $3.5 \text{ kg}\cdot\text{m}^2$ . In the shown configuration, the coordinates and velocity of the body are as follows:

$$\mathbf{r} = \begin{Bmatrix} 2.0 & 1.6 \end{Bmatrix}' \text{ m}, \phi = 30^\circ, \dot{\mathbf{r}} = \begin{Bmatrix} 0.1 & -0.3 \end{Bmatrix}' \text{ m/s}, \dot{\phi} = 0.5 \text{ rad/s}$$

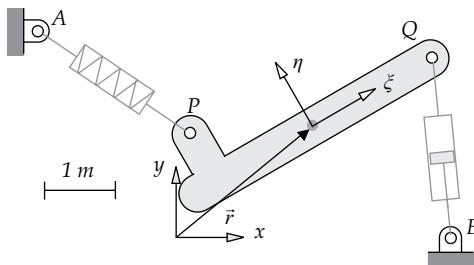
A spring is connected between points  $P$  and  $A$ , and a damper between points  $Q$  and  $B$ . The coordinates of these points are as follows:

$$\mathbf{s}^P = \begin{Bmatrix} -1.6 & 0.8 \end{Bmatrix}', \mathbf{s}^Q = \begin{Bmatrix} 2.0 & 0 \end{Bmatrix}', \mathbf{r}^A = \begin{Bmatrix} -2.0 & 3.0 \end{Bmatrix}', \mathbf{r}^B = \begin{Bmatrix} 4.0 & 0 \end{Bmatrix}' \text{ m}$$

The spring and the damper have the following characteristics:

$$k = 6.0 \text{ N/m}, {}^0L = 1.4 \text{ m}, d_c = 2.0 \text{ N}\cdot\text{s}/\text{m}$$

- a. Construct the equations of motion for this body.  
 b. Compute the accelerations.



- 4.12 In this problem, we make a comparison between the friction formulas of Eqs. (4.43) and (4.44) around the region of static friction and the Stribeck effect. Assume the coefficients of friction  $\mu_s = 0.8$ ,  $\mu_d = 0.5$ , and  $\mu_v = 0$ , and the normal force is  $f_N = 1.0 \text{ N}$ . Use the function M-files Friction\_A and Friction\_B, and compute the friction force from these two models for values of relative velocities starting from  $v_{i,j} = 0$  to  $v_{i,j} = 0.005 \text{ m/s}$  in increments of  $\Delta v_{i,j} = 0.0001 \text{ m/s}$ . Plot the results.
- a. For the two friction models, consider the following parameters:

For Eq. (4.43):  $v_s = 0.001 \text{ m/s}$ ,  $p = 2$ ,  $k_t = 10,000$

For Eq. (4.44):  $v_t = 0.0005 \text{ m/s}$

- b. Keep the parameters as in (a) except for  $v_s = 0.002 \text{ m/s}$ .
  - c. Keep the parameters as in (a) except for  $v_t = 0.001 \text{ m/s}$ .
- 

## References

1. Andersson, S., Söderberg, A., and Björklund, S., Friction Models for Sliding Dry, Boundary and Mixed Lubricated Contacts, *Tribol. Int.*, 40(4), 580–587 (2007).
2. Brown, P., and McPhee, J., A Continuous Velocity-Based Friction Model for Dynamics and Control with Physically Meaningful Parameters, *ASME J. Computational and Nonlinear Dynamics*, 11(5): 054502-1–054502-6 (2016).
3. Norton, R.L., *Design of Machinery*, 5th ed. New York, McGraw-Hill (2012).

# 5

---

## Vector Kinematics

---

Prior to the advent of computers and the appearance of the subject of multibody dynamics for analyzing mechanical systems, engineers were able to analyze and design simple mechanisms using vectors and graphical procedures. Even today, such methods and procedures can be found in most textbooks on mechanisms and machineries.

By definition, mechanisms are closed-chain systems with one degree of freedom (DoF). For each closed chain of a mechanism, vectors are defined to construct a vector loop equation. Each equation provides the relationship between magnitudes and angles of the corresponding vectors. The first and second time derivatives of a vector loop equation provide the velocity and acceleration equations. The traditional method of solving these equations is the graphical pencil-and-paper approach. Well-known methods for graphical velocity and acceleration analyses are the *instant centers of velocity*, and *velocity and acceleration polygons*.

The graphical method of solving vector loop equations is time consuming, and its accuracy is limited to the drawing and measurement errors. Analytical solution of vector loop equations by pencil and paper, in particular at the position level, is complicated due to the nonlinear nature of the equations. However, with a computer program, such nonlinear equations can be solved efficiently and accurately using numerical methods.

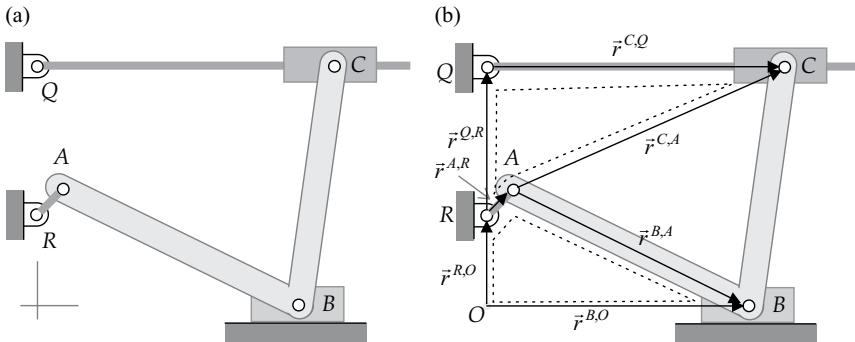
In this chapter, through several simple examples, we provide a review of the vector loop method for kinematics of planar mechanisms. We limit the discussion of the solution methods only to analytical and numerical procedures. Prior to discussing the vector loop method for closed-chain systems, we discuss a vector formulation for open-chain systems, which will become useful in some of the formulations in the upcoming chapters. Although the method of vector kinematics is applicable to most multibody problems, its use in developing general-purpose computer programs is limited. Another method known as the joint coordinates, as will be discussed in Chapter 9, will produce the same kinematic equations as the vector formulation, but in a systematic form that is suitable for general-purpose program development.

---

### 5.1 Use of Vectors

In kinematic analysis, vectors are used to describe the position, velocity, and acceleration of points and bodies. A *position* vector defines the position of one point with respect to another point, and the first and second time derivatives of the position vectors provide the velocity and acceleration vectors, respectively. Therefore, for kinematic analysis of a system, it is most important to identify and properly define all the necessary position vectors.

In closed-chain systems, particularly in mechanisms, one use of position vectors is to describe a closed chain in the form of a vector loop. As an example, consider the multi-loop

**FIGURE 5.1**

(a) A closed-chain system and (b) two of its loops are described by vectors.

system shown in Figure 5.1a. We can identify several kinematic loops in this mechanism. Two of the loops are shown in Figure 5.1b, and we have defined the necessary position vectors to describe the loops. Some of the defined vectors are stationary, some change magnitude, and some rotate when the mechanism is in motion. The points that are used for defining the vectors are the center of the pin joints, and one point,  $O$ , is defined on the ground. One vector,  $\vec{r}^{A,R}$ , is shared between the two loops. The upper loop is formed in vector form as

$$\vec{r}^{A,R} + \vec{r}^{C,A} = \vec{r}^{Q,R} + \vec{r}^{C,Q} \text{ or } \vec{r}^{A,R} + \vec{r}^{C,A} - \vec{r}^{C,Q} - \vec{r}^{Q,R} = 0$$

For the lower loop, we have

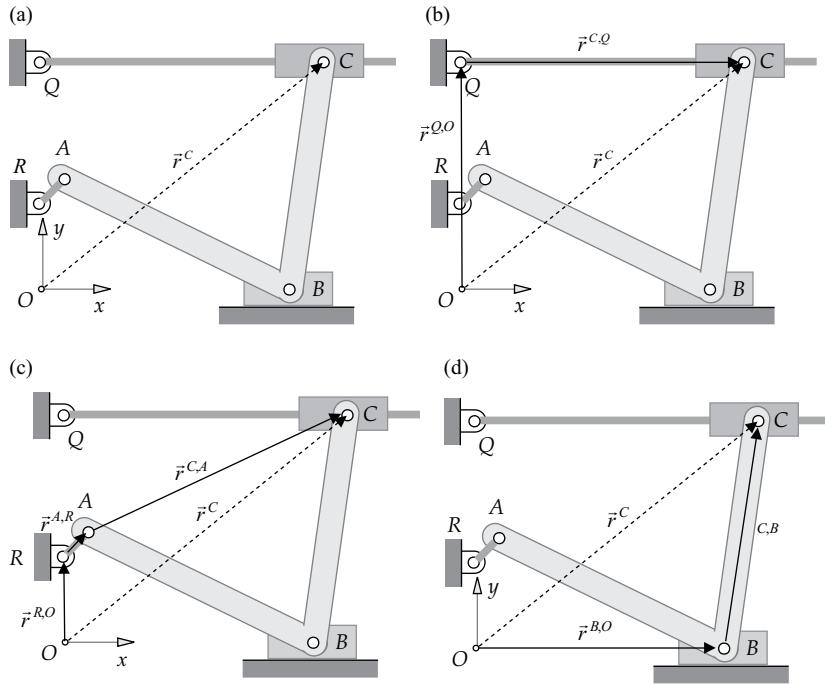
$$\vec{r}^{R,O} + \vec{r}^{A,R} + \vec{r}^{B,A} = \vec{r}^{B,O} \text{ or } \vec{r}^{R,O} + \vec{r}^{A,R} + \vec{r}^{B,A} - \vec{r}^{B,O} = 0$$

These are vector loop equations that are necessary for the kinematic analysis of this mechanism.

Another use of position vectors is for determining the coordinates of a point with respect to a defined reference frame. For example, assume that we are interested in the coordinates of point  $C$  with respect to a defined  $x$ - $y$  frame, which will be referred to as vector  $\vec{r}^C$ , as shown in Figure 5.2a. This vector can be represented as the sum of some of the position vectors that have already been defined for the mechanism. As shown in Figure 5.2b, we can set  $\vec{r}^C$  as the sum of vectors  $\vec{r}^{Q,O}$  and  $\vec{r}^{C,Q}$ , that is,  $\vec{r}^C = \vec{r}^{Q,O} + \vec{r}^{C,Q}$ . Otherwise, as shown in Figure 5.2c, we can determine  $\vec{r}^C$  as  $\vec{r}^C = \vec{r}^{R,Q} + \vec{r}^{A,R} + \vec{r}^{C,A}$ . Another possibility is  $\vec{r}^C = \vec{r}^{B,O} + \vec{r}^{C,B}$ , as shown in Figure 5.2d. The first and second time derivatives of any of these vector expressions can be used to determine the velocity and acceleration of  $C$ , denoted as  $\dot{\vec{r}}^C$  and  $\ddot{\vec{r}}^C$ , respectively.

### 5.1.1 Unit Vectors

A position vector can be described as a unit vector times the magnitude of the position vector. The angle of the position vector and its corresponding unit vector can be defined with respect to the global  $x$ -axis as an absolute angle, or with respect to another moving vector as a relative angle. Although in most problems we try to use absolute angles, in some cases a relative angle could be more useful. In either case, when we take the first

**FIGURE 5.2**

Coordinates of a point, such as C, as a sum of several position vectors can be obtained in more than one way.

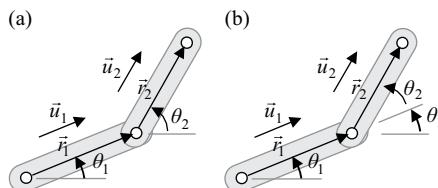
and second time derivatives of a unit vector, we need to pay attention whether its angle is absolute or relative.

Consider the two connected links shown in Figure 5.3a. The angles of the defined position vectors and their corresponding unit vectors are absolute. Therefore, their time derivatives are

$$\dot{\mathbf{u}}_i = \ddot{\mathbf{u}}_i \dot{\theta}_i, \quad \ddot{\mathbf{u}}_i = \ddot{\mathbf{u}}_i \ddot{\theta}_i - \mathbf{u}_i \dot{\theta}_i^2; \quad i = 1, 2 \quad (5.1)$$

For the same two links, one may define a relative angle for one of the links, as shown in Figure 5.3b. In this figure,  $\theta_1$  is an absolute angle, but  $\theta_2$  is defined with respect to the axis of  $\vec{r}_1$ , which means that the absolute angle of  $\vec{u}_2$  is  $\theta_1 + \theta_2$ . The time derivative  $\mathbf{u}_1$  is the same as in Eq. (5.1), but for  $\mathbf{u}_2$ , we have

$$\dot{\mathbf{u}}_2 = \ddot{\mathbf{u}}_2 (\dot{\theta}_1 + \dot{\theta}_2), \quad \ddot{\mathbf{u}}_2 = \ddot{\mathbf{u}}_2 (\ddot{\theta}_1 + \ddot{\theta}_2) - \mathbf{u}_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \quad (5.2)$$

**FIGURE 5.3**

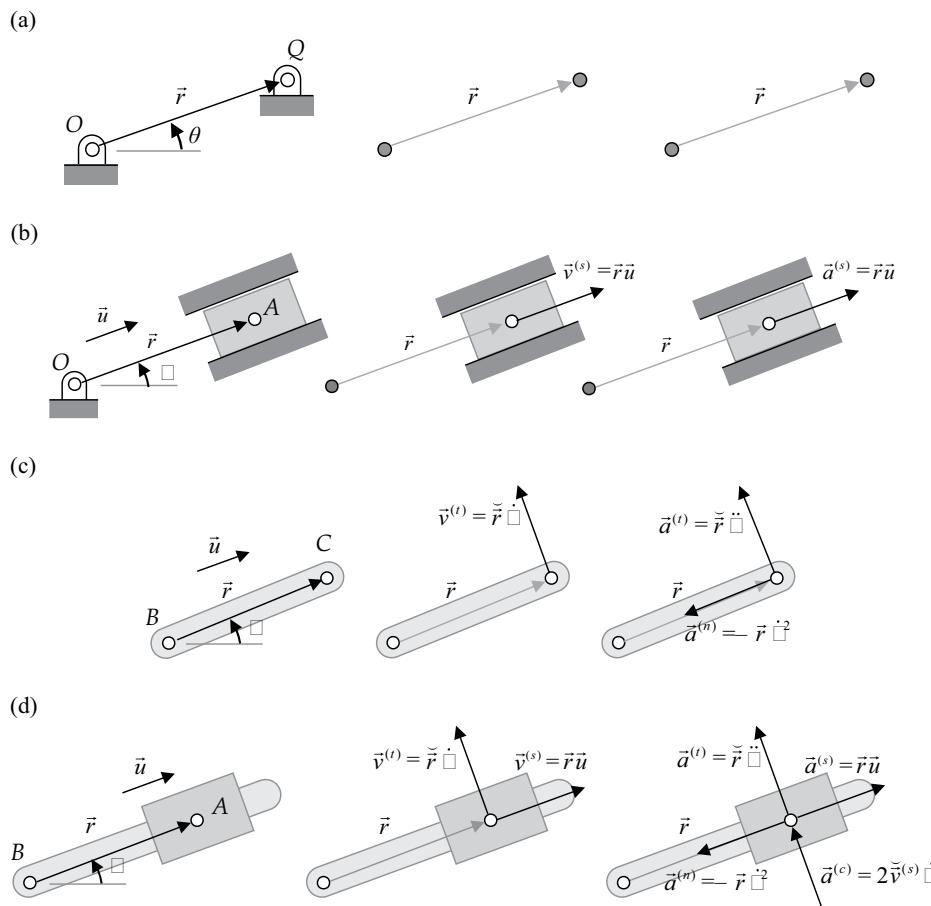
Axes of unit vectors can be defined by absolute or relative angles.

### 5.1.2 Types of Vectors

The position vectors that are used in describing the vector loop of the mechanism in Figure 5.1, or any other planar mechanism, can be categorized into four types: constant magnitude and angle, variable magnitude but constant angle, variable angle but constant magnitude, and variable magnitudes and angles. In the following discussion, we look at each type of vector and the corresponding velocity and acceleration vectors.

Vector  $\vec{r}$  shown in Figure 5.4a locates point  $Q$  with respect to point  $O$ . Since both points are fixed to the ground, this position vector has a constant magnitude and a constant angle, and therefore, its time derivatives are zero. In the examples of Figure 5.4, since each figure contains only one position vector, for notational simplification, the superscripts are omitted.

Figure 5.4b shows a vector that defines the position of a sliding block with respect to a nonmoving point. If vector  $\vec{r}$ , which is defined between points  $A$  and  $O$ , is parallel to the sliding axis, its angle remains a constant as the slider moves, but its magnitude becomes a variable. If we define a unit vector  $\vec{u}$  parallel to the position vector  $\vec{r}$ , the  $x-y$  components of vector  $\vec{r}$  and its time derivatives can be computed as



**FIGURE 5.4**

Four types of position vectors and their first and second time derivatives. For the depicted vectors, it is assumed that  $\theta$ ,  $\dot{\theta}$ ,  $\ddot{\theta}$ ,  $\vec{r}$ , and  $\ddot{r}$  are positive.

$$\begin{aligned}\mathbf{r} &= r\mathbf{u} \\ \dot{\mathbf{r}} &= \mathbf{v}^{(s)} = \dot{r}\mathbf{u} \\ \ddot{\mathbf{r}} &= \mathbf{a}^{(s)} = \ddot{r}\mathbf{u}\end{aligned}\tag{5.3}$$

where  $r$  denotes the magnitude of vector  $\vec{r}$ . We note that for this vector, both the velocity and acceleration vectors are along its axis, and therefore, they are called *slip* or *sliding* components.

The position vector  $\vec{r}$  in Figure 5.4c is defined between two points that are located on the same body. The distance between the two points is a constant, but due to the rotation of the link, the angle of the vector is a variable. The components of this vector and its time derivatives can be computed as

$$\begin{aligned}\mathbf{r} &= r\mathbf{u} \\ \dot{\mathbf{r}} &= \mathbf{v}^{(t)} = \dot{\theta}\tilde{\mathbf{r}} \\ \ddot{\mathbf{r}} &= \mathbf{a}^{(t)} + \mathbf{a}^{(n)} = \ddot{\theta}\tilde{\mathbf{r}} - \dot{\theta}^2\mathbf{r}\end{aligned}\tag{5.4}$$

The velocity vector is tangent to a circle centered at  $B$  with a radius  $r$ , which is called *tangential* component. This vector is perpendicular to the axis of the position vector. The acceleration vector contains two components: a *tangential* component perpendicular to the axis of the position vector and a *normal* component that is always in the opposite direction of the position vector.

A position vector with variable angle and variable magnitude is depicted in Figure 5.4d where the two end points of the vector are located on two different bodies that form a sliding joint. The components of the position vector  $\vec{r}$  and its time derivatives are expressed as

$$\begin{aligned}\mathbf{r} &= r\mathbf{u} \\ \dot{\mathbf{r}} &= \mathbf{v}^{(t)} + \mathbf{v}^{(s)} = \dot{\theta}\tilde{\mathbf{r}} + \dot{r}\mathbf{u} \\ \ddot{\mathbf{r}} &= \mathbf{a}^{(t)} + \mathbf{a}^{(n)} + \mathbf{a}^{(s)} + \mathbf{a}^{(c)} = \ddot{\theta}\tilde{\mathbf{r}} - \dot{\theta}^2\mathbf{r} + \ddot{r}\mathbf{u} + 2\dot{\theta}\tilde{\mathbf{v}}^{(s)}\end{aligned}\tag{5.5}$$

The velocity vector contains a *tangential* component perpendicular to  $\vec{r}$  and a *slip* component along the axis of  $\vec{r}$ . The acceleration vector consists of four components: the *tangential* and *Coriolis* components are perpendicular to  $\vec{r}$ , and the *normal* and *slip* components are along the axis of  $\vec{r}$ .

The depicted magnitudes of the velocity and acceleration vectors, and their components, in Figure 5.4 are arbitrary. For the directions of the depicted components, it is assumed that  $\theta$ ,  $\dot{\theta}$ ,  $\dot{r}$ , and  $\ddot{r}$  are positive. If any of these entities is negative, the direction of the corresponding component must be reversed, except for the normal component of acceleration  $\ddot{a}^{(n)} = -\ddot{r}\dot{\theta}^2$ , which is always in the opposite direction of the position vector  $\vec{r}$  regardless of the sign of  $\dot{\theta}$ .

In this textbook, we will refer to the variable magnitudes and angles of position vectors as *vector coordinates*. By properly defining a set of position vectors, the location and orientation of all the bodies of a multibody system should uniquely be determined.

## 5.2 Open-Chain Systems

In this section, we discuss the use of vectors in defining the position and orientation of a body in an open-chain system. In the open-chain system, if the vectors are defined properly, the total number of vector coordinates should be equal to the number of DoFs of that system. With such a set of vectors, we should be able to determine, through simple expressions, the coordinates of any point or the angle of any other vector in that system.

The sliding pendulum shown in Figure 5.5 has two DoFs. The slider-block can slide relative to the ground parallel to the  $x$ -axis, and the pendulum can rotate about a pin joint relative to the slider. We define vector  $\vec{r}^{B,Q}$ , parallel to the  $x$ -axis, to position the mass center of the block from point  $Q$  on the ground. The magnitude of this vector,  $\theta_1$ , serves as our first vector coordinate. A second vector,  $\vec{r}^{A,B}$ , can be defined along the axis of the pendulum positioning the tip of the pendulum with respect to point  $B$ . The angle of this vector,  $\theta_2$ , with respect to the  $x$ -axis (or any other well-defined axis) serves as the second vector coordinate. With these two vector coordinates, the kinematics of this system can be studied. For example, to determine the coordinates of point  $A$  in the  $x-y$  frame, we can write the following vector expression:

$$\vec{r}^{A,O} = \vec{r}^{Q,O} + \vec{r}^{B,Q} + \vec{r}^{A,B}$$

In algebraic form, this expression is expressed as

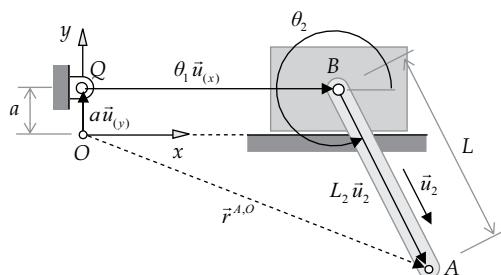
$$\mathbf{r}^{A,O} = \mathbf{r}^{Q,O} + \mathbf{r}^{B,Q} + \mathbf{r}^{A,B} \Rightarrow \mathbf{r}^{A,O} = a\mathbf{u}_{(y)} + \theta_1\mathbf{u}_{(x)} + L\mathbf{u}_2 \quad (5.6)$$

where  $\mathbf{u}_2 = [\cos \theta_2 \quad \sin \theta_2]'$ . The first and second time derivatives of Eq. (5.6) yield the velocity and acceleration of point  $A$ :

$$\dot{\mathbf{r}}^{A,O} = \dot{\theta}_1\mathbf{u}_{(x)} + L\dot{\mathbf{u}}_2 = \dot{\theta}_1\mathbf{u}_{(x)} + L\dot{\theta}_2\check{\mathbf{u}}_2 \quad (5.7)$$

$$\ddot{\mathbf{r}}^{A,O} = \ddot{\theta}_1\mathbf{u}_{(x)} + L\ddot{\theta}_2\check{\mathbf{u}}_2 - L\dot{\theta}_2^2\mathbf{u}_2 \quad (5.8)$$

For known values of  $\theta_1$  and  $\theta_2$ , and their first and second time derivatives, Eqs. (5.6)–(5.8) provide the coordinates, velocity, and acceleration of point  $A$ .



**FIGURE 5.5**

Defining translational and rotational coordinates based on the DoF.

As a second example, we consider the triple pendulum shown in Figure 5.6. In this system, the middle pendulum is pinned to the ground, and two side pendulums are attached to the “T-section” of the middle pendulum by two additional pin joints. Our objective is to define the minimum number of vector coordinates to study the kinematics of the mass centers of the three pendulums.

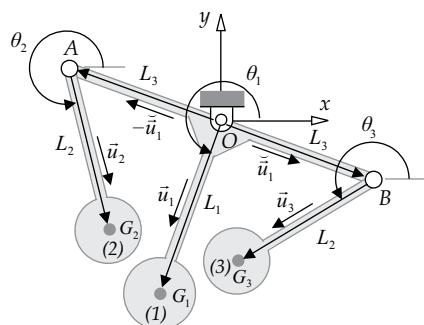
We construct five vectors as shown, where only three angles,  $\theta_1$ ,  $\theta_2$ , and  $\theta_3$ , are needed to define the orientation of all five vectors. We note that vectors  $\vec{r}^{A,O}$  and  $\vec{r}^{B,O}$  are perpendicular to  $\vec{r}^{G_1,O}$ , and remain perpendicular as the system moves. We construct three unit vectors along the arms of the three pendulums, and then write the following expressions for the coordinates of the three mass centers:

$$\begin{aligned}\mathbf{r}^{G_1,O} &= L_1 \mathbf{u}_1 \\ \mathbf{r}^{G_2,O} &= -L_3 \check{\mathbf{u}}_1 + L_2 \mathbf{u}_2 \\ \mathbf{r}^{G_3,O} &= L_3 \check{\mathbf{u}}_1 + L_2 \mathbf{u}_3\end{aligned}\tag{5.9}$$

The first and second time derivatives of these expressions provide the velocity and acceleration of the mass centers:

$$\begin{aligned}\dot{\mathbf{r}}^{G_1,O} &= L_1 \check{\mathbf{u}}_1 \dot{\theta}_1 \\ \dot{\mathbf{r}}^{G_2,O} &= L_3 \mathbf{u}_1 \dot{\theta}_1 + L_2 \check{\mathbf{u}}_2 \dot{\theta}_2 \\ \dot{\mathbf{r}}^{G_3,O} &= -L_3 \mathbf{u}_1 \dot{\theta}_1 + L_2 \check{\mathbf{u}}_3 \dot{\theta}_3 \\ \ddot{\mathbf{r}}^{G_1,O} &= L_1 \check{\mathbf{u}}_1 \ddot{\theta}_1 - L_1 \mathbf{u}_1 \dot{\theta}_1^2 \\ \ddot{\mathbf{r}}^{G_2,O} &= L_3 \mathbf{u}_1 \ddot{\theta}_1 + L_3 \check{\mathbf{u}}_1 \dot{\theta}_1^2 + L_2 \check{\mathbf{u}}_2 \ddot{\theta}_2 - L_2 \mathbf{u}_2 \dot{\theta}_2^2 \\ \ddot{\mathbf{r}}^{G_3,O} &= -L_3 \mathbf{u}_1 \ddot{\theta}_1 - L_3 \check{\mathbf{u}}_1 \dot{\theta}_1^2 + L_2 \check{\mathbf{u}}_3 \ddot{\theta}_3 - L_2 \mathbf{u}_3 \dot{\theta}_3^2\end{aligned}\tag{5.10}$$

We note that since this system has three DoFs, we only need three vector coordinates to construct the necessary expressions.



**FIGURE 5.6**

Angles are defined with respect to the  $x$ -axis.

The angles that we defined for the three pendulums in Figure 5.6 are *absolute* angles since they use the  $x$ -axis as their reference. Any of these angles could have been defined with respect to other well-defined axes. For example, for the two side pendulums, the angles  $\theta_2$  and  $\theta_3$  can be measured *relative* to the axis of the crossbar of pendulum 1, as shown in Figure 5.7, while leaving  $\theta_1$  as an absolute angle. The  $x$ - $y$  components of the three unit vectors are now computed as follows:

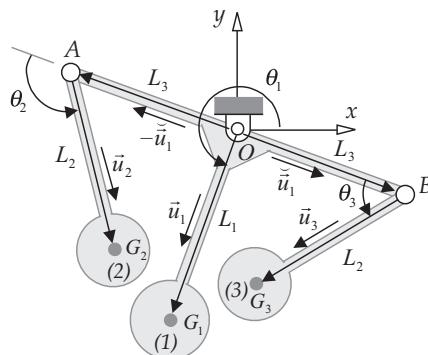
$$\mathbf{u}_1 = \begin{Bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{Bmatrix}, \quad \mathbf{u}_2 = \begin{Bmatrix} \cos(\theta_1 - \frac{\pi}{2} + \theta_2) \\ \sin(\theta_1 - \frac{\pi}{2} + \theta_2) \end{Bmatrix}, \quad \mathbf{u}_3 = \begin{Bmatrix} \cos(\theta_1 - \frac{\pi}{2} + \theta_3) \\ \sin(\theta_1 - \frac{\pi}{2} + \theta_3) \end{Bmatrix}$$

The expressions for the coordinates of the three mass centers are the same as those in Eq. (5.9). However, the velocity and acceleration expressions become

$$\begin{aligned} \dot{\mathbf{r}}^{G_1,O} &= L_1 \check{\mathbf{u}}_1 \dot{\theta}_1 \\ \dot{\mathbf{r}}^{G_2,O} &= L_3 \mathbf{u}_1 \dot{\theta}_1 + L_2 \check{\mathbf{u}}_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{\mathbf{r}}^{G_3,O} &= -L_3 \mathbf{u}_1 \dot{\theta}_1 + L_2 \check{\mathbf{u}}_3 (\dot{\theta}_1 + \dot{\theta}_3) \end{aligned} \quad (5.12)$$

$$\begin{aligned} \ddot{\mathbf{r}}^{G_1,O} &= L_1 \check{\mathbf{u}}_1 \ddot{\theta}_1 - L_1 \mathbf{u}_1 \dot{\theta}_1^2 \\ \ddot{\mathbf{r}}^{G_2,O} &= L_3 \mathbf{u}_1 \ddot{\theta}_1 + L_3 \check{\mathbf{u}}_1 \dot{\theta}_1^2 + L_2 \check{\mathbf{u}}_2 (\ddot{\theta}_1 + \ddot{\theta}_2) - L_2 \mathbf{u}_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \\ \ddot{\mathbf{r}}^{G_3,O} &= -L_3 \mathbf{u}_1 \ddot{\theta}_1 - L_3 \check{\mathbf{u}}_1 \dot{\theta}_1^2 + L_2 \check{\mathbf{u}}_3 (\ddot{\theta}_1 + \ddot{\theta}_3) - L_2 \mathbf{u}_3 (\dot{\theta}_1 + \dot{\theta}_3)^2 \end{aligned} \quad (5.13)$$

A comparison between the two sets of expressions that we derived for the triple pendulum reveals that those with the absolute angles have a much simpler form than those containing relative angles. Therefore, we make this as a general rule (or a suggestion) that if there is a choice between defining a coordinate either an absolute or a relative quantity, we should choose the absolute one. This suggestion is valid for both angles and magnitudes.



**FIGURE 5.7**

Angles at  $A$  and  $B$  are defined with respect to a moving axis on body (1).

From these examples, we conclude that for an open-chain system with  $n_{dof}$  DoFs, we need to define at least  $n_v = n_{dof}$  vector coordinates to fully determine the kinematics of the system. If the number of defined vector coordinates is greater than the number of DoF, that is,  $n_v > n_{dof}$ , then there must exist  $n_c = n_v - n_{dof}$  algebraic constraints that must be defined between the coordinates.

---

### 5.3 Closed-Chain Systems

Constructing vectors for a closed-chain system is similar to that for an open-chain system with the exception that the vectors in a closed chain must form a loop. The variable lengths and angles of the defined vectors form the vector coordinates for that system. The main difference with the open-chain system is that for the closed chain system, the number of defined vector coordinates will be greater than the number of system's DoFs, that is,  $n_v > n_{dof}$ . This means that there must exist  $n_c = n_v - n_{dof}$  algebraic constraints between the vector coordinates. These constraints are the result of the corresponding *vector loop equations*.

#### 5.3.1 Slider-Crank Mechanism

The slider-crank mechanism shown in Figure 5.8 is a one DoF system containing a single closed chain. We define three vectors as shown, resulting in three vector-coordinates:  $\theta_1$  and  $\theta_2$  are variable angles and  $\theta_3$  is a variable length. The three vectors form a loop that can be described analytically as the following vector loop equation:

$$L_1 \vec{u}_1 + L_2 \vec{u}_2 - \theta_3 \vec{u}_{(x)} = \vec{0}$$

Or algebraically we have

$$L_1 \mathbf{u}_1 + L_2 \mathbf{u}_2 - \theta_3 \mathbf{u}_{(x)} = \mathbf{0} \Rightarrow \begin{aligned} L_1 \cos \theta_1 + L_2 \cos \theta_2 - \theta_3 &= 0 \\ L_1 \sin \theta_1 + L_2 \sin \theta_2 &= 0 \end{aligned} \quad (5.14)$$

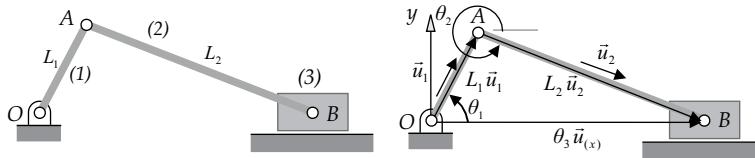
These equations state that the three vector coordinates,  $\theta_1$ ,  $\theta_2$ , and  $\theta_3$ , are not independent; therefore, the equations are called *constraints*. Here, we have two *nonlinear algebraic equations* and three variables. If we assign a value to one of the variables, we should be able to solve the two equations for the other two variables.

#### Example 5.1

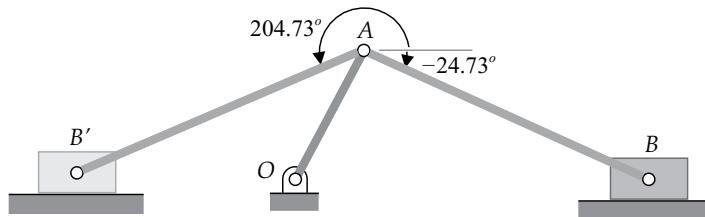
For the slider-crank example of Figure 5.8, assume the following constant lengths:  $L_1 = 0.12\text{ m}$ ,  $L_2 = 0.26\text{ m}$ . Solve the position constraint equations when  $\theta_1 = 65^\circ$ .

#### *Solution*

We substitute the constant data and the known value of the crank angle in Eq. (5.14):

**FIGURE 5.8**

A slider–crank mechanism and its vector loop representation.

**FIGURE 5.9**

Two solutions (assemblies) for the slider–crank mechanism.

$$0.12 \cos 65^\circ + 0.26 \cos \theta_2 - \theta_3 = 0$$

$$0.12 \sin 65^\circ + 0.26 \sin \theta_2 = 0$$

The second constraint contains only one unknown yielding  $\sin \theta_2 = -(0.12/0.26)\sin 65^\circ$ , which results in finding the angle  $\theta_2$ . Then, the first constraint equation can be solved for  $\theta_3$ .

**MATLAB/Chapter 5/Example \_ 5 \_ 1**

```
L1 = 0.12; L2 = 0.26; theta1 = 65*pi/180;
theta2 = asin(-L1*sin(theta1)/L2) .....
theta3 = L1*cos(theta1) + L2*cos(theta2) ....
```

```
theta2 =
-0.4316
theta3 =
0.2869
```

The solution for  $\theta_2$ , obtained from the `asin` function, is in radians. In degrees, we have  $\theta_2 = -24.73^\circ$  or  $360 - 24.73 = 335.27^\circ$ . This is one of the two solutions for the set of nonlinear position constraints. The second solution is  $\theta_2 = 180 + 24.73 = 204.73^\circ$ , which in turn yields  $\theta_3 = 0.185$  m. The two sets of answers represent two different slider–cranks, shown in Figure 5.9, that have identical link lengths.

For the position analysis of the slider–crank mechanism, we were able to solve the position constraints analytically due to the fact that both unknowns did not appear in both equations. In general, this is not the case for position constraints. It would be much simpler to solve the position constraints numerically rather than analytically, as we will see for a four-bar mechanism in Section 5.3.2.

To obtain the velocity constraints for the slider–crank, we take the time derivative of Eq. (5.14):

$$L_1\dot{\theta}_1\check{\mathbf{u}}_1 + L_2\dot{\theta}_2\check{\mathbf{u}}_2 - \dot{\theta}_3\mathbf{u}_{(x)} = \mathbf{0} \Rightarrow \begin{aligned} -L_1\dot{\theta}_1 \sin \theta_1 - L_2\dot{\theta}_2 \sin \theta_2 - \dot{\theta}_3 &= 0 \\ L_1\dot{\theta}_1 \cos \theta_1 + L_2\dot{\theta}_2 \cos \theta_2 &= 0 \end{aligned} \quad (5.15)$$

These constraints are linear in the velocities. If one of the velocities is assigned a value, the other two can be computed by solving the set of two *linear algebraic equations*. Assume that  $\dot{\theta}_1$  is known, then Eq. (5.15) can be expressed in matrix form as

$$\left[ \begin{array}{cc} L_2\check{\mathbf{u}}_2 & -\mathbf{u}_{(x)} \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_2 \\ \dot{\theta}_3 \end{array} \right\} = -L_1\dot{\theta}_1\check{\mathbf{u}}_1 \Rightarrow \left[ \begin{array}{cc} -L_2 \sin \theta_2 & -1 \\ L_2 \cos \theta_2 & 0 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_2 \\ \dot{\theta}_3 \end{array} \right\} = \left\{ \begin{array}{c} L_1 \sin \theta_1 \\ -L_1 \cos \theta_1 \end{array} \right\} \dot{\theta}_1 \quad (5.16)$$

### Example 5.1 (cont.)

Consider the first solution for the slider–crank mechanism:  $\theta_1 = 65^\circ$ ,  $\theta_2 = -24.73^\circ$ , and  $\theta_3 = 0.2869$  m. If  $\dot{\theta}_1 = 1.6$  rad/s, solve the velocity constraints of Eq. (5.16) for the remaining velocities.

```

...
theta1_d = 1.6;
u1 = [cos(theta1); sin(theta1)];
u2 = [cos(theta2); sin(theta2);
u1r = s_rot(u1); u2r = s_rot(u2);
C = [L2*s_rot(u2) [-1; 0]];
rhs_v = -L1*u1r*theta1_d;
solution_v = C\rhs_v;
theta2_d = solution_v(1) ..... theta2_d =
theta3_d = solution_v(2) ..... theta3_d =

```

If we perform the same analysis for the second assembly of the position constraints, we obtain the following velocities:  $\dot{\theta}_2 = 0.3436$  rad/s and  $\dot{\theta}_3 = -0.2114$  m/s.

For the acceleration constraints of the slider–crank, the time derivative of Eq. (5.15) provides the following:

$$L_1\ddot{\theta}_1\check{\mathbf{u}}_1 - L_1\dot{\theta}_1^2\mathbf{u}_1 + L_2\ddot{\theta}_2\check{\mathbf{u}}_2 - L_2\dot{\theta}_2^2\mathbf{u}_2 - \ddot{\theta}_3\mathbf{u}_{(x)} = \mathbf{0} \Rightarrow \begin{aligned} -L_1\ddot{\theta}_1 \sin \theta_1 - L_1\dot{\theta}_1^2 \cos \theta_1 - L_2\ddot{\theta}_2 \sin \theta_2 - L_2\dot{\theta}_2^2 \cos \theta_2 - \ddot{\theta}_3 &= 0 \\ L_1\ddot{\theta}_1 \cos \theta_1 - L_1\dot{\theta}_1^2 \sin \theta_1 + L_2\ddot{\theta}_2 \cos \theta_2 - L_2\dot{\theta}_2^2 \sin \theta_2 &= 0 \end{aligned} \quad (5.17)$$

These constraints are also a set of *linear algebraic equations*. If one of the accelerations is assigned a value, the other two can be solved for. For example, if  $\ddot{\theta}_1$  is known, Eq. (5.17) can be written as

$$\begin{bmatrix} L_2 \ddot{\mathbf{u}}_2 & -\mathbf{u}_{(x)} \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{Bmatrix} = -L_1 \ddot{\theta}_1 \ddot{\mathbf{u}}_1 + L_1 \dot{\theta}_1^2 \mathbf{u}_1 + L_2 \dot{\theta}_2^2 \mathbf{u}_2 \Rightarrow$$

$$\begin{bmatrix} -L_2 \sin \theta_2 & -1 \\ L_2 \cos \theta_2 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{Bmatrix} = \begin{Bmatrix} L_1 \ddot{\theta}_1 \sin \theta_1 + L_1 \dot{\theta}_1^2 \cos \theta_1 + L_2 \dot{\theta}_2^2 \cos \theta_2 \\ -L_1 \ddot{\theta}_1 \cos \theta_1 + L_1 \dot{\theta}_1^2 \sin \theta_1 + L_2 \dot{\theta}_2^2 \sin \theta_2 \end{Bmatrix} \quad (5.18)$$

**Example 5.1 (cont.)**

Following the velocity analysis of the slider–crank mechanism, if  $\ddot{\theta}_1 = 0$ , solve the acceleration constraints of Eq. (5.18) for the remaining accelerations.

```

...
theta1_dd = 0;
rhsa = -L1*u1r*theta1_dd + ...
        L1*u1*theta1_d^2 + L2*u2*theta2_d^2;
solution_a = C\rhsa;
theta2_dd = solution_a(1) ..... .
theta3_dd = solution_a(2) ..... .
theta2_dd =
1.1246
theta3_dd =
-0.0354

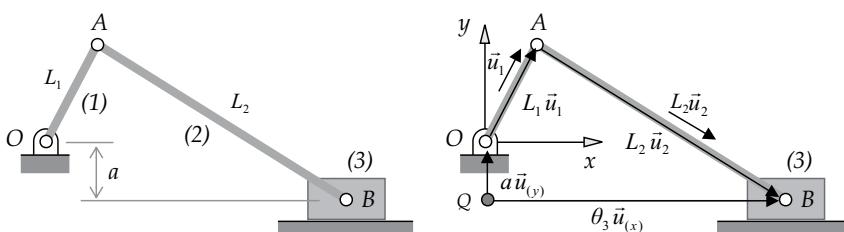
```

If we perform the same analysis for the second configuration of the slider–crank, we obtain the following accelerations:  $\ddot{\theta}_2 = -1.1246 \text{ rad/s}^2$  and  $\ddot{\theta}_3 = -0.2243 \text{ m/s}^2$ .

As a slight variation of the slider–crank mechanism of Figure 5.8, an offset slider–crank is shown in Figure 5.10. To construct the vector loop equation for this mechanism, we must first define a ground point, such as  $Q$ , in order to have a position vector for the slider–block along the axis of the sliding joint. We position point  $Q$  conveniently below point  $O$  resulting in a nonmoving vertical vector between the two points. Now we can construct the vector loop equation as

$$L_1 \vec{u}_1 + L_2 \vec{u}_2 - \theta_3 \vec{u}_{(x)} + a \vec{u}_{(y)} = \vec{0}$$

Or

**FIGURE 5.10**

An offset slider–crank mechanism and its vector loop representation.

$$L_1 \mathbf{u}_1 + L_2 \mathbf{u}_2 - \theta_3 \mathbf{u}_{(x)} + a \mathbf{u}_{(y)} = \mathbf{0} \Rightarrow \begin{aligned} L_1 \cos \theta_1 + L_2 \cos \theta_2 - \theta_3 &= 0 \\ L_1 \sin \theta_1 + L_2 \sin \theta_2 + a &= 0 \end{aligned} \quad (5.19)$$

Derivation of the velocity and acceleration constraints is the same as that for the non-offset slider–crank mechanism.

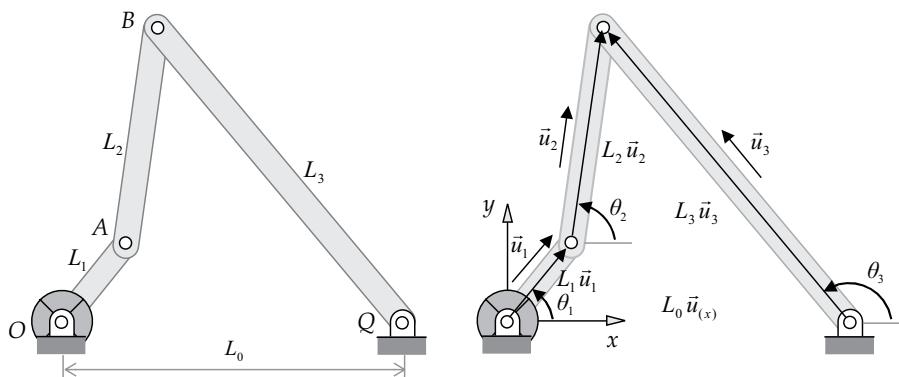
### 5.3.2 Four-Bar Mechanism

The four-bar mechanism shown in Figure 5.11 is a one DoF system containing one closed chain. We define four vectors as shown, where three of them have variable angles and one is a fixed vector. The four vectors yield the following vector loop equation:

$$L_1 \mathbf{u}_1 + L_2 \mathbf{u}_2 - L_3 \mathbf{u}_3 - L_0 \mathbf{u}_{(x)} = \mathbf{0} \Rightarrow \begin{aligned} L_1 \cos \theta_1 + L_2 \cos \theta_2 - L_3 \cos \theta_3 - L_0 &= 0 \\ L_1 \sin \theta_1 + L_2 \sin \theta_2 - L_3 \sin \theta_3 &= 0 \end{aligned} \quad (5.20)$$

These *constraints* are two nonlinear algebraic equations. If one of the angles is assigned a value, the two equations can be solved to determine the other two angles.

In general, position constraints are nonlinear equations that cannot be expressed in matrix form. This is true regardless of the type of coordinates used to formulate the constraints. In the slider–crank mechanism of Example 5.1, due to the simplicity of the constraints equations, we managed to solve the constraints analytically. However, in the four-bar mechanism, although there exists an analytical solution, it would be much simpler to solve the equations numerically.



**FIGURE 5.11**  
A four-bar mechanism and its vector loop representation.

**Example 5.2**

In the four-bar mechanism of Figure 5.11, the link lengths are as follows:  $L_0 = 0.2\text{ m}$ ,  $L_1 = 0.1\text{ m}$ ,  $L_2 = 0.3\text{ m}$ , and  $L_3 = 0.22\text{ m}$ . For  $\theta_1 = \pi/4$ , solve Eq. (5.20) for the unknowns.

**MATLAB/Chapter 5/Example \_ 5 \_ 2**

To solve the nonlinear constraints of Eq. (5.20), we take the advantage of the MATLAB® function `fsolve`. For this purpose, we formulate the nonlinear constraints in the following function M-file:

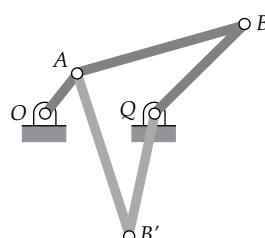
```
function Phi = fourbar(x, theta1)
% Array (x contains theta2 and theta3)
global L0 L1 L2 L3
    u1 = [cos(theta1); sin(theta1)];
    u2 = [cos(x(1)); sin(x(1))];
    u3 = [cos(x(2)); sin(x(2))];
    Phi = L1*u1 + L2*u2 - L3*u3 - [L0; 0];
```

In the following script, we provide the constant data, the known value of  $\theta_1$ , and our best estimates for the unknowns as  $\theta_2 = 0.5$  and  $\theta_3 = 1.3\text{ rad}$ . We direct the function `fsolve` to evaluate the constraints in the function M-file `fourbar`.

```
global L0 L1 L2 L3
L0 = 0.2; L1 = 0.1; L2 = 0.3; L3 = 0.22;
theta1 = pi/4;
theta23 = [0.5; 1.3]; % estimates
options = optimset('display', 'off');
angles = fsolve(@fourbar, theta23, ...
    options, theta1);
theta2 = angles(1) .....
theta3 = angles(2) .....
```

theta2 =	0.2721
theta3 =	0.7586

The solution converges to  $\theta_2 = 0.27\text{ rad}$  ( $15.6^\circ$ ) and  $\theta_3 = 0.76\text{ rad}$  ( $43.5^\circ$ ). Since we are solving two nonlinear algebraic equations, there must exist another set of answers. If we provide a different set of initial estimates for the unknowns, we may converge to a different solution. For example, for  $\theta_2 = 4.0$  and  $\theta_3 = 4.0\text{ rad}$  as our initial estimates, the solution converges to  $\theta_2 = 5.0\text{ rad}$  ( $287.1^\circ$ ) and  $\theta_3 = 4.5\text{ rad}$  ( $259.2^\circ$ ). The two sets of solutions of the four-bar mechanism are shown in Figure 5.12. Both sets of solutions are correct—we should choose the solution that represents the assembly of our mechanism.

**FIGURE 5.12**

The two possible configurations for the crank angle  $\theta_1 = 45^\circ$ .

The time derivative of Eq. (5.20) provides the velocity constraints. These constraints are linear in the velocities, and therefore, they can be expressed in matrix form:

$$L_1\bar{\mathbf{u}}_1\dot{\theta}_1 + L_2\bar{\mathbf{u}}_2\dot{\theta}_2 - L_3\bar{\mathbf{u}}_3\dot{\theta}_3 = \mathbf{0} \Rightarrow \begin{bmatrix} L_1\bar{\mathbf{u}}_1 & L_2\bar{\mathbf{u}}_2 & -L_3\bar{\mathbf{u}}_3 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} = \mathbf{0} \quad (5.21)$$

If one of the velocities has a known value, for example,  $\dot{\theta}_1$ , then Eq. (5.21) can be rearranged as

$$\begin{bmatrix} L_2\bar{\mathbf{u}}_2 & -L_3\bar{\mathbf{u}}_3 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} = -L_1\bar{\mathbf{u}}_1\dot{\theta}_1 \quad (5.22)$$

Here we have two algebraic equations that can be solved for two unknowns  $\dot{\theta}_2$  and  $\dot{\theta}_3$ .

### Example 5.2 (cont.)

Continuing with the first set of solution for the four-bar mechanism, assume that  $\dot{\theta}_1 = 1.5$  rad/s. Solve Eq. (5.22) for the other two unknown velocities.

```

...
u1 = [cos(theta1); sin(theta1)];
u2 = [cos(theta2); sin(theta2)];
u3 = [cos(theta3); sin(theta3)];
C = [L2*s_rot(u2) -L3*s_rot(u3)]; % Jacobian
rhs_v = -L1*s_rot(u1)*theta1_d;
solution_v = C\rhs_v;
theta2_d = solution_v(1) .....
theta3_d = solution_v(2) .....
theta2_d =
0.0286
theta3_d =
0.7161

```

The time derivative of the velocity constraints provides the acceleration constraints:

$$\begin{bmatrix} L_1\bar{\mathbf{u}}_1 & L_2\bar{\mathbf{u}}_2 & -L_3\bar{\mathbf{u}}_3 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{Bmatrix} = L_1\mathbf{u}_1\dot{\theta}_1^2 + L_2\mathbf{u}_2\dot{\theta}_2^2 - L_3\mathbf{u}_3\dot{\theta}_3^2 \quad (5.23)$$

If  $\ddot{\theta}_1$  is known, then the acceleration constraints can be rearranged as

$$\begin{bmatrix} L_2\bar{\mathbf{u}}_2 & -L_3\bar{\mathbf{u}}_3 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{Bmatrix} = -L_1\bar{\mathbf{u}}_1\ddot{\theta}_1 + L_1\mathbf{u}_1\dot{\theta}_1^2 + L_2\mathbf{u}_2\dot{\theta}_2^2 - L_3\mathbf{u}_3\dot{\theta}_3^2 \quad (5.24)$$

These two linear algebraic equations can be solved for the two unknown accelerations.

### Example 5.2 (cont.)

For  $\ddot{\theta}_1 = -1.2$  rad/s<sup>2</sup>, solve Eq. (5.24) for the other two accelerations.

```

...
thetal1_dd = -1.2;
rhsa = -L1*s_rot(u1)*thetal1_dd + ...
       L1*u1*thetal1_d^2 + ...
       L2*u2*thetal2_d^2 - L3*u3*thetal3_d^2;
solution_a = C\rhsa;
theta2_dd = solution_a(1) .....
theta3_dd = solution_a(2) .....

```

```

theta2_dd =
0.7779
theta3_dd =
0.3656

```

For the four-bar or any other mechanism, we might be interested in the kinematics of a point that is not part of the vector loop equation(s). For example, point  $P$  in Figure 5.13, which is attached to the coupler link (2), is the point of interest. This point can be positioned on link (2) with a constant angle  $\beta_2$  and a constant length  $L_{PA}$ . To determine the coordinates of this point in the  $x$ - $y$  frame, we first define the unit vector  $\bar{u}_{PA}$  as

$$\mathbf{u}_{PA} = \begin{bmatrix} \cos(\theta_2 + \beta_2) \\ \sin(\theta_2 + \beta_2) \end{bmatrix} \quad (5.25)$$

Then we express the coordinates of  $P$  as

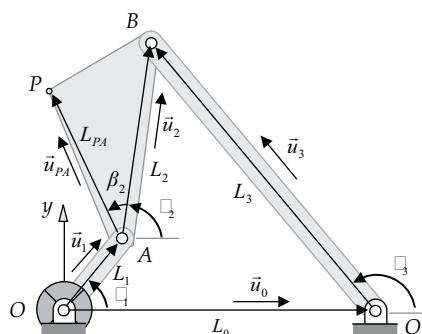
$$\vec{r}^P = \vec{r}^A + \vec{r}^{PA} \Rightarrow \mathbf{r}^P = L_1 \mathbf{u}_1 + L_{PA} \mathbf{u}_{PA} \quad (5.26)$$

If the position constraints have already been solved for the unknown coordinates, then everything on the right-hand side of Eq. (5.26) is known. Therefore, this expression can be evaluated for the coordinates of point  $P$ .

The velocity and acceleration expressions for this point of interest can be expressed as

$$\dot{\mathbf{r}}^P = L_1 \dot{\theta}_1 \bar{\mathbf{u}}_1 + L_{PA} \dot{\theta}_2 \bar{\mathbf{u}}_{PA} \quad (5.27)$$

$$\ddot{\mathbf{r}}^P = L_1 \ddot{\theta}_1 \bar{\mathbf{u}}_1 - L_1 \dot{\theta}_1^2 \mathbf{u}_1 + L_{PA} \ddot{\theta}_2 \bar{\mathbf{u}}_{PA} - L_{PA} \dot{\theta}_2^2 \mathbf{u}_{PA} \quad (5.28)$$



**FIGURE 5.13**

A coupler point  $P$  is defined on link (2).

**Example 5.2 (cont.)**

For a coupler point  $P$ , we have the following constants:  $L_{PA} = 0.2 \text{ m}$  and  $\beta_2 = 30^\circ$ . Determine the coordinates, velocity, and acceleration of this point.

```

...
Lpa = 0.2; beta2 = pi/6;
tb2 = theta2 + beta2;
upa = [cos(tb2); sin(tb2)];
rP = L1*u1 + Lpa*upa ..... rP =
0.2107
0.2136

rP_d = L1*theta1_d*s_rot(u1) + ... rP_d =
-0.1102
0.1101

rP_dd = L1*(theta1_dd*s_rot(u1) - ... rP_dd =
-0.1855
-0.1352
    theta1_d^2*u1) + ...
    Lpa*(theta2_dd*s_rot(upa) - ...
    theta2_d^2*upa) .....

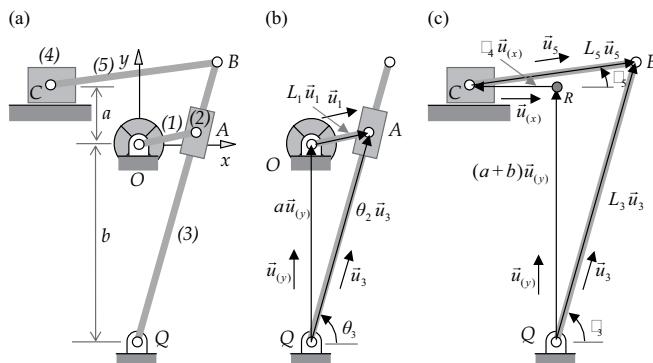
```

**5.3.3 Six-Bar Quick-Return Mechanism**

The six-bar mechanism shown in Figure 5.14a (adopted from Section 15.3) is a closed-chain system that consists of five moving bodies, five pin joints, and two sliding joints. The system has one DoF that is controlled by a motor. The constant lengths and distances are as follows:  $L_1$ ,  $L_3$ ,  $L_5$ ,  $a$ , and  $b$ . This system contains two loops, where for clarification purposes the loops are shown separately.

The first loop, shown in Figure 5.14b, is an inverted slider–crank where its motion is completely controlled by the motor; that is, the presence or absence of links (4) and (5) will not influence the kinematics of this loop. We define three vectors as shown and construct the following vector loop constraint:

$$a\mathbf{u}_{(y)} + L_1\mathbf{u}_1 - \theta_2\mathbf{u}_3 = \mathbf{0} \quad (5.29)$$

**FIGURE 5.14**

(a) A six-bar mechanism, and its (b) loop 1 and (c) loop 2.

where  $\theta_2$  is the variable distance between points  $Q$  and  $A$ .

The second loop, shown in Figure 5.14c, is an offset slider–crank where its motion is controlled by link (3) of the first slider–crank. In this loop, to construct a vector parallel to the axis of sliding joint of body (4), we define point  $R$  on the ground, where we end up with four vectors leading to a second vector loop equation:

$$(a+b)\mathbf{u}_{(y)} + \theta_4\mathbf{u}_{(x)} + L_5\mathbf{u}_5 - L_3\mathbf{u}_3 = \mathbf{0} \quad (5.30)$$

Note that the sign for a vector, in a vector loop equation, is based on whether we negotiate through a vector from the tail to the head or vice versa. For example, for the second loop, if we move through the loop clockwise, we go through vectors  $\vec{r}^{R,Q}$ ,  $\vec{r}^{C,R}$ , and  $\vec{r}^{B,C}$  from the tail to the head (positive signs), but for  $\vec{r}^{B,Q}$ , we negotiate it from the head to the tail (negative sign). Therefore, the unknown variables in a vector loop equation may be positive or negative quantities. For example, when the loop equations are solved for the unknown variables,  $\theta_4$  should end up as a negative quantity since  $\vec{r}^{C,R}$  is in the opposite direction of the unit vector  $\bar{\mathbf{u}}_{(x)}$ .

In the two sets of constraints, we have four algebraic equations and five variable coordinates:  $\theta_1$ ,  $\theta_3$ , and  $\theta_5$  are angles, and  $\theta_2$  and  $\theta_4$  are directional magnitudes. If the value of one of these variables is known, the four algebraic constraints can be solved for the other four. For example, if  $\theta_1$  is assigned a value, we can find the unknowns in two ways:

- a. Equations (5.29) and (5.30) can be solved as four simultaneous nonlinear algebraic equations.
- b. Equation (5.29) can be solved first for  $\theta_2$  and  $\theta_3$ , and then Eq. (5.30) is solved for  $\theta_4$  and  $\theta_5$ .

The two-step process in (b) is possible because the two sets of equations are *loosely coupled*. However, this is not the case for the constraints of any six-bar mechanism. There are six-bar mechanisms for which the two sets of constraints are *strongly coupled*, and therefore, they need to be solved simultaneously.

The first time derivative of the position constraints yields the velocity constraints:

$$L_1\bar{\mathbf{u}}_1\dot{\theta}_1 - \mathbf{u}_3\dot{\theta}_2 - \theta_2\bar{\mathbf{u}}_3\dot{\theta}_3 = \mathbf{0} \quad (5.31)$$

$$\mathbf{u}_{(x)}\dot{\theta}_4 + L_5\bar{\mathbf{u}}_5\dot{\theta}_5 - L_3\bar{\mathbf{u}}_3\dot{\theta}_3 = \mathbf{0} \quad (5.32)$$

For a known value of  $\dot{\theta}_1$ , we can first solve Eq. (5.31) for  $\dot{\theta}_2$  and  $\dot{\theta}_3$ , and then we can solve Eq. (5.32) for the remaining two velocities. We can also solve these two sets of equations simultaneously. The simultaneous set of velocity constraints can be expressed in matrix form as

$$\left[ \begin{array}{cc|cc|cc} L_1\bar{\mathbf{u}}_1 & -\mathbf{u}_3 & -\theta_2\bar{\mathbf{u}}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -L_3\bar{\mathbf{u}}_3 & \mathbf{u}_{(x)} & L_5\bar{\mathbf{u}}_5 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array} \right\} \quad (5.33)$$

The loosely coupled feature of the two sets of equations can be observed from the coefficient matrix of the velocity equations—the two sets of equations only share one unknown velocity,  $\dot{\theta}_3$ .

The acceleration constraints are obtained from the time derivative of Eq. (5.33):

$$\left[ \begin{array}{ccccc} L_1\ddot{\mathbf{u}}_1 & -\mathbf{u}_3 & -\theta_2\ddot{\mathbf{u}}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -L_3\ddot{\mathbf{u}}_3 & \mathbf{u}_{(x)} & L_5\ddot{\mathbf{u}}_5 \end{array} \right] \left\{ \begin{array}{c} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\theta}_4 \\ \ddot{\theta}_5 \end{array} \right\} = \left\{ \begin{array}{c} L_1\mathbf{u}_1\dot{\theta}_1^2 + 2\ddot{\mathbf{u}}_3\dot{\theta}_2\dot{\theta}_3 - \theta_2\mathbf{u}_3\dot{\theta}_3^2 \\ -L_3\mathbf{u}_3\dot{\theta}_3^2 + L_5\mathbf{u}_5\dot{\theta}_5^2 \end{array} \right\}$$
(5.34)

### 5.3.4 Six-Bar Dwell Mechanism

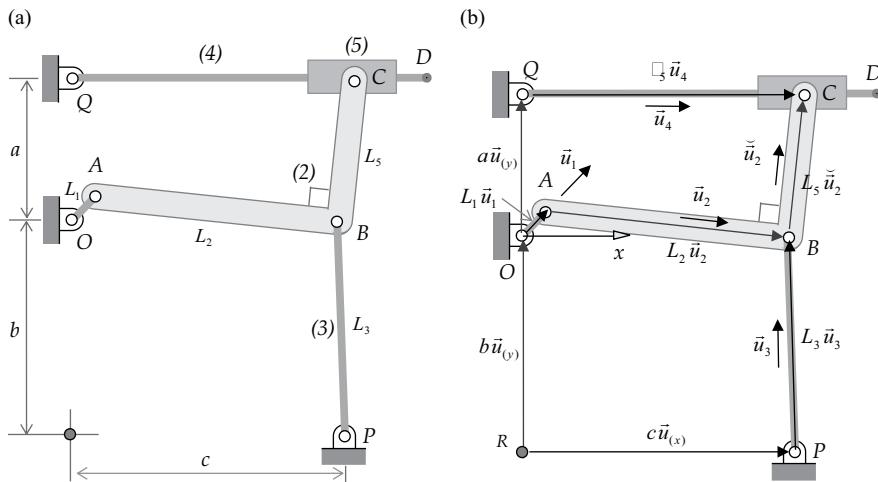
The six-bar system shown in Figure 5.15a is a dwell mechanism that is adopted from Section 15.4. When the crank, link (1), rotates with a constant angular velocity, the output link  $QD$  dwells. The mechanism contains two loops. The necessary vectors to construct vector loop equations are shown in Figure 5.15b. The vector coordinates  $\theta_1, \theta_2, \theta_3$ , and  $\theta_4$  are angles, and  $\theta_5$  is the distance between points  $Q$  and  $C$ .

The lower loop is a four-bar mechanism for which we can write the following vector loop equation:

$$L_1\mathbf{u}_1 + L_2\mathbf{u}_2 - L_3\mathbf{u}_3 - c\mathbf{u}_{(x)} + b\mathbf{u}_{(y)} = \mathbf{0} \quad (5.35)$$

For the second loop, we have

$$L_1\mathbf{u}_1 + L_2\mathbf{u}_2 + L_5\ddot{\mathbf{u}}_2 - \theta_5\mathbf{u}_4 - a\mathbf{u}_{(y)} = \mathbf{0} \quad (5.36)$$



**FIGURE 5.15**  
A six-bar dwell mechanism.

We may argue that there exists a third loop, *OQCBPRO*, which is correct. However, this loop is not independent from the first two loops. We can obtain the constraint equation associated with this third loop from combining the first two constraint equations.

The velocity constraints associated with Eqs. (5.35) and (5.36) can be expressed as four simultaneous equations in matrix form as

$$\left[ \begin{array}{ccccc} L_1\check{\mathbf{u}}_1 & L_2\check{\mathbf{u}}_2 & -L_3\check{\mathbf{u}}_3 & \mathbf{0} & \mathbf{0} \\ L_1\check{\mathbf{u}}_1 & L_2\check{\mathbf{u}}_2 & -L_5\mathbf{u}_2 & -\theta_5\check{\mathbf{u}}_4 & -\mathbf{u}_4 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array} \right\} \quad (5.37)$$

Then the acceleration constraints are obtained from the time derivative of the velocity equations.

### 5.3.5 Complete Kinematic Analysis

For most mechanisms, it is desirable to analyze the kinematics of a system through its complete range of motion. For this purpose, the position or orientation of the input link can be varied incrementally, and at each position, the coordinates, velocities, and accelerations of all the links are determined. The results from such a complete analysis can be reported as plots, animations, or other forms to assist in understanding the operation of that mechanism.

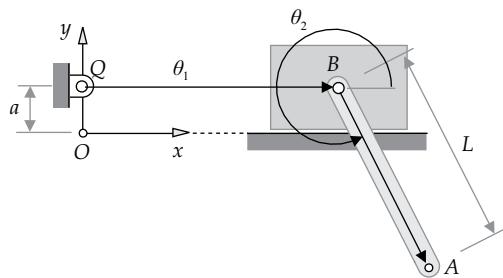
As an example, for the slider–crank mechanism of Example 5.1, the crank angle can be varied by increments of  $\Delta\theta_1 = 5^\circ$  or  $10^\circ$  from  $\theta_1 = 0$  to  $360^\circ$ . For every input angle, we can execute the program of Example 5.1 and save the results.

We should note that it may not physically be possible, for every mechanism, to rotate the crank through a complete revolution. For example, if a four-bar is not Grashof,\* then the mechanism has a limited range of motion. If we try to solve the position constraints for an angle of the crank outside its range, we will not be able to find a solution since it does not exist.

## 5.4 Problems

- 5.1 For the open-chain system shown, two vector coordinates are defined as  $\theta_1$  and  $\theta_2$ . Assume  $a = 0.01$  and  $L = 0.1$  m. Construct the recursive formulas and then develop a MATLAB program to determine the coordinates, velocities, and accelerations of point A. Assume  $\theta_1 = 0.15$  m,  $\theta_2 = 285^\circ$ ,  $\dot{\theta}_1 = -0.02$  m/s,  $\dot{\theta}_2 = 0.6$ , rad/s,  $\ddot{\theta}_1 = 0.12$  m/s<sup>2</sup>, and  $\ddot{\theta}_2 = -0.02$  rad/s<sup>2</sup>.

\* In a four-bar mechanism, if the sum of the shortest and the longest links is smaller than the sum of the lengths of the other two links, the linkage is said to be Grashof, meaning that at least one of the links can undergo a full rotation.



- 5.2 For each of the open-chain systems shown, define the necessary vectors and vector coordinates. For rotational coordinates, define absolute angles. Construct recursive position, velocity, and acceleration expressions for all the points of interest indicated on each figure, including the mass centers. Use the following dimensions for each system as applicable:

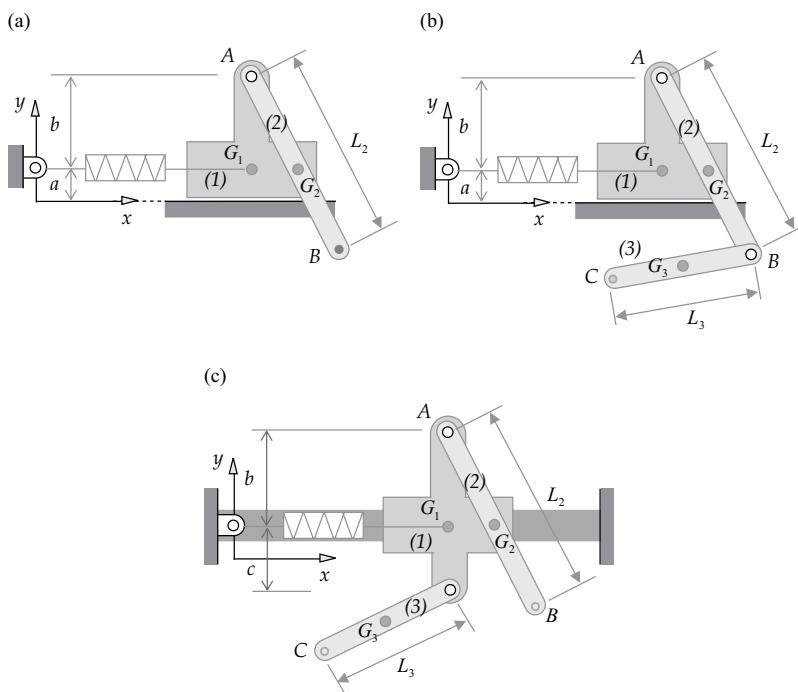
$$a = 0.01, \ b = 0.03, \ c = 0.02, \ L_2 = 0.06, \ L = 0.06 \text{ m}$$

Develop a MATLAB program to compute your expressions. Assume the following values where applicable:

$$\theta_1 = 0.15 \text{ m}, \dot{\theta}_1 = -0.02 \text{ m/s}, \ddot{\theta}_1 = 0.12 \text{ m/s}^2$$

$$\theta_2 = 285^\circ, \dot{\theta}_2 = 0.6 \text{ rad/s}, \text{ and } \ddot{\theta}_2 = -0.02 \text{ rad/s}^2$$

$$\theta_3 = 205^\circ, \dot{\theta}_3 = -0.5 \text{ rad/s}, \text{ and } \ddot{\theta}_3 = 0.05 \text{ rad/s}^2$$



- 5.3 For the triple pendulum of Figure 5.6 and the corresponding recursive formulas of Eqs. (5.9)–(5.11), develop a MATLAB program. Compute the formulas for the following constant and variable data:

$$L_1 = 0.20, L_2 = L_3 = 0.15 \text{ m}$$

$$\theta_1 = 255^\circ, \theta_2 = 120^\circ, \theta_3 = 215^\circ \text{ rad}$$

$$\dot{\theta}_1 = -0.2, \dot{\theta}_2 = 0.6, \dot{\theta}_3 = -0.5 \text{ rad/s}$$

$$\ddot{\theta}_1 = 0.03, \ddot{\theta}_2 = -0.02, \ddot{\theta}_3 = 0.05 \text{ rad/s}^2$$

- 5.4 For the triple pendulum of Figure 5.7 and the corresponding recursive formulas of Eqs. (5.9), (5.12), and (5.13), develop a MATLAB program. Compute the formulas for the following constant and variable data:

$$L_1 = 0.20, L_2 = L_3 = 0.15 \text{ m}$$

$$\theta_1 = 255^\circ, \theta_2 = 285^\circ, \theta_3 = 50^\circ \text{ rad}$$

$$\dot{\theta}_1 = -0.2, \dot{\theta}_2 = 0.8, \dot{\theta}_3 = 50^\circ \text{ rad/s}$$

$$\ddot{\theta}_1 = 0.03, \ddot{\theta}_2 = -0.05, \ddot{\theta}_3 = 0.02 \text{ rad/s}^2$$

- 5.5 Revise the MATLAB program of Example 5.1 to solve the position constraints of the slider–crank using MATLAB function `fssolve` (instead of the analytical solution).

- a. For  $\theta_1 = 65^\circ$ , assume the estimates for the other two coordinates as  $\theta_2 = -0.4 \text{ rad}$  and  $\theta_3 = 0.3 \text{ m}$ .
- b. For  $\theta_1 = 55^\circ$ , use the same estimates for the other two unknowns as in (a).

- 5.6 For the offset slider–crank shown in Figure 5.10, assume  $L_1 = 0.12 \text{ m}$ ,  $L_2 = 0.26 \text{ m}$ , and  $a = 0.1 \text{ m}$ .

- a. Construct the position, velocity, and acceleration constraints.
- b. Revise the MATLAB program from Example 5.1 for this offset slider–crank mechanism. Assume a constant angular velocity of  $\dot{\theta}_1 = 2\pi \text{ rad/s}$ .
- c. Execute the program for a complete revolution of the crank in  $5^\circ$  increments and save the computed results.
- d. Plot the velocity of the slider versus the crank angle and show that an offset slider–crank is a quick–return mechanism.
- e. Repeat (c) and (d) for  $a = 0.05 \text{ m}$  and  $a = 0$  (non-offset). Compare the plots of the three simulations. What do you conclude?

- 5.7 For the offset slider–crank of Figure 5.10, instead of using point  $Q$ , define a vector directly from  $O$  to  $B$  and then write the vector loop and the corresponding algebraic constraints.

- a. How many equations and how many variables do we have?
- b. For a known  $\theta_1$ , how can we solve these equations?

- 5.8 Revise the program of Example 5.2 for the four-bar by rotating the crank in  $5^\circ$  increments through a complete revolution. Save the coordinates of the coupler point.
- Perform three complete kinematic simulations for  $\beta_2 = 30^\circ$ ,  $\beta_2 = 15^\circ$ , and  $\beta_2 = -15^\circ$ .
  - Plot the path of point  $P$  for each case.
- 5.9 Develop a MATLAB program to solve the constraint equations of the six-bar mechanism of Section 5.3.3. Solve the position constraints using the MATLAB function `fsolve`. The constant data can be found in Section 15.3. Rotate the crank with a constant angular velocity or  $2\pi$  rad/s.
- Solve the constraints at each level (position, velocity, and acceleration) as a complete set.
  - Solve the constraints at each level as two sets of loosely coupled equations.
- 5.10 Develop a MATLAB program to solve the constraint equations of the six-bar dwell mechanism of Section 5.3.4. Solve the position constraints using the MATLAB function `fsolve`. Refer to Section 15.4 to obtain the necessary data. Rotate the crank with a constant angular velocity or  $2\pi$  rad/s.
- Solve the constraints at each level (position, velocity, and acceleration).
  - Show that link (4) dwells.
- 5.11 Consider the film-strip advancer system from Section 15.1.
- Define the necessary vectors and vector coordinates, construct the position constraints, and write the necessary expressions for the secondary point  $C$ .
  - Develop a MATLAB program to solve the constraint equations for a complete revolution of the crank. Plot the path of point  $P$ .
- 5.12 Consider the web-cutter mechanism from Section 15.2.
- Define the necessary vectors and vector coordinates; construct the position, velocity, and acceleration vector loop constraints; and write the necessary expressions for the secondary points  $C$  and  $D$ .
  - Develop a MATLAB program to solve the constraint equations for a complete revolution of the crank. Plot the paths of points  $C$  and  $D$ . Determine the velocity of these two points in the  $x$ -direction when they cut through the web.
- 5.13 Consider the double A-arm suspension system from Section 15.6.
- Define the necessary vectors and vector coordinates; construct the position, velocity, and acceleration vector loop constraints; and write the necessary expressions for the secondary point  $C$ .
  - Develop a MATLAB program to solve the constraint equations and to determine the kinematics of point  $C$ .
- 5.14 Consider the MacPherson suspension system from Section 15.7.
- Define the necessary vectors and vector coordinates; construct the position, velocity, and acceleration vector loop constraints; and write the necessary expressions for the secondary point  $C$ .
  - Develop a MATLAB program to solve the constraint equations and to determine the kinematics of point  $C$ .

# 6

---

## Free-Body Diagram

---

This chapter reviews the traditional concept of using a free-body diagram (FBD) as a basis to derive the equations of motion (EQMs) for planar multibody systems. As a review of this concept, we construct the free-body diagram (FBD) for several systems and discuss specific issues associated with each. For each FBD, we identify the applied and the reaction forces and torques, before constructing the EQMs. Finally, the EQMs are used to perform force analysis on closed-chain systems that represent single degree-of-freedom mechanisms. For such systems, it will be assumed that the motion of all the bodies is fully known, and the unknowns are the reaction forces and one applied force or torque.

---

### 6.1 FBD Examples

Free-body diagrams provide a simple and clear description of forces that act on the individual bodies of a multibody system, leading to the construction of the corresponding EQMs. For this process, all the forces that act on a body, whether applied or reaction, must be identified. Since applied forces could be functions of position and velocity of the bodies, for constructing the FBD of a system, all the coordinates and velocities must be known.

If there are no kinematic joints in a system, then there are no reaction forces, and therefore, we only need to determine the applied forces to construct the FBD. When a system contains kinematic joints, or any imposed algebraic conditions on its coordinates, in addition to the applied forces, we must also include in the FBD the reaction forces associated with the joints and the constraints.

#### 6.1.1 Two-Body System (Unconstrained)

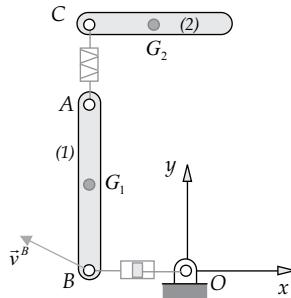
Consider the multibody system shown in Figure 6.1 containing two bodies. A spring connects the two bodies between points *A* and *C*, and a damper connects body (1) at *B* to the ground at point *O*. The gravity acts on the system in the negative *y*-direction. There are no kinematic joints in this system, and therefore, the system is called *unconstrained*. To construct the FBD for this system, we need to identify all of the applied forces from the spring, the damper, and the gravity.

##### Example 6.1

For the two-body system of Figure 6.1, consider the following data:

$$m_1 = 2.0 \text{ kg}, J_1 = 1.5 \text{ kg m}^2, m_2 = 1.5 \text{ kg}, J_2 = 0.5 \text{ kg m}^2$$

$$k = 100 \text{ N/m}, {}^0L = 2.0 \text{ m}, d_c = 20 \text{ N s x/m}$$

**FIGURE 6.1**

Two unconstrained bodies.

In the shown configuration, the following coordinates and velocities are provided:

$$\mathbf{r}_1^A = \begin{Bmatrix} -1.5 \\ 3.0 \end{Bmatrix} \text{ m}, \mathbf{r}_1^B = \begin{Bmatrix} -1.5 \\ 0 \end{Bmatrix} \text{ m}, \mathbf{r}_2^C = \begin{Bmatrix} -1.5 \\ 4.5 \end{Bmatrix} \text{ m}$$

$$\mathbf{r}_1^G = \begin{Bmatrix} -1.5 \\ 1.5 \end{Bmatrix} \text{ m}, \mathbf{r}_2^G = \begin{Bmatrix} -0.5 \\ 4.5 \end{Bmatrix} \text{ m}, \dot{\mathbf{r}}_1^B = \begin{Bmatrix} -2.0 \\ 1.0 \end{Bmatrix} \text{ m/s}$$

Construct (a) the FBDs and (b) the EQMs for this system.

### *Solution*

We can verify that the position and orientation of the two bodies in the figure agree with the given coordinates. The spring is along a vertical axis and the damper is horizontal. This means that we can easily do the calculations by hand. However, we perform the process of computing the spring and damper forces in a more systematic way.

#### MATLAB/Chapter 6/Example \_ 6 \_ 1

- (a) We first provide the constant data, and the coordinates and velocity of the points of interest. We define the unit vectors along the  $x$  and  $y$  axes, and then compute the gravitational forces,  $w_1$  and  $w_2$ . The gravitational forces are applied in the negative  $y$ -direction at the mass centers.

```
m1 = 2.0; J1 = 1.5; m2 = 1.5; J2 = 0.5;
k = 100; L0 = 2.0; dc = 20; g = 9.81;
rA1 = [-1.5; 3.0]; rB1 = [-1.5; 0];
rC2 = [-1.5; 4.5]; rB1_d = [-2.0; 1.0];
rG1 = [-1.5; 1.5]; rG2 = [-0.5; 4.5];
ux = [1; 0]; uy = [0; 1];
w1 = -g*m1*uy; .....
w2 = -g*m2*uy; .....
```

```
w1 =
0
-19.6200
w2 =
0
-14.7150
```

Next we determine the spring force by following the steps in Eqs. (4.22)–(4.25) (we can also use the function `pp_s`). The spring applies a pair of forces at  $A$  and  $C$  in the opposite directions along the axis of the spring. The sign of the computed force,  ${}^{(s)}f$ , clarifies whether the spring is in tension or compression.

```
d1 = rC2 - rA1; L1 = norm(d1); u1 = d1/L1;
fs = k*(L1 - L0) .....
```

```
fsA = fs*uy .....
```

```
fsC = -fs*uy .....
```

$fs =$	
	-50
$fsA =$	0
	-50
$fsC =$	0
	50

Since the computed value of the spring force is negative, the spring is compressed.

We note that the computed vectors of spring forces do not have components in the  $x$ -direction.

Next we compute the force of the damper, following Eqs. (4.22–4.24), (4.26), and (4.27) (we can also use the function `pp_sd`, while setting the spring stiffness to zero). The damper applies a pair of forces in opposite directions between points  $B$  and  $O$ . The computed sign of the force,  ${}^{(d)}f$ , determines whether the pair of forces should exhibit a pull or a push.

```
d2 = rB1; L2 = norm(d2); u2 = d2/L2;
L2_d = u2'*rB1_d;
fd = dc*L2_d;
fdb = fd*ux .....
```

$fdb =$	
	40
	0

It should be obvious that there is no need to determine the force that the damper applies at  $O$ .

We now construct the FBD for both bodies, as shown in Figure 6.2(a).

(b) Based on the FBD, we can construct the array of forces, but first we need to determine the moment associated with the spring and damper forces. For example, the moment of the spring force on body (2) with respect to the mass center can be obtained as  $\vec{s}_2^C \times {}^{(s)}\vec{f}_2^C$ , which we can compute it as  $\vec{s}'_2 {}^{(s)}\vec{f}_2^C$ , and so on. Therefore, we need to compute the moment arms first, which are shown in Figure 6.2(b).

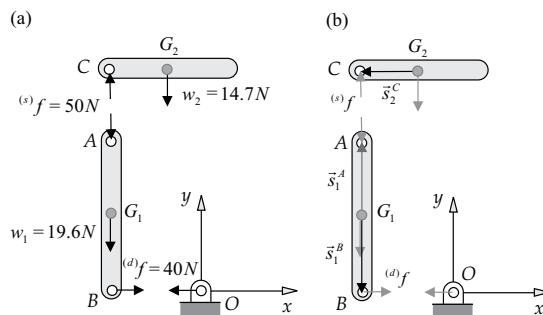


FIGURE 6.2

(a) The FBD and (b) the moment arms for forces acting at  $A$ ,  $B$ , and  $C$ .

```
sA1 = rA1 - rG1; sB1 = rB1 - rG1;
sC2 = rC2 - rG2;
```

We are now ready to construct the array of forces (forces and moments) for each body.

<pre>h1 = [(w1 + fsA + fdb)        (s_rot(sA1)'*fsA + s_rot(sB1)'*fdb)] ... h2 = [(w2 + fsC)        s_rot(sC2)'*fsC] .....</pre>	<pre>h1 = 40.0000 -69.6200 60.0000 0 35.2850 -50.0000</pre>
--	---

We construct the mass matrix, first as an array and then as a diagonal matrix.

<pre>M_diag = [m1 m1 ...            J1 m2 m2 J2]'; M = diag(M_diag) .....</pre>	<pre>M = 2.0   0   0   0   0   0 0   2.0   0   0   0   0 0   0   1.5   0   0   0 0   0   0   1.5   0   0 0   0   0   0   1.5   0 0   0   0   0   0   0.5</pre>
---	--

We have the mass matrix and the array of applied forces to form the EQMs:

$$\begin{bmatrix} 2.0 & & & & & \\ & 2.0 & & & & \\ & & 1.5 & & & \\ & & & 1.5 & & \\ & & & & 1.5 & \\ & & & & & 0.5 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\phi}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{\phi}_2 \end{bmatrix} = \begin{bmatrix} 40.000 \\ -69.620 \\ 60.000 \\ 0 \\ 35.285 \\ -50.000 \end{bmatrix}$$

These are six algebraic equations than can be solved for the six unknown accelerations.

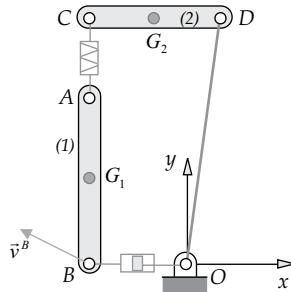
### 6.1.2 Two-Body System (Constrained)

This example, as shown in Figure 6.3, is a variation of the system in Figure 6.1, where a massless link is added between points *D* and *O*; otherwise, all the applied loads have remained the same. The massless link applies reaction forces at *D* and *O* in opposite directions along the axis of the link. We do not know the magnitude of the reaction force or whether the link is in compression or tension.

#### Example 6.2

Construct the EQMs for the system of Figure 6.3. This system is identical to the system of Example 6.1 with the exception of an added massless link between points *D* and *O*, having a length of  $L_c = 4.53\text{ m}$ . The coordinates of points *D* are found to be

$$\mathbf{r}_2^D = \begin{Bmatrix} 0.5 \\ 4.5 \end{Bmatrix} \text{ m}$$

**FIGURE 6.3**

A massless link is added to the two-body system of Example 6.1.

### Solution

The FBD of this system for the applied forces is the same as that in Example 6.1. A pair of reaction forces caused by the massless link is added to the FBD along the axis of the link acting at points  $D$  and  $O$ , as shown in Figure 6.4. The directions of the pair of reaction forces are arbitrary. The actual directions will be determined based on the sign of the directional magnitude  $\lambda_1$ .

#### MATLAB/Chapter 6/Example \_ 6 \_ 2

We continue with the MATLAB® program from Example 6.1 and add the following data:

$$\mathbf{rD2} = [0.5; 4.5]; \quad \mathbf{Lc} = 4.53;$$

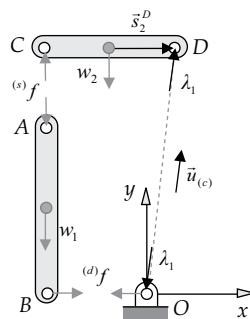
We construct a unit vector along the axis of the link and the moment arm for the reaction force acting at  $D$ . The unit vector will be used to form the pair of reaction forces as  $\vec{f}_2^D = \lambda_1 \vec{u}_{(c)}$  and  $\vec{f}_0^O = -\lambda_1 \vec{u}_{(c)}$ . The moment associated with the reaction force at  $D$  can be represented as  $\vec{n}_2 = \vec{s}_2^D \times \vec{f}_2^D = \lambda_1 \vec{s}_2^D \times \vec{u}_{(c)}$ .

$$\mathbf{uc} = \mathbf{rD2}/\text{norm}(\mathbf{rD2}) \quad \dots \dots \dots$$

$$\mathbf{sD2} = \mathbf{rD2} - \mathbf{rG2};$$

$$\mathbf{n2} = \mathbf{s\_rot}(\mathbf{sD2})' * \mathbf{uc} \quad \dots \dots \dots$$

$$\begin{aligned} \mathbf{uc} &= \\ &0.1104 \\ &0.9939 \\ \mathbf{n2} &= \\ &0.9939 \end{aligned}$$

**FIGURE 6.4**

Addition of reaction forces associated with the massless link.

We can now construct the EQMs as

$$\left\{ \begin{array}{c} 2.0 \\ 2.0 \\ 1.5 \\ 1.5 \\ 1.5 \\ 0.5 \end{array} \right| \left\{ \begin{array}{c} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\phi}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{\phi}_2 \end{array} \right\} = \left\{ \begin{array}{c} 40.000 \\ -69.620 \\ 60.000 \\ 0 \\ 35.285 \\ -50.000 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0.1104 \\ 0.9939 \\ 0.9939 \end{array} \right\} \lambda_1$$

These are six algebraic equations and seven unknowns: six unknown accelerations and one unknown multiplier. To solve these equations for the unknowns, additional information is needed.

### 6.1.3 Sliding Pendulum

Consider the sliding pendulum shown in Figure 6.5. This is a *constrained* system due to the presence of kinematic joints. Body (1), the slider, is connected to the ground by a frictionless sliding joint, and to body (2), the pendulum, by a pin joint. A spring is attached between the ground and body (1), and the gravity acts on the system.

#### Example 6.3

For the sliding pendulum of Figure 6.5, consider the following data:

$$m_1 = 5.0 \text{ m}, J_1 = 4.0 \text{ kg m}^2, m_2 = 2.0 \text{ kg}, J_2 = 0.2 \text{ kg m}^2, k = 20 \text{ N/m}, {}^0L = 0.6 \text{ m}$$

In the configuration shown in the figure, the following coordinates are provided:

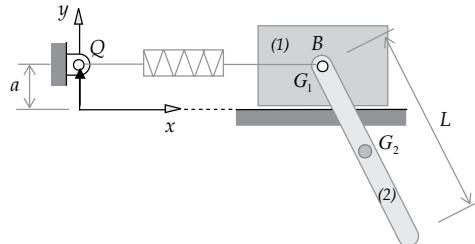
$$\mathbf{r}_1^B = \mathbf{r}_1^G = \left\{ \begin{array}{c} 1.0 \\ 0.2 \end{array} \right\} \text{ m}, \mathbf{r}_2^G = \left\{ \begin{array}{c} 1.25 \\ -0.233 \end{array} \right\} \text{ m}, a = 0.2 \text{ m}$$

Construct (a) the FBD and (b) the EQM for this system.

#### Solution

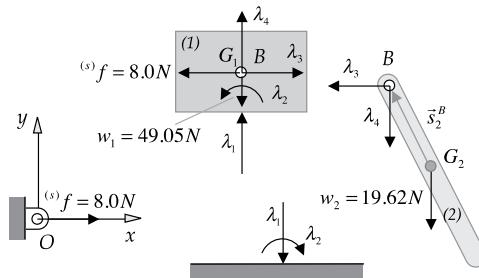
Due to the simplicity of the system, we can perform the necessary calculations by hand. The gravitational forces are as follows:  $w_1 = (5.0)(9.81) = 49.05 \text{ N}$  and  $w_2 = (2.0)(9.81) = 19.62 \text{ N}$ . The spring having a deformed length of  $L = 1.0 \text{ m}$  produces a force  ${}^{(s)}f = 20(1.0 - 0.6) = 8.0 \text{ N}$  that acts on body (1) in the negative  $x$ -direction.

The reaction force and torque associated with the sliding joint between the ground and the slider are denoted as  $\lambda_1$  and  $\lambda_2$ , respectively. The components of the reaction force for the



**FIGURE 6.5**

A sliding pendulum.



**FIGURE 6.6**  
FBD for the sliding pendulum.

pin joint are marked as  $\lambda_3$  and  $\lambda_4$ . The reaction forces and torques are indexed consecutively for easy reference. The applied and reaction forces are depicted on the FBD of Figure 6.6. In this figure, the reaction forces and torque that act on the ground are also shown.

Based on the FBD, we construct the arrays of applied and reaction force, and then the EQMs as

$$\left[ \begin{array}{ccccc} 5.0 & & & & \\ & 5.0 & & & \\ & & 4.0 & & \\ & & & 2.0 & \\ & & & & 2.0 \\ & & & & & 0.2 \end{array} \right] \left\{ \begin{array}{c} \dot{x}_1 \\ \dot{y}_1 \\ \ddot{\phi}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \ddot{\phi}_2 \end{array} \right\} = \left[ \begin{array}{c} -8.0 \\ -49.05 \\ 0 \\ 0 \\ -19.62 \\ 0 \end{array} \right] + \left[ \begin{array}{c} \lambda_3 \\ \lambda_1 + \lambda_4 \\ \lambda_2 \\ -\lambda_3 \\ -\lambda_4 \\ 0.433\lambda_3 + 0.25\lambda_4 \end{array} \right]$$

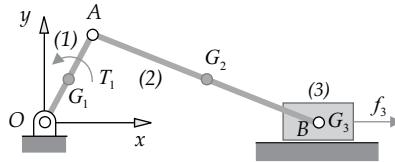
We note that the axes of all the forces that act on body (1) pass through the mass center, and therefore, they do not have moment arms. The only torque that acts on this body is the reaction torque  $\lambda_2$ . The reaction force from the pin joint that acts on body (2) has a moment arm  $\vec{s}_2^B$ , which results in a moment in the rotational equation of this body. The components of the moment arm can be computed as  $\vec{s}_2^B = \mathbf{r}_1^G - \mathbf{r}_2^G = \{-0.25 \quad 0.433\}'$  m. The moment associated with the reaction force is determined as  $\vec{s}_2^B(-\lambda_{3,4}) = \{-0.433 \quad -0.25\} \begin{Bmatrix} -\lambda_3 \\ -\lambda_4 \end{Bmatrix}$  N.

The array of reaction forces can be expressed in another form:

$$\left\{ \begin{array}{c} \lambda_3 \\ \lambda_1 + \lambda_4 \\ \lambda_2 \\ -\lambda_3 \\ -\lambda_4 \\ 0.433\lambda_3 + 0.25\lambda_4 \end{array} \right\} \Rightarrow \left[ \begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0.433 & 0.25 \end{array} \right] \left\{ \begin{array}{c} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{array} \right\}$$

$$\Rightarrow \left[ \begin{array}{cc|c} \mathbf{u}_{(y)} & 0 & \mathbf{I} \\ 0 & 1 & 0 \\ 0 & 0 & -\mathbf{I} \\ 0 & 0 & -\vec{s}_2^B \end{array} \right] \left\{ \begin{array}{c} \lambda_1 \\ \lambda_2 \\ \lambda_{3,4} \end{array} \right\}$$

The coefficient matrix of the reaction forces, as we will see in Chapter 7, is the transpose of the constraint Jacobian matrix.

**FIGURE 6.7**

Several types of applied loads acting on a slider–crank mechanism.

### 6.1.4 Slider-Crank Mechanism

Consider the slider–crank mechanism shown in Figure 6.7 on which an applied torque acts on link (1), an applied force acts on link (3), and the gravity acts on all three links. In the following example, we learn the steps that need to be followed for constructing the FBD and EQM for the system.

#### Example 6.4

We consider the slider–crank from Example 5.1. The following values for the mass and moment of inertia of the three bodies are given:

$$m_1 = 1.0 \text{ kg}, J_1 = 0.001 \text{ kg m}^2, m_2 = 2.0 \text{ kg}, J_2 = 0.01 \text{ kg m}^2, m_3 = 4.0 \text{ kg}, J_3 = 0.02 \text{ kg m}^2$$

A force with the magnitude 50 N acts on the slider in the positive horizontal direction, and an unknown torque acts on link (1). In the configuration where the crank angle is 65°, construct (a) the FBD and (b) the EQM.

#### Solution

(a) We continue with the MATLAB program from Example 5.1. For the given angle of the crank, we have already solved the constraint equations for the other unknowns.

#### MATLAB/Chapter 6/Example \_ 6 \_ 4

We add to the program the inertial data and the known applied force.

```

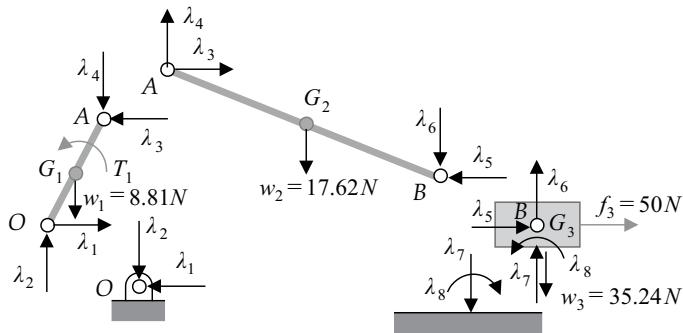
...
m1 = 1.0; J1 = 0.001; m2 = 2.0; J2 = 0.01;
m3 = 4.0; J3 = 0.02;
g = 9.81; uy = [0; 1]; fB = [50; 0];

```

With the gravitational forces  $w_1 = 9.81 \text{ N}$ ,  $w_2 = 19.62 \text{ N}$ , and  $w_3 = 39.24 \text{ N}$ , we construct the FBD as shown in Figure. 6.8.

(b) The EQM can be constructed based on the FBD as

$$\mathbf{M}\ddot{\mathbf{c}} = \left\{ \begin{array}{c} -w_1 \mathbf{u}_{(y)} \\ T_1 \\ -w_2 \mathbf{u}_{(y)} \\ 0 \\ -w_3 \mathbf{u}_{(y)} + \mathbf{f}_3 \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} \lambda_{1,2} - \lambda_{3,4} \\ \check{s}_1'^O \lambda_{1,2} - \check{s}_1'^A \lambda_{3,4} \\ \lambda_{3,4} - \lambda_{5,6} \\ \check{s}_2'^A \lambda_{3,4} - \check{s}_2'^B \lambda_{5,6} \\ \lambda_{5,6} + \mathbf{u}_{(y)} \lambda_7 \\ \lambda_8 \end{array} \right\}$$



**FIGURE 6.8**  
FBD for the slider–crank mechanism.

In this equation, due to the simplicity of the mass matrix, we have not shown the left-hand side in expanded form. The array of reaction forces can be expressed as a coefficient matrix times the array of the components of reaction forces:

$${}^{(r)}\mathbf{h} = \left[ \begin{array}{c|c|c|c|c} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \check{\mathbf{s}}_1'^O & -\check{\mathbf{s}}_1'^A & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \check{\mathbf{s}}_2'^A & -\check{\mathbf{s}}_2'^B & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{u}_{(y)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{array} \right] \left[ \begin{array}{c} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_7 \\ \lambda_8 \end{array} \right] \Rightarrow \mathbf{D}'\boldsymbol{\lambda}$$

where

$$\lambda_{1,2} = \left\{ \begin{array}{c} \lambda_1 \\ \lambda_2 \end{array} \right\}, \lambda_{3,4} = \left\{ \begin{array}{c} \lambda_3 \\ \lambda_4 \end{array} \right\}, \lambda_{5,6} = \left\{ \begin{array}{c} \lambda_5 \\ \lambda_6 \end{array} \right\}$$

The coefficient matrix of the reaction forces, as we will see in Chapter 7, is the transpose of the so-called Jacobian matrix,  $\mathbf{D}$ , associated with the kinematic constraints.

We now construct the array of *known* applied forces. We note that none of these forces have moment arms.

```
w1 = -g*m1*uy; w2 = -g*m2*uy; w3 = -g*m3*uy;
h_a = [w1; 0; w2; 0; (w3 + fB); 0] ..... .
```

```
h_a =
0
-9.8100
0
0
-19.6200
0
50.0000
-39.2400
0
```

Since we do not have a value for the applied torque on the crank, we cannot include it in the computed array of applied forces.

For the coefficient matrix of reaction forces, we compute the moment arms.

```
sO1 = -(L1/2)*u1; sO1r = s_rot(sO1);
sA1 = (L1/2)*u1; sA1r = s_rot(sA1);
sA2 = -(L2/2)*u2; sA2r = s_rot(sA2);
sB2 = (L2/2)*u2; sB2r = s_rot(sB2);
```

The computed values of the moment arms are as follows:

$$\begin{aligned}\mathbf{s}_2^O &= \begin{Bmatrix} -0.0254 \\ -0.0544 \end{Bmatrix} \text{m}, \quad \mathbf{s}_1^A = \begin{Bmatrix} 0.0254 \\ 0.0544 \end{Bmatrix} \text{m}, \\ \mathbf{s}_2^A &= \begin{Bmatrix} -0.1181 \\ 0.0544 \end{Bmatrix} \text{m}, \quad \mathbf{s}_2^B = \begin{Bmatrix} 0.1181 \\ -0.0544 \end{Bmatrix} \text{m}\end{aligned}$$

Next we construct the coefficient matrix of the reaction forces.

```
I2 = eye(2); Z2 = zeros(2);
Z12 = zeros(1,2); Z21 = zeros(2,1);
Dt = [I2 -I2 Z2 Z21 Z21
      sO1r' -sA1r' Z12 0 0
      Z2 I2 -I2 Z21 Z21
      Z12 sA2r' -sB2r' 0 0
      Z2 Z2 I2 uy Z21
      Z12 Z12 Z12 0 1] .....
Dt =
1.000 0 -1.000 0 0 0 0 0
0 1.000 0 -1.000 0 0 0 0
0.054 -0.025 0.054 -0.025 0 0 0 0
0 0 1.000 0 -1.000 0 0 0
0 0 0 1.000 0 -1.000 0 0
0 0 -0.054 -0.118 -0.054 -0.118 0 0
0 0 0 0 1.000 0 0 0
0 0 0 0 0 1.000 1.000 0
0 0 0 0 0 0 0 1.000
```

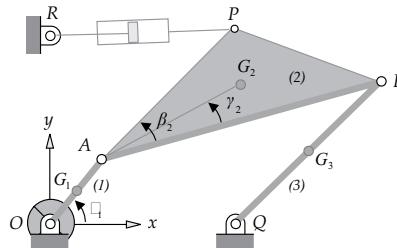
The diagonal mass matrix can be constructed as an array.

```
M_diag = [m1 m1 J1 m2 m2 J2 m3 m3 J3]'; M = diag(M_diag);
```

We have now evaluated all the known arrays and matrices of the EQMs.

### 6.1.5 Four-Bar Mechanism

The four-bar mechanism is another example of a closed-chain system that we can consider for analysis. For the four-bar linkage shown in Figure 6.9, there is a motor between link (1) and the ground about the axis of pin joint  $O$ , a damper connects point  $P$  of link (2) to point  $R$  on the ground, and the gravity also acts on the system. The motor applies a torque  $(\omega)_T$  on link (1) and an opposite torque but with the same magnitude on the ground. We also know that the crank has a known angular velocity and acceleration at a given orientation.



**FIGURE 6.9**  
A four-bar linkage with a motor and a damper.

### Example 6.5

Most of the geometric data for this mechanism are provided in Example 5.2. The position of the mass center of link (2) is defined by the length  $L_{G_2, A} = 0.16$  m and  $\gamma_2 = 12^\circ$ . Additional data for the mass and moment of inertia of the links are as follows:  $m_1 = 1.0\text{ kg}$ ,  $J_1 = 0.001\text{ kg m}^2$ ,  $m_2 = 6.0\text{ kg}$ ,  $J_2 = 0.1\text{ kg m}^2$ ,  $m_3 = 4.0\text{ kg}$ , and  $J_3 = 0.02\text{ kg m}^2$ . The damper has a damping coefficient of  $d_c = 125\text{ N s/m}$ . The attachment point of the damper to the ground, point R, has a distance from O as  $L^{R, O} = 0.2\text{ m}$ . The torque of the motor that is applied on link (1) is unknown. In the configuration shown,  $\theta_1 = \pi/4$ ,  $\dot{\theta}_1 = 1.5\text{ rad/s}$ ,  $\ddot{\theta}_1 = -1.2\text{ rad/s}^2$ . Construct the FBD and the EQM.

#### Solution

For the given values of the angle, velocity, and acceleration of link (1), we can determine all the needed coordinates and velocities from the program we developed in Example 5.2. To determine the force of the damper, we need the coordinates and velocity of point P, which were found to be

$$\mathbf{r}_2^P = \begin{Bmatrix} 0.2107 \\ 0.2136 \end{Bmatrix} \text{ m}, \quad \dot{\mathbf{r}}_2^P = \begin{Bmatrix} -0.1102 \\ 0.1101 \end{Bmatrix} \text{ m/s}$$

#### MATLAB/Chapter 6/Example \_ 6 \_ 5

We add the following data to the program from Example 5.2:

```

...
LG2A = 0.16; gamma2 = 12*pi/180;
m1 = 1.0; J1 = 0.001; m2 = 6.0; J2 = 0.1;
m3 = 3.0; J3 = 0.015;
g = 9.81; dc = 125; LRO = 0.2; uy = [0; 1];

```

To compute the force of the damper, we follow Eqs. (4.22)–(4.24) and (4.28)–(4.30):

```

rR = [0; LRO];
d = rP - rR; L = norm(d); u = d/L;
L_d = u'*rP_d ..... .
f_d = dc*L_d ..... .
fP2 = -f_d*u ..... .

```

```

L_d =
-0.1028
f_d =
-12.8555
fP2 =
12.8288
0.8273

```

The computed value for  $\dot{L}$  is negative, meaning that the damper is shortening.

Therefore, it must oppose the motion by applying a pair of *push* forces on the attachment points. The FBD is constructed as shown in Figure 6.10.

The EQMs can now be constructed as

$$\mathbf{M}\ddot{\mathbf{c}} = \left\{ \begin{array}{l} -w_1 \mathbf{u}_{(y)} \\ (a)T \\ -w_2 \mathbf{u}_{(y)} + (d)\mathbf{f}_2^P \\ \check{\mathbf{s}}_2'^P (d)\mathbf{f}_2^P \\ -w_3 \mathbf{u}_{(y)} \\ 0 \end{array} \right\} + \left\{ \begin{array}{l} \lambda_{1,2} - \lambda_{3,4} \\ \check{\mathbf{s}}_1'^O \lambda_{1,2} - \check{\mathbf{s}}_1'^A \lambda_{3,4} \\ \lambda_{3,4} - \lambda_{5,6} \\ \check{\mathbf{s}}_2'^A \lambda_{3,4} - \check{\mathbf{s}}_2'^B \lambda_{5,6} \\ \lambda_{5,6} + \lambda_{7,8} \\ \check{\mathbf{s}}_3'^B \lambda_{5,6} + \check{\mathbf{s}}_3'^Q \lambda_{7,8} \end{array} \right\} \quad (6.1)$$

The array of reaction forces can be expressed in matrix form as

$$(r)\mathbf{h} = \left[ \begin{array}{c|c|c|c} \mathbf{I} & -\mathbf{I} & 0 & 0 \\ \check{\mathbf{s}}_1'^O & -\check{\mathbf{s}}_1'^A & 0 & 0 \\ \hline 0 & \mathbf{I} & -\mathbf{I} & 0 \\ 0 & \check{\mathbf{s}}_2'^A & -\check{\mathbf{s}}_2'^B & 0 \\ \hline 0 & 0 & \mathbf{I} & \mathbf{I} \\ 0 & 0 & \check{\mathbf{s}}_3'^B & \check{\mathbf{s}}_3'^Q \end{array} \right] \left\{ \begin{array}{l} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_{7,8} \end{array} \right\} \Rightarrow \mathbf{D}'\boldsymbol{\lambda}$$

where

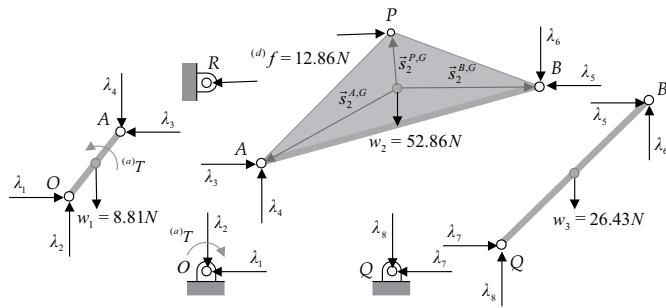
$$\lambda_{1,2} = \left\{ \begin{array}{l} \lambda_1 \\ \lambda_2 \end{array} \right\}, \lambda_{1,2} = \left\{ \begin{array}{l} \lambda_1 \\ \lambda_2 \end{array} \right\}, \lambda_{3,4} = \left\{ \begin{array}{l} \lambda_3 \\ \lambda_4 \end{array} \right\}, \lambda_{5,6} = \left\{ \begin{array}{l} \lambda_5 \\ \lambda_6 \end{array} \right\}, \lambda_{7,8} = \left\{ \begin{array}{l} \lambda_7 \\ \lambda_8 \end{array} \right\}$$

To evaluate the arrays and matrices of the EQM, we need to compute the moment arms for the damping force and the reaction forces.

```
rA = L1*u1; rB = rA + L2*u2; rQ = [L0; 0];
uG2A = [cos(theta2 + gamma2); sin(theta2 + gamma2)];
sG2 = LG2A*uG2A;
rG1 = (L1/2)*u1; rG2 = rA + sG2; rG3 = rQ + (L3/2)*u3;
sO1 = -(L1/2)*u1; sO1r = s_rot(sO1);
sA1 = (L1/2)*u1; sA1r = s_rot(sA1);
sA2 = rA - rG2; sA2r = s_rot(sA2);
sB2 = rB - rG2; sB2r = s_rot(sB2);
sP2 = rP - rG2; sP2r = s_rot(sP2);
sB3 = (L3/2)*u3; sB3r = s_rot(sB3);
sQ3 = -(L3/2)*u3; sQ3r = s_rot(sQ3);
```

The computed moment arms are as follows:

$$\begin{aligned} \mathbf{s}_1^O &= \left\{ \begin{array}{l} -0.0354 \\ -0.0354 \end{array} \right\} \text{m}, \mathbf{s}_1^A = \left\{ \begin{array}{l} 0.0354 \\ 0.0354 \end{array} \right\} \text{m}, \mathbf{s}_2^A = \left\{ \begin{array}{l} -0.1418 \\ -0.0741 \end{array} \right\} \text{m}, \mathbf{s}_2^B = \left\{ \begin{array}{l} 0.1472 \\ 0.0688 \end{array} \right\} \text{m}, \\ \mathbf{s}_2^P &= \left\{ \begin{array}{l} -0.0018 \\ 0.0688 \end{array} \right\} \text{m}, \mathbf{s}_3^B = \left\{ \begin{array}{l} 0.0798 \\ 0.0757 \end{array} \right\} \text{m}, \mathbf{s}_3^Q = \left\{ \begin{array}{l} -0.0798 \\ -0.0757 \end{array} \right\} \text{m} \end{aligned}$$

**FIGURE 6.10**

The FBD for the four-bar mechanism.

Next we evaluate the array of applied forces, without including the applied torque on link (1) since it is unknown.

$h\_a = [w_1; 0; (w_2 + fP2); sP2r' * fP2; w_3; 0] \dots$	$h\_a =$ 0 $-9.8100$ 0 $12.8288$ $-58.0327$ $-0.8837$ 0 $-29.4300$ 0
---	---

The coefficient matrix for the reaction forces is evaluated next:

```

I2 = eye(2); Z2 = zeros(2);
Z12 = zeros(1,2); Z21 = zeros(2,1);
Dt = [I2 -I2 Z2 Z2
      sO1r' -sA1r' Z12 Z12
      Z2 I2 -I2 Z2
      Z12 sA2r' -sB2r' Z12
      Z2 Z2 I2 I2
      Z12 Z12 sB3r' sQ3r'] .....

```

$Dt =$ 1.000 0 -1.000 0 0 0 0 0 0 1.000 0 -1.000 0 0 0 0 $0.035 -0.035 0.035 -0.035 0 0 0 0$ 0 0 1.000 0 -1.000 0 0 0 0 0 0 1.000 0 -1.000 0 0 0 0 0.074 -0.142 0.006 -0.147 0 0 0 0 0 0 1.000 0 1.000 0 0 0 0 0 0 1.000 0 1.000 0 0 0 0 -0.076 0.079 0.076 -0.079
---

Finally, we construct the mass matrix as an array before converting it to a diagonal matrix.

```
M_diag = [m1 m1 J1 m2 m2 J2 m3 m3 J3]'; M = diag(M_diag);
```

We have evaluated all the known arrays and matrices in the EQM. We will determine the unknown applied torque and reaction forces in Section 6.4.

## 6.2 Equations of Motion

In the examples of Section 6.1, we derived the EQMs for several simple multibody systems. A comparison between these equations reveals that every set of EQMs contains a mass (or inertia) matrix, an array of accelerations, an array of applied forces, and an array of reaction forces except for the systems that do not have kinematic joints. Therefore, we can generalize the construction of these equations for unconstrained and constrained multibody systems.

Let us assume that there are  $n_b$  bodies in the system of interest. For an *unconstrained* system, where there are no kinematic joints, for each body of the system, we can write the EQMs based on Eq. (4.15) to form  $3n_b$  equations as

$$\mathbf{M}_i \ddot{\mathbf{c}}_i = {}^{(a)}\mathbf{h}_i; \quad i = 1, \dots, n_b \quad (6.2)$$

Appending these equations yields the EQMs for the system in a single matrix equation as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} \quad (6.3)$$

where  $\mathbf{M}$  is a  $3n_b \times 3n_b$  constant and diagonal mass matrix,  $\ddot{\mathbf{c}}$  is a  $3n_b \times 1$  array of accelerations, and  ${}^{(a)}\mathbf{h}$  is a  $3n_b \times 1$  array of applied forces.

For a system of  $n_b$  *constrained* bodies, a system that contains kinematic joints or any form of conditions on its coordinates, Eq. (6.2) must be revised to include the reaction forces that act on a body as

$$\mathbf{M}_i \ddot{\mathbf{c}}_i = {}^{(a)}\mathbf{h}_i + {}^{(r)}\mathbf{h}_i; \quad i = 1, \dots, n_b \quad (6.4)$$

Appending these equations results in the EQMs for the system that can be expressed in a single matrix equation as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + {}^{(r)}\mathbf{h} \Rightarrow \mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + \mathbf{D}'\boldsymbol{\lambda} \quad (6.5)$$

where  ${}^{(r)}\mathbf{h}$  is a  $3n_b \times 1$  array of reaction forces. Assuming that there are  $n_c = 3n_b - n_{dof}$  constraints in the system,  $\mathbf{D}'$  is a  $3n_b \times n_c$  matrix, and  $\boldsymbol{\lambda}$  is an  $n_c \times 1$  array containing all components of the reaction forces and reaction torques.

## 6.3 Force Analysis

A classical method of force analysis, based on the FBD and the EQMs of a system, can be applied to a mechanism (a closed-chain system of one degree of freedom) to determine an unknown applied force or torque that has resulted in a specified motion. The specified motion of a mechanism is obtained from a kinematic analysis as discussed in Section 5.3. Knowing the coordinates, velocities, and accelerations for all the links of a mechanism, the EQMs can be solved to determine the reaction forces and the unknown torque of a motor

or the unknown force of an actuator. This method of force analysis, which is called inverse dynamics, will be discussed in more detail in Chapter 12. In this section, we review this classical method of force analysis through two examples.

### 6.3.1 Slider-Crank Mechanism

We continue with the slider–crank mechanism from Section 6.1.4. Assume that the angular velocity and acceleration of the crank, at a given angle, are known. Based on this information, a kinematic analysis provides the position, velocity, and acceleration of other two links, as seen in Section 5.3.1. With known kinematics of the system, the constructed EQMs can be solved to determine the unknown applied torque and the reaction forces.

#### Example 6.6

As a continuation of Example 6.4, assume that for a crank angle of  $65^\circ$ , the angular velocity of the crank is  $1.6 \text{ rad/s}$  and constant. Determine the unknown applied torque that acts on the crank.

#### Solution

For the given angle, velocity, and acceleration of the crank, a kinematic analysis provides the coordinates, velocities, and accelerations for all the links, including the acceleration of the mass centers. We have already constructed the EQMs in Example 6.4 as

$$\left\{ \begin{array}{c} m_1 \ddot{\mathbf{r}}_1 \\ J_1 \ddot{\phi}_1 \\ m_2 \ddot{\mathbf{r}}_2 \\ J_2 \ddot{\phi}_2 \\ m_3 \ddot{\mathbf{r}}_3 \\ J_3 \ddot{\phi}_3 \end{array} \right\} = \left\{ \begin{array}{c} -w_1 \mathbf{u}_{(y)} \\ T_1 \\ -w_2 \mathbf{u}_{(y)} \\ 0 \\ -w_3 \mathbf{u}_{(y)} + \mathbf{f}_3^B \\ 0 \end{array} \right\} + \left[ \begin{array}{c|c|c|c|c|c} \mathbf{I} & -\mathbf{I} & 0 & 0 & 0 & 0 \\ \check{\mathbf{s}}_1^O & -\check{\mathbf{s}}_1^A & 0 & 0 & 0 & 1 \\ \hline 0 & \mathbf{I} & -\mathbf{I} & 0 & 0 & 0 \\ 0 & \check{\mathbf{s}}_2^A & -\check{\mathbf{s}}_2^B & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{I} & \mathbf{u}_{(y)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right] \left\{ \begin{array}{c} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_7 \\ \lambda_8 \end{array} \right\}$$

Since the accelerations of the mass centers are known, the EQMs can be rearranged as

$$\left[ \begin{array}{c|c|c|c|c|c} \mathbf{I} & -\mathbf{I} & 0 & 0 & 0 & 0 \\ \check{\mathbf{s}}_1^O & -\check{\mathbf{s}}_1^A & 0 & 0 & 0 & 1 \\ \hline 0 & \mathbf{I} & -\mathbf{I} & 0 & 0 & 0 \\ 0 & \check{\mathbf{s}}_2^A & -\check{\mathbf{s}}_2^B & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{I} & \mathbf{u}_{(y)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right] \left\{ \begin{array}{c} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_7 \\ \lambda_8 \\ T_1 \end{array} \right\} = \left\{ \begin{array}{c} m_1 \ddot{\mathbf{r}}_1 + w_1 \mathbf{u}_{(y)} \\ J_1 \ddot{\phi}_1 \\ m_2 \ddot{\mathbf{r}}_2 + w_2 \mathbf{u}_{(y)} \\ J_2 \ddot{\phi}_2 \\ m_3 \ddot{\mathbf{r}}_3 + w_3 \mathbf{u}_{(y)} - \mathbf{f}_3^B \\ J_3 \ddot{\phi}_3 \end{array} \right\}$$

In this equation, we have moved all the unknowns and their coefficients to the left-hand side, and moved all the known terms to the right-hand side. In this process, we have also appended the unknown torque,  $T_1$ , to the array of reaction forces by adding a ninth column to the matrix of coefficients. With this arrangement, we have constructed nine linear algebraic equations that can be solved for the nine unknowns—eight reaction force components and torque, and one applied torque.

**MATLAB/Chapter 6/Example \_ 6 \_ 6**

In an extension to the program from Example 6.4, we first compute the coordinates of the mass centers and points *A* and *B*. We then compute the acceleration of the mass centers, construct the array of accelerations, and then evaluate the array of mass times accelerations.

```

...
rG1 = (L1/2)*u1; rA = L1*u1;
rG2 = rA + (L2/2)*u2; rB = [theta3; 0];
rG1_dd = s_rot(rG1)*theta1_dd - rG1*theta1_d^2;
rG2_dd = s_rot(rA)*theta1_dd - rA*theta1_d^2 - ...
          sA2r*theta2_dd + sA2*theta2_d^2;
rG3_dd = [theta3_dd; 0];
c_dd = [rG1_dd; theta1_dd; rG2_dd; theta2_dd; ...
          rG3_dd; theta3_dd];
Mcdd = M_diag.*c_dd;
```

We add a ninth column to the coefficient matrix of the reaction forces, containing all zeros with the exception of the third row, which is a “1”.

```
column9 = zeros(9,1); column9(3) = 1.0;
Dt9 = [Dt column9];
```

We solve the equations for the unknowns, where the ninth unknown is the applied torque on the crank.

```
sol = Dt9\ (Mcdd - h_a);
T_a = sol(9) ..... T_a =
7.3810
```

In this process, we have also solved for the reaction forces that act between the links at this configuration of the mechanism.

### 6.3.2 Four-Bar Mechanism

The four-bar mechanism from Section 6.1.5 is considered here for force analysis, where the velocity and acceleration of the crank, at a given orientation, are provided. The main objective is to determine the torque of the motor that causes the specified motion of the crank. Based on the known motion of the crank, a kinematic analysis provides the position, velocity, and acceleration of the other links of the mechanism, as was performed in Section 5.3.2. With known kinematics of the system, the EQMs are solved to determine one unknown applied torque and the reaction forces.

#### Example 6.7

For the four-bar mechanism from Example 6.5, assume that for a crank angle of  $\theta_1 = \pi/4$  rad, the angular velocity and acceleration are  $\dot{\theta}_1 = 1.5$  rad/s and  $\ddot{\theta}_1 = -1.2$  rad/s<sup>2</sup>, respectively. Determine the unknown applied torque that acts on the crank.

#### *Solution*

For the given angle, velocity, and acceleration of the crank, a kinematic analysis can be performed to determine the angular acceleration of the links and the acceleration of the mass centers. The constructed EQMs from Example 6.5 are

$$\begin{bmatrix} m_1 \ddot{\mathbf{r}}_1 \\ J_1 \ddot{\phi}_1 \\ m_2 \ddot{\mathbf{r}}_2 \\ J_2 \ddot{\phi}_2 \\ m_3 \ddot{\mathbf{r}}_3 \\ J_3 \ddot{\phi}_3 \end{bmatrix} = \begin{bmatrix} -w_1 \mathbf{u}_{(y)} \\ {}^{(a)T} \\ -w_2 \mathbf{u}_{(y)} + {}^{(d)}\mathbf{f}_2^P \\ \tilde{\mathbf{s}}_2' P {}^{(d)}\mathbf{f}_2^P \\ -w_3 \mathbf{u}_{(y)} \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{s}}_1'^O & -\tilde{\mathbf{s}}_1'^A & 0 & 0 \\ 0 & \mathbf{I} & -\mathbf{I} & 0 \\ 0 & \tilde{\mathbf{s}}_2'^A & -\tilde{\mathbf{s}}_2'^B & 0 \\ 0 & 0 & 0 & \mathbf{I} \\ 0 & 0 & 0 & \tilde{\mathbf{s}}_3'^B \end{bmatrix} \begin{bmatrix} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_{7,8} \end{bmatrix}$$

With known accelerations, the EQMs can be rearranged as

$$\begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{s}}_1'^O & -\tilde{\mathbf{s}}_1'^A & 0 & 0 & 1 \\ 0 & \mathbf{I} & -\mathbf{I} & 0 & 0 \\ 0 & \tilde{\mathbf{s}}_2'^A & -\tilde{\mathbf{s}}_2'^B & 0 & 0 \\ 0 & 0 & \mathbf{I} & \mathbf{I} & 0 \\ 0 & 0 & \tilde{\mathbf{s}}_3'^B & \tilde{\mathbf{s}}_3'^Q & 0 \end{bmatrix} \begin{bmatrix} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_{7,8} \\ {}^{(a)T} \end{bmatrix} = \begin{bmatrix} m_1 \ddot{\mathbf{r}}_1 + w_1 \mathbf{u}_{(y)} \\ J_1 \ddot{\phi}_1 \\ m_2 \ddot{\mathbf{r}}_2 + w_2 \mathbf{u}_{(y)} - {}^{(d)}\mathbf{f}_2^P \\ J_2 \ddot{\phi}_2 - \tilde{\mathbf{s}}_2' P {}^{(d)}\mathbf{f}_2^P \\ m_3 \ddot{\mathbf{r}}_3 + w_3 \mathbf{u}_{(y)} \\ J_3 \ddot{\phi}_3 \end{bmatrix}$$

In this equation, we have moved all the unknowns and their coefficients to the left-hand side, and all the known terms to the right-hand-side. One additional column is appended to the coefficient matrix of the reaction forces in order to include the torque  ${}^{(a)T}$  as the ninth unknown. The resultant nine linear algebraic equations can be solved for the nine unknowns.

#### MATLAB/Chapter 6/Example \_ 6 \_ 7

We extend the program from Example 6.5 by computing the acceleration of the mass centers, constructing the array of accelerations, and then evaluating the array of mass times accelerations.

```
...
rG1_dd = s_rot(rG1)*theta1_dd - rG1*theta1_d^2;
rG2_dd = s_rot(rA)*theta1_dd - rA*theta1_d^2 - ...
          sA2r*theta2_dd + sA2*theta2_d^2;
rG3_dd = -s_rot(sQ3)*theta3_dd + sQ3*theta3_d^2;
c_dd    = [rG1_dd; theta1_dd; rG2_dd; ...
           theta2_dd; rG3_dd; theta3_dd];
Mcdd   = M_diag.*c_dd
```

We add a ninth column to the coefficient matrix of reaction forces containing zeros with the exception of the third row being a "1".

```
column9 = zeros(9,1); column9(3) = 1.0;
Dt9 = [Dt column9];
```

We solve the equations for the unknowns, where the ninth unknown is the applied torque on the crank.

```
sol = Dt9 \ (Mcdd - h_a);
T_a = sol(9) ..... | T_a =
6.6577
```

In this process, we have also found the reaction forces, which could be reported if needed.

### 6.3.3 Generalization of Force Analysis

The process of force analysis that was applied to a slider–crank and a four-bar mechanism in Sections 6.3.1 and 6.3.2, respectively, can be generalized for any multibody system with a *single* degree of freedom. We assume that the motion of *one* of the coordinates of the system is fully defined by a simple known time function, or *driver*, having continuous first and second time derivatives as

$$c_i = f(t), \dot{c}_i = \dot{f}(t), \ddot{c}_i = \ddot{f}(t) \quad (6.6)$$

where  $c_i$  is  $x_i$ ,  $y_i$ , or  $\phi_i$  of one of the bodies, and  $t$  is the time parameter. With the known driver equation, we perform a kinematic analysis at a given time  $t$ , as discussed in Section 5.3, to determine all the coordinates, velocities, and accelerations. Then we compute the acceleration of the mass centers.

With known accelerations, the EQMs for a system of  $n_b$  bodies from Eq. (6.5) can be rearranged as

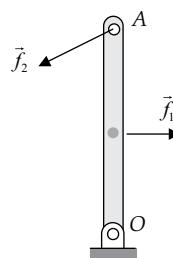
$$\mathbf{D}'\lambda + \mathbf{e}x = \mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h} \Rightarrow [\mathbf{D}' \quad \mathbf{e}] \begin{Bmatrix} \lambda \\ x \end{Bmatrix} = \mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h} \quad (6.7)$$

where the unknown applied force or torque, denoted as  $x$ , is moved out of the array of applied forces,  ${}^{(a)}\mathbf{h}$ , and included with the unknown reaction forces and torques. The coefficient  $\mathbf{e}$  denotes a  $3n_b \times 1$  zero array with one or possibly more nonzero coefficients in the row(s) associated with the EQM(s) in which  $x$  belongs. The coefficient matrix  $[\mathbf{D}' \quad \mathbf{e}]$  is  $3n_b \times 3n_b$ , that is, square, and therefore, Eq. (6.7) can be solved for the  $3n_b$  unknowns yielding the solution for the reaction forces as well as the unknown applied force or torque. This type of force analysis is presented in more detail in Chapter 12 as inverse dynamic analysis.

## 6.4 Problems

In the following problems, do not consider gravitational force unless it is stated.

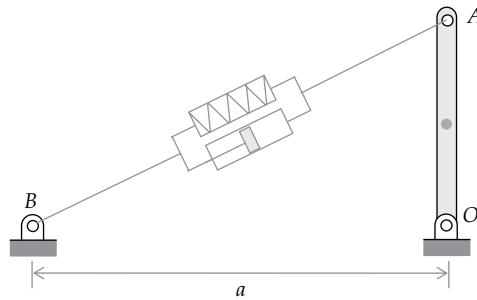
- 6.1 The rod shown is 0.3 m long and is pinned to the ground at one end. The rod has a mass  $m = 4.0 \text{ kg}$  and a moment of inertia  $J = 0.03 \text{ Nm}^2$ . A horizontal force of magnitude  $f_1 = 1.5 \text{ N}$  acts on the mass center and another force  $\mathbf{f}_2 = \{-2.0 \quad -1.0\}' \text{ N}$  acts at point A.



- Construct the FBD for the rod.
- Construct the EQMs.

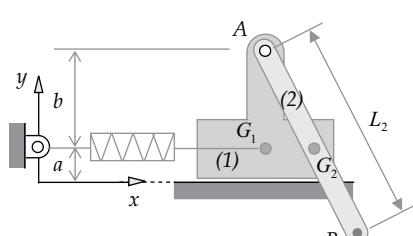
6.2 The rod shown is 0.3 m long and is pinned to the ground at one end. The rod has a mass  $m = 4.0 \text{ kg}$  and a moment of inertia  $J = 0.03 \text{ N m}^2$ . A spring-damper connects the other end of the rod to the ground as shown, where  $a = 0.6 \text{ m}$ . The force element has the following characteristics:  $k = 100 \text{ N/m}$ ,  $^0L = 0.8 \text{ m}$ ,  $d_c = 20 \text{ N s/m}$ . In the configuration shown, the rod has an angular velocity of  $1.0 \text{ rad/s}$  clockwise.

- Construct the FBD for the rod.
- Construct the EQMs.

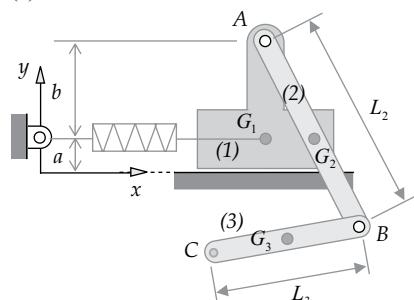


6.3 For each of the systems shown, consider the gravity as an applied force. Assume a deformed length for the spring  $L = 0.07 \text{ m}$ . Use the following data as applicable:

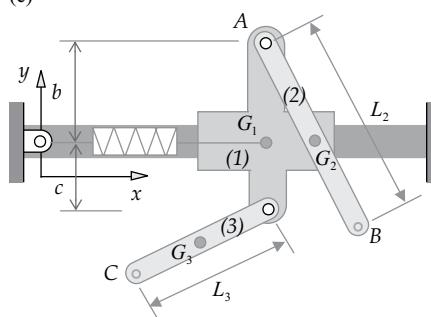
(a)



(b)



(c)



$$a = 0.01, b = 0.03, c = 0.02, L_2 = 0.06, L_3 = 0.04 \text{ m}$$

$$k = 100 \text{ N/m}, {}^0L = 0.08 \text{ m}$$

$$m_1 = 0.3, m_2 = 0.2, m_3 = 0.15 \text{ kg}, J_1 = 0.05, J_2 = 0.001, J_3 = 0.0008 \text{ kg m}^2$$

In the configurations shown, construct the FBD and the EQM.

- 6.4 For the offset slider–crank mechanism shown, assume the following data:

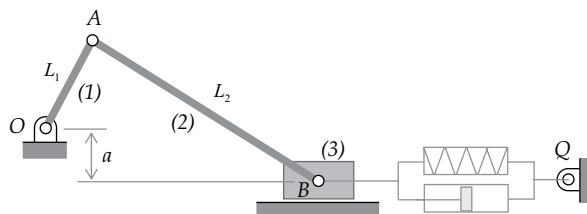
$$a = 0.05, L_1 = 0.12, L_2 = 0.26 \text{ m}$$

$$m_1 = 1.0, m_2 = 2.0, m_3 = 4.0 \text{ kg}, J_1 = 0.001, J_2 = 0.01, J_3 = 0.02 \text{ kg m}^2$$

The mass centers are at the geometric center of each link. A spring-damper that is connected between points  $B$  and  $Q$  has a deformed length of  $L = 0.25 \text{ m}$ , and it has the following characteristics:

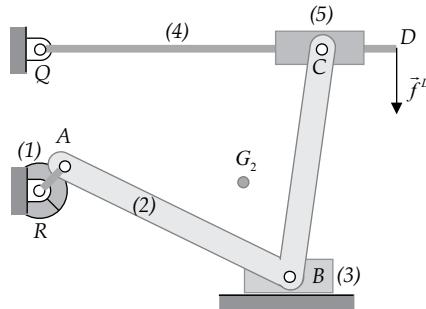
$$k = 80 \text{ L/m}, {}^0L = 0.2 \text{ m}, d_c = 5 \text{ N s/m}$$

In the configuration shown, the slider–block is moving to the left with a speed of  $\dot{x} = 2.4 \text{ m/s}$ . Construct the FBD and the EQM.



- 6.5 Revise the program for the slider–crank mechanism of Example 6.6. Perform a complete kinematic analysis by rotating the crank through a complete revolution in increments of  $\Delta\theta_1 = 5^\circ$ . Save the solution for the unknown torque that acts on the crank at each orientation. Plot the computed torque versus the crank angle.
- 6.6 Revise the program for the four-bar mechanism of Example 6.7 assuming a constant angular velocity of  $\dot{\theta}_1 = 2\pi \text{ rad/s}$  for the crank. Perform a complete kinematic analysis by rotating the crank through a full revolution, starting from  $\theta_1 = 45^\circ$  in increments of  $\Delta\theta_1 = 5^\circ$ . Save the solution for the unknown torque that acts on the crank at each orientation. Plot the computed torque versus the crank angle.
- 6.7 For the film-strip advancer system from Section 15.1, construct the FBD in a configuration when point  $P$  is engaged with the film strip.
- 6.8 For the web-cutter mechanism from Section 15.2, construct the FBD in a configuration when the blades are in the process of cutting the web. Consider the gravity acting on the system.

- 6.9 For the double A-arm suspension system from Section 15.6, construct the FBD. Assume the tire is resting on the ground having a deformed radius as stated, and consider the gravity as an applied force.
- 6.10 For the MacPherson suspension system from Section 15.7, construct the FBD. Assume the tire is resting on the ground having a deformed radius as stated. Consider the gravity acting on the system.
- 6.11 For the six-bar quick-return mechanism of Section 15.3, construct the FBD.
- 6.12 For the six-bar dwell mechanism of Section 15.4, construct the FBD.
- 6.13 In this six-bar mechanism, a motor rotates link (1) and a force  $\vec{f}^D$  acts at point D of link (4). The gravity also acts on the system. The mass centers are at the geometric centers of each link, except for link (2) which is at  $G_2$ . Construct the FBD for the links of this system.



# 7

---

## *Body-Coordinate Formulation*

---

Body-coordinate formulation is a method suitable for implementation in a computer program; the method is not recommended for the pencil and paper approach. This method can systematically generate the kinematic and dynamic equations of motion for a wide variety of mechanical systems. The number of equations generated by this method is by far much larger than those generated by other formulations for the same multibody system.

In the body-coordinate formulation, equal number of coordinates is defined for each rigid body, regardless of the kinematic joints that may exist in the system. Kinematic constraints describing the joints are defined between the body coordinates. The mass matrix, the array of applied forces, and the array of reaction forces are constructed to complete the equations of motion for a multibody system. The equations of motion that are constructed by the body-coordinate method are exactly the same as those equations generated based on the free-body-diagram approach, as discussed in Chapter 6. The main difference is that the body-coordinate method generates the equations systematically without the need to construct a free-body diagram (FBD) first.

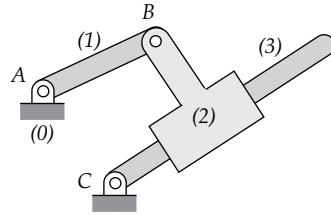
Based on the body-coordinate formulation, a general-purpose computer program is presented in Chapter 8. The program will be used to model a variety of planar systems and simulate their dynamics.

---

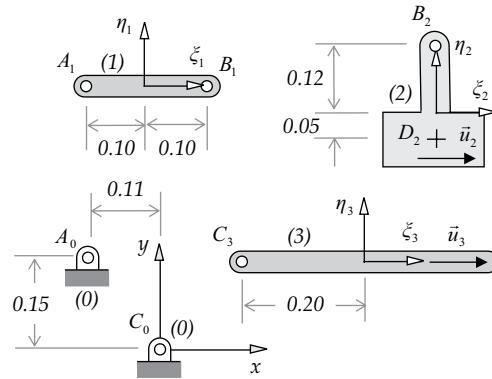
### 7.1 General Procedure

The process of describing a multibody system as a collection of interconnected bodies can be illustrated with a simple generic example. Assume the planar multibody system shown in Figure 7.1. Because at this stage the system is considered only for kinematic modeling, no force elements are shown in the figure. We apply the following steps as a rule of thumb in the kinematic modeling process.

- We assign indices (numbers) to each body. If the ground is part of the system, we assign it index (0). We number the other bodies consecutively as (1), (2), (3), and so forth.
- We separate the bodies at the kinematic joints and position them in a convenient orientation, such as the bodies shown in Figure 7.2.
- We attach a reference frame  $x-y$  to body (0) at a convenient location; in this example, the origin is attached to the pin joint at C.
- We attach body-fixed  $\xi-\eta$  frames to each moving body. The origin of these frames can be located anywhere on the corresponding bodies. However, since for dynamic analysis the origin of these frames must be positioned at the mass center of each body, as it will be discussed in Section 7.3, we should do the same for kinematic

**FIGURE 7.1**

A generic planar system containing three moving bodies and the ground.

**FIGURE 7.2**

Mechanism of Figure 7.1 viewed as an unconstrained system.

analysis. The  $\xi$ -axis of each frame can be oriented in any desired direction. It is a good practice to orient the  $\xi$ -axis (or the  $\eta$ -axis) along a geometric axis (or an existing joint axis, if any) on the body.

The three moving bodies and the ground, without considering the existence of the joints, form an unconstrained system. The  $x$  and  $y$  coordinates of the origin of each  $\xi$ - $\eta$  frame describe the translational coordinates of that body. The angle between each  $\xi$ -axis and the  $x$ -axis describes the rotational coordinate of that body. These coordinates are arranged in three arrays as

$$\mathbf{c}_1 = \begin{Bmatrix} x_1 \\ y_1 \\ \phi_1 \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} x_2 \\ y_2 \\ \phi_2 \end{Bmatrix}, \mathbf{c}_3 = \begin{Bmatrix} x_3 \\ y_3 \\ \phi_3 \end{Bmatrix}$$

These nine coordinates are variables; that is, when the bodies move, the coordinates find different values. Obviously, if we enforce the presence of the kinematic joints, these variables no longer remain independent from one another. In the following sections, we learn how to describe the kinematic constraints representing a variety of kinematic joints.

- We define the location of each joint on its corresponding body. In our example, the pin joint  $A$  is connected between bodies (0) and (1). The coordinates of  $A$  on these two bodies (i.e., the coordinates of  $A_0$  and  $A_1$ ) are

$$\mathbf{r}_0^A = \mathbf{s}_0^A = \begin{Bmatrix} -0.11 \\ 0.15 \end{Bmatrix} \text{m}, \quad \mathbf{s}_1^A = \begin{Bmatrix} -0.10 \\ 0 \end{Bmatrix} \text{m}$$

Because  $A_0$  is a point on the ground, we define its  $x$ - $y$  coordinates, but  $A_1$  is attached to a moving body; therefore, we state its  $\xi$ - $\eta$  coordinates.

For the pin joint at  $B$ , we determine the coordinates of points  $B_1$  and  $B_2$  on their corresponding bodies:

$$\mathbf{s}_1^B = \begin{Bmatrix} 0.10 \\ 0 \end{Bmatrix} \text{m}, \quad \mathbf{s}_2^B = \begin{Bmatrix} 0 \\ 0.12 \end{Bmatrix} \text{m}$$

Similarly, for the pin joint at  $C$ , we have

$$\mathbf{r}_0^C = \mathbf{s}_0^C = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \text{m}, \quad \mathbf{s}_3^C = \begin{Bmatrix} -0.20 \\ 0 \end{Bmatrix} \text{m}$$

For a sliding joint, we define one point and one unit vector on (or parallel to) the axis of the joint on each body. In our example, we define point  $D_2$  on body (2), point  $C_3$  and vector  $\vec{u}_2$  on body (2), and vector  $\vec{u}_3$  on body (3):

$$\mathbf{s}_2^D = \begin{Bmatrix} 0 \\ -0.05 \end{Bmatrix} \text{m}, \quad \mathbf{u}_2 = \begin{Bmatrix} 1.0 \\ 0 \end{Bmatrix} \text{m}, \quad \mathbf{s}_3^C = \begin{Bmatrix} -0.20 \\ 0 \end{Bmatrix} \text{m}, \quad \mathbf{u}_3 = \begin{Bmatrix} 1.0 \\ 0 \end{Bmatrix} \text{m}$$

Points  $D_2$  and  $C_3$  could have been defined anywhere on the axis. We chose  $D_2$  at a location that was most convenient. Point  $C_3$  was chosen because it had already been already defined for the pin joint.

## 7.2 Kinematic Joints

In this section, we present the body-coordinate formulation of constraint equations for several commonly used kinematic joints. We adopt the notation that a single constraint equation is denoted by the *lightface* character  $\Phi$ . If there is more than one algebraic equation in a constraint, we denote it by the *boldface* character  $\Phi$ . The constraint symbol may carry a left superscript in parentheses with two entries, such as  ${}^{(type,n)}\Phi$ . The first entry states the constraint *type*—for example,  $r$  for *revolute* and  $t$  for *translational*. The second entry,  $n$ , denotes the *number* of algebraic equations in a constraint. If the array of coordinates is defined as  $\mathbf{c}$ , then the position constraints are expressed in general form as

$$\Phi \equiv \Phi(\mathbf{c}) = \mathbf{0} \tag{7.1}$$

Most kinematic constraints are nonlinear in the coordinates. For a system of  $n_b$  bodies, the array of coordinates  $\mathbf{c}$  contains  $n_b$  three-arrays as

$$\mathbf{c} = \begin{Bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_{n_b} \end{Bmatrix} \quad (7.2)$$

where  $\mathbf{c}_i = \{x_i \ y_i \ \phi_i\}'$ ,  $i = 1, \dots, n_b$ .

In most analyses, the constraints on the velocities and accelerations are also needed. Velocity and acceleration constraints are obtained by taking the first and second time derivatives of the position constraints. The arrays of velocities and accelerations associated with the array of coordinates  $\mathbf{c}$  are denoted as

$$\dot{\mathbf{c}} = \begin{Bmatrix} \dot{\mathbf{c}}_1 \\ \vdots \\ \dot{\mathbf{c}}_{n_b} \end{Bmatrix}, \ddot{\mathbf{c}} = \begin{Bmatrix} \ddot{\mathbf{c}}_1 \\ \vdots \\ \ddot{\mathbf{c}}_{n_b} \end{Bmatrix} \quad (7.3)$$

Assuming that the parameter *time* does not appear explicitly in the constrain equations, the time derivative of Eq. (7.1) yields the velocity constraints in general form as

$$\dot{\Phi} = \mathbf{D}\dot{\mathbf{c}} = \mathbf{0} \quad (7.4)$$

These constraints are *linear* in the velocities where the coefficient matrix  $\mathbf{D}$  is the *Jacobian matrix*.

The acceleration constraints are linear in the accelerations, which are expressed as

$$\ddot{\Phi} = \mathbf{D}\ddot{\mathbf{c}} + \dot{\mathbf{D}}\dot{\mathbf{c}} = \mathbf{0} \quad (7.5)$$

The coefficient matrix of the acceleration array is the same Jacobian matrix. The product  $\dot{\mathbf{D}}\dot{\mathbf{c}}$  contains quadratic terms in velocities. The acceleration constraints can be expressed in another useful form as

$$\mathbf{D}\ddot{\mathbf{c}} = \boldsymbol{\gamma} \quad (7.6)$$

where

$$\boldsymbol{\gamma} = -\dot{\mathbf{D}}\dot{\mathbf{c}} \quad (7.7)$$

The array  $\boldsymbol{\gamma}$  is called the *right-hand side* array of acceleration constraints.

Equations (7.4) and (7.5) reveal that if we have the Jacobian matrix,  $\mathbf{D}$ , and the array of quadratic velocity terms,  $\boldsymbol{\gamma}$ , we can construct the velocity and acceleration constraints. Therefore, for any kinematic joint, it would be sufficient to derive the position constraints, that is, Eq. (7.1), matrix  $\mathbf{D}$ , and array  $\boldsymbol{\gamma}$ .\*

---

\* In the MATLAB® program of Chapter 8, constraint equations, matrix  $\mathbf{D}$ , and array  $\boldsymbol{\gamma}$  are provided in a library of M-files for a variety of kinematic joints.

In the following subsections, we derive the velocity and acceleration constraints for some commonly used kinematic joints. In these derivations, several formulas from Chapter 3 are used repeatedly. These are Eqs. (3.12), (3.13), (3.16), and (3.17) that are restated here for easy reference:

$$\dot{\mathbf{s}} = \check{\mathbf{s}}\dot{\phi}, \ddot{\mathbf{s}} = \dot{\check{\mathbf{s}}} = -\mathbf{s}\dot{\phi} \quad (7.8)$$

$$\dot{\mathbf{r}}^P = \dot{\mathbf{r}} + \dot{\mathbf{s}}^P = \dot{\mathbf{r}} + \check{\mathbf{s}}^P\dot{\phi} \quad (7.9)$$

$$\ddot{\mathbf{s}} = \check{\mathbf{s}}\ddot{\phi} + \dot{\check{\mathbf{s}}}\dot{\phi} = \check{\mathbf{s}}\ddot{\phi} - \mathbf{s}\dot{\phi}^2 \quad (7.10)$$

$$\ddot{\mathbf{r}}^P = \ddot{\mathbf{r}} + \check{\mathbf{s}}^P\ddot{\phi} + \dot{\check{\mathbf{s}}}\dot{\phi} = \ddot{\mathbf{r}} + \check{\mathbf{s}}^P\ddot{\phi} - \mathbf{s}\dot{\phi}^2 \quad (7.11)$$

Other useful formulas are the first and second time derivatives of the position vector  $\mathbf{d} = \mathbf{r}_i^P - \mathbf{r}_j^P$ :

$$\dot{\mathbf{d}} = \dot{\mathbf{r}}_i^P - \dot{\mathbf{r}}_j^P = \dot{\mathbf{r}}_i + \check{\mathbf{s}}_i^P\dot{\phi}_i - \dot{\mathbf{r}}_j - \check{\mathbf{s}}_j^P\dot{\phi}_j \quad (7.12)$$

$$\ddot{\mathbf{d}} = \ddot{\mathbf{r}}_i^P - \ddot{\mathbf{r}}_j^P = \ddot{\mathbf{r}}_i + \check{\mathbf{s}}_i^P\ddot{\phi}_i - \mathbf{s}_i^P\dot{\phi}_i^2 - \ddot{\mathbf{r}}_j - \check{\mathbf{s}}_j^P\ddot{\phi}_j + \mathbf{s}_j^P\dot{\phi}_j^2 \quad (7.13)$$

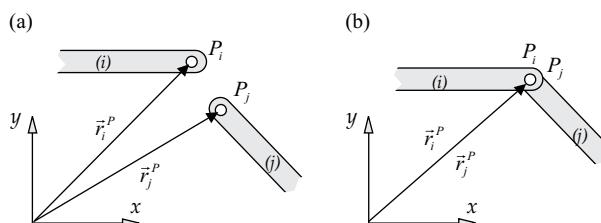
### 7.2.1 Revolute (Pin) Joint

Constraint equations for a revolute joint in the body-coordinate formulation can be constructed easily. As shown in Figure 7.3a, for a pin joint to exist between points  $P_i$  and  $P_j$  on bodies (i) and (j), their  $x$ - $y$  coordinates must be equal, that is,  $\vec{r}_j^P = \vec{r}_i^P$ . Therefore, the constraint equation for a pin joint is

$${}^{(r,2)}\Phi = \mathbf{r}_i^P - \mathbf{r}_j^P = \mathbf{0} \quad (7.14)$$

where the left-superscript “ $r$ ” is for revolute and “2” denotes the number of algebraic conditions introduced by this joint. To have a better understanding on how the coordinates of bodies (i) and (j) have become dependent on each other through this kinematic joint, we use Eq. (3.10) to express Eq. (7.14) in matrix form as

$$\left\{ \begin{array}{c} x_i \\ y_i \end{array} \right\} + \left[ \begin{array}{cc} \cos\phi_i & -\sin\phi_i \\ \sin\phi_i & \cos\phi_i \end{array} \right] \left\{ \begin{array}{c} \xi_i^P \\ \eta_i^P \end{array} \right\} - \left\{ \begin{array}{c} x_j \\ y_j \end{array} \right\} - \left[ \begin{array}{cc} \cos\phi_j & -\sin\phi_j \\ \sin\phi_j & \cos\phi_j \end{array} \right] \left\{ \begin{array}{c} \xi_j^P \\ \eta_j^P \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\}$$



**FIGURE 7.3**

Two bodies (a) before and (b) after the pin joint constraints are enforced.

Enforcing these conditions forms a pin joint between the two bodies as shown in Figure 7.3b. The two constraints in Eq. (7.14) reduce the degree of freedom (DoF) between the two bodies by 2.

The velocity constraints for a revolute joint are obtained from the time derivative of Eq. (7.14) as

$$\begin{aligned} {}^{(r,2)}\dot{\Phi} = \dot{\mathbf{r}}_i^P - \dot{\mathbf{r}}_j^P = \mathbf{0} \quad \Rightarrow \quad {}^{(r,2)}\dot{\Phi} = \dot{\mathbf{r}}_i + \check{\mathbf{s}}_i^P \dot{\phi}_i - \dot{\mathbf{r}}_j - \check{\mathbf{s}}_j^P \dot{\phi}_j = \mathbf{0} \\ \Rightarrow \left[ \begin{array}{cc|cc} \mathbf{I} & \check{\mathbf{s}}_i^P & -\mathbf{I} & -\check{\mathbf{s}}_j^P \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{array} \right\} = \mathbf{0} \end{aligned} \quad (7.15)$$

From these velocity constraints, we extract the Jacobian matrix, expressed in two separate submatrices:

$${}^{(r,2)}\mathbf{D}_i = \left[ \begin{array}{cc} \mathbf{I} & \check{\mathbf{s}}_i^P \end{array} \right], \quad {}^{(r,2)}\mathbf{D}_j = \left[ \begin{array}{cc} -\mathbf{I} & -\check{\mathbf{s}}_j^P \end{array} \right] \quad (7.16)$$

The time derivative of Eq. (7.15) yields the acceleration constraints:

$$\begin{aligned} {}^{(r,2)}\ddot{\Phi} = \left[ \begin{array}{cc|cc} \mathbf{I} & \check{\mathbf{s}}_i^P & -\mathbf{I} & -\check{\mathbf{s}}_j^P \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}_i \\ \ddot{\phi}_i \\ \ddot{\mathbf{r}}_j \\ \ddot{\phi}_j \end{array} \right\} + \left[ \begin{array}{cc|cc} \mathbf{0} & \dot{\check{\mathbf{s}}}_i^P & \mathbf{0} & -\dot{\check{\mathbf{s}}}_j^P \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{array} \right\} = \mathbf{0} \\ \Rightarrow \left[ \begin{array}{cc|cc} {}^{(r,2)}\mathbf{D}_i & {}^{(r,2)}\mathbf{D}_j & \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}_i \\ \ddot{\phi}_i \\ \ddot{\mathbf{r}}_j \\ \ddot{\phi}_j \end{array} \right\} + \dot{\check{\mathbf{s}}}_i^P \dot{\phi}_i - \dot{\check{\mathbf{s}}}_j^P \dot{\phi}_j = \mathbf{0} \end{aligned} \quad (7.17)$$

The right-hand side of the acceleration constraints for a revolute joint is

$${}^{(r,2)}\boldsymbol{\gamma} = -\dot{\check{\mathbf{s}}}_i^P \dot{\phi}_i + \dot{\check{\mathbf{s}}}_j^P \dot{\phi}_j \quad (7.18)$$

The right-hand-side array of accelerations for any kinematic joint can be expressed in different forms. For example, for the revolute joint,  $-\dot{\check{\mathbf{s}}}_i^P \dot{\phi}_i + \dot{\check{\mathbf{s}}}_j^P \dot{\phi}_j$  can also be written as  $\mathbf{s}_i^P \dot{\phi}_i^2 - \mathbf{s}_j^P \dot{\phi}_j^2$ . When we evaluate these terms in a program, if the time derivative of the vectors (such as  $\dot{\mathbf{s}}_i^P$  and  $\dot{\mathbf{s}}_j^P$ ) has already been computed, we may consider using the first expression instead of the second expression.

### 7.2.2 Translational (Sliding) Joint

A translational joint allows two bodies to slide relative to one another along the axis of the joint. To construct a translational joint between bodies  $(i)$  and  $(j)$ , we describe one point and one unit vector along the sliding axis on each body as shown in Figure 7.4a. These are points  $P_i$  and  $P_j$  and unit vectors  $\vec{u}_i$  and  $\vec{u}_j$ . Vector  $\vec{d}$  is defined by connecting  $P_i$  to  $P_j$  as  $\vec{d} = \vec{r}_i^P - \vec{r}_j^P$ . The necessary and sufficient conditions for the two bodies not to have relative translation perpendicular to the sliding axis, and not to have relative rotation, are obtained by enforcing  $\vec{u}_j$ ,  $\vec{u}_i$ , and  $\vec{d}$  to remain parallel:

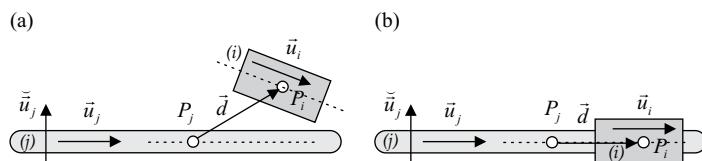
$${}^{(t,2)}\Phi = \begin{Bmatrix} \vec{u}'_j \mathbf{d} \\ \vec{u}'_j \mathbf{u}_i \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \quad (7.19)$$

where the left-superscript “ $t$ ” is for translational and “ $2$ ” is for the number of equations, removing two relative DoFs. If these constraints are enforced, a sliding joint is formed between the two bodies as illustrated in Figure 7.4b.

The time derivative of Eq. (7.19) provides the velocity constraints as

$$\begin{aligned} {}^{(t,2)}\dot{\Phi} &= \begin{Bmatrix} \dot{\vec{u}}'_j \mathbf{d} + \vec{u}'_j \dot{\mathbf{d}} \\ \dot{\vec{u}}'_j \mathbf{u}_i + \vec{u}'_j \dot{\mathbf{u}}_i \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \Rightarrow \begin{Bmatrix} -\mathbf{u}'_j \mathbf{d} \dot{\phi}_j + \vec{u}'_j (\dot{\vec{r}}_i^P - \dot{\vec{r}}_j^P) \\ -\mathbf{u}'_j \mathbf{u}_i \dot{\phi}_j + \vec{u}'_j \vec{u}_i \dot{\phi}_i \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \\ &\Rightarrow \begin{Bmatrix} -\mathbf{u}'_j \mathbf{d} \dot{\phi}_j + \vec{u}'_j (\dot{\vec{r}}_i + \vec{s}_i^P \dot{\phi}_i - \dot{\vec{r}}_j - \vec{s}_j^P \dot{\phi}_j) \\ \dot{\phi}_i - \dot{\phi}_j \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \\ &\Rightarrow \begin{bmatrix} \vec{u}'_j & \mathbf{u}'_j \mathbf{s}_i^P & -\vec{u}'_j & -\mathbf{u}'_j (\mathbf{s}_j^P + \mathbf{d}) \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \dot{\vec{r}}_i \\ \dot{\phi}_i \\ \dot{\vec{r}}_j \\ \dot{\phi}_j \end{bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \end{aligned} \quad (7.20)$$

The second position constraint, stating that  $\vec{u}_i$  and  $\vec{u}_j$  must remain parallel, resulted in the constraint  $\dot{\phi}_i - \dot{\phi}_j = 0$ —no relative rotation between the two bodies. The Jacobian matrix from the velocity constraints can be split into two submatrices as



**FIGURE 7.4**

Two bodies (a) before and (b) after the sliding joint constraints are enforced.

$${}^{(t,2)}\mathbf{D}_i = \begin{bmatrix} \bar{\mathbf{u}}'_j & \bar{\mathbf{u}}'_j \mathbf{s}_i^P \\ \mathbf{0} & 1 \end{bmatrix}, {}^{(t,2)}\mathbf{D}_j = \begin{bmatrix} -\bar{\mathbf{u}}'_j & -\bar{\mathbf{u}}'_j (\mathbf{s}_j^P + \mathbf{d}) \\ \mathbf{0} & -1 \end{bmatrix} \quad (7.21)$$

The time derivative of Eq. (7.20) yields the acceleration constraints as

$${}^{(t,2)}\ddot{\Phi} = \left[ \begin{array}{c|c} {}^{(t,2)}\mathbf{D}_i & {}^{(t,2)}\mathbf{D}_j \end{array} \right] \begin{Bmatrix} \ddot{\mathbf{r}}_i \\ \ddot{\phi}_i \\ \ddot{\mathbf{r}}_j \\ \ddot{\phi}_j \end{Bmatrix} + \left[ \begin{array}{c|c} {}^{(t,2)}\dot{\mathbf{D}}_i & {}^{(t,2)}\dot{\mathbf{D}}_j \end{array} \right] \begin{Bmatrix} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \quad (7.22)$$

The time derivative of the second row of the Jacobian matrix yields all zeros. The time derivative of the first row becomes

$$\left[ \begin{array}{c|c} \dot{\mathbf{u}}'_j & \dot{\mathbf{u}}'_j \mathbf{s}_i^P + \mathbf{u}'_j \dot{\mathbf{s}}_i^P \\ \hline -\dot{\mathbf{u}}'_j & -\dot{\mathbf{u}}'_j (\mathbf{s}_j^P + \mathbf{d}) - \mathbf{u}'_j (\dot{\mathbf{s}}_j^P + \dot{\mathbf{d}}) \end{array} \right]$$

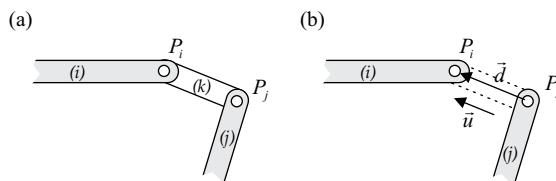
When we multiply this row by the array of velocities, we can take advantage of  $\dot{\phi}_i = \dot{\phi}_j$  to simplify the expression for the array  $\boldsymbol{\gamma}$  as

$${}^{(t,2)}\boldsymbol{\gamma} = \begin{Bmatrix} (\bar{\mathbf{u}}'_j(\mathbf{r}_i - \mathbf{r}_j)\dot{\phi}_i + 2\mathbf{u}'_j(\dot{\mathbf{r}}_i - \dot{\mathbf{r}}_j))\dot{\phi}_i \\ 0 \end{Bmatrix} \Rightarrow {}^{(t,2)}\boldsymbol{\gamma} = \begin{Bmatrix} \bar{\mathbf{u}}'_j(\mathbf{r}_i - \mathbf{r}_j)\dot{\phi}_i - 2\dot{\mathbf{u}}'_j(\dot{\mathbf{r}}_i - \dot{\mathbf{r}}_j) \\ 0 \end{Bmatrix} \quad (7.23)$$

### 7.2.3 Revolute–Revolute Joint

The total number of coordinates and constraint equations that describe a mechanical system can be reduced if some of the joints and bodies are combined and represented as composite joints. This technique simplifies the analytical formulation without changing the kinematic characteristics of a system. For example, consider the three bodies connected by two revolute joints shown in Figure 7.5a. This system requires nine coordinates (three per body) and four constraint equations (two per joint). Therefore, this system has  $9 - 4 = 5$  DoFs. This system may be represented by the kinematically equivalent system shown in Figure 7.5b where body  $(k)$  and the two revolute joints are combined to form a *revolute–revolute* joint as a link. This configuration requires only six coordinates for bodies  $(i)$  and  $(j)$ , and one constraint equation. Therefore, this equivalent system has  $6 - 1 = 5$  DoFs—the same as the original system.

To formulate the constraint equation for a revolute–revolute joint, we take advantage of the length of the link being a known constant  $L$ . As shown in Figure 7.5b, the centers of



**FIGURE 7.5**

(a) Three bodies connected by two revolute joints and (b) the equivalent composite joint.

the two pin joints are denoted as points  $P_i$  and  $P_j$ . Vector  $\vec{d} = \vec{r}_i^P - \vec{r}_j^P$  connecting these two points must keep a constant length. Therefore, the constraint equation is written as

$${}^{(r-r,1)}\Phi = \mathbf{d}'\mathbf{d} - L^2 = 0$$

This is a well-defined constraint with a time derivative  ${}^{(r-r,1)}\dot{\Phi} = 2\mathbf{d}'\dot{\mathbf{d}} = 0$ . However, we notice that there is a coefficient "2" that will continue to appear in front of every term in the Jacobian matrix and also on the right-hand side of the acceleration term. To eliminate the appearance of this coefficient in every term, we can express the constraint as

$${}^{(r-r,1)}\Phi = \frac{1}{2}(\mathbf{d}'\mathbf{d} - L^2) = 0 \quad (7.24)$$

Then the time derivative of this constraint, by referring to Eq. (7.12), yields the velocity constraint as

$${}^{(r-r,1)}\dot{\Phi} = \mathbf{d}'\dot{\mathbf{d}} = 0 \Rightarrow \left[ \begin{array}{cc|cc} \mathbf{d}' & \mathbf{d}'\tilde{\mathbf{s}}_i^P & -\mathbf{d}' & -\mathbf{d}'\tilde{\mathbf{s}}_j^P \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{array} \right\} = 0 \quad (7.25)$$

The right-hand side of the acceleration constraint is determined from the time derivative of the velocity constraint:

$${}^{(r-r,1)}\gamma = -\dot{\mathbf{d}}'\dot{\mathbf{d}} - \mathbf{d}'(\dot{\tilde{\mathbf{s}}}_i^P \dot{\phi}_i - \dot{\tilde{\mathbf{s}}}_j^P \dot{\phi}_j) \quad (7.26)$$

We can express these constraints in a slightly different form by defining a unit vector  $\vec{u} = \vec{d}/L$  and use it to replace one of the  $\vec{d}$  vectors in the constraint equations, which results in the following revised form of the constraints:

$${}^{(r-r,1)}\Phi = \frac{1}{2}(\mathbf{u}'\mathbf{d} - L) = 0 \quad (7.27)$$

$${}^{(r-r,1)}\dot{\Phi} = \mathbf{u}'\dot{\mathbf{d}} = 0 \Rightarrow {}^{(r-r,1)}\dot{\Phi} = \left[ \begin{array}{cc|cc} \mathbf{u}' & \mathbf{u}'\tilde{\mathbf{s}}_i^P & -\mathbf{u}' & -\mathbf{u}'\tilde{\mathbf{s}}_j^P \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{array} \right\} = 0 \quad (7.28)$$

$${}^{(r-r,1)}\mathbf{D}_i = \left[ \begin{array}{cc} \mathbf{u}' & \mathbf{u}'\tilde{\mathbf{s}}_i^P \end{array} \right], \quad {}^{(r-r,1)}\mathbf{D}_j = \left[ \begin{array}{cc} -\mathbf{u}' & -\mathbf{u}'\tilde{\mathbf{s}}_j^P \end{array} \right] \quad (7.29)$$

$${}^{(r-r,1)}\gamma = -\dot{\mathbf{u}}'\dot{\mathbf{d}} - \mathbf{u}'(\dot{\tilde{\mathbf{s}}}_i^P \dot{\phi}_i - \dot{\tilde{\mathbf{s}}}_j^P \dot{\phi}_j) \Rightarrow {}^{(r-r,1)}\gamma = -\dot{\mathbf{u}}'\dot{\mathbf{d}} + \tilde{\mathbf{u}}'(\dot{\tilde{\mathbf{s}}}_i^P \dot{\phi}_i - \dot{\tilde{\mathbf{s}}}_j^P \dot{\phi}_j) \quad (7.30)$$

One advantage of the revised form, which may not be apparent at this point, is that the elements of the Jacobian submatrices contain a unit vector that will make the interpretation of reaction forces simpler. This issue will be discussed further in Section 7.4.1.

### 7.2.4 Revolute–Translational Joint

Another type of composite joint is the *revolute–translational* or *pin-in-slot* joint. Figure 7.6 schematically illustrates three cases, where a revolute and a translational joint connect three bodies. Similar to the revolute–revolute joint formulation, this system can be modeled as a revolute–translational joint between bodies  $(i)$  and  $(j)$  by eliminating the middle body  $(k)$ . Since there are two relative DoFs between the two bodies, we only need one constraint equation. To construct the constraint, the center of the pin joint is denoted as point  $P_j$ , and point  $P_i$  and unit vector  $\vec{u}_i$  are defined along the translational axis on body  $(i)$ . The constraint equation is found by requiring that the projection of vector  $\vec{d} = \vec{r}_i^P - \vec{r}_j^P$  onto an axis perpendicular to  $\vec{u}_i$  keep a length equal to  $L$ :

$${}^{(r-t,1)}\Phi = \check{\mathbf{u}}'_i \mathbf{d} - L = 0 \quad (7.31)$$

In this equation, the length  $L$  must be considered a directional scalar; that is, it may be a positive or a negative quantity. For example, if we examine the vectors in Figure 7.6a,  $L$  must be considered a positive quantity. But if the projection of  $\vec{d}$  onto  $\check{\vec{u}}_i$  is in the opposite direction of  $\check{\vec{u}}_i$ , as in Figure 7.6b, then  $L$  must be considered a negative quantity. If the pin joint is located directly on the axis of the sliding joint as in Figure 7.6c, then  $L = 0$ ; therefore, its directional magnitude is no longer an issue.

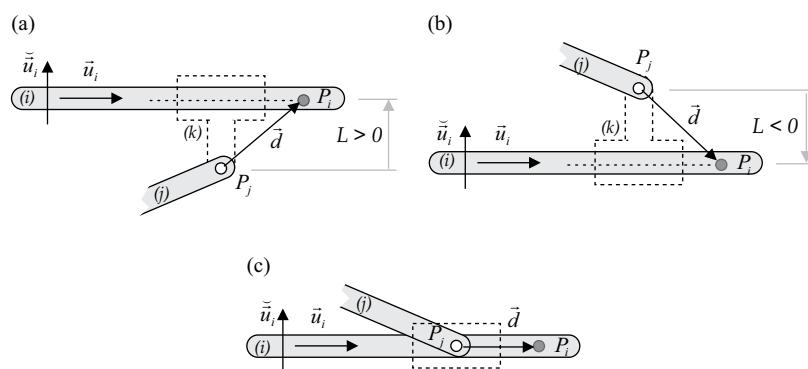
We obtain the velocity constraint from the time derivative of Eq. (7.31):

$${}^{(r-t,1)}\dot{\Phi} = \check{\mathbf{u}}'_i \dot{\mathbf{d}} + \mathbf{d}' \dot{\check{\mathbf{u}}}_i = 0 \Rightarrow \left[ \begin{array}{cc|cc} \check{\mathbf{u}}'_i & \mathbf{u}'_i(\mathbf{s}_i^P - \mathbf{d}) & -\check{\mathbf{u}}'_i & -\mathbf{u}'_i \mathbf{s}_j^P \\ \vdots & \vdots & \vdots & \vdots \end{array} \right] \begin{Bmatrix} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \mathbf{r}_j \\ \dot{\phi}_j \end{Bmatrix} = 0 \quad (7.32)$$

The following Jacobian submatrices are extracted from the velocity constraint:

$${}^{(r-t,1)}\mathbf{D}_i = \left[ \begin{array}{cc} \check{\mathbf{u}}'_i & \mathbf{u}'_i(\mathbf{s}_i^P - \mathbf{d}) \end{array} \right], \quad {}^{(r-t,1)}\mathbf{D}_j = \left[ \begin{array}{cc} -\check{\mathbf{u}}'_i & -\mathbf{u}'_i \mathbf{s}_j^P \end{array} \right] \quad (7.33)$$

Furthermore, the right-hand side of the acceleration constraint is found to be



**FIGURE 7.6**

Three bodies connected by a revolute and a translational joint can be modeled as an equivalent revolute–translational joint, where three possible cases are shown in (a), (b), and (c).

$$(r-t,1) \gamma = \dot{\mathbf{u}}'_i(\mathbf{d}\dot{\phi}_i + 2\ddot{\mathbf{d}}) - \mathbf{u}'_i(\dot{\mathbf{s}}_i^P\dot{\phi}_i - \dot{\mathbf{s}}_j^P\dot{\phi}_j) \quad (7.34)$$

### 7.2.5 Rigid Joint

The connection between two bodies that are rigidly attached to each other is called a *rigid joint* (also called *bracket* or *welded joint*). Such a joint eliminates all three relative DoFs, which can be described analytically in more than one way. One formulation can define a revolute joint between the two bodies, such as at points  $P_i$  and  $P_j$ , as shown in Figure 7.7a, to eliminate two DoFs. A third constraint, such as  $\phi_i - \phi_j = \theta$ , where  $\theta$  is a constant angle, can be defined to eliminate the relative rotation.

In another formulation, instead of introducing a revolute joint, we eliminate the relative translation between the two bodies differently. Here we note that any vector defined between the two bodies does not change its components with respect to either body as they translate and rotate. To enforce such conditions, we define a vector  $\vec{d}$  connecting the two origins as shown in Figure 7.7b. At initial time, we compute the components of this vector with respect to one of the body-fixed frames, for example,  $\xi_j - \eta_j$ , as

$$\vec{d}_j = {}^0\mathbf{A}'_j({}^0\mathbf{r}_i - {}^0\mathbf{r}_j)$$

And for the angles we determine a relative angle as

$$\theta = {}^0\phi_i - {}^0\phi_j$$

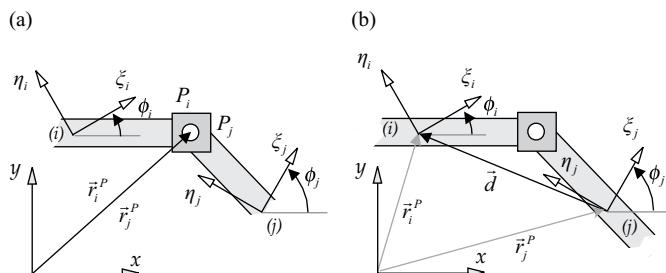
Now we can construct the rigid joint constraints as

$${}^{(rigid,3)}\Phi = \begin{Bmatrix} \mathbf{r}_i - (\mathbf{r}_j + \mathbf{A}_j \vec{d}_j) \\ \phi_i - \phi_j - \theta \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ 0 \end{Bmatrix} \quad (7.35)$$

The Jacobian submatrices are formed as

$${}^{(rigid,3)}\mathbf{D}_i = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, {}^{(rigid,3)}\mathbf{D}_j = \begin{bmatrix} -\mathbf{I} & -\vec{d}_j \\ \mathbf{0} & -1 \end{bmatrix} \quad (7.36)$$

where  $\mathbf{d}_j = \mathbf{A}_j \vec{d}_j$ . The right-hand side of the acceleration constraints is expressed as



**FIGURE 7.7**

A rigid joint between two bodies can be modeled in two different forms as depicted in (a) and (b).

$$\overset{(rigid,3)}{\gamma} = \begin{Bmatrix} -\mathbf{d}_j \dot{\phi}_j^2 \\ 0 \end{Bmatrix} \quad (7.37)$$

### 7.2.6 Simple Constraints

A simple constraint is a condition imposed on one or more coordinates of a body. For example, if the  $y$ -coordinate of a typical body ( $i$ ) must remain equal to 0.25 m, we write a simple constraint as  $y_i - 0.25 = 0$ . Since a body has three coordinates, any of the following simple constraints can be imposed on a body:

$${}^{(x,1)}\Phi = x_i - c_1 = 0$$

$${}^{(y,1)}\Phi = y_i - c_2 = 0 \quad (7.38)$$

$${}^{(\phi,1)}\Phi = \phi_i - c_3 = 0$$

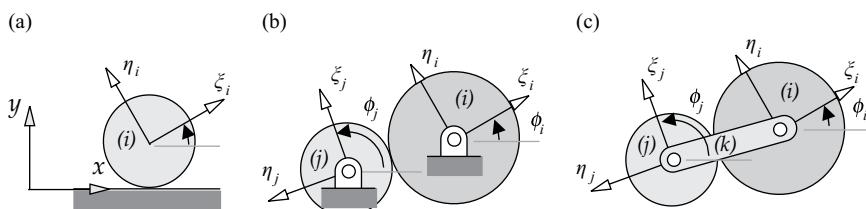
where  $c_1$ ,  $c_2$ , and  $c_3$  are constants. The velocity and acceleration constraints for these simple constraints have trivial forms, which yield

$$\begin{aligned} {}^{(x,1)}\mathbf{D}_i &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} = 0, {}^{(x,1)}\gamma = 0 \\ {}^{(y,1)}\mathbf{D}_i &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} = 0, {}^{(y,1)}\gamma = 0 \\ {}^{(\phi,1)}\mathbf{D}_i &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = 0, {}^{(\phi,1)}\gamma = 0 \end{aligned} \quad (7.39)$$

### 7.2.7 Circular Disc

A rigid circular disc rolling on a flat surface, as shown in Figure 7.8a, could represent a wheel rolling on the ground. To ensure that the disc rolls without slipping, a constraint for no-slip must be enforced. Assuming that a constraint, such as  $y_i - R_i = 0$ , where  $R_i$  is the radius of the disc, has already defined to keep the disc in contact with the ground, the no-slip condition can be viewed as the relationship between the rotational and translational velocities of the disc:  $R_i \dot{\phi}_i = -\dot{x}_i$ . We note that when the disc rolls counterclockwise (positive  $\dot{\phi}_i$ ), it translates to the left (negative  $\dot{x}_i$ ). The integral of this relationship provides the position constraint for the *no-slip* condition as

$${}^{(d-ns,1)}\Phi = R_i(\phi_i - {}^0\phi_i) + (x_i - {}^0x_i) = 0 \quad (7.40)$$



**FIGURE 7.8**

(a) A disc rolling on the ground; (b) two discs pinned to the ground; (c) two discs pinned to a third body.

where  ${}^0x_i$  and  ${}^0\phi_i$  are the initial  $x$  and  $\phi$  coordinates of the disc. The corresponding Jacobian matrix and the right-hand side of the acceleration constraint are

$${}^{(d-ns,1)}\mathbf{D}_i = \begin{bmatrix} 1 & 0 & R_i \end{bmatrix} = 0, {}^{(d-ns,1)}\gamma = 0 \quad (7.41)$$

A similar constraint can be derived for no-slip condition between two rigid circular discs. As shown in Figure 7.8b, the two discs are pinned to the ground at their centers. The contact points of the two discs must have the same velocity; therefore,  $R_i \dot{\phi}_i = -R_j \dot{\phi}_j$ , where  $R_i$  and  $R_j$  are the radii. The integral of this relationship provides the position constraint as

$${}^{(dd-ns,1)}\Phi = R_i(\phi_i - {}^0\phi_i) + R_j(\phi_j - {}^0\phi_j) = 0 \quad (7.42)$$

where  ${}^0\phi_i$  and  ${}^0\phi_j$  are the initial values. If the two discs are pinned to a third body instead of the ground, as shown in Figure 7.8c, the angle of the third body must also be included in the constraint equation.

The no-slip constraints of Eqs. (7.40) and (7.42) can be revised to fit other applications. For example, the flat horizontal surface of the first model could be replaced by another shape. This may represent a wheel going over a bump, for example.

### 7.2.8 Driver Constraints

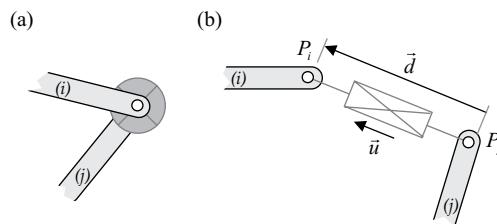
A driver constraint can represent a *rotary* motor or a *linear* actuator, where instead of specifying its torque or its force, we specify its kinematics as a known time function. For a rotary motor acting about the axis of a pin joint between bodies ( $i$ ) and ( $j$ ), as shown in Figure 7.9a, a driver constraint can be defined as

$${}^{(dr-r,1)}\Phi = \phi_i - \phi_j - f(t) = 0 \quad (7.43)$$

The first and second time derivatives of this function are

$${}^{(dr-r,1)}\dot{\Phi} = \dot{\phi}_i - \dot{\phi}_j - \dot{f}(t) = 0, {}^{(dr-r,1)}\ddot{\Phi} = \ddot{\phi}_i - \ddot{\phi}_j - \ddot{f}(t) = 0 \quad (7.44)$$

As an example, the time function could be  $f(t) = {}^0\theta + \omega t$ , where  ${}^0\theta$  is the initial relative angle between the two bodies, and  $\omega$  is the required relative angular velocity. Since for this function  $\ddot{f}(t) = 0$ , the angular velocity is a constant.



**FIGURE 7.9**

(a) A rotary motor and (b) a linear actuator.

For a linear actuator between bodies  $(i)$  and  $(j)$ , as shown in Figure 7.9b, the distance between points  $P_i$  and  $P_j$  is defined as a time function. The constraint for this driver is similar to that of a revolute–revolute joint, with the exception that the length is a variable instead of a constant, that is,  $L = f(t)$ . If we define vector  $\bar{d}$  between the two attachment points, and then define a unit vector  $\bar{u} = \bar{d}/f(t)$ , we can adopt the constraints from the revolute–revolute joint and revise them for a linear actuator as

$${}^{(dr-l,1)}\Phi = \frac{1}{2}(\mathbf{u}' \mathbf{d} - f(t)) = 0 \quad (7.45)$$

$$\begin{aligned} {}^{(dr-l,1)}\dot{\Phi} = \mathbf{u}' \dot{\mathbf{d}} - \dot{f}(t) = 0 \Rightarrow {}^{(dr-l,1)}\dot{\Phi} = \left[ \begin{array}{cc|cc} \mathbf{u}' & \mathbf{u}' \check{\mathbf{s}}_i^P & -\mathbf{u}' & -\mathbf{u}' \check{\mathbf{s}}_j^P \end{array} \right] \begin{Bmatrix} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \\ \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{Bmatrix} - \dot{f}(t) = 0 \end{aligned} \quad (7.46)$$

$${}^{(dr-l,1)}\mathbf{D}_i = \left[ \begin{array}{cc} \mathbf{u}' & \mathbf{u}' \check{\mathbf{s}}_i^P \end{array} \right], \quad {}^{(dr-l,1)}\mathbf{D}_j = \left[ \begin{array}{cc} -\mathbf{u}' & -\mathbf{u}' \check{\mathbf{s}}_j^P \end{array} \right] \quad (7.47)$$

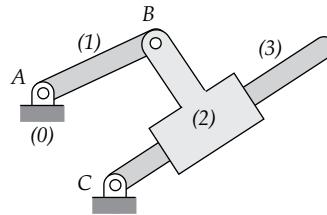
$${}^{(dr-l,1)}\gamma = -\dot{\mathbf{u}}' \dot{\mathbf{d}} - \mathbf{u}' (\dot{\check{\mathbf{s}}}_i^P \dot{\phi}_i - \dot{\check{\mathbf{s}}}_j^P \dot{\phi}_j) + \ddot{f}(t) \Rightarrow {}^{(dr-l,1)}\gamma = -\dot{\mathbf{u}}' \dot{\mathbf{d}} + \check{\mathbf{u}}' (\dot{\check{\mathbf{s}}}_i^P \dot{\phi}_i - \dot{\check{\mathbf{s}}}_j^P \dot{\phi}_j) + \ddot{f}(t) \quad (7.48)$$

### 7.2.9 System Jacobian

The Jacobian matrix for a multibody system is constructed from the Jacobian submatrices of its joints. In this subsection, we use a simple example to show the process of constructing the Jacobian matrix in generic form. We consider the multibody system in Figure 7.10. This system contains three pin joints and one sliding joint, which yield eight algebraic constraints (two per joints). For the moment, let us assume that the ground, body  $(0)$ , is a moving body; therefore, we have 12 coordinates (3 per body) and the system Jacobian matrix should be  $8 \times 12$ . Assume that the joint constraints are assembled in the following order:  $A$ ,  $B$ ,  $T$ , and  $C$ . The Jacobian matrix for the pin joint  $A$ , between bodies  $(0)$  and  $(1)$ , can be obtained as two  $2 \times 3$  submatrices from Eq. (7.17). These submatrices, denoted by  ${}^A\mathbf{D}_0$  and  ${}^A\mathbf{D}_1$ , occupy the top two rows and the left six columns of the overall  $8 \times 12$  Jacobian matrix. Similarly, the submatrices for other joints are placed in their corresponding rows and columns to form the following matrix:

$$\left[ \begin{array}{c|c|c|c} {}^A\mathbf{D}_0 & {}^A\mathbf{D}_1 & & \\ \hline & {}^B\mathbf{D}_1 & {}^B\mathbf{D}_2 & \\ \hline & & {}^T\mathbf{D}_2 & {}^T\mathbf{D}_3 \\ \hline & {}^C\mathbf{D}_0 & & {}^C\mathbf{D}_3 \end{array} \right]$$

In this matrix, each generic row represents two actual rows, and each generic column represents three actual columns. The blanks represent zero entries. Note that a sliding joint and a pin joint have the same generic form. So far, we have considered body  $(0)$  as a moving body in order to make the process of constructing the overall Jacobian matrix more

**FIGURE 7.10**

A generic multibody system.

easily understood. Since body (0) is a nonmoving body (i.e., its velocities and accelerations are always zero), we can remove its corresponding columns from the Jacobian matrix. This yields an  $8 \times 9$  system Jacobian matrix as

$$\mathbf{D} = \begin{bmatrix} {}^A\mathbf{D}_1 & & \\ \hline {}^B\mathbf{D}_1 & {}^B\mathbf{D}_2 & \\ \hline & {}^T\mathbf{D}_2 & {}^T\mathbf{D}_3 \\ \hline & & {}^C\mathbf{D}_3 \end{bmatrix}$$

If the process of constructing the overall Jacobian matrix is well understood, there will be no need to first include the ground as a moving body and then remove its corresponding columns—the Jacobian matrix could be constructed in its final form directly.

### 7.3 Unconstrained Equations of Motion

The equations of motion for a multibody system in the body-coordinate formulation are exactly those discussed in Section 6.2. For an *unconstrained* system, that is, a system of bodies interconnected by force elements without the presence of kinematic joints or constraints, the equations of motion from Eq. (6.3) are

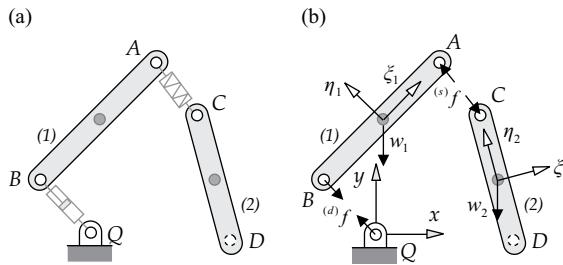
$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} \quad (7.49)$$

where  $\mathbf{M}$  is the mass matrix,  $\ddot{\mathbf{c}}$  is the array of accelerations, and  ${}^{(a)}\mathbf{h}$  is the array of applied forces. If the multibody system contains  $n_b$  bodies, then the mass matrix  $\mathbf{M}$  should be  $3n_b \times 3n_b$ , whereas  $\ddot{\mathbf{c}}$  and  ${}^{(a)}\mathbf{h}$  should be  $3n_b \times 1$  arrays. We should note that since there are no constraints present in these equations, the number of DoFs of the system is  $n_{dof} = 3n_b$ .

Equation (7.49) can be solved as a set of linear algebraic equations to find the accelerations. Obviously, since  $\mathbf{M}$  is a constant diagonal matrix, its inverse can be determined easily.

#### Example 7.1

A system of two unconstrained bodies is presented in Figure 7.11a. A spring connects the bodies at  $A$  and  $C$ , and a damper connects point  $B$  of body (1) to the ground at  $Q$ . The gravity acts on the system in the negative  $y$ -direction. The lengths of links (1) and (2) are 0.3 and 0.24 m, respectively, and the other constants are

**FIGURE 7.11**

(a) Two unconstrained bodies and (b) the corresponding FBD.

$$m_1 = 0.2 \text{ kg}, J_1 = 0.03 \text{ kg m}^2, m_2 = 0.15 \text{ kg}, J_2 = 0.02 \text{ kg m}^2$$

$$k = 50 \text{ N/m}, {}^0L = 0.2 \text{ m}, d_c = 20 \text{ N s/m}$$

For the instant shown, the bodies have the following coordinates and velocities:

$$\begin{aligned} \mathbf{r}_1 &= \begin{Bmatrix} 0.02 \\ 0.2 \end{Bmatrix} \text{ m}, \phi_1 = 45^\circ, \dot{\mathbf{r}}_1 = \begin{Bmatrix} -0.05 \\ 0.1 \end{Bmatrix} \text{ m/s}, \dot{\phi}_1 = -0.3 \text{ rad/s} \\ \mathbf{r}_2 &= \begin{Bmatrix} 0.22 \\ 0.1 \end{Bmatrix} \text{ m}, \phi_2 = 15^\circ, \dot{\mathbf{r}}_2 = \begin{Bmatrix} -0.09 \\ 0.13 \end{Bmatrix} \text{ m/s}, \dot{\phi}_2 = 0.07 \text{ rad/s} \end{aligned}$$

(This system is the same as the system in Example 6.1, except for the given positions and velocities). Construct the mass matrix and the array of forces, and then determine the accelerations.

### Solution

The FBD for this system is shown in Figure 7.11b, and reference frames are defined for the ground and the two bodies. Based on the given link lengths, and how we have attached the body-fixed frames to each body, we can state the constant coordinates of each point as

$$\mathbf{s}_1^A = \begin{Bmatrix} 0.15 \\ 0 \end{Bmatrix} \text{ m}, \mathbf{s}_1^B = \begin{Bmatrix} -0.15 \\ 0 \end{Bmatrix} \text{ m}, \mathbf{s}_2^C = \begin{Bmatrix} 0 \\ 0.12 \end{Bmatrix} \text{ m}, \mathbf{r}_Q^0 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \text{ m}$$

We perform the necessary computation with the following MATLAB program.

#### MATLAB/Chapter 7/Example 7 \_ 1

We provide the constant data to the program.

```
m = [0.2 0.15]; J = [0.03 0.02]; g = 9.81; uy = [0; 1];
k = 50; L0 = 0.2; dc = 20;
s_A1_local = [ 0.15; 0]; s_B1_local = [-0.15; 0];
s_C2_local = [ 0; 0.12]; r_Q0 = [ 0; 0];
```

We provide the given coordinate and velocity for the shown configuration.

```
r1 = [ 0.02; 0.2]; phi1 = pi/4;
r2 = [ 0.22; 0.1]; phi2 = 15*pi/180;
```

```
r1_d = [-0.05; 0.1] ; phi1_d = -0.3;
r2_d = [-0.09; 0.13]; phi2_d = 0.07;
```

We compute the rotational transformation matrices  $A_1$  and  $A_2$ ; the  $x-y$  components of  $\vec{s}_1^A, \vec{s}_1^B$ , and  $\vec{s}_2^C$ ; and the  $x-y$  coordinates of points A, B, and C.

```
A1 = A_matrix(phi1);
s_A1 = A1*s_A1_local;
s_B1 = A1*s_B1_local;
s_C2 = A2*s_C2_local;
r_A1 = r1 + s_A1;
r_B1 = r1 + s_B1;
r_C2 = r2 + s_C2;
```

Next we compute the velocity of point B.

```
r_B1_d = r_Point_d(r1_d, s_B1, phi1_d);
```

We compute a vector  $\vec{d}$  between A and C, and another vector between B and Q, and compute its time derivative.

```
d1 = r_A1 - r_C2;
d2 = r_B1 - r_Q0;
d2_d = r_B1_d;
```

We can now compute the force of the spring and the force of the damper:

```
f_s = pp_s(d1, k, L0) .....
f_d = pp_sd(d2, d2_d, dc) .....
f_A1 = -f_s;
f_C2 = f_s;
f_B1 = -f_d;
```

```
f_s =
 2.5766
 -3.6945
f_d =
 -2.0600
 2.2483
```

Next we compute the moment associated with each force:

```
ns1 = s_rot(s_A1)'*f_A1 .....
ns2 = s_rot(s_C2)'*f_C2 .....
nd1 = s_rot(s_B1)'*f_B1 .....
```

```
ns1 =
 0.6652
ns2 =
 -0.1839
nd1 =
 0.4570
```

We also compute the vectors representing the weights:

```
w1 = -m(1)*g*uy;
w2 = -m(2)*g*uy;
```

Next we construct the array of forces and the mass matrix:

```
h = [(f_A1 + f_B1 + w1)
      (ns1 + nd1)
      (f_C2 + w2)
      ns2] .....
m_array = [m(1); m(1); J(1); ...
            m(2); m(2); J(2)];
M = diag(m_array);
```

```
h =
 -0.5166
 -0.5158
 1.1221
 2.5766
 -5.1660
 -0.1839
```

Now we can determine the accelerations.

```
acc = M\h .....
```

```
acc =
-2.5832
-2.5789
37.4038
17.1775
-34.4401
-9.1957
```

The translational and rotational acceleration of each body are found to be

$$\ddot{\mathbf{r}}_1 = \begin{Bmatrix} -2.58 \\ -2.58 \end{Bmatrix} \text{ m/s}^2, \ddot{\phi}_1 = 37.40 \text{ rad/s}^2, \ddot{\mathbf{r}}_2 = \begin{Bmatrix} 17.18 \\ -34.44 \end{Bmatrix} \text{ m/s}^2, \ddot{\phi}_2 = -9.19 \text{ rad/s}^2$$

For any unconstrained multibody system, we can apply the same process as in Example 7.1.

---

## 7.4 Constrained Equations of Motion

In Section 7.2, we discussed that in the presence of kinematic joints, we must consider the dependency of the coordinates, velocities, and accelerations through Eqs. (7.1), (7.4), and (7.5), respectively. These constraint equations are repeated here, for easy reference, as

$$\Phi \equiv \Phi(\mathbf{c}) = 0 \quad (7.50)$$

$$\dot{\Phi} = \mathbf{D}\dot{\mathbf{c}} = 0 \quad (7.51)$$

$$\ddot{\Phi} = \mathbf{D}\ddot{\mathbf{c}} + \dot{\mathbf{D}}\dot{\mathbf{c}} = 0 \quad (7.52)$$

Furthermore, for a system of *constrained bodies*, the equations of motion must contain the reaction forces associated with the kinematic joints. These reaction forces were included in the equations of motion of Eq. (6.5), based on the FBD of a system, as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + {}^{(r)}\mathbf{h} \quad (7.53)$$

Therefore, for a constrained multibody system, the complete set of equations of motion must be considered as Eqs. (7.50) through (7.53). We note that if there are  $n_c$  independent constraints in Eq. (7.50), the number of DoFs of the system is  $n_{dof} = 3 \times n_b - n_c$ .

Knowing the coordinates and velocities of a system at any given time, the constraints of Eq. (7.50), the Jacobian matrix  $\mathbf{D}$ , the array  $\mathbf{D}\dot{\mathbf{c}}$ , and the array of applied forces,  ${}^{(a)}\mathbf{h}$ , can be evaluated. The arrays that need to be determined numerically are the array of accelerations,  $\ddot{\mathbf{c}}$ , and the array of reaction forces,  ${}^{(r)}\mathbf{h}$ .

In Section 6.1, we learned how to apply Newton's law of action-reaction to represent reaction forces and torques between two contacting bodies, and then we incorporated these forces and torques in the corresponding FBD. This was a *manual* process in which we had to draw FBD for each body of the system. In contrast, in the body-coordinate formulation, a *systematic* process can construct the array of reaction forces. The developed systematic process can enable a computer program to automatically construct the complete set of equations of motion for a multibody system.

The array of reaction forces can be described in a different form, by the method of *Lagrange multipliers*, borrowed from the optimization theory, as

$${}^{(r)}\mathbf{h} = \mathbf{D}'\lambda \quad (7.54)$$

where  $\mathbf{D}$  is the system Jacobian matrix, as in Eqs. (7.51) and (7.52), and  $\lambda$  is an array of coefficients, known as *Lagrange multipliers*. The number of multipliers, that is, the dimension of array  $\lambda$ , is equal to the number of constraints  $n_c$  (or the number of rows in  $\mathbf{D}$ ). In other words, for each constraint, there is one Lagrange multiplier. The immediate advantage of describing the array  $(r)\mathbf{h}$  as  $\mathbf{D}'\lambda$  is that the number of elements in  $(r)\mathbf{h}$  is  $3 \times n_b$  (unknowns), but the number of unknown multipliers in the array  $\lambda$  is  $n_c$ , where  $n_c < 3 \times n_b$ ; that is, we have reduced the number of unknowns in the equations of motion. With the new representation of the array of reaction forces, Eq. (7.53) is expressed as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + \mathbf{D}'\lambda \quad (7.55)$$

To solve these equations for the unknown accelerations and Lagrange multipliers, we append Eq. (7.52) to Eq. (7.55) to form the following set of equations:

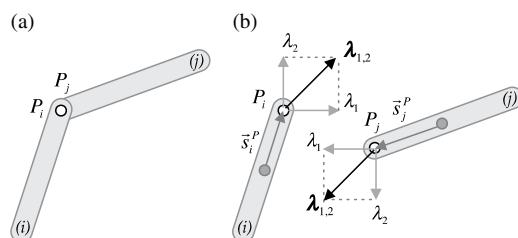
$$\left[ \begin{array}{cc} \mathbf{M} & -\mathbf{D}' \\ \mathbf{D} & \mathbf{0} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{c}} \\ \lambda \end{array} \right\} = \left\{ \begin{array}{c} {}^{(a)}\mathbf{h} \\ -\dot{\mathbf{D}}\dot{\mathbf{c}} \end{array} \right\} \quad (7.56)$$

Since the coefficient matrix on the left-hand side is square, the set of equations can be solved for the unknown. Further detailed discussion on solving these equations, either as an *inverse dynamic analysis* or as a *forward dynamic analysis*, is left for Chapters 12 and 13.

#### 7.4.1 Reaction Forces and Lagrange Multipliers

In this section, we investigate the validity of Eq. (7.54) when applied to kinematic joints, which expresses the array of reaction forces in terms of the transpose of the constraint Jacobian matrix and the array of Lagrange multipliers. We should note that this issue has already been discussed in Section 4.4.2 for some commonly used joints.

For the revolute joint connecting two bodies shown in Figure 7.12a, the reaction forces act between points  $P_i$  and  $P_j$ , as shown in the FBD of Figure 7.12b. The two orthogonal



**FIGURE 7.12**

(a) Two links connected by a revolute joint; (b) the corresponding reaction forces.

components of the reaction force can be expressed as a vector of reaction force  $\lambda_{1,2} = \{\lambda_1 \ \lambda_2\}'$ . Since these forces have moment arms,  $\vec{s}_i^P$  and  $\vec{s}_j^P$ , about their corresponding mass centers, the array of reaction force/moment for each body can be expressed as

$$(r)\mathbf{h}_i = \begin{Bmatrix} \lambda_{1,2} \\ \vec{s}_i' \lambda_{1,2} \end{Bmatrix} = \begin{bmatrix} \mathbf{I} \\ \vec{s}_i' \end{bmatrix} \lambda_{1,2} \Rightarrow (r)\mathbf{h}_i = {}^{(r,2)}\mathbf{D}'_i \lambda_{1,2} \quad (7.57)$$

$$(r)\mathbf{h}_j = \begin{Bmatrix} -\lambda_{1,2} \\ -\vec{s}_j' \lambda_{1,2} \end{Bmatrix} = \begin{bmatrix} -\mathbf{I} \\ -\vec{s}_j' \end{bmatrix} \lambda_{1,2} \Rightarrow (r)\mathbf{h}_j = {}^{(r,2)}\mathbf{D}'_j \lambda_{1,2} \quad (7.58)$$

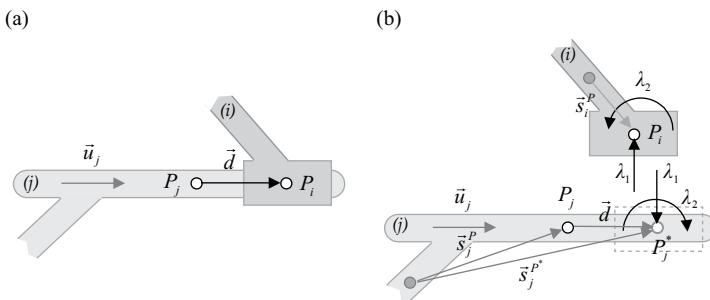
We note that the coefficient matrices for  $\lambda_{1,2}$  are exactly the transpose of the Jacobian submatrices for the revolute joint constraints, as in Eq. (7.16).

For the translational joint between two bodies shown in Figure 7.13a, we consider the reaction forces and torques shown in the FBD of Figure 7.13b. A point  $P_j^*$ , at which the reaction force on body (j) acts, is defined which coincides with  $P_j$ . The pair of reaction forces  $\lambda_1$  is perpendicular to the joint axis. These forces have their own moment arms about their corresponding body mass centers. On body (i) the moment arm is  $\vec{s}_i^P$ , and on body (j) it is  $\vec{s}_j^P$ , which can be expressed as  $\vec{s}_j^P = \vec{s}_j^P + \vec{d}$ . The array of reaction force/moment for each body can be expressed as

$$(r)\mathbf{h}_i = \begin{Bmatrix} \bar{\mathbf{u}}_j \lambda_1 \\ \mathbf{s}_i' \bar{\mathbf{u}}_j \lambda_1 + \lambda_2 \end{Bmatrix} = \begin{bmatrix} \bar{\mathbf{u}}_j & \mathbf{0} \\ \mathbf{s}_i' \bar{\mathbf{u}}_j & 1 \end{bmatrix} \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix} \Rightarrow (r)\mathbf{h}_i = {}^{(t,2)}\mathbf{D}'_i \lambda_{1,2} \quad (7.59)$$

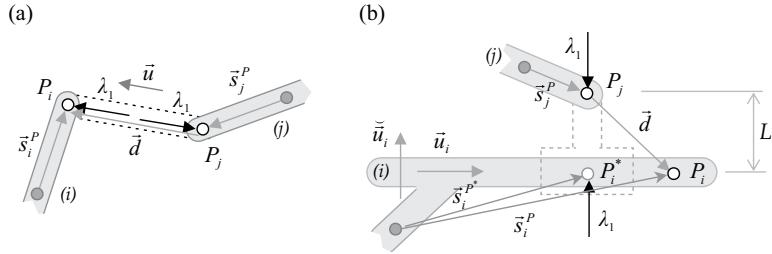
$$(r)\mathbf{h}_j = \begin{Bmatrix} -\bar{\mathbf{u}}_j \lambda_1 \\ -(\mathbf{s}_j^P + \mathbf{d})' \bar{\mathbf{u}}_j \lambda_1 - \lambda_2 \end{Bmatrix} = \begin{bmatrix} -\bar{\mathbf{u}}_j & \mathbf{0} \\ -(\mathbf{s}_j^P + \mathbf{d})' \bar{\mathbf{u}}_j & -1 \end{bmatrix} \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix} \Rightarrow (r)\mathbf{h}_j = {}^{(t,2)}\mathbf{D}'_j \lambda_{1,2} \quad (7.60)$$

Similar to the revolute joint, we conclude that the coefficient matrices for  $\lambda_{1,2}$  are exactly the transpose of the Jacobian submatrices for the translational joint constraints as expressed in Eq. (7.21).



**FIGURE 7.13**

(a) Two links connected by a translational joint; (b) the corresponding reaction forces and torques.

**FIGURE 7.14**

Reaction forces for (a) a revolute–revolute joint and (b) a revolute–translational joint.

For the revolute–revolute joint of Figure 7.14a, the reaction forces are shown along the axis of the massless link. Considering the moment arm of each reaction force, the array of reaction forces for each body can be constructed which leads to the same conclusion as the other joints:

$${}^{(r)}\mathbf{h}_i = \begin{bmatrix} \mathbf{u} \\ \bar{\mathbf{s}}_i^P \mathbf{u} \end{bmatrix} \lambda_1 \Rightarrow {}^{(r)}\mathbf{h}_i = {}^{(r-r,1)}\mathbf{D}'_i \lambda_1 \quad (7.61)$$

$${}^{(r)}\mathbf{h}_j = \begin{bmatrix} -\mathbf{u} \\ -\bar{\mathbf{s}}_j^P \mathbf{u} \end{bmatrix} \lambda_1 \Rightarrow {}^{(r)}\mathbf{h}_j = {}^{(r-r,1)}\mathbf{D}'_j \lambda_1 \quad (7.62)$$

For a revolute–translational joint, as shown in Figure 7.14b, the reaction forces between the two bodies are along an axis perpendicular to the sliding axis that passes through the center of the pin joint. The array of reaction force for body (j) can easily be constructed as

$${}^{(r)}\mathbf{h}_j = \begin{Bmatrix} -\bar{\mathbf{u}}_i \lambda_1 \\ -\bar{\mathbf{s}}_j^P \bar{\mathbf{u}}_i \lambda_1 \end{Bmatrix} = \begin{bmatrix} -\bar{\mathbf{u}}_i \\ -\bar{\mathbf{s}}_j^P \bar{\mathbf{u}}_i \end{bmatrix} \lambda_1 \Rightarrow {}^{(r)}\mathbf{h}_j = {}^{(r-t,1)}\mathbf{D}'_j \lambda_1 \quad (7.63)$$

The point of application of the reaction force on body (i) is \$P\_i^\*\$. The moment arm of this force is \$\bar{s}\_i^P = \bar{s}\_i^P - \bar{d} - L\bar{u}\_i\$, where \$L\$ is the offset length of the massless bracket. The corresponding moment with respect to the mass center of body (i) can be determined as

$$\begin{aligned} n_i &= \bar{\mathbf{s}}_i^P (\bar{\mathbf{u}}_i \lambda_1) = (\bar{\mathbf{s}}_i^P - \bar{\mathbf{d}}' - L\bar{\mathbf{u}}'_i) \bar{\mathbf{u}}_i \lambda_1 \\ &= (\bar{\mathbf{s}}_i^P - \bar{\mathbf{d}}') \bar{\mathbf{u}}_i \lambda_1 = (\mathbf{s}_i^P - \mathbf{d}') \mathbf{u}_i \lambda_1 = \mathbf{u}'_i (\mathbf{s}_i^P - \mathbf{d}) \lambda_1 \end{aligned}$$

Now the array of reaction force/moment for body (i) can be written as

$${}^{(r)}\mathbf{h}_i = \begin{Bmatrix} \bar{\mathbf{u}}_i \lambda_1 \\ \bar{\mathbf{s}}_i^P \bar{\mathbf{u}}_i \lambda_1 \end{Bmatrix} = \begin{bmatrix} \bar{\mathbf{u}}_i \\ \mathbf{u}'_i (\mathbf{s}_i^P - \mathbf{d}) \end{bmatrix} \lambda_1 \Rightarrow {}^{(r)}\mathbf{h}_i = {}^{(r-t,1)}\mathbf{D}'_i \lambda_1 \quad (7.64)$$

For our four fundamental joints, we have observed that the Lagrange multipliers represent exactly the reaction force or the reaction torque associated with a joint. This may not always be the case. For example, for the revolute–revolute joint, if instead of using Eq. (7.27) we consider the constraint  ${}^{(r-r,1)}\Phi = \mathbf{d}'\mathbf{d} - L^2 = 0$ , the Jacobian submatrices would be

$${}^{(r-r,1)}\mathbf{D}_i = \begin{bmatrix} 2\mathbf{d}' & 2\mathbf{d}'\tilde{\mathbf{s}}_i^P \end{bmatrix}, \quad {}^{(r-r,1)}\mathbf{D}_j = \begin{bmatrix} -2\mathbf{d}' & -2\mathbf{d}'\tilde{\mathbf{s}}_j^P \end{bmatrix}$$

With these matrices, the array of reaction force for the two bodies become

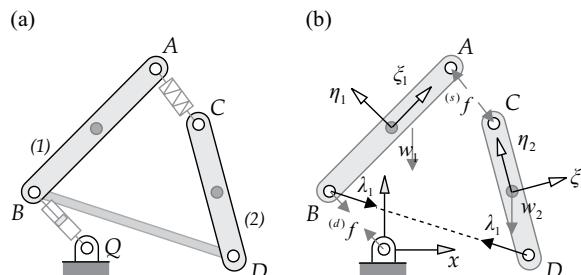
$${}^{(r-r,1)}\mathbf{h}_i = \begin{bmatrix} 2\mathbf{d} \\ 2\mathbf{d}'\tilde{\mathbf{s}}_i^P \end{bmatrix} \lambda_1, \quad {}^{(r-r,1)}\mathbf{h}_j = \begin{bmatrix} -2\mathbf{d} \\ -2\mathbf{d}'\tilde{\mathbf{s}}_j^P \end{bmatrix} \lambda_1$$

With this presentation, the reaction force acting on bodies  $(i)$  and  $(j)$  are  $2\mathbf{d}\lambda_1$  and  $-2\mathbf{d}\lambda_1$ , respectively. Since the magnitude of  $2\mathbf{d}$  is not necessarily “one,”  $\lambda_1$  is only a coefficient and not necessarily the directional magnitude of the reaction force. In the submatrices that we have derived for the commonly used joints, the coefficient for a Lagrange multiplier is either a one or a unit vector. In those cases, the multipliers are exactly the directional magnitudes of the reaction forces or torques.

### Example 7.2

This system is a variation of Example 7.1. As shown in Figure 7.15a, a massless rod having a length of  $L = 0.355$  m is attached between points  $D$  and  $B$ ; otherwise, no other changes have been made.

For the given coordinates and velocities as in Example 7.1, construct the mass matrix, the array of applied forces, the Jacobian matrix, and the right-hand side of the acceleration constraint. Evaluate the constraints at the coordinate and velocity levels to determine whether the given coordinates and velocities satisfy the constraints.



**FIGURE 7.15**

(a) Two constrained bodies and (b) the corresponding FBD.

**Solution**

The construction and evaluation of the mass matrix and the array of applied forces are the same as in Example 7.1. For the added revolute–revolute joint, we need to state the length of the link,  $L = 0.355\text{ m}$ , and the coordinates of point  $D$  on body (2) as  $\mathbf{s}_2^D = \begin{Bmatrix} 0 & -0.12 \end{Bmatrix}' \text{ m}$ . This constraint provides reaction forces as shown in Figure 7.15b.

**MATLAB/Chapter 7/Example \_ 7 \_ 2**

We add the following data to the program of Example 7.1:

```
...  
s_D2_local = [ 0; -0.12]; L = 0.355;
```

We determine the  $x$ - $y$  components of  $\vec{s}_2^D$ , the  $x$ - $y$  coordinates of point  $D$ , and the velocity of point  $D$ :

```
s_D2 = A2*s_D2_local; r_D2 = r2 + s_D2;  
r_D2_d = r_Point_d(r2_d, s_D2, phi2_d);
```

We construct a vector  $\vec{d}$  between  $B$  and  $D$ , and its time derivative, and a unit vector:

```
d3 = r_B1 - r_D2; d3_d = r_B1_d - r_D2_d;  
u3 = de/L;
```

We can now evaluate the constraint for the revolute–revolute joint from Eq. (7.27):

Phi = (u3'*d3 - L)/2 .....	Phi =
	-4.3136e-04

The computed value for the constraint is very close to zero. That means the coordinate values were not given randomly.

Next, we construct the Jacobian matrix, based on Eq. (7.29) as

$$(r-r,1)\mathbf{D}_1 = \begin{bmatrix} \mathbf{u}_3' & \mathbf{u}_3' \check{\mathbf{s}}_1^B \end{bmatrix}, \quad (r-r,1)\mathbf{D}_2 = \begin{bmatrix} -\mathbf{u}_3' & -\mathbf{u}_3' \check{\mathbf{s}}_2^D \end{bmatrix}$$

```
D = [u3' u3'*s_rot(s_B1) -u3' -u3'*s_rot(s_D2)] .....  
D =  
-0.9496 0.3094 -0.1335 0.9496 -0.3094 0.1005
```

We construct the array of velocities and evaluate Eq. (7.51) for violation at the velocity level:

c_d = [r1_d; phil1_d; r2_d; phi2_d]; Phi_d = D*c_d .....	Phi_d =
	-1.7269e-04

We note that the velocity constraint is also satisfied.

Next we compute the necessary vectors for evaluating the right-hand side of the acceleration constraint, based on Eq. (7.30), as

$$(r-r,1)\gamma = -\dot{\mathbf{u}}_3' \dot{\mathbf{d}}_3 + \ddot{\mathbf{u}}_3' (\check{\mathbf{s}}_1^B \dot{\phi}_1 - \check{\mathbf{s}}_2^D \dot{\phi}_2)$$

d3_d = r_B1_d - r_D2_d; u3_d = d3_d/L; s_B1_d = s_rot(s_B1)*phil1_d; s_D2_d = s_rot(s_D2)*phi2_d; gamma = -u3_d'*d3_d + s_rot(u3)'* ... (s_B1_d*phil1_d - s_D2_d*phi2_d) ....	gamma =
	0.0064

We can now construct Eq. (7.56) and solve for the accelerations and one Lagrange multiplier representing the reaction force along the massless rod.

```
solution = [M -D'  
           D  0 ] \ [h; gamma] ..... solution =  
5.8784  
-5.3360  
45.3366  
5.8953  
-30.7640  
-18.1473  
-1.7821
```

Accelerations and the Lagrange multiplier are

$$\ddot{\mathbf{r}}_1 = \begin{Bmatrix} 5.88 \\ -5.34 \end{Bmatrix} \text{ m/s}^2, \ddot{\phi}_1 = 45.34 \text{ rad/s}^2, \ddot{\mathbf{r}}_2 = \begin{Bmatrix} 5.90 \\ -30.76 \end{Bmatrix} \text{ m/s}^2,$$

$$\ddot{\phi}_2 = -18.15 \text{ rad/s}^2, \lambda = -1.78$$

### Example 7.3

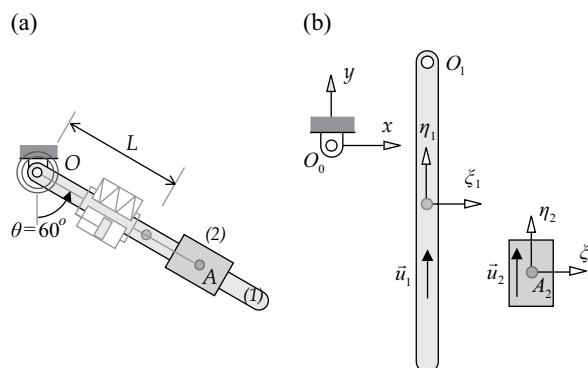
The variable-length pendulum shown in Figure 7.16a contains two bodies, a pin joint, a sliding joint, a rotational spring about the pin joint at  $O$ , and a point-to-point spring-damper between point  $O$  and the mass center of body (2) at  $A$ . Gravity acts on the system. The following constant data are provided:

$$L = 0.5 \text{ m}, m_1 = 2.0 \text{ kg}, J_1 = 0.04 \text{ kg m}^2, m_2 = 1.0 \text{ kg}, J_2 = 0.1 \text{ kg m}^2$$

$${}^{(r)}k = 6 \text{ N m/rad}, {}^0\theta = 0.0 \text{ (rotational spring)}$$

$$k = 20 \text{ N/m}, {}^0L = 0.7 \text{ m}, d_c = 5 \text{ N s/m} \text{ (point-to-point spring-damper)}$$

The individual bodies are shown in Figure 7.16b. At the initial time, the system is at rest where the pendulum makes a  $60^\circ$  angle with the vertical axis, and the mass center of the



**FIGURE 7.16**

(a) The variable-length pendulum and (b) its corresponding FBD.

slider is 0.8m from point  $O$ . Determine the acceleration and the Lagrange multipliers at the initial time.

### Solution

#### MATLAB/Chapter 7/Example 7 \_ 3

We enter the constant data:

```
s_O1_local = [0; 0.5]; u1_local = [0; 1]; u2_local = [0; 1];
theta0 = 0; k_r = 6;
L0 = 0.7; k = 20; dc = 5;
m = [2; 1]; J = [0.04; 0.1]; g = 9.81; uy = [0; 1];
```

Based on the initial orientation of the pendulum and the position of the slider, we determine the following initial values for the body coordinates (body velocities are zero):

```
r1 = [0.4330; -0.2500]; phi1 = pi/3;
r2 = [0.6928; -0.4000]; phi2 = pi/3;
r1_d = [0; 0]; phi1_d = 0; r2_d = [0; 0]; phi2_d = 0;
```

For constructing the Jacobian matrix and the right-hand side of the acceleration constraints, we first compute the following matrices and vectors:

```
A1 = A_matrix(phi1); A2 = A_matrix(phi2);
s_O1 = A1*s_O1_local;
u1 = A1*u1_local; u2 = A2*u2_local;
d = r2 - r1; d_d = r2_d - r1_d; u1r = s_rot(u1);
```

We can now construct the Jacobian matrix for the revolute and translational joints. For the revolute joint, we refer to Eq. (7.16) to construct the following submatrix:

$${}^{(r,2)}\mathbf{D}_1 = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{s}}_1^O \end{bmatrix}$$

Since the other body is the ground, it does not contribute a submatrix to the Jacobian matrix. For the translational joint, we use Eq. (7.21) to form the following submatrices:

$${}^{(t,2)}\mathbf{D}_2 = \begin{bmatrix} \tilde{\mathbf{u}}_1' & 0 \\ 0 & 1 \end{bmatrix}, {}^{(t,2)}\mathbf{D}_1 = \begin{bmatrix} -\tilde{\mathbf{u}}_1' & -\mathbf{u}_1' \mathbf{d} \\ 0 & -1 \end{bmatrix}$$

```
D = [eye(2) s_rot(s_O1) zeros(2,3)
      -u1r' -u1'*d u1r' 0
      0 0 -1 0 0 1] .....
```

```
D =
1.0000 0 -0.2500 0 0 0
0 1.0000 -0.4330 0 0 0
0.5000 0.8660 0.3000 -0.5000 -0.8660 0
0 0 -1.0000 0 0 1.0000
```

We also construct the right-hand side array of acceleration constraints for the revolute and translational joints, by referring to Eqs. (7.18) and (7.23). At the initial time, since all the velocities are zero, all the entries of this array should be zeros.

```
gamma = [s_O1*phi1_d^2
         (u1r'*d*phi1_d + 2*u1'*d_d)*phi1_d
         0];
```

We construct the mass matrix and compute the torques of the torsional and the force of the point-to-point springs, and the gravitational forces:

$M = [m(1) \ m(1) \ J(1) \ m(2) \ m(2) \ J(2)]';$	$T_r =$	
$T_r = r_s(phi_1, k_r, theta_0) \ \dots \dots \dots$	$6.2832$	
$T1 = -T_r;$	$f_sd =$	
$f_sd = pp_sd(r2, r2_d, k, L0, dc) \ \dots$	$1.7317$	
$f2 = -f_sd;$	$w1 = -m(1)*g*uy; w2 = -m(2)*g*uy;$	$-0.9998$

Next we construct the array of applied forces:

$f_a = [w1; T1; w2 + f2; 0] \ \dots \dots \dots$	$f_a =$
	0
	-19.6200
	-6.2832
	-1.7317
	-8.8102
	0

The mass matrix and the Jacobian matrix are used to construct a coefficient matrix, and the arrays of applied forces and gamma are appended to form the right-hand side array. The equations are solved for the accelerations and Lagrange multipliers:

$DMD = [diag(M) \ -D'$	$c_dd =$
$\quad D \ zeros(4)]$	-4.2140
$rhs = [f_a; gamma]$	-7.2988
$solution = DMD\rhs;$	-16.8559
$c_dd = solution(1:6) \ \dots \dots \dots \dots \dots \dots$	-4.2262
$Lag = solution(7:10) \ \dots \dots \dots \dots \dots \dots$	-13.1307
	-16.8559
	$Lag =$
	-10.9224
	0.7018
	4.9889
	-1.6856

The first two multipliers are the reaction forces at the pin joint, and the last two are the reaction force and torque at the sliding joint.

## 7.5 Total Energy

Monitoring the total energy of a system that is in motion can be useful for various purposes, specifically for determining if there are any errors in the constructed set of equations of motion. The total energy of a conservative system must remain unchanged during a dynamic simulation. Therefore, for such a system, if the total energy does not remain a constant, which would indicate the existence of at least one error in the model. To perform

a similar test on a nonconservative system, any energy dissipating components in the model, such as dampers and frictions, can temporarily be inactivated.

To compute the total energy of a system, we must compute its kinetic energy and its potential energy associated with the springs, gravity, and any constant applied forces. The kinetic energy for all of the bodies of a system, according to Eq. (4.53), can be computed as

$$T = \frac{1}{2} \dot{\mathbf{c}}' \mathbf{M} \dot{\mathbf{c}} \quad (7.65)$$

To compute the potential energy associated with gravity, if included in a model, we sum Eq. (4.55) for all the bodies as

$${}^{(g)}V = \sum_{i=1}^{n_b} \mathbf{w}_i' \mathbf{r}_i \quad (7.66)$$

For any point-to-point or rotational springs in a model, we compute Eqs. (4.57) and (4.58), and include them in the total energy of the system as

$$E = T + \sum_{i=1}^{n_b} {}^{(g)}V_i + \sum_{j=1}^{n_{springs}} {}^{(s)}V_j + \dots \quad (7.67)$$

This equation can be implemented in a program to compute the total energy of a multi-body system.

## 7.6 Problems

7.1 Points *A* and *B* are defined on bodies (1) and (2). Consider the following data:

$$\mathbf{s}_2^B = \begin{Bmatrix} 0.8 \\ -0.5 \end{Bmatrix}, \mathbf{s}_2^B = \begin{Bmatrix} -0.3 \\ 1.2 \end{Bmatrix}, \mathbf{r}_1 = \begin{Bmatrix} 2.0 \\ 1.5 \end{Bmatrix}, \phi_1 = 30^\circ, \mathbf{r}_2 = \begin{Bmatrix} 4.5 \\ 2.3 \end{Bmatrix}, \phi_2 = 135^\circ$$

- a. Determine the components of vector  $\vec{d}$  that connects point *A* to point *B*.
- b. Determine the components of a unit vector along  $\vec{d}$ .

7.2 Vectors  $\vec{a}$  is attached to body (1) and vectors  $\vec{b}$  and  $\vec{c}$  are attached to body (2). The following data are provided:

$$\mathbf{a}_1 = \begin{Bmatrix} 0.1 \\ -0.4 \end{Bmatrix}, \mathbf{b}_2 = \begin{Bmatrix} -0.5 \\ 0.2 \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} -0.2 \\ -0.4 \end{Bmatrix}$$

$$\mathbf{r}_1 = \begin{Bmatrix} 2.0 \\ 1.5 \end{Bmatrix}, \phi_1 = 30^\circ, \mathbf{r}_2 = \begin{Bmatrix} 4.5 \\ 2.3 \end{Bmatrix}, \phi_2 = 135^\circ$$

- a. Vectors  $\vec{a}$  and  $\vec{c}$  are supposed to be perpendicular. Determine the amount of constraint violation.

- b. Vectors  $\vec{a}$  and  $\vec{b}$  are supposed to be parallel. Determine the amount of constraint violation.

7.3 Points  $A$  and  $B$  are defined on bodies (1) and (2). Consider the following data:

$$\mathbf{s}_1^A = \begin{Bmatrix} 1.5 \\ 1.2 \end{Bmatrix}, \mathbf{s}_2^B = \begin{Bmatrix} 2.0 \\ 0.6 \end{Bmatrix}, \mathbf{r}_1 = \begin{Bmatrix} 2.0 \\ 1.5 \end{Bmatrix}, \phi_1 = 30^\circ, \mathbf{r}_2 = \begin{Bmatrix} 4.5 \\ 2.3 \end{Bmatrix}$$

The two points should coincide to form a revolute joint. Determine the amount of constraint violation.

7.4 Expand the following constraints and Jacobian matrices by expressing them in terms of the coordinates of the two bodies:

- a. Translational joint; Eqs. (7.19) and (7.21)
- b. Revolute–revolute joint; Eqs. (7.27) and (7.29)
- c. Revolute–translational joint; Eqs. (7.31) and (7.33)

7.5 Simplify the constraints, the Jacobian matrices, and the right-hand side of the acceleration constraints for the following joints between body ( $i$ ) and the ground:

- a. Revolute joint
- b. Translational joint
- c. Revolute–revolute joint
- d. Revolute–translational joint

7.6 Point  $P_i$  is defined on body ( $i$ ). For the following conditions, in expanded form, derive the constraints, the Jacobian matrices, and the right-hand side of the acceleration constraints:

- a. Keep  $x_i^P$  to be a constant  $c_1$ .
- b. Keep  $y_i^P$  to be a constant  $c_2$ .

7.7 Two axes on bodies ( $i$ ) and ( $j$ ) must be kept perpendicular to each other. The axis on body ( $i$ ) is defined by a unit vector having local components  $\mathbf{u}_i = \{0.6 \ -0.8\}'$ . The axis on body ( $j$ ) passes through two points,  $B$  and  $C$ , having local coordinates  $\mathbf{s}_j^B = \{0.5 \ 0.4\}'$  and  $\mathbf{s}_j^C = \{1.3 \ -0.2\}'$ .

- a. Write the constraint equation(s) in terms of the coordinates of the bodies.
- b. Write the velocity constraint(s).
- c. Determine the entries of the Jacobian matrix.

7.8 Vectors  $\vec{s}_1$  and  $\vec{s}_2$  are attached to bodies (1) and (2) having local components

$$\mathbf{s}_1 = \{1.2 \ -0.5\}' \text{ and } \mathbf{s}_2 = \{-0.3 \ 0.8\}'. \text{ An array of coordinates is defined as } \mathbf{c} = \{x_1 \ y_1 \ \phi_1 \ x_2 \ y_2 \ \phi_2\}'.$$

- a. If  $\phi_1 = 30^\circ$  and  $\phi_2 = 45^\circ$ , evaluate the entries of the Jacobian matrix for the constraint  $\Phi = \mathbf{s}_1' \mathbf{s}_2$ .
- b. If  $x_1 = 6.2$ ,  $y_1 = 1$ ,  $\phi_1 = 30^\circ$ ,  $x_2 = -1.9$ ,  $y_2 = 2.3$ , and  $\phi_2 = 45^\circ$ , evaluate the entries of the Jacobian matrix for the constraint  $\Phi = \vec{s}_1' \mathbf{d}$ , where vector  $\vec{d} = \vec{r}_i - \vec{r}_j$  connects the two origins.

7.9 The body-fixed coordinates of revolute joints  $A$  and  $B$  of a four-bar mechanism, and the global coordinates of the bodies are given as

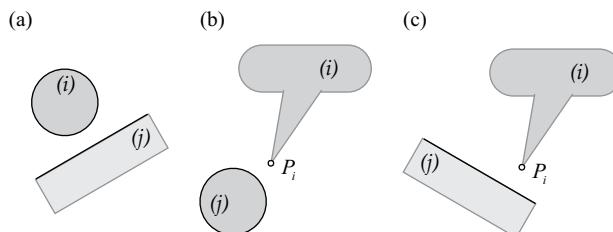
$$\mathbf{s}_1^A = \begin{Bmatrix} 0 \\ 1.5 \end{Bmatrix}, \mathbf{s}_2^A = \begin{Bmatrix} -2.2 \\ 0 \end{Bmatrix}, \mathbf{s}_2^B = \begin{Bmatrix} 2.2 \\ 0 \end{Bmatrix}, \mathbf{s}_3^B = \begin{Bmatrix} 2.0 \\ 0 \end{Bmatrix}$$

$$\mathbf{c}_1 = \begin{Bmatrix} -2.0 \\ 2.5 \\ -12^\circ \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} 0.5 \\ 3.6 \\ -8^\circ \end{Bmatrix}, \mathbf{c}_3 = \begin{Bmatrix} 1.6 \\ 1.7 \\ 56^\circ \end{Bmatrix}$$

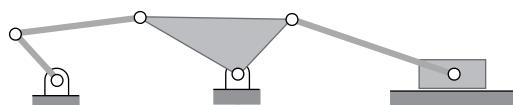
Are the constraint equations for these two joints violated or not?

- 7.10 Two bodies, (i) and (j), can translate and rotate in the plane. For the following cases, describe the necessary constraint(s) to keep the two bodies in contact:
- A circle with known radius to remain in contact with the surface (line) of another body but free to roll or slide
  - Point  $P$  on body (i) to remain in contact with the circle but free to slide
  - Point  $P$  on body (i) to remain in contact with the surface (line) of the other body but free to slide

Assume that a flat surface (line) extends without limit at either end. *Hint:* To define a specific line on a body, in the body-fixed frame, define a point on the line and a unit vector perpendicular to the line.



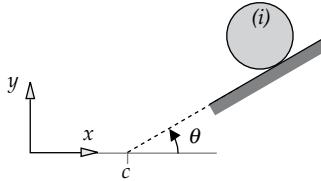
- 7.11 Consider modeling the mechanism shown for kinematic analysis by the body-coordinate formulation. Determine the *minimum* number of bodies, coordinates, and constrain equations that are needed if
- Only revolute and translational joint constraints are to be used.
  - Revolute, translational, revolute–revolute, and revolute–translational joints are all available for formulation.
- 7.12 In the translational joint constraints of Eqs. (7.19)–(7.23), it is possible for vector



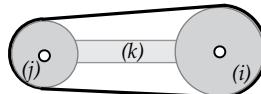
$\vec{d}$  to become a zero vector as the bodies slide relative to each other. Does this cause any numerical difficulty in the analysis? *Hint:* Numerical difficulty can be encountered when the Jacobian matrix of the constraint loses rank.

- 7.13 A disc can roll on an inclined flat surface as shown. Derive the necessary constraints for

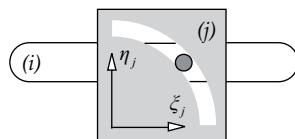
- a. Roll with slip
- b. Roll without slip



- 7.14 Derive the constraint equation for two discs that are connected by a belt while attached to a third body by pin joints. Determine the Jacobian matrix and the right-hand side of the acceleration constraint. Consider two cases:
- a. Body (k) is the ground.
  - b. Body (k) can also translate and rotate.



- 7.15 Derive the constraint and the Jacobian matrix for a point-follower joint where the slot on body (j) describes a circle of radius  $R$  with its center at the origin of the body-fixed frame.



- 7.16 Derive the velocity constraint equation(s) to keep the translational speed of a body equal to  $s$  (a constant) along a known direction denoted by a unit vector  $\vec{u}$ . Consider two cases:
- a. Vector  $\vec{u}$  is fixed to the ground.
  - b. Vector  $\vec{u}$  is fixed to the body.

Are the constraints holonomic or non-holonomic?

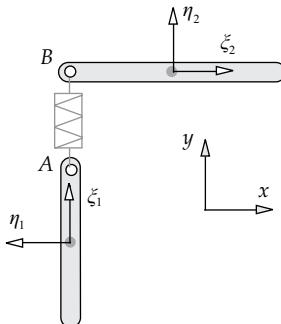
- 7.17 What is the size of the Jacobian matrix for a four-bar linkage modeled by three moving bodies and four revolute joints? Show the location of the nonzero entries in the matrix. What percentage of the elements of the matrix is nonzero?
- 7.18 What is the size of the Jacobian matrix for a slider-crank mechanism in each of the following cases? The mechanism is modeled by
- a. Three moving bodies, three revolute joints, and one translational joint
  - b. Two moving bodies, one revolute joint, one revolute–revolute joint, and one translational joint
  - c. Two moving bodies, two revolute joints, and one revolute–translational joint

For each case, show the location of the nonzero entries in the corresponding Jacobian matrix, and determine the percentage of the nonzero elements of the matrix.

- 7.19 Two bodies are connected by a point-to-point spring as shown. The body-fixed coordinates of the spring attachment points and the coordinates of the two bodies are

$$\mathbf{s}_1^A = \begin{Bmatrix} 0.11 \\ 0 \end{Bmatrix}, \mathbf{s}_2^B = \begin{Bmatrix} -0.15 \\ 0 \end{Bmatrix}, \mathbf{c}_1 = \begin{Bmatrix} -0.20 \\ -0.05 \\ 90^\circ \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} -0.05 \\ 0.20 \\ 0 \end{Bmatrix}$$

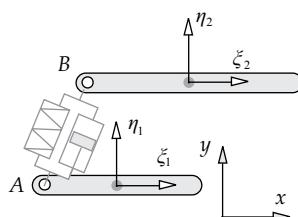
The inertial and spring data are given as follows:  $m_1 = 0.2 \text{ kg}$ ,  $J_1 = 0.03 \text{ kg m}^2$ ,  $m_2 = 0.15 \text{ kg}$ ,  $J_2 = 0.02 \text{ kg m}^2$ ,  $k = 50 \text{ N/m}$ , and  ${}^0L = 0.1 \text{ m}$ . The gravity acts on the system. Construct the equations of motion.



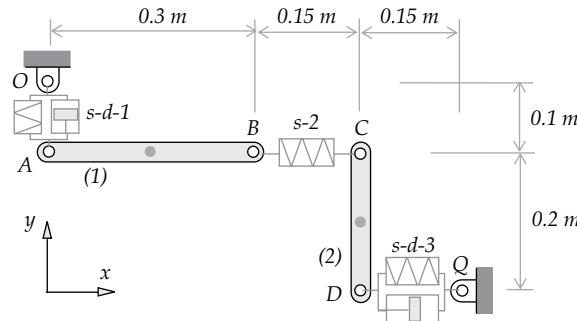
- 7.20 Two bodies are connected by a point-to-point spring-damper element as shown. The attachment points have the following body-fixed coordinates:  $\mathbf{s}_1^A = \{-0.11 \ 0\}'$  and  $\mathbf{s}_2^B = \{-0.15 \ 0\}'$ . In the shown configuration, the two bodies have the following coordinates and velocities (angular velocities in rad/s):

$$\mathbf{c}_1 = \begin{Bmatrix} -0.15 \\ 0.05 \\ 0 \end{Bmatrix}, \dot{\mathbf{c}}_1 = \begin{Bmatrix} 0.04 \\ -0.06 \\ 0.1 \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} -0.05 \\ 0.20 \\ 0 \end{Bmatrix}, \dot{\mathbf{c}}_2 = \begin{Bmatrix} -0.01 \\ 0.03 \\ -0.2 \end{Bmatrix}$$

The inertial and spring data are given as follows:  $m_1 = 0.2 \text{ kg}$ ,  $J_1 = 0.03 \text{ kg m}^2$ ,  $m_2 = 0.15 \text{ kg}$ ,  $J_2 = 0.02 \text{ kg m}^2$ ,  $k = 50 \text{ N/m}$ ,  ${}^0L = 0.1 \text{ m}$ , and  $d_c = 30 \text{ N s/m}$ . Construct the equations of motion.



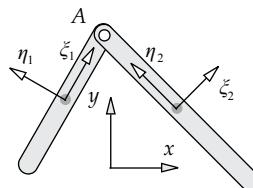
- 7.21 Two rods with uniform mass distribution are connected to each other and to the ground by point-to-point spring and damper elements as shown. The corresponding data are given as follows:  $m_1 = 0.4\text{kg}$ ,  $J_1 = 0.3\text{ kg m}^2$ ,  $m_2 = 0.3\text{kg}$ ,  $J_2 = 0.1\text{ kg m}^2$ ,  $k_1 = 40\text{ N/m}$ ,  ${}^0L_1 = 0.12\text{ m}$ ,  $d_{c-1} = 12\text{ Ns/m}$ ,  $k_2 = 60\text{ N/m}$ ,  ${}^0L_2 = 0.2\text{ m}$ ,  $k_3 = 25\text{ N/m}$ ,  ${}^0L_3 = 0.13\text{ m}$ , and  $d_{c-3} = 24\text{ Ns/m}$ . Obtain any necessary geometric data from the figure. Assume that the only nonzero velocities are  $\dot{\gamma}_1 = -0.3$  and  $\dot{\phi}_2 = -0.15$ . Construct the equations of motion.



- 7.22 Two bodies are connected by a revolute joint as shown. The joint attachment points are at  $s_1^A = \begin{Bmatrix} 0.11 & 0 \end{Bmatrix}'\text{ m}$  and  $s_2^A = \begin{Bmatrix} 0 & 0.15 \end{Bmatrix}'\text{ m}$ . A force of 10N acts in the negative  $x$ -direction at the mass center of body (1), and another force of 15N acts on the mass center of body (2) in the positive  $y$ -direction. In the shown configuration, the two bodies have the following coordinates and velocities (angular velocities in rad/s):

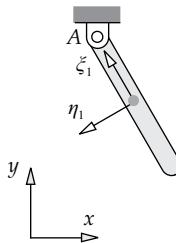
$$\mathbf{c}_1 = \begin{Bmatrix} -0.07 \\ 0.10 \\ 60^\circ \end{Bmatrix}, \dot{\mathbf{c}}_1 = \begin{Bmatrix} 0.04 \\ -0.06 \\ 0.1 \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} 0.10 \\ 0.09 \\ 45^\circ \end{Bmatrix}, \dot{\mathbf{c}}_2 = \begin{Bmatrix} 0.01 \\ -0.08 \\ -0.2 \end{Bmatrix}$$

We have the following inertial data:  $m_1 = 0.2\text{kg}$ ,  $J_1 = 0.03\text{ kg m}^2$ ,  $m_2 = 0.15\text{kg}$ , and  $J_2 = 0.02\text{ kg m}^2$ . Construct the equations of motion. Check the position and velocity constraints for any violations.

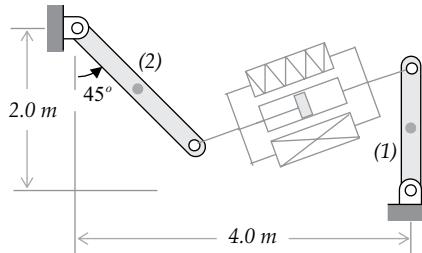


- 7.23 For the single pendulum shown, construct the equations of motion using the following constant data:  $r_0^A = \begin{Bmatrix} 0.1 & 0.3 \end{Bmatrix}'\text{ m}$ ,  $s_1^A = \begin{Bmatrix} 0.11 & 0 \end{Bmatrix}'\text{ m}$ ,  $m_1 = 0.2\text{kg}$ , and  $J_1 = 0.03\text{ kg m}^2$ . Consider the gravity and evaluate the variable elements of the equations of motion for the following coordinates and velocities (angular velocity in rad/s):

$$\mathbf{c}_1 = \begin{Bmatrix} 0.15 & 0.20 & 120^\circ \end{Bmatrix}', \dot{\mathbf{c}}_1 = \begin{Bmatrix} -0.19 & -0.11 & -2.0 \end{Bmatrix}'$$



- 7.24 Two rods are pinned to the ground as shown. The lengths of the rods are 1.5 and 2.0m, respectively. A spring–damper–actuator element is connected between the free ends of the rods having the following characteristics:  $k = 100\text{ N/m}$ ,  ${}^0L = 2\text{ m}$ ,  $d_c = 20\text{ Ns/m}$ , and  ${}^{(a)}f = 30\text{ N}$ . In the shown configuration, the angular velocities of the two bodies are, respectively,  $1.0\text{ rad/s}$  CW, and  $2.5\text{ rad/s}$  CCW. Gravity acts on the system. Construct the equations of motion for this system. Assume  $m_1 = 0.8$ ,  $m_2 = 1.2\text{ kg}$ ,  $J_1 = 0.15$ , and  $J_2 = 0.40\text{ kg m}^2$ .



- 7.25 A point-to-point spring–damper–actuator element is connected between two bodies at point A on link (1) and point B on link (2). The two points are positioned in their respective  $\xi$ – $\eta$  frame as

$$\mathbf{s}_1^A = \begin{Bmatrix} -0.25 \\ 0.12 \end{Bmatrix} \text{ m}, \mathbf{s}_2^B = \begin{Bmatrix} 0.15 \\ 0.20 \end{Bmatrix} \text{ m}$$

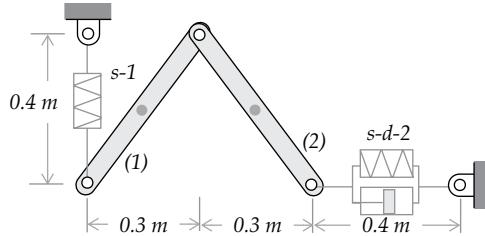
The actuator pulls on the bodies with a force of  $4.5\text{ N}$ . The spring has an undeformed length  ${}^0L = 0.35\text{ m}$  and a stiffness  $k = 30\text{ N/m}$ . The damping coefficient is  $d_c = 15\text{ N s/m}$ . At a given instant, the two bodies have the following coordinates and velocities (angular data in rad and rad/s):

$$\mathbf{c}_1 = \begin{Bmatrix} 0.32 \\ 0.23 \\ -0.55 \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} -0.12 \\ 0.16 \\ 0.60 \end{Bmatrix}, \dot{\mathbf{c}}_1 = \begin{Bmatrix} -0.10 \\ 0.21 \\ 0.0 \end{Bmatrix}, \dot{\mathbf{c}}_2 = \begin{Bmatrix} 0.18 \\ 0.09 \\ -0.20 \end{Bmatrix}$$

Assume  $m_1 = 1.5\text{ kg}$ ,  $J_1 = 2.0\text{ kg m}^2$ ,  $m_2 = 1.3\text{ kg}$ , and  $J_2 = 1.8\text{ kg m}^2$ . Construct the equations of motion.

- 7.26 Two rods are connected to each other by a pin joint and connected to the ground by spring–damper elements as shown. The gravity acts on the system. We have the following inertial and spring–damper data:  $m_1 = m_2 = 6\text{ kg}$ ,  $J_1 = J_2 = 12.5\text{ kg m}^2$ ,

$k_1 = 20 \text{ N/m}$ ,  ${}^0L_1 = 5 \text{ m}$ ,  $k_2 = 30 \text{ N/m}$ ,  ${}^0L_2 = 4.5 \text{ m}$ , and  $d_{\dot{\phi}_2} = 6 \text{ N s/m}$ . The following velocity values are known:  $\dot{x}_1 = 0.5$ ,  $\dot{y}_1 = -0.2 \text{ m/s}$ ,  $\dot{\phi}_1 = 0.6$ , and  $\dot{\phi}_2 = -0.3 \text{ rad/s}$ . Determine the remaining velocities and then construct the equations of motion.



- 7.27 For each of the systems shown, consider the gravity as an applied force. Use the following data for each system as applicable:

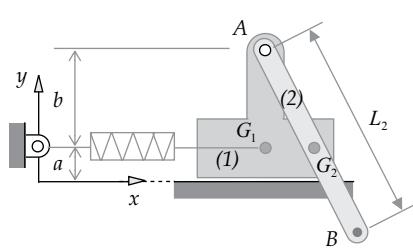
$$a = 0.01, b = 0.03, c = 0.02, L_2 = 0.06, L_3 = 0.04 \text{ m}$$

$$k = 100 \text{ N/m}, {}^0L = 0.08 \text{ m}$$

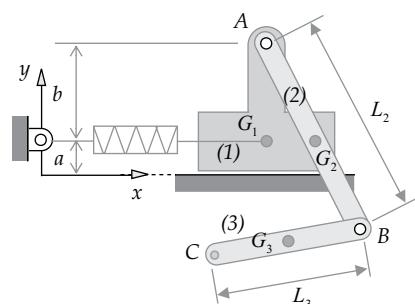
$$m_1 = 0.3, m_2 = 0.2, m_3 = 0.15 \text{ kg}, J_1 = 0.05, J_2 = 0.001, J_3 = 0.0008 \text{ kg m}^2$$

Assume the deformed length of the spring to be  ${}^0L = 0.07 \text{ m}$ . In the shown configurations, construct the equations of motion. Extract values for any angles from the figures.

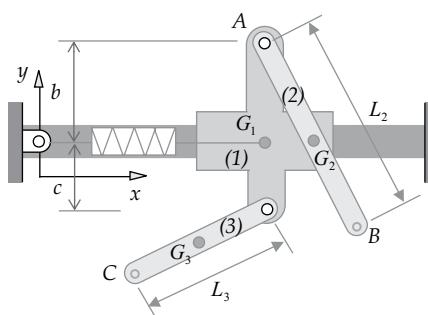
(a)



(b)



(c)



# 8

---

## *Body-Coordinate Simulation Program*

---

The body-coordinate formulation, as discussed in Chapter 7, is a method suitable for implementation in a computer program. In this chapter, we present a MATLAB® program that is based on the body-coordinate formulation. The dynamic analysis program with body coordinates (DAP\_BC) is a general-purpose program that is not developed for a specific problem—it can simulate the dynamics of a variety of problems based on the description and the data provided by the user. The program contains most of the features that have been discussed in Chapter 7, such as kinematic joints and force elements. A user can modify the program to include new features and capabilities. The program has been developed with the following objectives in mind:

- Suitability for instructional purposes
- As an aid for learning multibody dynamics
- Program simplicity (at the cost of computational efficiency)
- A balance between the ease of understanding the program structure versus implementing the more advanced capabilities of MATLAB
- Ease of program modifications by a user

In its present form, the program can be used to model a multibody containing

- As many bodies as needed
- Force elements such as
  - Point-to-point spring-damper-actuators
  - Rotational spring-damper-actuators
  - Constant forces and torques that can act directly on bodies
  - Gravitational force
  - User-defined forces
- Kinematic joints such as
  - Revolute (pin)
  - Translational (sliding)
  - Revolute-revolute
  - Revolute-translational
  - Rolling disc
  - Rigid
- Driver constraints such as
  - Relative rotation (a motor)
  - Relative translation (an actuator)
- Several defined functions

The program is capable of performing kinematic, inverse dynamic, and forward dynamic analyses by integrating the equations of motion (refer to Chapters 12 and 13). The program can easily be revised to perform static equilibrium analysis (refer to Chapter 14). The program can correct the user-provided initial conditions on the coordinates and velocities in order to satisfy the kinematic constraints (refer to Chapter 14). An animation program that provides a simple animated stick-drawing of the dynamic response of the analyzed model accompanies the analysis program.

For detailed description on how to use this program, the reader should refer to the User Manual in Appendix B in the downloaded folder. In this appendix, through several examples, the reader is guided how to describe a multibody model for analysis and how to take advantage of different features and capabilities of the program.

## 8.1 Application Examples

In this section, some of the example models that accompany the DAP \_ BC program are reviewed. The MATLAB statements in the M-files that describe a model are discussed in the manual but not in the textbook. Although some of the examples are very simple, the purpose of discussing them is to review the modeling aspects of as many features of the program as possible.

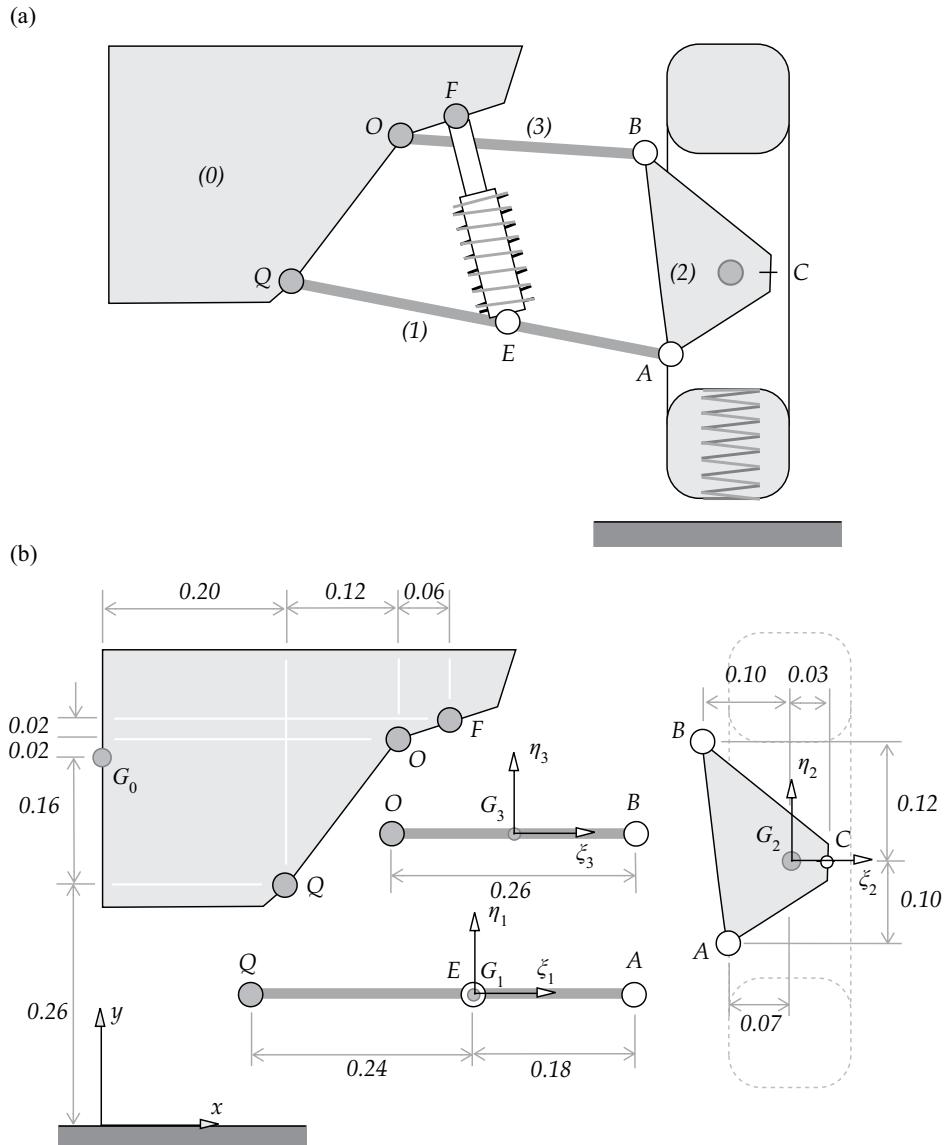
In each example, the multibody model is described with as much detail as needed. The body indices are marked in parentheses as (1), (2), etc. The points are marked in square brackets as [1], [2], etc. The unit vectors are identified in double square brackets as [[1]], [[2]], etc.

The program DAP \_ BC and the accompanying models are subject to revisions (and possible corrections) over time. Therefore, the downloaded materials may not be exactly identical to the corresponding listings and discussions prior to the publication of the book.

### 8.1.1 Double A-Arm Suspension

**MATLAB/Chapter 8/DAP \_ BC/Models/AA**

This example, as shown in Figure 8.1a, is the planar representation of a double A-arm suspension system, where its detailed description and the corresponding data can be found in Section 15.6. The model assumes that the main frame of the car is stationary. The moving bodies are indexed (1), (2), and (3). A nonmoving  $x$ - $y$  frame is defined on the ground in a convenient location, and three body-fixed frames are defined with their origins at the mass centers as shown in Figure 8.1b. Points [1] through [11] are defined at the attachment points of the revolute joints, spring-damper, and a point of interest C on the wheel. The suspension spring-damper is represented by a point-to-point force element. The model requires

**FIGURE 8.1**

The double A-arm suspension in its (a) assembled and (b) representation for modeling with the body-coordinate formulation.

a nonstandard point-to-point force element, representing the radial force of the tire, to be active only when the tire is in contact with the ground.

We extract the following constant coordinates from Figure 8.1b for points [1] through [11] with respect to their corresponding frames (all in meters):

$$\mathbf{s}_1^Q = \begin{Bmatrix} -0.24 \\ 0 \end{Bmatrix}, \mathbf{s}_1^A = \begin{Bmatrix} 0.18 \\ 0 \end{Bmatrix}, \mathbf{s}_1^E = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

$$\begin{aligned}\mathbf{s}_2^A &= \begin{Bmatrix} -0.07 \\ -0.10 \end{Bmatrix}, \mathbf{s}_2^B = \begin{Bmatrix} -0.10 \\ 0.12 \end{Bmatrix}, \mathbf{s}_2^C = \begin{Bmatrix} 0.03 \\ 0 \end{Bmatrix} \\ \mathbf{s}_3^B &= \begin{Bmatrix} 0.13 \\ 0 \end{Bmatrix}, \mathbf{s}_3^O = \begin{Bmatrix} -0.13 \\ 0 \end{Bmatrix} \\ \mathbf{r}_0^O = \mathbf{s}_0^O = \mathbf{s}_0^O &= \begin{Bmatrix} 0.32 \\ 0.40 \end{Bmatrix}, \mathbf{r}_0^Q = \mathbf{s}_0^Q = \mathbf{s}_0^Q = \begin{Bmatrix} 0.20 \\ 0.26 \end{Bmatrix}, \mathbf{r}_0^F = \mathbf{s}_0^F = \mathbf{s}_0^F = \begin{Bmatrix} 0.38 \\ 0.43 \end{Bmatrix}\end{aligned}$$

To determine whether the tire is in contact with the ground, we compute a parameter  $\delta = y_2 - {}^0R$ , where  ${}^0R = 0.35$  m is the undeformed radius of the tire. If  $\delta > 0$ , the tire does not contact the ground, and therefore, no tire force is applied to body (2). But if  $\delta < 0$ , the tire force is computed as a point-to-point spring-damper, vertically attached between point C and the ground, and the force is applied on body (2).

The model assumes that at initial time, the wheel is not in contact with the ground. For the moving bodies, we must provide initial conditions on the coordinates and velocities. The initial conditions must satisfy the constraints at the coordinates and velocities. If we are not certain that the provided initial conditions are correct, we should ask the program to correct them for us. Since in this model it is assumed that at initial time, the bodies are at rest, all the velocities are simply set to zeros.

### Exercise 8.1

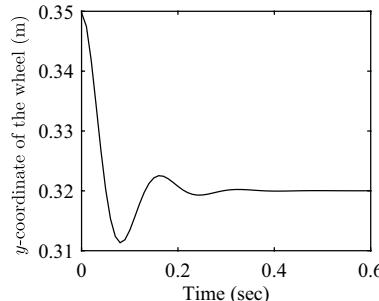
Use the following set of initial conditions for the coordinates (angles must be defined in radians).

$$\mathbf{r}_1 = \begin{Bmatrix} 0.4398 \\ 0.2512 \end{Bmatrix} \text{m}, \phi_1 = 357.9^\circ, \mathbf{r}_2 = \begin{Bmatrix} 0.6817 \\ 0.3498 \end{Bmatrix} \text{m}, \phi_2 = 4.5^\circ, \mathbf{r}_3 = \begin{Bmatrix} 0.4463 \\ 0.4308 \end{Bmatrix} \text{m}, \phi_3 = 373.7^\circ$$

Execute dap for 1.0 s and then observe the animation. Plot the  $y$ -coordinate of body (2) versus time.

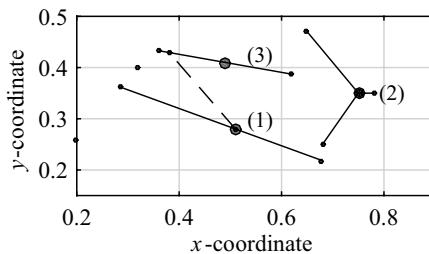
Since these coordinates satisfy the position constraints, there is no need to ask the program to correct them. However, if we ask the program to correct the coordinates, we should get the same results back.

The animation clarifies that there are no constraint violations at the initial time. The plot of the  $y$ -coordinate of body (2) versus time is shown in Figure 8.2. The plot



**FIGURE 8.2**

Vertical displacement of the wheel.

**FIGURE 8.3**

Constraints at the pin joints are violated.

reveals that the center of the wheel drops from a height of 0.35 m and settles at approximately 0.32 m.

### Exercise 8.2

Use the following set of initial values for the coordinates:

$$\mathbf{r}_1 = \begin{Bmatrix} 0.51 \\ 0.28 \end{Bmatrix} \text{m}, \phi_1 = 340^\circ, \mathbf{r}_2 = \begin{Bmatrix} 0.75 \\ 0.35 \end{Bmatrix} \text{m}, \phi_2 = 0, \mathbf{r}_3 = \begin{Bmatrix} 0.49 \\ 0.41 \end{Bmatrix} \text{m}, \phi_3 = 350^\circ$$

Execute dap without correcting the initial conditions. Set the final time to zero—this will evaluate the equations of motion for only  $t = 0$ —and then animate the results that contain only one frame. What do you observe?

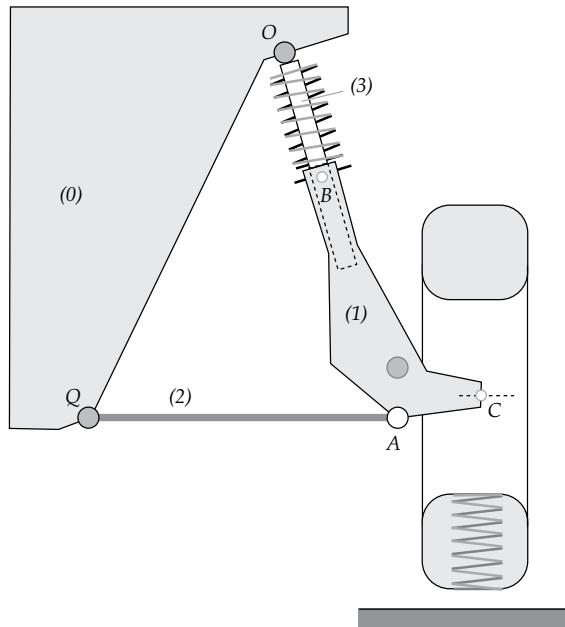
We observe that all of the constraints are violated; the bodies are separated at the pin joints as shown in Figure 8.3. We may simulate the model again, but this time we should ask the program to correct the initial conditions.

#### 8.1.2 MacPherson Suspension

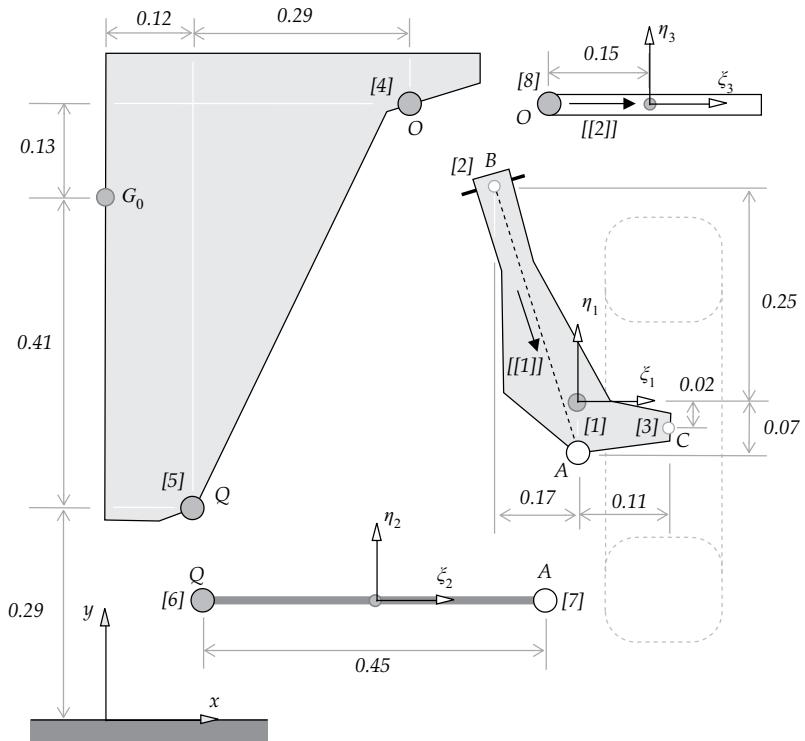
**MATLAB/Chapter 8/DAP \_ BC/Models/MP \_ A, MP \_ B, MP \_ C**

This model is the frontal view of a MacPherson suspension system as shown in Figure 8.4. Detailed description and the corresponding data for this model can be found in Section 15.7. Similar to the double A-arm suspension system model, we perform a simulation with this model from an initial state where the wheel/tire is not in contact with the ground. We construct three different models for this system demonstrating how to take advantage of composite joints for reducing the size of a model. The models contain three, two, and one moving bodies, respectively. Although these models are constructed with different number of bodies and joints, they are *kinematically identical*.

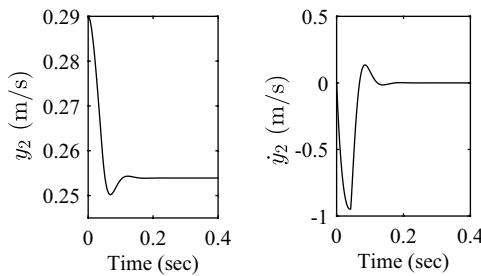
**Models/MP \_ A:** The individual bodies, the defined points, and the unit vectors of this model are shown in Figure 8.5. This model consists of three moving bodies, three pin joints, and one translational joint that connects bodies (2) and (3). This model results in nine coordinates and eight constraints. The spring-damper representing the tire is modeled similar to that in the double A-arm suspension model.



**FIGURE 8.4**  
A quarter-car MacPherson suspension system.



**FIGURE 8.5**  
The model for the quarter-car MacPherson with three bodies.



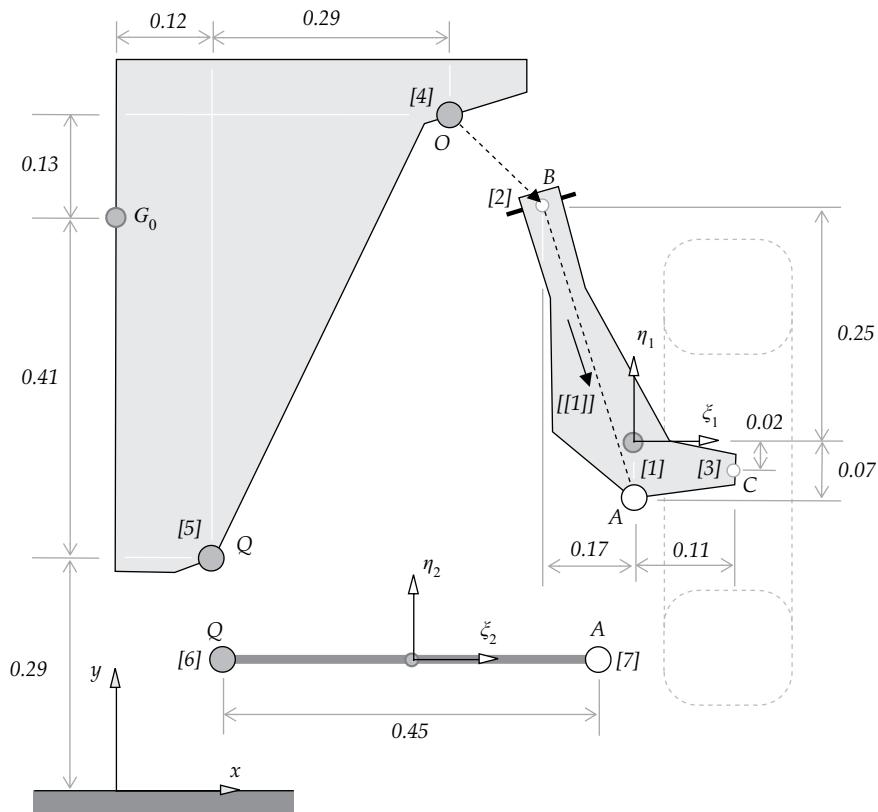
**FIGURE 8.6**  
Vertical displacement and velocity of the wheel/tire.

### Exercise 8.3

Review the input M-files for this model and simulate its dynamic response for 0.4 s. View the animation, and then plot the coordinate and velocity of body (2) (or point C) versus time.

The plotted results are shown in Figure 8.6.

**Models/MP\_B:** The individual bodies, the defined points, and the unit vector of this model are shown in Figure 8.7. This model consists of two moving bodies, two pin joints,



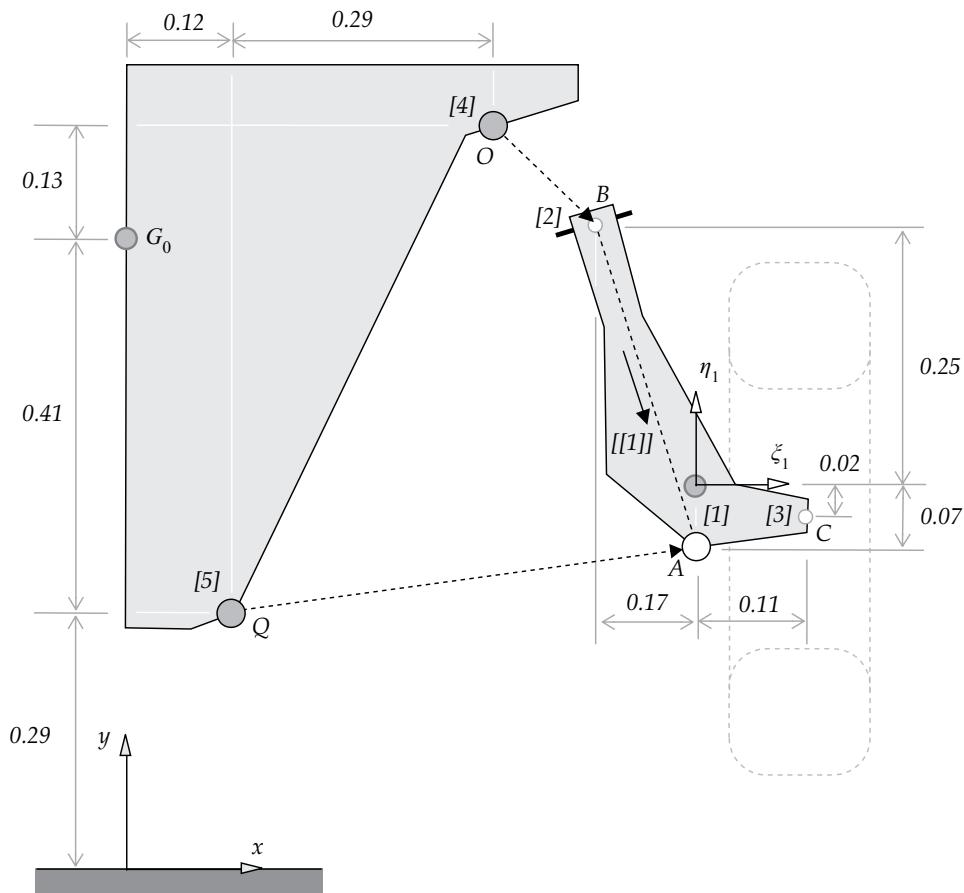
**FIGURE 8.7**  
The model for the quarter-car MacPherson with two bodies.

and one revolute-translational joint, where body (3) from the first model (MP \_ A) has been combined as a massless link with the pin joint  $O$  and the translational joint to form a revolute-translational joint. This model yields six coordinates and five constraints. The program constructs a vector between points  $O$  and  $B$ , and another vector between  $B$  and  $A$ , and then it enforces the two vectors to stay parallel.

#### Exercise 8.4

Review the input M-files for this model and simulate its dynamic response for 0.4 s. View the animation, and then plot the coordinate and velocity of body (2) (or point  $C$ ) versus time. The plotted results should be practically identical to those shown in Figure 8.6.

**Models/MP \_ C:** This model consists of one moving body, one revolute-revolute joint, and one revolute-translational joint as shown in Figure 8.8. Body (2) from the previous models (MP \_ B) has been combined with the pin joints  $Q$  and  $A$  as a massless link to form a revolute-revolute joint. This model results in three coordinates and two constraints. The



**FIGURE 8.8**

The model for the quarter-car MacPherson with one moving body.

program constructs a vector between points  $Q$  and  $A$  and enforces the length of the vector to remain a constant (0.45 m).

### Exercise 8.5

Review the input M-files for this model and simulate its dynamic response for 0.4 seconds. View the animation, and then plot the coordinate and velocity of body (1) (or point C) versus time. The plotted results should be practically identical to those shown in Figure 8.6.

#### 8.1.3 Cart

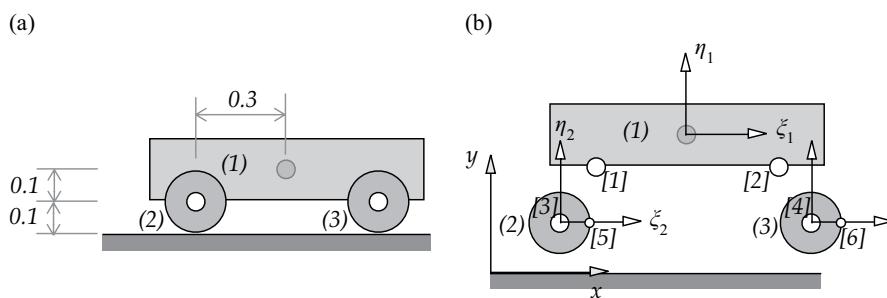
**MATLAB/Chapter 8/DAP\_BC/Models/Cart\_A, Cart\_B, Cart\_C, Cart\_D**

This model, as shown in Figure 8.9a, represents a cart containing a main body and two wheels that are connected to the main body by revolute joints. The wheels are modeled as discs rolling on the flat ground without slipping. A driver motor rotates one of the wheels causing the cart to move forward. The individual bodies, the reference frames, and the defined points are shown in Figure 8.9b. The main objective of this example is to show how to represent a motor either as a driver constraint or by a torque-speed function.

**Models/Cart \_ A:** In this model, it is assumed that the rear wheel, body (2), rolls with a constant angular velocity of  $2\pi$  rad/s in the clockwise direction. To enforce this condition, we employ a *relative rotation* constraint as the driver, which refers to function "a" for its analytical description and parameters. This function is based on the driver described in Eq. (7.43) as

$${}^{(dr-r,1)}\Phi = \phi_i - \phi_j - f(t) = 0$$

In our model, the body indices ( $i$ ) and ( $j$ ) are (2) and (0), respectively, and the driver function  $f(t) = -2\pi t$  assumes that at  $t = 0$ ,  $\phi_i = 0$ . Note that since the angular velocity of the wheel is  $-2\pi$  rad/s at  $t = 0$ , we need to provide correct initial velocities for all the three bodies, or ask the program to correct them for us.



**FIGURE 8.9**

(a) A wheeled cart; (b) the corresponding mutibody representation.

### Exercise 8.6

Review the input M-files for this model and note that the initial velocities are all zeros. Execute dap and ask the program to correct the initial conditions. Then execute anim and observe the animation of the response.

When you ask the program to correct the initial conditions, you receive the corrected values as follows:

```
Corrected coordinates
x           y           phi
0.5         0.2         0
0.2         0.1         0
0.8         0.1         0
Corrected velocities
x-dot       y-dot       phi-dot
0.62832     0           0
0.62832     0           -6.2832
0.62832     0           -6.2832
```

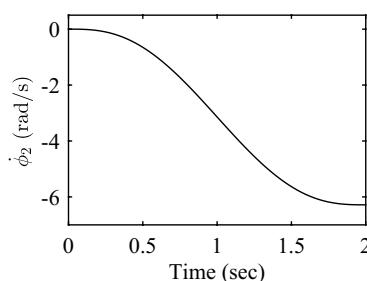
We note that the coordinate values have not been changed, but the velocities have.

### Exercise 8.7

Execute dap but do not ask the program to correct the initial conditions; that is, keep all initial velocities at zero. Perform an animation of the response. What do you observe? Why?

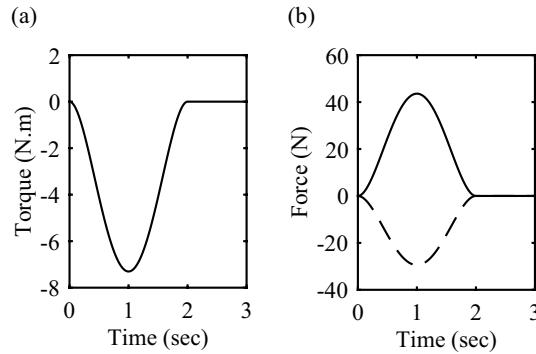
We observe that nothing moves, even though we have a driver stating that the angular velocity of body (2) is  $-2\pi$  rad/s. The reason is that when we solve the equations of motion, we only enforce the second time derivative of the constraints as in Eq. (7.52), including the driver constraint. The second derivative of our driver constraint enforces  $\ddot{\phi}_2 = 0$ . So, if we start the simulation with zero angular velocity, the driver acceleration  $\dot{\phi}_2 = 0$  will keep the velocities unchanged at zeros.

**Models/Cart \_ B:** This model is the same as the model Cart \_ A except for the driver function referring to *function "c"* instead of "a". Function "c" allows the angular velocity of body (2) to start from zero and reach the desired speed of  $-2\pi$  rad/s in a specified period of time, which is set to 2.0 s in our model. The general description of this function at the velocity level is shown in Figure 8.10. With this type of driver, all the velocities can be set to zeros at the initial time.



**FIGURE 8.10**

First time derivative of function "c".

**FIGURE 8.11**

(a) Reaction torque associated with the driver constraint; (b) no-slip reaction forces associated with the rear driver wheel (solid line) and the front driven wheel (dashed line).

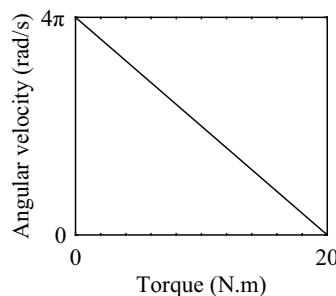
### Exercise 8.8

Review the input M-files for this model. Execute `dap`, and then execute `anim` and observe the animation of the response.

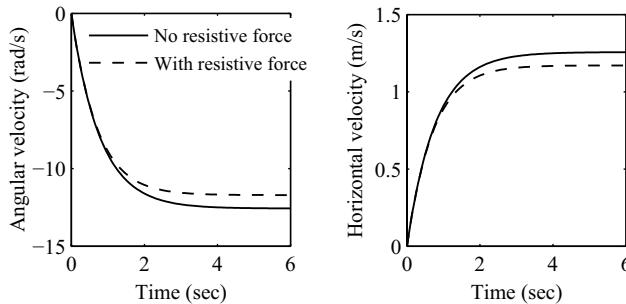
Execute `post` and plot the angular velocity of body (2) versus time. The plot should reveal that the angular velocity has been varied from zero to the final value in the described time period as depicted in Figure 8.10.

We may also plot the Lagrange multiplier associated with some of the constraints. The multiplier associated with the driver constraint is shown in Figure 8.11a, which is a torque. The reaction forces associated with the no-slip constraints of the rear and front wheels are shown in Figure 8.11b. We note that the reaction force on the rear wheel is positive since this is the driver wheel, but for the driven wheel at the front, it is negative.

**Models/Cart \_ C:** This is a variation of the previous cart models, where an applied torque has replaced the driver constraint on the rear wheel. It is assumed that the torque is generated by an electric motor with a torque–speed characteristic shown in Figure 8.12 (refer to Section 4.5.2). This motor can provide a maximum torque of 20 N m at zero angular velocity and can reach a maximum angular velocity of  $4\pi$  rad/s at zero load.

**FIGURE 8.12**

Torque–speed characteristic of a permanent magnet motor.

**FIGURE 8.13**

(a) Angular velocity of a wheel and (b) the linear velocity of the Cart.

**Exercise 8.9**

Simulate the dynamic response of this model for 6.0 s. Plot the angular velocity of the rear wheel and the velocity of the main body in the  $x$ -direction versus time.

The two plots are shown in Figure 8.13. We observe that the Cart is initially at rest, but after approximately 3 s, it reaches a constant speed—the wheel reaches the specified angular velocity of  $-4\pi$  rad/s. At this speed, the motor does not provide a torque since there is no resistive load on the vehicle.

**Models/Cart \_ D:** In this model, we add a resistive force to the model of Cart \_ C. Assume that the forward motion of this vehicle is opposed by aerodynamic forces that act on body (1). The aerodynamic force can be represented by a simple model as  $(^{(aero)}f = cv^2)$ , where  $c$  is a damping coefficient as a function of drag and surface area, and  $v$  is the speed of the vehicle.

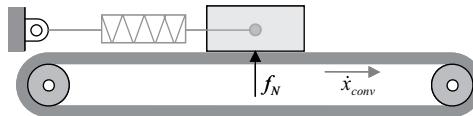
**Exercise 8.10**

Simulate the response of the model with the resistive force for 6.0 s. Plot the angular velocity of the rear wheel and the velocity of the main body in the  $x$ -direction versus time, and compare the results against those obtained without resistive force.

The results are shown in Figure 8.13 (dashed lines), superimposed on the results with no resistive force. We observe a reduction in the final speed of the Cart.

**8.1.4 Conveyor Belt and Friction****MATLAB/Chapter 8/DAP \_ BC/Models/CB**

This simple example is provided to show how to use a friction model as discussed in Section 4.5. The spring-mass system shown in Figure 8.14 is a typical example that has been used in literature for testing and comparing different friction formulas. The block rests on a conveyor belt that moves with a constant speed. As the block moves with the belt, it stretches the spring until the force of the spring overcomes the friction force and pulls the block back, and the process is repeated. For this system, let us consider the friction model of Eq. (4.43) and the following data:

**FIGURE 8.14**

A spring-mass system moving with a conveyor belt.

$$m = 1 \text{ kg}, k = 10 \text{ N/m}, {}^0L = 0.5 \text{ m}, g = 9.81 \text{ m/s}^2, \dot{x}_{conv} = 0.1 \text{ m/s}$$

$$\mu_s = 0.2, \mu_d = 0.15, \mu_v = 0, k_t = 10,000, v_s = 0.001 \text{ m/s}$$

In this model, it is also assumed that the velocity of the block at the initial time is zero. We note that the normal reaction force on the block is a constant and equal to the weight of the block.

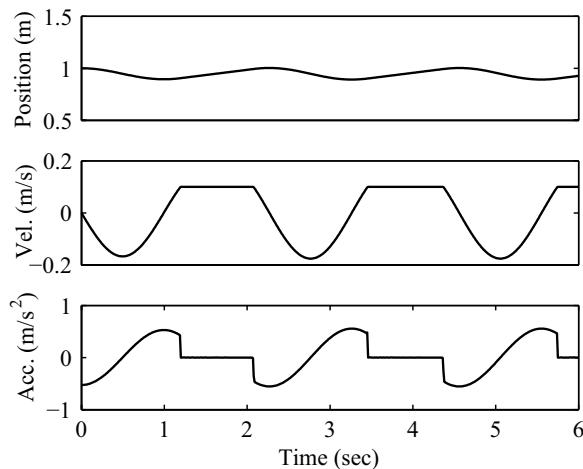
### Exercise 8.11

Simulate the response of the model for 6.0 s. Plot the coordinate, velocity, and acceleration of the block in the  $x$ -direction versus time.

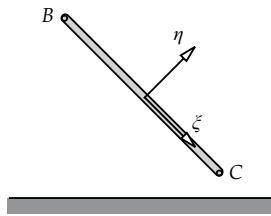
The results are shown in Figure 8.15. When we execute `dap`, we should notice that the simulation takes much longer time than most other simulations. This is due to the phase when the block is in stiction; that is, it is moving practically with the same speed as the belt. In this phase, the integration algorithm takes very small time steps.

### Exercise 8.12

Repeat Exercise 8.11, but this time use the friction model of Eq. (4.44). Use the function M-file `Friction_B` instead of `Friction_A`. Assume  $v_t = 0.001 \text{ m/s}$ . Compare the results against those from Exercise 8.11.

**FIGURE 8.15**

Position, velocity, and acceleration of the block in the  $x$ -direction.



**FIGURE 8.16**  
A rod moving vertically to impact the ground.

### 8.1.5 Rod Impacting Ground

**MATLAB/Chapter 8/DAP \_ BC/Models/Rod**

This simple example demonstrates the use of the continuous contact/impact model as described in Section 11.2, Eq. (11.42). The system is a slender rod that moves vertically toward the ground as shown in Figure 8.16. The rod has a length of 2.0 m, a mass of 1.0 kg, and a moment of inertia of  $0.01 \text{ kg m}^2$ . The rod is oriented at  $45^\circ$  angle, its mass center is 1.0 m off the ground, and its velocity at this orientation is 6.0 m/s downward in the vertical direction with zero angular velocity. For the contact model, the following parameters are used:

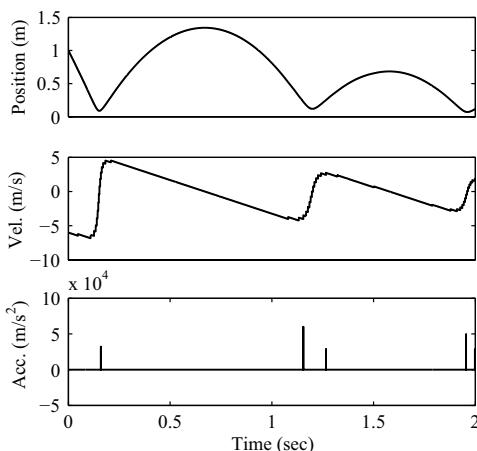
$$k = 10^{11} \text{ N/m}^{-0.5} \text{ and } e = 0.95$$

In this model, the  $y$ -coordinates of points B and C are monitored. As soon as either point penetrates the ground, the impact model determines a normal force and applies it to that point.

#### Exercise 8.13

Simulate the response of the model for 2.0 s. Plot the coordinate, velocity, and acceleration of the mass center of the rod in the  $y$ -direction versus time.

The results are shown in Figure 8.17. The acceleration plot clearly shows that as point B or C impacts the ground, a sudden jump in the vertical acceleration occurs. This causes



**FIGURE 8.17**  
Vertical position, velocity, and acceleration of the mass center of the rod versus time.

a high-frequency oscillatory velocity response. This impact does not affect the lateral position of the rod since no friction at the contact points is considered in this model.

Simulation of the example in Exercise 8.13 shows that the integration algorithm takes very small time steps the moment the rod contacts the ground, resulting in a large computation time. This is due to the sudden application of a large contact force that causes a sudden jump in the acceleration of the rod.

If we simulate the problem of Exercise 8.13 for a longer period of time, to allow the rod to come to rest on the ground, the integration time steps will remain very small. This phenomenon is one of the drawbacks of this type of contact models. Such models represent a contact reasonably well when the relative velocity of the contact points is not too small. When the bodies begin to remain in contact with each other, a simpler linear spring-damper contact model may be more suitable to replace the original contact model.

---

## 8.2 Problems

- 8.1 In the models for the double A-arm and MacPherson quarter-car suspension systems, we used a nonstandard point-to-point force element to determine the radial force of the tire only when the tire is in contact with the ground. Develop such a unilateral spring-damper as a standard force element in the program.
- 8.2 In some simulations, we may want to know the amount of time it takes MATLAB to perform certain computation. To find out how much time the program spends on integrating the equations of motion, we can include the following statements in the M-file `dap` (or in other programs) before and after invoking the integrator `ode45`:

```
% Start simulation time clock
t0_clock = clock;
% Integrate
[T, zT] = ode45(@BC_eqsmo, Tspan, z);
% Show elapsed simulation time
elapsed_time = etime(clock,t0_clock);
disp(['Runtime (seconds): ', num2str(elapsed_time)])
```

- 8.3 After completing a simulation, we may decide to continue with the simulation from the point that the first simulation ended. In other words, results from the last step of one simulation could be used as the initial conditions for another simulation. Provide this capability to the program.
- 8.4 Revise the program to include the constraint violation stabilization terms, as an option, in the equations of motion (refer to Section 13.3.1).
- 8.5 Develop a version of the program (or include this revision as an option in the program) to solve the equations of motion with the penalty method from Section 13.3.3.
- 8.6 Develop a version of the program to solve the equations of motion based on the momentum method from Section 13.3.4.

- 8.7 Revise the program by adding the capability to perform static equilibrium analysis with the fictitious damping method as in Eq. (14.5). This feature can be included in the program as another type of force element.
- 8.8 Revise the program to add the capability of performing static equilibrium analysis by solving the equilibrium equations of Eq. (14.4). The set of algebraic equilibrium equations can be solved by MATLAB's `fso1ve` function.
- 8.9 The initial condition correction module in the program adjusts all of the coordinates and velocities in order to satisfy the constraints according to the formulas in Eqs. (14.7) and (14.9). In some simulations, we may want one or more of the coordinates (and the corresponding velocities) to keep their stated initial condition. Add such capability to the M-file `ic_correct.m`. The program should ask the user for the coordinate(s) that should remain unchanged. The program should then add, for each specified coordinate, the necessary simple constraint and its corresponding row in the Jacobian matrix before implementing Eqs. (14.7) and (14.9). After the initial conditions are corrected, the temporary constraint(s) should be discarded.

# 9

---

## *Joint-Coordinate Formulation*

---

The body-coordinate formulation from Chapter 7 provides a simple method to systematically formulate the equations of motion for a multibody system. The formulation is suitable for implementation in a computer program but not for manual calculations. One negative feature of the body-coordinate formulation is that it generates a large number of equations even for a simple system. In contrast to the body-coordinate formulation, we learned in Chapter 5 that the vectorial formulation constructs a small set of equations representing the kinematics of a multibody system. However, the shortcomings of the vectorial formulation are that (a) we need to derive the necessary equations, matrices, and arrays analytically, which are not suitable for a systematic process; and (b) the method, in its presented form, is only suitable for kinematic analysis, and not for forward dynamics (Chapter 13).

In this chapter, we discuss the method of joint-coordinate formulation that has most of the advantages of both the body-coordinate and vectorial formulations. The joint-coordinate method provides a *systematic* process that transforms the large set of equations of motion from the body-coordinate formulation to a smaller or possibly a minimal set of equations. Fewer number of equations results in a more efficient analysis as far as the computational time is concerned. We will find certain similarities between the kinematic equations obtained from this method and those obtained from the vectorial method.

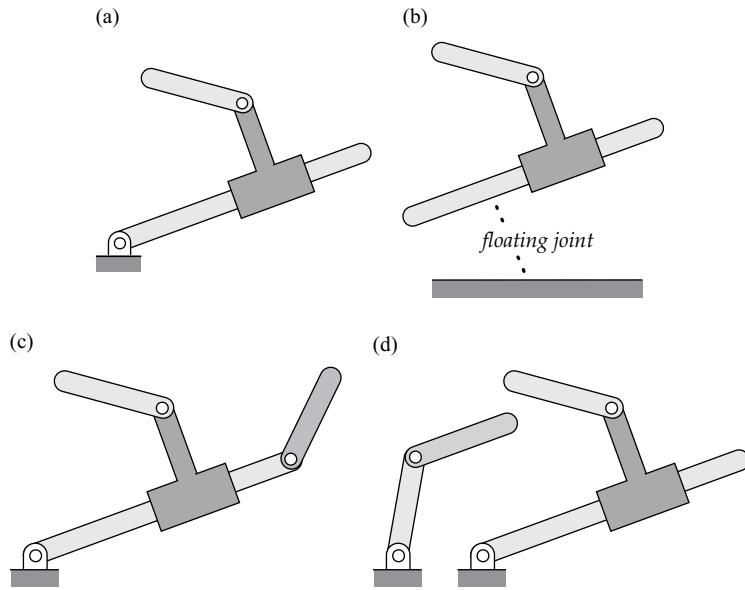
The joint-coordinate method is first presented for open-chain systems before extending it to closed-chain systems. A MATLAB® program based on this method will be discussed at the conclusion of this chapter.

---

### 9.1 Definitions

The interconnectivity of bodies of a multibody system via kinematic joints is called the *structure* (or the *topology*) of that system. The *structure* of an open-chain system is analogous to a tree containing a root, branches, and leaves. The generic system shown in Figure 9.1a consists of three links, three joints, and the ground. The ground is always the *root* (or the *base*). This system contains only one *branch*. If we move from the root through the branch (i.e., move from a joint to a link and then to another joint and so on), we will end up at a *leaf*—the last link in a branch.

If a system is not connected to the ground by a kinematic joint, such as the system in Figure 9.1b, it is referred to as a *floating system*. For such a system, we still consider the ground as the root, but we assume an imaginary joint connects the ground to one of the bodies. This imaginary joint is called a *floating joint* that allows three degrees of freedom (DoFs) between the ground and the body; that is, this joint does not eliminate any DoF from the system.

**FIGURE 9.1**

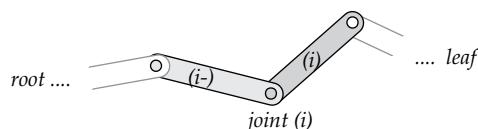
Various topologies of open-chain systems: (a) a grounded system; (b) a floating system; (c) a system with two branches; (d) system with two trees.

A system may contain more than one branch. For example, the system shown in Figure 9.1c contains two branches. Each branch ends at a leaf; therefore, there are as many leaves in a system as the number of branches. A system may contain more than one tree, such as the system shown in Figure 9.1d.

In a topological view of a tree, let us consider two connected bodies in a branch as shown in Figure 9.2. The body that is closer to the leaf of that branch is referred to as body  $(i)$ , and the body closer to the root is referred to as body  $(i-)$ . If we move through the branch from the root toward a leaf, we enter body  $(i-)$  before getting to body  $(i)$ . The joint that connects these two bodies will be given the same index as body  $(i)$ . We say that this joint belongs to body  $(i)$  and not to body  $(i-)$ . Since in an open-chain system, there are as many joints as the number of bodies (this includes the floating joint(s) in a system), each body owns only one joint.

### 9.1.1 Joint Coordinate and Joint Reference Point

A joint coordinate, as its name suggests, describes the DoF that a joint allows between two bodies. Therefore, the joint coordinate for a revolute joint must be an angle and that for

**FIGURE 9.2**

The joint connecting bodies  $(i)$  and  $(i-)$  belongs to body  $(i)$ .

a translational joint must be a length. Defining joint coordinates between bodies is very similar to that of the vectorial formulation in Chapter 5.

In this section, we look at the definition of joint coordinates for several commonly used planar joints and derive the corresponding recursive kinematic formulas. Here the term *recursive* means that if we know the coordinates of body  $(i-)$  and the joint coordinate  $(i)$ , we should be able to determine the coordinates of body  $(i)$ . We first derive the necessary recursive formulas for coordinate transformation, and then develop the formulas for the velocity transformation.

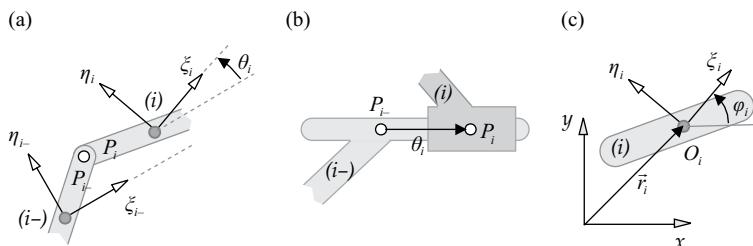
The joint coordinate for the *revolute* joint  $(i)$  shown in Figure 9.3a is defined as the relative angle between bodies  $(i)$  and  $(i-)$  denoted as  $\theta_i$ . This angle can be measured between any two axes on the bodies. However, it would be much simpler to have the angle between the  $\xi$ -axes of the two body-fixed frames to represent  $\theta_i$ . Point  $P_i$ , which is the center of the pin joint on body  $(i)$ , will be referred to as the *reference point* for this joint.\*

For a *sliding* joint, the joint coordinate  $\theta_i$  is defined as the relative distance between two points on bodies  $(i)$  and  $(i-)$  along the joint axis, as shown in Figure 9.3b. The reference point for this joint is point  $P_i$ .

For a floating joint between a body and the ground, three joint coordinates must be defined since this is a fictitious joint that does not eliminate any of the body's DoF. These joint coordinates can be the same as the coordinates of body  $(i)$ ;  $\theta_i = \{x_i \quad y_i \quad \phi_i\}'$  as shown in Figure 9.3c. The reference point for this joint is the origin of the body  $(i)$ , that is,  $O_i$ .

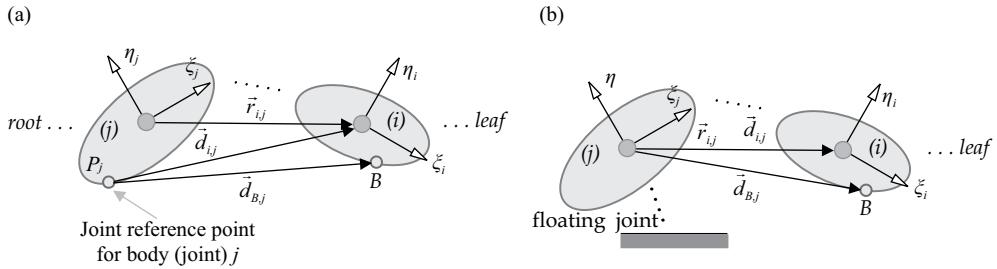
In Chapter 7, we discussed two composite joints: revolute–revolute and revolute–translational. The main purpose for introducing these joints in the body-coordinate formulation was to reduce the number of coordinates and constraints in a model. Introducing these composite joints in the joint-coordinate formulation will not result in much (or any) reduction in the number of equations or coordinates. Therefore, we do not discuss these composite joints in our joint-coordinate recursive formulation. However, a composite joint can be considered as a *cut-joint* in a closed-chain system.

Three typical position vectors that will appear in the upcoming kinematic formulas are depicted in Figure 9.4a. Vector  $\vec{r}_{i,j}$  connects the mass center of body  $(j)$  to the mass center of body  $(i)$ . This vector can simply be evaluated as



**FIGURE 9.3**  
Joint coordinates for (a) a pin joint, (b) a sliding joint, and (c) a floating joint.

\* At this stage, it may not be clear why we need to define a *reference point* for a joint. The use of the joint reference points should become clear when we formulate the Jacobian matrix for a cut-joint.

**FIGURE 9.4**

Three useful position vectors for (a) a general case and (b) a floating joint.

$$\begin{aligned}\phi_1 &= \theta_1 \\ \mathbf{r}_1 &= -\mathbf{s}_1^A \\ \phi_2 &= \phi_1 \\ \mathbf{r}_2 &= \mathbf{r}_1 + \mathbf{u}_1 \theta_2\end{aligned}$$

Another vector definition is vector  $\vec{d}_{i,j}$  that connects the joint reference point of body (or joint)  $(j)$  to the mass center of body  $(i)$ . This vector is simply evaluated as

$$\mathbf{d}_{i,j} = \mathbf{r}_i - \mathbf{r}_j^P$$

A third type of vector,  $\vec{d}_{B,j}$ , connects the joint reference point  $(j)$  to a typical point  $B$  on body  $(i)$ . We should note that if body  $(j)$  is the owner of a floating joint, vectors  $\vec{r}_{i,j}$  and  $\vec{d}_{i,j}$  become the same, as shown in Figure 9.4b, since the reference point for a floating joint is at the mass center of its body.

### 9.1.2 Recursive Kinematics

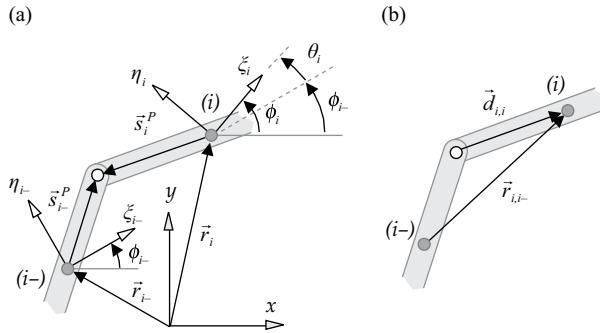
In this section, we derive recursive formulas to compute the coordinates and velocity of body  $(i)$ , assuming that the coordinates and velocity of body  $(i-)$  and those for joint  $(i)$  are known. The formulas are derived for revolute and translational joints.

**Revolute joint:** For a revolute joint, as shown in Figure 9.5a, the rotational and translational coordinates of body  $(i)$  can be expressed as

$$\begin{aligned}\phi_i &= \phi_{i-} + \theta_i \\ \mathbf{r}_i &= \mathbf{r}_{i-} + \mathbf{s}_{i-}^P - \mathbf{s}_i^P\end{aligned}\tag{9.1}$$

We call this a recursive formula because the coordinates of body  $(i)$  can be determined following several systematic steps as follows:

- Compute the  $x-y$  components of vector  $\vec{s}_{i-}^P$ .  $\mathbf{A}_{i-} \mathbf{s}_{i-}^P \Rightarrow \mathbf{s}_{i-}^P$
- Determine the rotational coordinate of body  $(i)$ .  $\phi_{i-} + \theta_i \Rightarrow \phi_i$
- Determine the rotational transformation matrix of body  $(i)$ .  $\phi_i \Rightarrow \mathbf{A}_i$
- Compute the  $x-y$  components of vector  $\vec{s}_i^P$ .  $\mathbf{A}_i \mathbf{s}_i^P \Rightarrow \mathbf{s}_i^P$
- Compute the  $x-y$  coordinates of body  $(i)$ .  $\mathbf{r}_{i-} + \mathbf{s}_{i-}^P - \mathbf{s}_i^P \Rightarrow \mathbf{r}_i$

**FIGURE 9.5**

(a) Graphical description of the joint coordinate for a revolute joint and (b) the required vectors for the recursive kinematics.

We use the time derivative of Eq. (9.1) to determine recursive formulas for evaluating the velocity of body (i):

$$\begin{aligned}\dot{\phi}_i &= \dot{\phi}_{i-} + \dot{\theta}_i \\ \dot{\mathbf{r}}_i &= \dot{\mathbf{r}}_{i-} + \bar{\mathbf{s}}_{i-}^P \dot{\phi}_{i-} - \bar{\mathbf{s}}_i^P \dot{\theta}_i = \dot{\mathbf{r}}_{i-} + (\bar{\mathbf{s}}_{i-}^P - \bar{\mathbf{s}}_i^P) \dot{\phi}_{i-} - \bar{\mathbf{s}}_i^P \dot{\theta}_i\end{aligned}$$

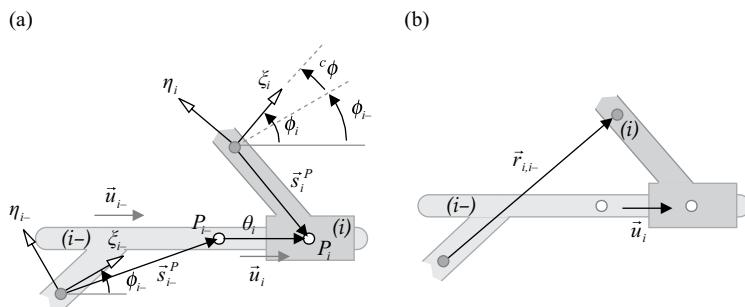
In matrix form, these equations can be expressed as

$$\left\{ \begin{array}{l} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \bar{\mathbf{r}}_{i,i-} \\ \mathbf{0} & 1 \end{array} \right] \left\{ \begin{array}{l} \dot{\mathbf{r}}_{i-} \\ \dot{\phi}_{i-} \end{array} \right\} + \left[ \begin{array}{c} \bar{\mathbf{d}}_{i,i} \\ 1 \end{array} \right] \dot{\theta}_i \quad (9.2)$$

where  $\mathbf{r}_{i,i-} = \mathbf{s}_{i-}^P - \mathbf{s}_i^P$  and  $\mathbf{d}_{i,i} = -\mathbf{s}_i^P$  as shown in Figure 9.5b.

**Translational joint:** For a translational joint, as shown in Figure 9.6a, the coordinate formulas can be written as

$$\begin{aligned}\phi_i &= \phi_{i-} + {}^c\phi \\ \mathbf{r}_i &= \mathbf{r}_{i-} + \mathbf{s}_{i-}^P + \theta_i \mathbf{u}_{i-} - \mathbf{s}_i^P\end{aligned} \quad (9.3)$$

**FIGURE 9.6**

(a) Graphical description of the joint coordinate for a translational joint and (b) the required vectors for the recursive kinematics.

where  ${}^c\phi$  is a constant angle. The recursive steps for determining the coordinates of body ( $i$ ) are as follows:

- Compute the  $x$ - $y$  components of vector  $\bar{\mathbf{s}}_{i-}^P$ .  $\mathbf{A}_{i-}\mathbf{s}_{i-}^P \Rightarrow \mathbf{s}_{i-}^P$
- Determine the rotational coordinate of body ( $i$ ).  $\phi_{i-} + {}^c\phi \Rightarrow \phi_i$
- Determine the rotational transformation matrix of body ( $i$ ).  $\phi_i \Rightarrow \mathbf{A}_i$
- Compute the  $x$ - $y$  components of vector  $\bar{\mathbf{s}}_i^P$ .  $\mathbf{A}_i\mathbf{s}_i^P \Rightarrow \mathbf{s}_i^P$
- Compute the  $x$ - $y$  components of vector  $\bar{\mathbf{u}}_i = \bar{\mathbf{u}}_{i-}$ .  $\mathbf{A}_{i-}\bar{\mathbf{u}}_{i-} \Rightarrow \mathbf{u}_{i-}$  or  $\mathbf{A}_i\bar{\mathbf{u}}_i \Rightarrow \mathbf{u}_i$
- Compute the  $x$ - $y$  coordinates of body ( $i$ ).  $\mathbf{r}_{i-} + \mathbf{s}_{i-}^P + \theta_i \mathbf{u}_i - \mathbf{s}_i^P \Rightarrow \mathbf{r}_i$

The velocity of body ( $i$ ) is determined as

$$\begin{aligned}\dot{\phi}_i &= \dot{\phi}_{i-} \\ \dot{\mathbf{r}}_i &= \dot{\mathbf{r}}_{i-} + \bar{\mathbf{s}}_{i-}^P \dot{\phi}_{i-} + \theta_i \bar{\mathbf{u}}_i \dot{\phi}_i + \mathbf{u}_i \dot{\theta}_i - \bar{\mathbf{s}}_i^P \dot{\phi}_i = \dot{\mathbf{r}}_{i-} + (\bar{\mathbf{s}}_{i-}^P + \theta_i \bar{\mathbf{u}}_i - \bar{\mathbf{s}}_i^P) \dot{\phi}_{i-} + \mathbf{u}_i \dot{\theta}_i\end{aligned}$$

Or, in matrix form, we have

$$\left\{ \begin{array}{l} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \bar{\mathbf{r}}_{i,i-} \\ \mathbf{0} & 1 \end{array} \right] \left\{ \begin{array}{l} \dot{\mathbf{r}}_{i-} \\ \dot{\phi}_{i-} \end{array} \right\} + \left[ \begin{array}{l} \mathbf{u}_i \\ 0 \end{array} \right] \dot{\theta}_i \quad (9.4)$$

where  $\mathbf{r}_{i,i-} = \mathbf{s}_{i-}^P + \theta_i \mathbf{u}_i - \mathbf{s}_i^P$  as depicted in Figure 9.6b.

The recursive formulas for the two fundamental joints show that to determine the coordinates of body ( $i$ ), we need to systematically perform several computational steps. The coordinate formulas cannot be put in matrix form because the equations are nonlinear. In contrast, the velocity equations are linear in the velocities, and therefore, they can be expressed in matrix form. There are certain similarities between the velocity formulas of the two types of joints. The coefficient matrix for the velocity of body ( $i$ ) is the same for both joints. The difference is in the coefficient matrix for the joint velocity.

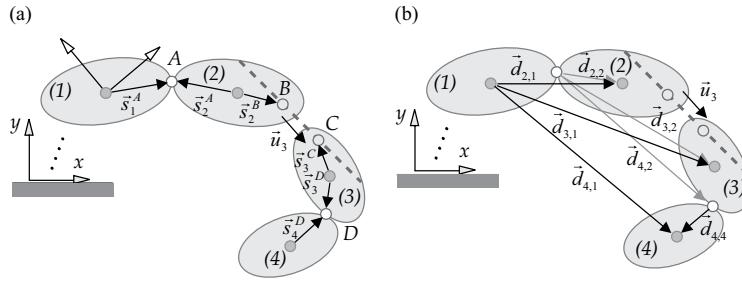
## 9.2 Open-Chain Systems

The recursive formulas developed in Section 9.1 can now be applied to a multibody system. In this section, we consider only open chains. We demonstrate the general procedure through several examples.

Let us consider the schematic presentation of a multibody system shown in Figure 9.7a. There is a revolute joint between bodies (1) and (2), a translational joint between bodies (2) and (3), and another revolute joint between bodies (3) and (4). With the defined body-fixed frames, it is assumed that the following constant data are known:

$$\mathbf{s}_1^A, \mathbf{s}_2^A, \mathbf{s}_2^B, \mathbf{s}_3^C, \mathbf{s}_3^D, \mathbf{s}_4^D, \mathbf{u}_3 \text{ (or } \mathbf{u}_2\text{)}, {}^c\phi_{3,2}$$

The process begins with moving from the ground (the root) through a branch, toward a leaf. Since this is a floating system, we consider a floating joint between body (1) and the ground. We define the following joint coordinates:  $\mathbf{r}_1$  and  $\phi_1$  for the floating joint,  $\theta_2$  for the revolute joint at  $A$ ,  $\theta_3$  for the translational joint, and  $\theta_4$  for the revolute joint at  $D$ . With

**FIGURE 9.7**

(a) Graphical description of a floating system and (b) the required vectors for the recursive kinematics.

these joint coordinates, the recursive process for computing the body coordinates can be constructed as follows (refer to Eqs. (9.1) and (9.3)):

- Knowing  $\mathbf{r}_1$  and  $\phi_1$ , compute  $\phi_1 \Rightarrow \mathbf{A}_1; \mathbf{A}_1 \mathbf{s}_1^A \Rightarrow \mathbf{s}_1^A$ .
- Knowing  $\theta_2$ , compute  $\phi_1 + \theta_2 \Rightarrow \phi_2 \Rightarrow \mathbf{A}_2; \mathbf{A}_2 \mathbf{s}_2^A \Rightarrow \mathbf{s}_2^A$ ;  
 $\mathbf{r}_1 + \mathbf{s}_1^P - \mathbf{s}_2^P \Rightarrow; \mathbf{A}_2 \mathbf{s}_2^B \Rightarrow \mathbf{s}_2^B$ .
- Knowing  $\theta_3$ , compute  $\phi_2 + {}^c\phi_{3,2} \Rightarrow \phi_3 \Rightarrow \mathbf{A}_3; \mathbf{A}_3 \mathbf{s}_3^C \Rightarrow \mathbf{s}_3^C; \mathbf{A}_3 \mathbf{u}_3 \Rightarrow \mathbf{u}_3$ ;  
 $\mathbf{r}_1 + \mathbf{s}_1^B + \theta_3 \mathbf{u}_3 - \mathbf{s}_1^C \Rightarrow \mathbf{r}_3; \mathbf{A}_3 \mathbf{s}_3^D \Rightarrow \mathbf{s}_3^D$
- Knowing  $\theta_4$ , compute  $\phi_3 + \theta_4 \Rightarrow \phi_4 \Rightarrow \mathbf{A}_4; \mathbf{A}_4 \mathbf{s}_4^D \Rightarrow \mathbf{s}_4^D; \mathbf{r}_3 + \mathbf{s}_3^D - \mathbf{s}_4^D \Rightarrow \mathbf{r}_4$ .

At this point, all the coordinates have been computed, and in the process, the  $x$ - $y$  components of all position vectors between the bodies are also computed.

By referring to Eqs. (9.2) and (9.4), the velocity equations are constructed as follows:

$$\begin{aligned}
 \dot{\phi}_1 &= \dot{\phi}_1 \\
 \dot{\mathbf{r}}_1 &= \dot{\mathbf{r}}_1 \\
 \dot{\phi}_2 &= \dot{\phi}_1 + \dot{\theta}_2 \\
 \dot{\mathbf{r}}_2 &= \dot{\mathbf{r}}_1 + \check{\mathbf{r}}_{2,1}\dot{\phi}_1 + \check{\mathbf{d}}_{2,2}\dot{\theta}_2 \\
 \dot{\phi}_3 &= \dot{\phi}_2 \\
 \dot{\mathbf{r}}_3 &= \dot{\mathbf{r}}_2 + \check{\mathbf{r}}_{3,2}\dot{\phi}_2 + \mathbf{u}_3\dot{\theta}_3 \\
 \dot{\phi}_4 &= \dot{\phi}_3 + \dot{\theta}_4 \\
 \dot{\mathbf{r}}_4 &= \dot{\mathbf{r}}_3 + \check{\mathbf{r}}_{4,3}\dot{\phi}_3 + \check{\mathbf{d}}_{4,4}\dot{\theta}_4
 \end{aligned} \tag{9.5}$$

A forward substitution process transforms these equations to the following form, where, on the right-hand side, all the body velocities are expressed in terms of the joint velocities (note that for body (1), the body velocities and the joint velocities are the same):

$$\begin{aligned}
 \dot{\phi}_1 &= \dot{\phi}_1 \\
 \dot{\mathbf{r}}_1 &= \dot{\mathbf{r}}_1 \\
 \dot{\phi}_2 &= \dot{\phi}_1 + \dot{\theta}_2 \\
 \dot{\mathbf{r}}_2 &= \dot{\mathbf{r}}_1 + \check{\mathbf{r}}_{2,1}\dot{\phi}_1 + \check{\mathbf{d}}_{2,2}\dot{\theta}_2 \\
 \dot{\phi}_3 &= \dot{\phi}_1 + \dot{\theta}_2
 \end{aligned}$$

$$\begin{aligned}\dot{\mathbf{r}}_3 &= \dot{\mathbf{r}}_1 + (\check{\mathbf{r}}_{2,1} + \check{\mathbf{r}}_{3,2})\dot{\phi}_1 + (\check{\mathbf{d}}_{2,2} + \check{\mathbf{r}}_{3,2})\dot{\theta}_2 + \mathbf{u}_3\dot{\theta}_3 \\ \dot{\phi}_4 &= \dot{\phi}_1 + \dot{\theta}_2 + \dot{\theta}_4 \\ \dot{\mathbf{r}}_4 &= \dot{\mathbf{r}}_1 + (\check{\mathbf{r}}_{2,1} + \check{\mathbf{r}}_{3,2} + \check{\mathbf{r}}_{4,3})\dot{\phi}_1 + (\check{\mathbf{d}}_{2,2} + \check{\mathbf{r}}_{3,2} + \check{\mathbf{r}}_{4,3})\dot{\theta}_2 + \mathbf{u}_3\dot{\theta}_3 + \check{\mathbf{d}}_{4,4}\dot{\theta}_4\end{aligned}$$

We define the following position vectors (for the floating joint,  $\mathbf{d}_{i,1} = \mathbf{r}_{i,1}$ ) as shown in Figure 9.7b:

$$\begin{aligned}\mathbf{d}_{3,1} &= \mathbf{r}_{3,1} = \mathbf{r}_{2,1} + \mathbf{r}_{3,2} \\ \mathbf{d}_{3,2} &= \mathbf{d}_{2,2} + \mathbf{r}_{3,2} \\ \mathbf{d}_{4,1} &= \mathbf{r}_{4,1} = \mathbf{r}_{2,1} + \mathbf{r}_{3,2} + \mathbf{r}_{4,3} \\ \mathbf{d}_{4,2} &= \mathbf{d}_{2,2} + \mathbf{r}_{3,2} + \mathbf{r}_{4,3}\end{aligned}$$

With these position vectors, the velocity equations find a simpler form, expressed in matrix form as

$$\left[ \begin{array}{c} \dot{\mathbf{r}}_1 \\ \dot{\phi}_1 \\ \dot{\mathbf{r}}_2 \\ \dot{\phi}_2 \\ \dot{\mathbf{r}}_3 \\ \dot{\phi}_3 \\ \dot{\mathbf{r}}_4 \\ \dot{\phi}_4 \end{array} \right] = \left[ \begin{array}{cc|c|c|c} \mathbf{I} & \check{\mathbf{d}}_{1,1} & & & \\ 0 & 1 & & & \\ \hline \mathbf{I} & \check{\mathbf{d}}_{2,1} & \check{\mathbf{d}}_{2,2} & & \\ 0 & 1 & 1 & & \\ \hline \mathbf{I} & \check{\mathbf{d}}_{3,1} & \check{\mathbf{d}}_{3,2} & \mathbf{u}_3 & \\ 0 & 1 & 1 & 0 & \\ \hline \mathbf{I} & \check{\mathbf{d}}_{4,1} & \check{\mathbf{d}}_{4,2} & \mathbf{u}_3 & \check{\mathbf{d}}_{4,4} \\ 0 & 1 & 1 & 0 & 1 \end{array} \right] \left[ \begin{array}{c} \mathbf{r}_1 \\ \phi_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{array} \right] \quad (9.6)$$

In the coefficient matrix, we have added a vector  $\check{\mathbf{d}}_{1,1}$  to the first row, second column. The addition of this vector does not change the matrix because  $\check{\mathbf{d}}_{1,1}$  is a zero vector—it is a vector that connects the mass center of body (1) to the joint reference point of the floating joint (1), which is the mass center. This vector is added to provide consistency in the appearance of the floating joint matrices.

The coefficient matrix in Eq. (9.6) is called the *velocity transformation matrix*, denoted as matrix **B**:

$$\mathbf{B} = \left[ \begin{array}{cc|c|c|c} \mathbf{I} & \check{\mathbf{d}}_{1,1} & & & \\ 0 & 1 & & & \\ \hline \mathbf{I} & \check{\mathbf{d}}_{2,1} & \check{\mathbf{d}}_{2,2} & & \\ 0 & 1 & 1 & & \\ \hline \mathbf{I} & \check{\mathbf{d}}_{3,1} & \check{\mathbf{d}}_{3,2} & \mathbf{u}_3 & \\ 0 & 1 & 1 & 0 & \\ \hline \mathbf{I} & \check{\mathbf{d}}_{4,1} & \check{\mathbf{d}}_{4,2} & \mathbf{u}_3 & \check{\mathbf{d}}_{4,4} \\ 0 & 1 & 1 & 0 & 1 \end{array} \right] \quad (9.7)$$

This matrix is at the heart of the transformation of the equations of motion from the body-coordinate to the joint-coordinate formulation. The rows of this matrix follow the body indices, and the columns represent the joint indices. The first two columns (actually the

**TABLE 9.1**

Block Matrices for Three Commonly Used Joints and Their Time Derivatives

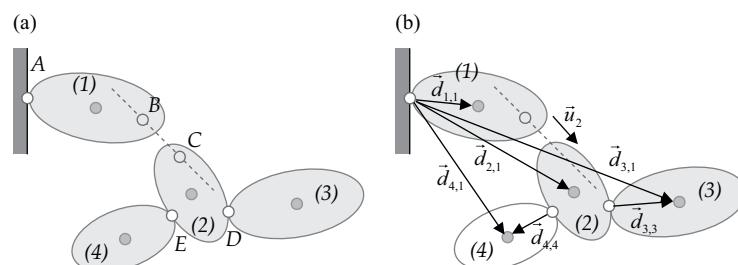
Joint type	Block matrix	Size	Time derivative of block matrix	
Floating	$\mathbf{F}_{i,j} = \begin{bmatrix} \mathbf{I} & \check{\mathbf{d}}_{i,j} \\ \mathbf{0} & 1 \end{bmatrix}$	$3 \times 3$	$\dot{\mathbf{F}}_{i,j} = \begin{bmatrix} \mathbf{0} & \dot{\check{\mathbf{d}}}_{i,j} \\ \mathbf{0} & 0 \end{bmatrix}$	(9.8)
Revolute	$\mathbf{R}_{i,j} = \begin{bmatrix} \check{\mathbf{d}}_{i,j} \\ 1 \end{bmatrix}$	$3 \times 1$	$\dot{\mathbf{R}}_{i,j} = \begin{bmatrix} \dot{\check{\mathbf{d}}}_{i,j} \\ 0 \end{bmatrix}$	(9.9)
Translational	$\mathbf{T}_{i,j} = \begin{bmatrix} \mathbf{u}_j \\ 0 \end{bmatrix}$	$3 \times 1$	$\dot{\mathbf{T}}_{i,j} = \begin{bmatrix} \dot{\mathbf{u}}_j \\ 0 \end{bmatrix}$	(9.10)

first three algebraic columns) belong to the floating joint, followed by the revolute joint at  $A$ , then the translational joint, and finally the last column belongs to the revolute joint at  $D$ . A brief observation of this matrix reveals that each type of joint has its own specific so-called *block matrix*. These matrices are listed in Table 9.1 where the subscript “ $i,j$ ” refers to body ( $i$ ) and joint ( $j$ ). If we construct the velocity transformation matrix  $\mathbf{B}$  for any open-chain system, regardless of the number of bodies and the placement of the joints, the block matrices will have exactly the same form as those in Table 9.1.

Another important observation is the location of block matrices in matrix  $\mathbf{B}$ . Block matrices in a typical row belonging to body ( $i$ ) reveal the joints that need to be navigated through if we move from the ground to that body. As an example, the block matrices associated with body (2), in our example, indicate that to reach this body from the ground, we have to go through the floating joint and the revolute joint  $A$ . Similarly, for body (4) we have to go through the floating joint, the revolute joint  $A$ , the translational joint, and finally through the revolute joint  $D$ . This structure suggests that matrix  $\mathbf{B}$  could be constructed directly based on the topology of the open-chain system, without taking the time derivative of the recursive coordinate equations.

### Example 9.1

Construct the  $\mathbf{B}$  matrix for the system shown in Figure 9.8a. There are revolute joints at  $A, D$ , and  $E$ , and a translational joint between bodies (1) and (2).

**FIGURE 9.8**

(a) Graphical description of a system with two branches and (b) the required vectors for the recursive kinematics.

**Solution**

Based on the topology of the system, the  $\mathbf{B}$  matrix can be constructed as

$$\mathbf{B} = \begin{bmatrix} \mathbf{R}_{1,1} & & & \\ \mathbf{R}_{2,1} & \mathbf{T}_{2,2} & & \\ \mathbf{R}_{3,1} & \mathbf{T}_{3,2} & \mathbf{R}_{3,3} & \\ \mathbf{R}_{4,1} & \mathbf{T}_{4,2} & & \mathbf{R}_{4,4} \end{bmatrix} = \begin{bmatrix} \check{\mathbf{d}}_{1,1} & & & \\ 1 & & & \\ \check{\mathbf{d}}_{2,1} & \mathbf{u}_2 & & \\ 1 & 0 & & \\ \check{\mathbf{d}}_{3,1} & \mathbf{u}_2 & \check{\mathbf{d}}_{3,3} & \\ 1 & 0 & 1 & \\ \check{\mathbf{d}}_{4,1} & \mathbf{u}_2 & & \mathbf{d}_{4,4} \\ 1 & 0 & & 1 \end{bmatrix}$$

The vectors that are used in this matrix are shown in Figure 9.8b.

**Example 9.2**

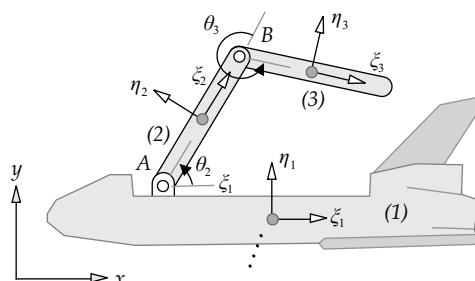
A simplified planar version of the robotic arm of the space shuttle is depicted in Figure 9.9. Determine the recursive coordinate formulas and the  $\mathbf{B}$  matrix.

**Solution**

Because the orbiting shuttle is not connected to the ground, we assume a floating joint between its main body, body (1), and a nonmoving reference frame  $x$ - $y$ . The joint coordinates for this floating joint are  $\mathbf{r}_1$  and  $\phi_1$ , the same as the body coordinates of body (1). For the pin joint at A, we consider a relative angle  $\theta_2 = \phi_2 - \phi_1$ , and for the pin joint at B, we consider another relative angle  $\theta_3 = \phi_3 - \phi_2$ .

For the coordinate transformation, we have

$$\begin{aligned} \phi_1 &= \phi_1 \\ \mathbf{r}_1 &= \mathbf{r}_1 \\ \phi_2 &= \phi_1 + \theta_2 \\ \mathbf{r}_2 &= \mathbf{r}_1 + \mathbf{s}_1^A - \mathbf{s}_2^A \\ \phi_3 &= \phi_2 + \theta_3 \\ \mathbf{r}_3 &= \mathbf{r}_2 + \mathbf{s}_2^B - \mathbf{s}_3^B \end{aligned} \tag{a}$$

**FIGURE 9.9**

An example of a floating multibody system.

The **B** matrix is constructed as

$$\mathbf{B} = \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{F}_{2,1} & \mathbf{R}_{2,2} & \mathbf{0} \\ \mathbf{F}_{3,1} & \mathbf{R}_{3,2} & \mathbf{R}_{3,3} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \check{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 \\ \mathbf{I} & \check{\mathbf{d}}_{2,1} & \check{\mathbf{d}}_{2,2} & \mathbf{0} \\ 0 & 1 & 1 & 0 \\ \mathbf{I} & \check{\mathbf{d}}_{3,1} & \check{\mathbf{d}}_{3,2} & \check{\mathbf{d}}_{3,3} \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad (9.11)$$

### 9.2.1 Absolute Angle

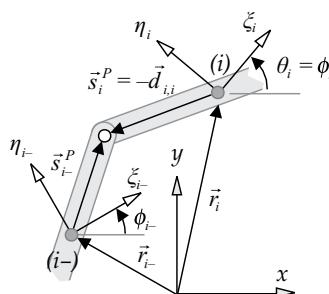
The joint coordinate for a revolute joint, as we implemented in Eq. (9.1), is defined as a relative angle between the two connected bodies. It is also possible to define an absolute angle to represent the joint coordinate for a revolute joint. An absolute angle may not exhibit a true meaning of the joint coordinate; however, it can result in a kinematic formulation identical to that obtained from the vectorial formulation, which could be viewed as a simplifying feature.

With an absolute joint coordinate defined for the revolute joint ( $i$ ), as shown in Figure 9.10, the rotational and translational coordinates of body ( $i$ ) can be described as

$$\begin{aligned} \phi_i &= \theta_i \\ \mathbf{r}_i &= \mathbf{r}_{i-} + \mathbf{s}_{i-}^P - \mathbf{s}_i^P \end{aligned} \quad (9.12)$$

$$\begin{aligned} \dot{\phi}_i &= \dot{\theta}_i \\ \dot{\mathbf{r}}_i &= \dot{\mathbf{r}}_{i-} + \check{\mathbf{s}}_{i-}^P \dot{\phi}_{i-} - \check{\mathbf{s}}_i^P \dot{\theta}_i \end{aligned} \Rightarrow \left\{ \begin{array}{l} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \check{\mathbf{s}}_{i-}^P \\ \mathbf{0} & 0 \end{array} \right] \left\{ \begin{array}{l} \dot{\mathbf{r}}_{i-} \\ \dot{\phi}_{i-} \end{array} \right\} + \left[ \begin{array}{c} \check{\mathbf{d}}_{i,i} \\ 1 \end{array} \right] \dot{\theta}_i \quad (9.13)$$

A comparison between this last equation and Eq. (9.2) reveals that the coefficient matrix for the joint velocity  $\dot{\theta}_i$  has not changed, but the coefficient matrix of the velocity of body ( $i-$ ) has changed. The overall effect of this change on the **B** matrix can be best demonstrated with an example.



**FIGURE 9.10**

Recursive kinematics for a revolute joint with an absolute joint coordinate.

**Example 9.2 (cont.)**

For the robotic arm of the space shuttle, consider an absolute angle for the revolute joint  $B$ . Derive the corresponding  $\mathbf{B}$  matrix.

**Solution**

Keeping the joint coordinate at revolute joint  $A$  as a relative angle, but using an absolute angle  $\theta = \phi_3$  at  $B$ , results in the following coordinate transformations:

$$\begin{aligned}\dot{\phi}_1 &= \dot{\phi}_1 \\ \dot{\mathbf{r}}_1 &= \dot{\mathbf{r}}_1 \\ \dot{\phi}_2 &= \dot{\phi}_1 + \dot{\theta}_2 \\ \dot{\mathbf{r}}_2 &= \dot{\mathbf{r}}_1 + \dot{\mathbf{s}}_1^A - \dot{\mathbf{s}}_2^A \\ \dot{\phi}_3 &= \theta_3 \\ \dot{\mathbf{r}}_3 &= \dot{\mathbf{r}}_2 + \dot{\mathbf{s}}_2^B - \dot{\mathbf{s}}_3^B\end{aligned}\tag{b}$$

The first time derivative of Eq. (b) is

$$\begin{aligned}\dot{\phi}_1 &= \dot{\phi}_1 & \dot{\phi}_1 &= \dot{\phi}_1 \\ \dot{\mathbf{r}}_1 &= \dot{\mathbf{r}}_1 & \dot{\mathbf{r}}_1 &= \dot{\mathbf{r}}_1 \\ \dot{\phi}_2 &= \dot{\phi}_1 + \dot{\theta}_2 & \dot{\phi}_2 &= \dot{\phi}_1 + \dot{\theta}_2 \\ \dot{\mathbf{r}}_2 &= \dot{\mathbf{r}}_1 + \dot{\mathbf{s}}_1^A \dot{\phi}_1 - \dot{\mathbf{s}}_2^A (\dot{\phi}_1 + \dot{\theta}_2) & \Rightarrow \dot{\mathbf{r}}_2 &= \dot{\mathbf{r}}_1 + (\dot{\mathbf{s}}_1^A - \dot{\mathbf{s}}_2^A) \dot{\phi}_1 - \dot{\mathbf{s}}_2^A \dot{\theta}_2 \\ \dot{\phi}_3 &= \dot{\theta}_3 & \dot{\phi}_3 &= \dot{\theta}_3 \\ \dot{\mathbf{r}}_3 &= \dot{\mathbf{r}}_2 + \dot{\mathbf{s}}_2^B (\dot{\phi}_1 + \dot{\theta}_2) - \dot{\mathbf{s}}_3^B \dot{\theta}_3 & \dot{\mathbf{r}}_3 &= \dot{\mathbf{r}}_1 + (\dot{\mathbf{s}}_1^A - \dot{\mathbf{s}}_2^A + \dot{\mathbf{s}}_2^B) \dot{\phi}_1 - (\dot{\mathbf{s}}_2^A - \dot{\mathbf{s}}_2^B) \dot{\theta}_2 - \dot{\mathbf{s}}_3^B \dot{\theta}_3\end{aligned}$$

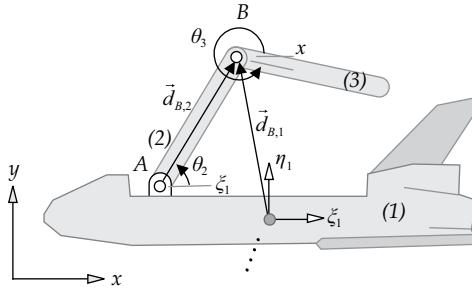
The new  $\mathbf{B}$  matrix is constructed as

$$\mathbf{B} = \begin{bmatrix} \mathbf{I} & \check{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 \\ \mathbf{I} & \check{\mathbf{d}}_{2,1} & \check{\mathbf{d}}_{2,2} & \mathbf{0} \\ 0 & 1 & 1 & 0 \\ \mathbf{I} & \check{\mathbf{d}}_{B,1} & \check{\mathbf{d}}_{B,2} & \check{\mathbf{d}}_{3,3} \\ 0 & 0 & 0 & 1 \end{bmatrix}\tag{9.14}$$

where  $\check{\mathbf{d}}_{B,1} = \check{\mathbf{s}}_1^A - \check{\mathbf{s}}_2^A + \check{\mathbf{s}}_2^B$  and  $\check{\mathbf{d}}_{B,2} = -\check{\mathbf{s}}_2^A + \check{\mathbf{s}}_2^B$  as shown in Figure 9.11.

Comparing the  $\mathbf{B}$  matrices from Eqs. (9.11) and (9.14) shows that the difference is in the rows associated with body (3):

$$\mathbf{B}_{(rel)} = \left[ \begin{array}{cccc} \vdots & & & \\ - & \check{\mathbf{d}}_{3,1} & \check{\mathbf{d}}_{3,2} & \check{\mathbf{d}}_{3,3} \\ - & 1 & 1 & 1 \end{array} \right], \quad \mathbf{B}_{(abs)} = \left[ \begin{array}{cccc} \vdots & & & \\ - & \mathbf{I} & \check{\mathbf{d}}_{B,1} & \check{\mathbf{d}}_{B,2} & \check{\mathbf{d}}_{3,3} \\ - & 0 & 0 & 0 & 1 \end{array} \right]$$



**FIGURE 9.11**  
Defining an absolute joint coordinate at  $B$ .

Since we did not change the joint coordinates for the floating and revolute joints at  $A$ , there are no changes in the rows associated with bodies (1) and (2). The last row of  $\mathbf{B}_{(rel)}$  states that the rotational velocity of body (3), as well as its rotational coordinate, depends on the joint velocities  $\dot{\phi}_1$  and  $\dot{\theta}_2$ . In contrast, the last row of  $\mathbf{B}_{(abs)}$  indicates that the rotational velocity of body (3) does not depend on the joint velocities  $\dot{\phi}_1$  and  $\dot{\theta}_2$  ("ones" have become "zeros"). This change in the dependency is also reflected on the associated  $\vec{d}_{i,j}$  vectors that have become  $\vec{d}_{B,j}$  vectors. Therefore, when we use an absolute angle, these subtle changes must carefully be incorporated in the  $\mathbf{B}$  matrix, if the matrix is constructed based on the topology and the associated block matrices.

In the kinematic description of a multibody system containing revolute joints, we may use either the relative joint-coordinate formulations of Eqs. (9.1) and (9.2) or those of Eqs. (9.12) and (9.13). In most derivations of this chapter, we consider relative angles for revolute joints unless it is stated otherwise.

### 9.2.2 Equations of Motion

For an open-chain system containing  $n_b$  bodies and  $n_{dof}$  DoFs, we define  $n_{dof}$  joint coordinates as

$$\boldsymbol{\theta} = \begin{Bmatrix} \theta_1 \\ \vdots \\ \theta_{n_{dof}} \end{Bmatrix} \quad (9.15)$$

We then write recursive expressions describing the body coordinates as functions of the joint coordinates:

$$\mathbf{c} = \mathbf{c}(\boldsymbol{\theta}) \quad (9.16)$$

As we have already seen in Section 9.2, the time derivative of the coordinate transformations yields the velocity transformation as

$$\dot{\mathbf{c}} = \mathbf{B}\dot{\boldsymbol{\theta}} \quad (9.17)$$

The acceleration transformation is obtained from the time derivative of the velocity transformation as

$$\ddot{\mathbf{c}} = \mathbf{B}\ddot{\boldsymbol{\theta}} + \dot{\mathbf{B}}\dot{\boldsymbol{\theta}} \quad (9.18)$$

There are no constraints associated with the joint coordinates as long as the number of defined coordinates is equal to the number of system's DoFs. We can verify this claim easily at the velocity level by substituting Eq. (9.17) in the velocity constraints of Eq. (7.4):

$$\dot{\Phi} = \mathbf{D}\dot{\mathbf{c}} = \mathbf{0} \Rightarrow \mathbf{D}\mathbf{B}\dot{\boldsymbol{\theta}} = \mathbf{0} \quad (9.19)$$

Since there are as many joint velocities in  $\dot{\boldsymbol{\theta}}$  as the number of DoFs, the elements of  $\dot{\boldsymbol{\theta}}$  are independent; that is, they can be assigned arbitrary values. Hence, Eq. (9.19) can only be valid if

$$\mathbf{D}\mathbf{B} = \mathbf{0} \quad (9.20)$$

This means that the rows of  $\mathbf{D}$  and the columns of  $\mathbf{B}$  are orthogonal. This is a very important and useful characteristic of matrix  $\mathbf{B}$ .

The equations of motion for a multibody system containing kinematic joints were derived in Eq. (7.53) as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + \mathbf{D}'\boldsymbol{\lambda} \quad (9.21)$$

We substitute Eq. (9.18) in Eq. (9.21) and premultiply the resulting equation by  $\mathbf{B}'$  to obtain

$$\mathbf{B}'\mathbf{M}(\mathbf{B}\ddot{\boldsymbol{\theta}} + \dot{\mathbf{B}}\dot{\boldsymbol{\theta}}) = \mathbf{B}'{}^{(a)}\mathbf{h} + \mathbf{B}'\mathbf{D}'\boldsymbol{\lambda}$$

According to Eq. (9.20),  $\mathbf{B}'\mathbf{D}' = \mathbf{0}$ , the equations of motion become

$$\mathbf{M}\ddot{\boldsymbol{\theta}} = {}^{(a)}\mathbf{h} \quad (9.22)$$

where

$$\mathbf{M} = \mathbf{B}'\mathbf{M}\mathbf{B} \quad (9.23)$$

$${}^{(a)}\mathbf{h} = \mathbf{B}'\left({}^{(a)}\mathbf{h} - \mathbf{M}\dot{\mathbf{B}}\dot{\boldsymbol{\theta}}\right) \quad (9.24)$$

Where matrix  $\mathbf{M}$  is the new mass matrix and vector  ${}^{(a)}\mathbf{h}$  is the new array of applied and velocity-dependent inertial forces.

We note that to evaluate the array of forces for the equations of motion, we need the time derivative of the velocity transformation matrix, or the product  $\dot{\mathbf{B}}\dot{\boldsymbol{\theta}}$ . Because the block matrices are known for the commonly used joints, the time derivative of the block matrices can be expressed in analytical form, as provided in Table 9.1.

We observe that the transformation of the equations of motion from the body coordinates to the joint coordinates, for open-chain systems, eliminates the reaction forces from the equations. To compute the reaction forces, if needed, after determining the joint

acceleration from Eq. (9.22), Eq. (9.18) can be evaluated to determine the body accelerations. Then the body accelerations are substituted in Eq. (9.21) to obtain

$$\mathbf{D}'\lambda = \mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h}$$

Since  $\mathbf{D}'$  is a rectangular matrix containing more rows than columns, to solve this equation for the Lagrange multipliers, we premultiply the equation by  $\mathbf{D}$  to obtain

$$\mathbf{D}\mathbf{D}'\lambda = \mathbf{D}(\mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h})$$

The coefficient matrix  $\mathbf{D}\mathbf{D}'$  is square, and therefore, the equation can be solved for the Lagrange multipliers.\* The array of reaction forces is then determined as

$${}^{(r)}\mathbf{h} = \mathbf{D}'\lambda \quad (9.25)$$

### Example 9.3

The variable-length pendulum shown in Figure 9.12a contains two bodies, a pin joint, a sliding joint, a rotational spring about the pin joint at  $O$ , and a point-to-point spring-damper between point  $O$  and the mass center of body (2) at  $A$ . Gravity acts on the system. The following constant data are provided:

$$L = 0.5 \text{ m}, m_1 = 2.0 \text{ kg}, J_1 = 0.04 \text{ kg} \cdot \text{m}^2, m_2 = 1.0 \text{ kg}, J_2 = 0.1 \text{ kg} \cdot \text{m}^2$$

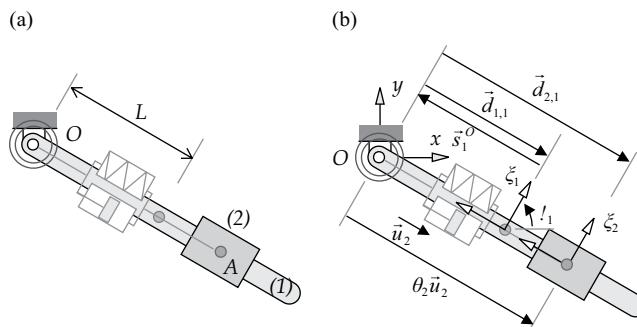
$${}^{(r)}k = 6 \text{ N} \cdot \text{m/rad}, {}^0\theta = 0.0 \text{ (rotational spring)}$$

$$k = 20 \text{ N/m}, {}^0L = 0.7 \text{ m}, d_c = 5 \text{ N} \cdot \text{s/m} \text{ (point-to-point spring-damper)}$$

Two joint coordinates are defined as shown in Figure 9.12b. At the initial time, the values of the joint coordinates and velocities are given as

$$\theta_1 = \pi/3 \text{ rad}, \theta_2 = 0.8 \text{ m}, \dot{\theta}_1 = 0, \dot{\theta}_2 = 0$$

Determine the joint acceleration at the initial time.



**FIGURE 9.12**

(a) A variable-length pendulum and (b) the defined joint coordinates and vectors.

\* It is assumed that all of the kinematic constraints are independent.

**Solution**

The following MATLAB program performs the necessary computations.

**MATLAB/Chapter 9/Example \_ 9 \_ 3**

We state the constant data:

```
s_O1_local = [0; 0.5]; u2_local = [0; -1]; uy = [0; 1];
k_r = 6; theta0 = 0;
k = 20; L0 = 0.7; dc = 5;
m = [2; 1]; J = [0.04; 0.1]; g = 9.81;
```

The two defined joint coordinates are  $\theta_1 = \phi_1$  for the rotation of body (1), and  $\theta_2$  for the relative translation between the two bodies.

```
theta = [pi/3; 0.8]; theta_d = [0; 0];
```

The recursive coordinate formulas are as follows:

$$\begin{aligned}\phi_1 &= \theta_1 \\ \mathbf{r}_1 &= -\mathbf{s}_1^A \\ \phi_2 &= \phi_1 \\ \mathbf{r}_2 &= \mathbf{u}_2 \theta_2\end{aligned}$$

```
phi1 = theta(1); A1 = A_matrix(phi1);
r1 = -A1*s_O1_local;
phi2 = phi1; A2 = A1;
u2 = A2*u2_local;
r2 = theta(2)*u2;
c = [r1; phi1; r2; phi2] ..... c =
0.4330
-0.2500
1.0472
0.6928
-0.4000
1.0472
```

The velocity transformation matrix is constructed as

$$\mathbf{B} = \left[ \begin{array}{cc} \mathbf{R}_{1,1} & \mathbf{0} \\ \mathbf{R}_{2,1} & \mathbf{T}_{2,2} \end{array} \right] = \left[ \begin{array}{c|c} \check{\mathbf{d}}_{1,1} & \mathbf{0} \\ \hline 1 & 0 \\ \check{\mathbf{d}}_{2,1} & \mathbf{u}_2 \\ \hline 1 & 0 \end{array} \right] = \left[ \begin{array}{c|c} \check{\mathbf{r}}_1 & \mathbf{0} \\ \hline \check{\mathbf{r}}_2 & \mathbf{u}_2 \\ \hline 1 & 0 \end{array} \right]$$

```
B = [s_rot(r1) [0; 0];
      1          0
      s_rot(r2) u2
      1          0] ..... B =
0.2500      0
0.4330      0
1.0000      0
0.4000  0.8660
0.6928 -0.5000
1.0000      0
```

We compute body velocities:

```
c_d = B*theta_d      .....  
c_d =  
0  
0  
0  
0  
0  
0  
0
```

The product  $\dot{\mathbf{B}}\dot{\theta}$  is expressed as

$$\dot{\mathbf{B}}\dot{\boldsymbol{\theta}} = \left\{ \begin{array}{c} \dot{\mathbf{r}}_1 \dot{\theta}_1 \\ 0 \\ \dot{\mathbf{r}}_2 \dot{\theta}_1 + \dot{\mathbf{u}}_2 \dot{\theta}_2 \\ 0 \end{array} \right\}$$

```

r1_d = c_d(1:2); r2_d = c_d(4:5);
u2_d = s_rot(u2)*c_d(6);
Bdtd = [s_rot(r1_d)*theta_d(1)
         0
         s_rot(r2_d)*theta_d(1) + ...
         s_rot(u2_d)*theta_d(2)
0] ..... Bdtd =
         0
         0
         0
         0
         0
         0
         0

```

We construct the body-coordinate mass matrix as an array:

$$M = [m(1) \ m(1) \ J(1) \ m(2) \ m(2) \ J(2)]$$

The array of body-coordinate applied loads contains the gravitational forces on the two bodies, the torque of the rotational spring that acts on body (1), and the force of the spring-damper that acts on body (2). Since this force acts directly at the mass center, there is no moment associated with it.

$$^{(a)}\mathbf{h} = \begin{cases} \mathbf{w}_1 \\ -k_r(\theta_1 - {}^0\theta) \\ \mathbf{w}_2 - \left(k(\theta_2 - {}^0L) + d_c \dot{\theta}_2\right)\mathbf{u}_2 \\ 0 \end{cases}$$

```
w1 = -m(1)*g*uy; w2 = -m(2)*g*uy;
T_r = r_s(theta(1), k_r, theta0) .....
T1 = -T_rd
f_sd = pp_sd(r2, r2_d, k, L0, dc) .....
f2 = -f_sd;

T_r =
6.2832
f_sd =
1.7321
1.0000
```

```
h_a = [w1; T1; w2 + f2; 0] .....
```

```
h_a =
0
-19.6200
-6.2832
-1.7321
-8.8100
0
```

The joint-coordinate mass matrix and the array of applied loads are computed next:

```
M_joint = B'*diag(M)*B .....
```

```
M_joint =
1.2800 0.0000
0.0000 1.0000
```

```
h_joint = B'*(h_a - diag(M)*Bdtd) .....
```

```
h_joint =
-21.5755
2.9050
```

Now the joint accelerations can be determined:

```
theta_dd = M_joint\h_joint .....
```

```
theta_dd =
-16.8558
2.9050
```

If necessary, one could compute the body accelerations using Eq. (9.18).

It would be interesting to construct the mass matrix and the array of forces in analytical form for observation. If we perform the operations of Eqs. (9.23) and (9.24) analytically, we get

$$\begin{aligned} \mathbf{M} &= \mathbf{B}' \mathbf{M} \mathbf{B} = \begin{bmatrix} m_1 L^2 + J_1 + m_2 \theta_2^2 + J_2 & 0 \\ 0 & m_2 \end{bmatrix} \\ {}^{(a)} \mathbf{h} &= \mathbf{B}' ({}^{(a)} \mathbf{h} - \mathbf{M} \dot{\mathbf{B}} \dot{\theta}) = \begin{Bmatrix} \check{\mathbf{r}}'_1 \mathbf{w}_1 + \check{\mathbf{r}}'_2 \mathbf{w}_2 - k_r (\theta_1 - {}^0 \theta) \\ \mathbf{u}'_2 \mathbf{w}_2 - (k(\theta_2 - {}^0 L) + d_c \dot{\theta}_2) \end{Bmatrix} \end{aligned}$$

Hence, the equations of motion for this system can be expressed as

$$\begin{bmatrix} m_1 L^2 + J_1 + m_2 \theta_2^2 + J_2 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} \check{\mathbf{r}}'_1 \mathbf{w}_1 + \check{\mathbf{r}}'_2 \mathbf{w}_2 - k_r (\theta_1 - {}^0 \theta) \\ \mathbf{u}'_2 \mathbf{w}_2 - (k(\theta_2 - {}^0 L) + d_c \dot{\theta}_2) \end{Bmatrix}$$

In this particular example, the off-diagonal elements of the mass matrix are zeros. However, for more complex problems, the off-diagonal elements of the mass matrix are not necessarily zeros. In any case, we can make the following observations:

- The first row of this equation represents a rotational equation of motion where the term  $m_1 L^2 + J_1 + m_2 \theta_2^2 + J_2$  is the moment of inertia of the combined bodies about O. This moment of inertia is not constant since  $\theta_2$  is a variable.
- The torque of the rotational spring could have been included directly in the joint-coordinate equations of motion associated with  $\theta_1$ —this torque only affects the rotation of the system.

- The gravitational forces acting on the two bodies have their own moment arms about point  $O$ .
  - The second equation represents the sliding motion of body (2) along the axis of the sliding joint. The force of the spring-damper acts only on body (2). This force could have been included directly in the joint-coordinate equations of motion associated with  $\theta_2$ —this force does not directly influence the rotation of the system.
- 

### 9.3 Closed-Chain Systems

The topology of multibody systems containing closed chains cannot be described as a tree. In a closed chain, or a loop, if we start navigating from one body through a joint to another body and so on, we will end up at the body we started from. Therefore, the first step is to transform the system to an open-chain system through a process called *cut-joint*.

The cut-joint process starts by temporarily removing one joint from each loop. This temporary removal of joints converts the system to a single or multiple open-chain trees. We refer to the resulting system as the *cut system* and the removed joints as the *cut-joints*. If the system contains only one loop, we need to remove only one joint.

As an example, consider the closed-chain system of Figure 9.13a. There are four joints in this single-loop system, and therefore, there are four possible cut-joints as shown in Figure 9.13b–e. It is totally our choice to choose which joint should be cut from a loop. As a second example, consider the six-bar mechanism shown in Figure 9.14a that contains two loops. Any of the joints in this system can be considered for cutting. Two out of several such cut choices are shown in Figure 9.14b and c.

After the cut-joint process is completed, for the resultant open-chain system, we define the joint coordinates and the corresponding transformation expressions as discussed in Section 9.2. We then put the cut-joints back in the system by introducing constraints between the joint coordinates. The cut-joint constraints result in a small number of Lagrange multipliers that will remain in the final set of equations of motion.

#### 9.3.1 Cut-Joint Constraints

For a closed-chain system, let us assume that the corresponding open-chain system, after the cut-joints are removed, is represented by a set of joint coordinates as in Eq. (9.15). For this cut open-chain system, the coordinate, velocity, and acceleration transformations are determined as in Eqs. (9.16)–(9.18); in other words, we construct the coordinate transformation formulas, matrix  $B$ , and the product  $B\dot{\theta}$ .

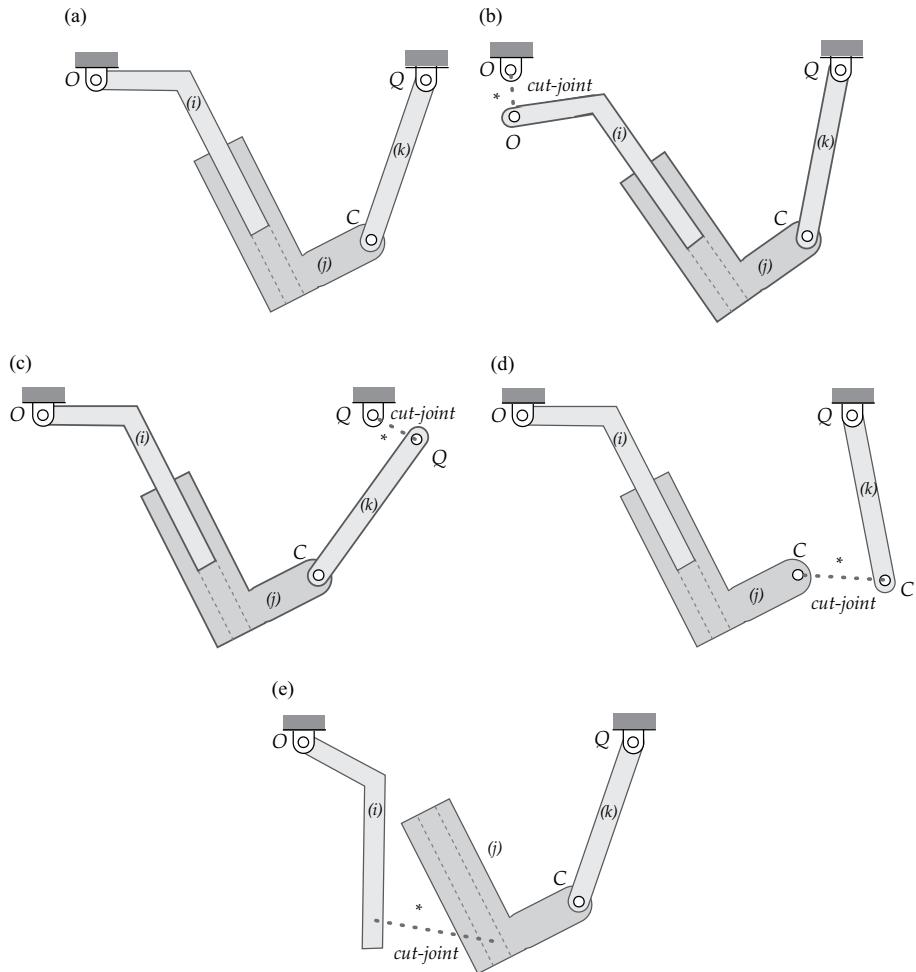
When the cut-joints are put back into the system, we consider the constraint equations for these joints from the body-coordinate formulation, expressed as

$${}^*\Phi(\mathbf{c}) = {}^*\Phi(\mathbf{c}(\boldsymbol{\theta})) = \mathbf{0} \quad (9.26)$$

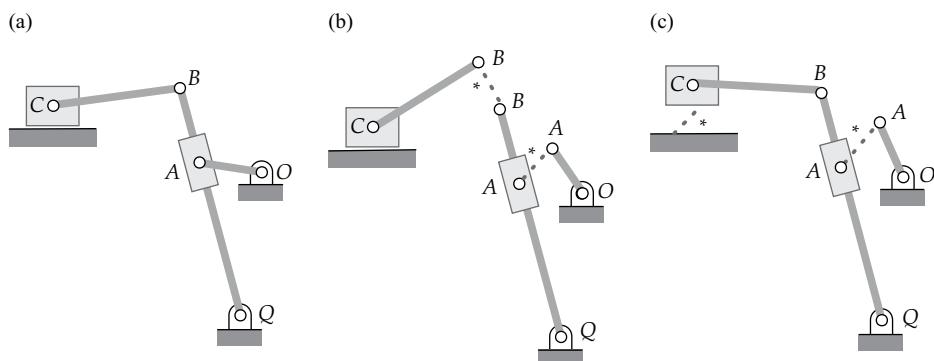
The left superscript  $*$  indicates an entity associated with the cut-joints.

For the velocity constraints, we have

$${}^*\dot{\Phi} = {}^*D\dot{\mathbf{c}} = \mathbf{0} \Rightarrow {}^*\dot{\Phi} = {}^*D\mathbf{B}\dot{\boldsymbol{\theta}} = \mathbf{C}\dot{\boldsymbol{\theta}} = \mathbf{0} \quad (9.27)$$

**FIGURE 9.13**

(a) A closed chain system and (b–e) four possible open-chain systems as the result of the cut-joint process.

**FIGURE 9.14**

(a) A multi-loop mechanism and (b and c) two of its corresponding cut open-chain systems.

where

$$\mathbf{C} = {}^*\mathbf{DB} \quad (9.28)$$

Matrix  $\mathbf{C}$  is the transformed Jacobian matrix of the cut-joints. The acceleration constraints for the cut-joints become

$${}^*\ddot{\Phi} = \mathbf{C}\ddot{\theta} + \dot{\mathbf{C}}\dot{\theta} = \mathbf{0} \quad (9.29)$$

where

$$\dot{\mathbf{C}} = {}^*\mathbf{DB} + {}^*\dot{\mathbf{DB}} \quad (9.30)$$

The array  $\dot{\mathbf{C}}\dot{\theta}$  can also be expressed as

$$\dot{\mathbf{C}}\dot{\theta} = {}^*\mathbf{DB}\dot{\theta} + {}^*\dot{\mathbf{DB}}\dot{\theta} = {}^*\mathbf{DB}\dot{\theta} + {}^*\dot{\mathbf{D}}\dot{\mathbf{c}} = {}^*\mathbf{D}\{\dot{\mathbf{B}}\dot{\theta}\} - {}^*\gamma \quad (9.31)$$

This expression reveals that if the Jacobian matrix and the right-hand side of the acceleration constraints for the cut-joints are available in the body-coordinate formulation,  $\dot{\mathbf{C}}\dot{\theta}$  can easily be computed.

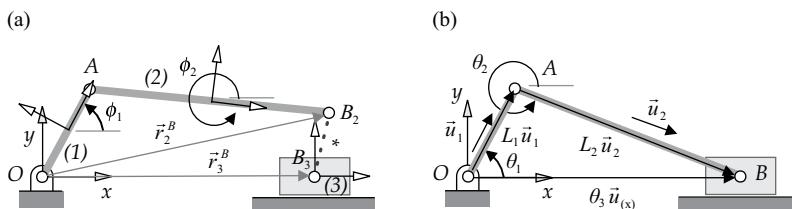
The velocity and acceleration constraints of Eqs. (9.27) and (9.29) require the Jacobian matrix  $\mathbf{C}$  and its time derivative  $\dot{\mathbf{C}}$  (or the product  $\dot{\mathbf{C}}\dot{\theta}$ ). Equations (9.28) and (9.30) provide the means to evaluate these matrices. However, a close look at the cut-joint constraints reveals a simpler process for such evaluations.

A comparison between the cut-joint constraints of the joint-coordinate formulation and its vector-loop representation from the vectorial formulation of Chapter 5 reveals that the two methods yield identical constraint equations. Therefore, if the joint coordinates were selected to be the same as the vectorial variables, the resulting  $\mathbf{C}$  and  $\dot{\mathbf{C}}$  matrices would be identical as well. This means that instead of using Eqs. (9.28) and (9.30), we can obtain these entities from the vector-loop approach as long as the selected coordinates between the two formulations are the same.

#### Example 9.4

Let us consider the slider–crank example shown in Figure 9.15 for the joint-coordinate formulation. If we cut the revolute joint at  $B$ , as shown in Figure 9.15a, we have the following cut-joint constraints:

$${}^*\Phi = \mathbf{r}_2^B - \mathbf{r}_3^B = \mathbf{0} \quad (a)$$



**FIGURE 9.15**

The closed chain of a slider–crank mechanism is described by (a) a cut-joint and (b) a vector loop.

The two vectors  $\mathbf{r}_2^B$  and  $\mathbf{r}_3^B$  can be described in terms of the unit vectors shown in Figure 9.15b as  $\mathbf{r}_2^B = L_1\mathbf{u}_1 + L_2\mathbf{u}_2$  and  $\mathbf{r}_3^B = \theta_3\mathbf{u}_{(x)}$ . The cut-joint constraint of Eq. (a) can now be expressed as

$${}^*\Phi = L_1\mathbf{u}_1 + L_2\mathbf{u}_2 - \theta_3\mathbf{u}_{(x)} = \mathbf{0} \quad (\text{b})$$

This is exactly the vector-loop constraint that we can derive for this slider-crank mechanism based on the description of the vectors shown in Figure 9.15b.

To obtain the  $\mathbf{C}$  matrix from the vector-loop equation, referring to the angles that are defined in Figure 9.15b, we take the time derivative of Eq. (b):

$${}^*\dot{\Phi} = L_1\check{\mathbf{u}}_1\dot{\theta}_1 + L_2\check{\mathbf{u}}_2\dot{\theta}_2 - \dot{\theta}_3\mathbf{u}_{(x)} = \mathbf{0}$$

This equation leads to the  $\mathbf{C}$  matrix as

$$\mathbf{C} = \begin{bmatrix} L_1\check{\mathbf{u}}_1 & L_2\check{\mathbf{u}}_2 & -\mathbf{u}_{(x)} \end{bmatrix} \quad (\text{c})$$

To obtain the  $\mathbf{C}$  matrix from the joint-coordinate formulation, we assume that the joint coordinates are selected as those shown in Figure 9.15b. Note that the joint coordinate for revolute joint  $A$  is an absolute angle and not a relative angle, and therefore, the coordinate and velocity transformations, and the  $\mathbf{B}$  matrix are

$$\begin{array}{ll} \mathbf{r}_1 = \frac{L_1}{2}\mathbf{u}_1 & \dot{\mathbf{r}}_1 = \frac{L_1}{2}\check{\mathbf{u}}_1\dot{\theta}_1 \\ \dot{\phi}_1 = \theta_1 & \dot{\phi}_1 = \dot{\theta}_1 \\ \mathbf{r}_2 = L_1\mathbf{u}_1 + \frac{L_2}{2}\mathbf{u}_2 & \dot{\mathbf{r}}_2 = L_1\check{\mathbf{u}}_1\dot{\theta}_1 + \frac{L_2}{2}\check{\mathbf{u}}_2\dot{\theta}_2 \\ \dot{\phi}_2 = \theta_2 & \dot{\phi}_2 = \dot{\theta}_2 \\ \mathbf{r}_3 = \mathbf{u}_{(x)}\theta_3 & \dot{\mathbf{r}}_3 = \mathbf{u}_{(x)}\dot{\theta}_3 \\ \dot{\phi}_3 = 0 & \dot{\phi}_3 = 0 \end{array}, \mathbf{B} = \begin{bmatrix} \frac{L_1}{2}\check{\mathbf{u}}_1 & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 \\ L_1\check{\mathbf{u}}_1 & \frac{L_2}{2}\check{\mathbf{u}}_2 & \mathbf{0} \\ 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{u}_{(x)} \\ 0 & 0 & 0 \end{bmatrix}$$

For the cut-joint, we can write the body-coordinate Jacobian matrix as

$${}^*\mathbf{D} = \left[ \begin{array}{cc|cc|cc} \mathbf{0} & \mathbf{0} & \mathbf{I} & \frac{L_2}{2}\check{\mathbf{u}}_2 & -\mathbf{u}_{(x)} & \mathbf{0} \end{array} \right]$$

We then determine the  $\mathbf{C}$  matrix as

$$\mathbf{C} = {}^*\mathbf{D}\mathbf{B} = \left[ \begin{array}{ccc} L_1\check{\mathbf{u}}_1 & \frac{L_2}{2}\check{\mathbf{u}}_2 + \frac{L_2}{2}\check{\mathbf{u}}_2 & -\mathbf{u}_{(x)} \end{array} \right] \Rightarrow \mathbf{C} = \begin{bmatrix} L_1\check{\mathbf{u}}_1 & L_2\check{\mathbf{u}}_2 & -\mathbf{u}_{(x)} \end{bmatrix}$$

which is the same as what we derived from the vector loop process. It should also be clear that the product  $\mathbf{C}\dot{\theta}$  obtained by either method would be the same.

### 9.3.2 Equations of Motion

In a closed-chain system, we can divide the joints into two groups—those that remain in the open-chain portion and those that are cut. Based on this grouping, we express the body-coordinate equations of motion for the system as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + \mathbf{D}'\lambda + {}^*\mathbf{D}'{}^*\lambda \quad (9.32)$$

where the reaction forces are split into two sets according to the grouping of the joints:  $\mathbf{D}'\lambda$  represents the reaction forces at the joints in the open-chain portion; and  ${}^*\mathbf{D}'{}^*\lambda$  represents the reaction forces at the cut-joints.

To transform the equations of motion to the joint coordinates, we substitute Eq. (9.18) in Eq. (9.4) and premultiply the resultant by  $\mathbf{B}'$ :

$$\mathbf{B}'\mathbf{M}(\mathbf{B}\ddot{\theta} + \dot{\mathbf{B}}\dot{\theta}) = \mathbf{B}'\left({}^{(a)}\mathbf{h} + \mathbf{D}'\lambda + {}^*\mathbf{D}'{}^*\lambda\right)$$

Rearranging the terms, noting that  $\mathbf{B}'\mathbf{D}' = \mathbf{0}$  based on Eq. (9.20), and using Eq. (9.28) result in

$$\mathbf{M}\ddot{\theta} = {}^{(a)}\mathbf{h} + \mathbf{C}'{}^*\lambda \quad (9.33)$$

where the mass matrix,  $\mathbf{M}$ , and the array of applied forces,  ${}^{(a)}\mathbf{h}$ , are the same as those in the corresponding cut open-chain system defined in Eqs. (9.23) and (9.24). The term  $\mathbf{C}'{}^*\lambda$  represents the reaction forces and/or torques at the cut-joints. If all the joint coordinates and velocities are known at a given time, the joint accelerations can be found if we append the acceleration constraints from Eq. (9.29) to Eq. (9.33) to form the following:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{C}' \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\theta} \\ {}^*\lambda \end{Bmatrix} = \begin{Bmatrix} {}^{(a)}\mathbf{h} \\ -\dot{\mathbf{C}}\dot{\theta} \end{Bmatrix} \quad (9.34)$$

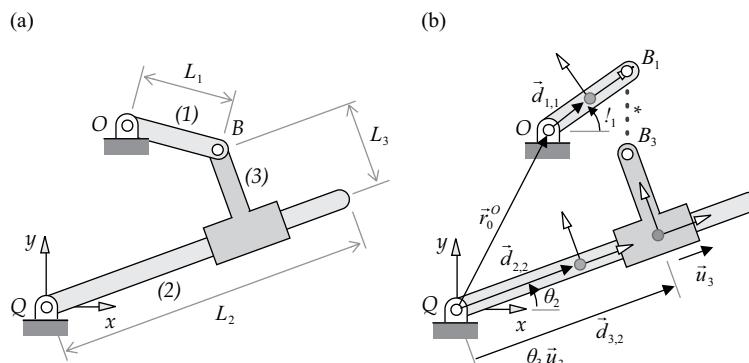
The preceding equation contains as many algebraic equations as the unknowns.

### Example 9.5

Consider the inverted slider–crank mechanism shown in Figure 9.16a. The cut system, as shown in Figure 9.16b, contains two trees: one body in the first tree and two bodies in the second tree. The following constant data are provided:

$$\mathbf{r}_0^O = \begin{Bmatrix} 0.15 & 0.33 \end{Bmatrix}' \text{ m}, L_1 = 0.208 \text{ m}, L_2 = 0.6 \text{ m}, L_3 = 0.195 \text{ m}$$

$$m_1 = 0.2 \text{ kg}, J_1 = 0.004 \text{ kg} \cdot \text{m}^2, m_2 = 1.0 \text{ kg}, J_2 = 0.1 \text{ kg} \cdot \text{m}^2, m_3 = 0.5 \text{ kg}, J_3 = 0.05 \text{ kg} \cdot \text{m}^2$$



**FIGURE 9.16**

(a) An inverted slider–crank mechanism and (b) its corresponding cut open-chain system.

Gravity acts on the system. For the cut system, three joint coordinates are defined as shown. Since the slider–crank has only one DoF, the following values are provided at the initial time:

$$\theta_1 = 345^\circ \text{ (or } -15^\circ\text{)}, \dot{\theta}_1 = \pi \text{ rad/s}$$

Determine the joint accelerations at the initial time.

We solve this problem performing the following steps:

- First, for the cut open-chain system, we construct all the necessary arrays and matrices.
- For the cut-joint, for comparison purposes, we construct the necessary arrays and matrices in two ways:
  - We apply the systematic joint-coordinate formulation, as in Eqs. (9.26)–(9.30), to construct the arrays and matrices associated with the cut-joint.
  - We follow the traditional vector-loop method to construct the cut-joint arrays and matrices that are needed for the equations of motion.

### **Solution (Cut Open-Chain)**

Based on the known initial conditions for  $\theta_1$  and  $\dot{\theta}_1$ , we must determine the initial conditions for the other two joint coordinates. How we determine the initial conditions for the remaining variables will be discussed in Section 9.3.4. At this point, let us assume that the values have been determined as

$$\theta_2 = 0.2149 \text{ rad}, \dot{\theta}_2 = 1.4453 \text{ rad/s}, \theta_3 = 0.4017 \text{ m}, \dot{\theta}_3 = 0.5816 \text{ m/s}$$

### **Example 9.5 (cont.)**

#### **MATLAB/Chapter 9/Example \_ 9 \_ 5**

We state the constant data:

```
r_O = [0.15; 0.33]; L1 = 0.208; L2 = 0.6; L3 = 0.195;
s_O1_local = [-L1/2; 0]; s_B1_local = [L1/2; 0];
s_Q1_local = [-L2/2; 0]; s_B3_local = [0; L3];
u3_local = [1; 0]; uy = [0; 1];
m = [0.2; 1.0; 0.5]; J = [0.004; 0.1; 0.05]; g = 9.81;
```

We also state the initial conditions:

```
theta    = [345*pi/180; 0.2149; 0.4017];
theta_d = [pi; 1.4453; 0.5816];
```

The recursive coordinate formulas are written as

$$\begin{aligned} \phi_1 &= \theta_1; & \mathbf{r}_1 &= \mathbf{r}_0^O - \mathbf{s}_1^O \\ \phi_2 &= \theta_2; & \mathbf{r}_2 &= -\mathbf{s}_2^Q \\ \phi_3 &= \theta_3; & \mathbf{r}_3 &= \theta_3 \mathbf{u}_3 \end{aligned} \tag{a}$$

```
phi1 = theta(1); A1 = A_matrix(phi1);
s_O1 = A1*s_O1_local; r1 = r_O - s_O1;
phi2 = theta(2); A2 = A_matrix(phi2); r2 = -A2*s_Q2_local;
phi3 = phi2; A3 = A2; u3 = A3*u3_local; r3 = theta(3)*u3;
c = [r1; phi1; r2; phi2; r3; phi3];
```

The velocity transformation matrix is constructed as

$$\mathbf{B} = \begin{bmatrix} \mathbf{R}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{2,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{3,2} & \mathbf{T}_{3,3} \end{bmatrix} = \begin{bmatrix} \check{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 \\ \mathbf{0} & \check{\mathbf{d}}_{2,2} & \mathbf{0} \\ 0 & 1 & 0 \\ \mathbf{0} & \check{\mathbf{d}}_{3,2} & \mathbf{u}_3 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \check{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 \\ \mathbf{0} & \check{\mathbf{r}}_2 & \mathbf{0} \\ 0 & 1 & 0 \\ \mathbf{0} & \check{\mathbf{r}}_3 & \mathbf{u}_3 \\ 0 & 1 & 0 \end{bmatrix} \quad (b)$$

```

z21 = [0; 0];
B = [s_rot(-s_O1) z21 z21
      1           0   0
      z21       s_rot(r2) z21
      0           1   0
      z21       s_rot(r3) u3
      0           1   0] ...    B =
      0.0269      0   0
      0.1005      0   0
      1.0000      0   0
      0   -0.0640  0
      0   0.2931  0
      0   1.0000  0
      0   -0.0857  0.9770
      0   0.3925  0.2132
      0   1.0000  0

```

We compute the body velocities:

$$\mathbf{c}_d = \mathbf{B} * \theta_d;$$

The product  $\dot{\mathbf{B}}\dot{\theta}$  is expressed as

$$\dot{\mathbf{B}}\dot{\theta} = \left\{ \begin{array}{l} \dot{\check{\mathbf{d}}}_{1,1}\dot{\theta}_1 \\ 0 \\ \dot{\check{\mathbf{d}}}_{2,2}\dot{\theta}_2 \\ 0 \\ \dot{\check{\mathbf{d}}}_{3,2}\dot{\theta}_2 + \dot{\mathbf{u}}_3\dot{\theta}_3 \\ 0 \end{array} \right\} = \left\{ \begin{array}{l} \dot{\check{\mathbf{r}}}_1\dot{\theta}_1 \\ 0 \\ \dot{\check{\mathbf{r}}}_2\dot{\theta}_2 \\ 0 \\ \dot{\check{\mathbf{r}}}_3\dot{\theta}_2 + \dot{\mathbf{u}}_3\dot{\theta}_3 \\ 0 \end{array} \right\}$$

```

r1_d = c_d(1:2); r2_d = c_d(4:5);
r3_d = c_d(7:8);
u3_d = s_rot(u3)*c_d(9);
Bdtd = [s_rot(r1_d)*theta_d(1)
         0
         s_rot(r2_d)*theta_d(2)
         0
         s_rot(r3_d)*theta_d(2) + ...
         u3_d*theta_d(3)

```

```
0] ..... Bdtd =
-0.9915
0.2657
0
-0.6123
-0.1336
0
-1.1783
1.4636
0
```

The body-coordinate mass matrix is constructed in an array form as

```
M = [m(1) m(1) J(1) m(2) m(2) J(2)];
```

The body-coordinate array of applied loads is computed next:

```
w1 = -m(1)*g*uy; w2 = -m(2)*g*uy; w3 = -m(3)*g*uy;
h_a = [w1; 0; w2; 0; w3; 0];
```

The joint-coordinate mass matrix and array of applied loads are as follows:

<pre>M_joint = B'*diag(M)*B .....</pre> <pre>h_joint = B'*(h_a - diag(M)*Bdtd) ..</pre>	<pre>M_joint = 0.0062 0 0 0 0.3207 0 0 0 0.5000</pre> <pre>h_joint = -0.1971 -5.1380 -0.6264</pre>
--	---

### Solution A (Closed-Chain)

We continue with the solution and construct the necessary arrays and matrices for the cut-joint using Eqs. (9.26)–(9.30).

#### MATLAB/Chapter 9/Example\_9\_5\_a

In the open-chain system, the three joint coordinates are independent. However, when we put the pin joint back between points  $B_1$  and  $B_3$ , the joint coordinates become dependent through the following constraint:

$$(r,2)^* \Phi = \mathbf{r}_1^B - \mathbf{r}_3^B = \mathbf{0} \Rightarrow (r,2)^* \Phi = \mathbf{r}_1 + \mathbf{s}_1^B - \mathbf{r}_3 - \mathbf{s}_3^B = \mathbf{0} \quad (c)$$

These equations are expressed in terms of the body coordinates. If we substitute coordinate transformation expressions in these constraints, they become functions of the joint coordinates. This substitution is performed numerically because the constraints and the transformation expressions are nonlinear in the coordinates.

The body-coordinate velocity constraints for the cut-joint are

$$(r,2)^* \dot{\Phi} = \left[ \begin{array}{cc|cc|cc} \mathbf{I} & \check{\mathbf{s}}_1^B & \mathbf{0} & \mathbf{0} & -\mathbf{I} & -\check{\mathbf{s}}_3^B \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}_1 \\ \dot{\phi}_1 \\ \hline \dot{\mathbf{r}}_2 \\ \dot{\phi}_2 \\ \hline \dot{\mathbf{r}}_3 \\ \dot{\phi}_3 \end{array} \right\} = \mathbf{0}$$

This provides the Jacobian matrix as

$${}^*D = \left[ \begin{array}{c|cc|cc|cc} I & \check{s}_1^B & 0 & 0 & -I & -\check{s}_3^B \end{array} \right]$$

We can now compute the  $C$  matrix as  ${}^*DB$ .

```
s_B3 = A3*s_B3_local; s_B1 = A1*s_B1_local;
Dstar = [eye(2) s_rot(s_B1) zeros(2,3) ...
          -eye(2) -s_rot(s_B3)] .....
Dstar =
1.0000      0  0.0269    0    0    0 -1.0000      0  0.1905
      0  1.0000  0.1005    0    0    0      0 -1.0000  0.0416
C = Dstar*B .....
C =
0.0538    0.2762   -0.9770
0.2009   -0.3509   -0.2132
```

We construct the right-hand side of the acceleration constraints as

$$\begin{aligned} -\dot{C}\dot{\theta} &= -({}^*DB + {}^*DB)\dot{\theta} \\ &= -({}^*DB\dot{\theta} + {}^*DB\dot{\theta}) \end{aligned}$$

Since we have already computed the product  $\dot{B}\dot{\theta}$ , we only need to compute  ${}^*\dot{D}$ , which is

$$\begin{aligned} {}^*\dot{D} &= \left[ \begin{array}{c|cc|cc|cc} 0 & \dot{s}_1^B & 0 & 0 & 0 & -\dot{s}_3^B \end{array} \right] \\ &= \left[ \begin{array}{c|cc|cc|cc} 0 & -s_1^B \dot{\phi}_1 & 0 & 0 & 0 & s_3^B \dot{\phi}_3 \end{array} \right] \end{aligned}$$

```
Dstar_d = [zeros(2) -s_B1*c_d(3) zeros(2,3) ...
           zeros(2) s_B3*c_d(9)];
Cdtd = D*Bdtd + Dstar_d*B*theta_d .....
Cdtd =
-0.8915
-0.5343
```

We now construct the coefficient matrix and the right-hand side array for Eq. (9.34):

```
CMC = [M_joint -C'
       C zeros(2)] ..
CMC =
0.0062      0      0 -0.0538 -0.2009
      0  0.3207      0 -0.2762  0.3509
      0      0  0.5000  0.9770  0.2132
0.0538  0.2762 -0.9770      0      0
0.2009 -0.3509 -0.2132      0      0
rhs = [h_joint
       -Cdtd] .....
rhs =
-0.1971
-5.1380
-0.6264
0.8915
0.5343
```

We can now solve for the joint accelerations and Lagrange multipliers:

```

sol = CMC\rhs;
theta_dd = sol(1:3) .....
LagMult = sol(4:5) .....

```

theta_dd =
-26.8578
-13.1827
-6.1188
LagMult =
2.6085
-0.5418

### Example 9.5 (cont.)

#### Solution B (Closed-Chain):

We construct the necessary arrays and matrices for the cut-joint using the vector-loop constraints as in Chapter 5.

#### MATLAB/Chapter 9/Example \_ 9 \_ 5 \_ b

For the closed chain, as shown in Figure 9.17, a vector-loop equation and its first time derivative are expressed as

$$\begin{aligned} \mathbf{r}^O + L_1 \mathbf{u}_1 - L_3 \check{\mathbf{u}}_3 - \theta_3 \mathbf{u}_3 &= \mathbf{0} \\ L_1 \check{\mathbf{u}}_1 \dot{\theta}_1 + L_3 \mathbf{u}_3 \dot{\theta}_2 - \mathbf{u}_3 \dot{\theta}_3 - \theta_3 \check{\mathbf{u}}_3 \dot{\theta}_2 &= \mathbf{0} \end{aligned}$$

We extract the **C** matrix from the velocity constraint as

$$\mathbf{C} = \begin{bmatrix} L_1 \check{\mathbf{u}}_1 & (L_3 \mathbf{u}_3 - \theta_3 \check{\mathbf{u}}_3) & -\mathbf{u}_3 \end{bmatrix}$$

The product  $\dot{\mathbf{C}}\dot{\theta}$  is expressed as

$$\dot{\mathbf{C}}\dot{\theta} = -L_1 \mathbf{u}_1 \dot{\theta}_1^2 + (L_3 \check{\mathbf{u}}_3 + \theta_3 \mathbf{u}_3) \dot{\theta}_2^2 - 2 \check{\mathbf{u}}_3 \dot{\theta}_2 \dot{\theta}_3$$

```

u1 = [cos(theta(1)); sin(theta(1))];
u2 = [cos(theta(2)); sin(theta(2))];
u1r = s_rot(u1); u3r = s_rot(u3);
C = [L1*u1r (L3*u3-theta(3)*u3r) -u3] .

```

```

Cdtd = -L1*u1*theta_d(1)^2 + ...
(L3*u3r + theta(3)*u3)* ...
theta_d(2)^2 - ...
2*u3r*theta_d(2)*theta_d(3) ...

```

```

C =
0.0538 0.2762 -0.9770
0.2009 -0.3509 -0.2132

```

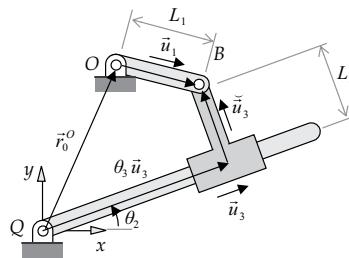
```

Cdtd =
0.8915
-0.5343

```

We note that matrix **C** and the product  $\dot{\mathbf{C}}\dot{\theta}$  are exactly the same as those evaluated with the systematic joint-coordinate formulation.

We now construct the coefficient matrix and the right-hand side array. Then we solve for the joint accelerations and Lagrange multipliers.

**FIGURE 9.17**

Vector loop representation of the cut-joint.

```

CMC = [M_joint -C'
       C zeros(2)];
rhs = [h_joint; -Cdtd];
sol = CMC\rhs;
theta_dd = sol(1:3)
.....
LagMult = sol(4:5)
.....

```

```

theta_dd =
-26.8578
-13.1827
-6.1188
LagMult =
2.6085
-0.5418

```

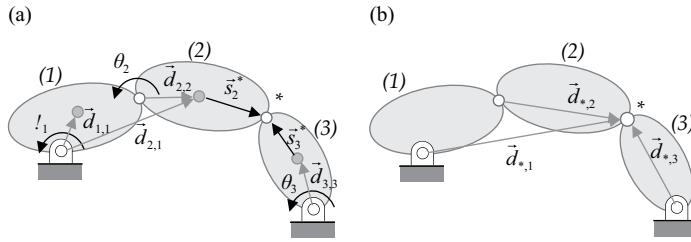
This example has illustrated that the entities associated with a cut-joint, such as matrix  $\mathbf{C}$  and the product  $\dot{\mathbf{C}}\dot{\theta}$ , can be constructed either by following the joint-coordinate process or by the traditional vector-loop method. The joint-coordinate process is well suited for general-purpose program development, whereas the vector-loop method is more adequate for small-scale problems.

### 9.3.3 Jacobian Matrix

We have shown that the Jacobian matrix  $\mathbf{C}$  for a loop associated with a cut-joint can be obtained numerically from the product of  ${}^*\mathbf{D}$  and  $\mathbf{B}$  matrices, as stated in Eq. (9.28). Since both these matrices contain a large number of zero entries, for systems with a large number of bodies, the numerical process may not be very efficient. Therefore, a more direct construction and evaluation of the  $\mathbf{C}$  matrix (and  $\dot{\mathbf{C}}\dot{\theta}$ ) would be desirable. In this section, we show that an analytical product of  ${}^*\mathbf{D}$  and  $\mathbf{B}$  can reveal a process for constructing  $\mathbf{C}$  directly based on the topology of a loop. This direct process will be demonstrated through several simple examples.

Let us consider the system shown in Figure 9.18a, which schematically presents a four-bar mechanism. The system's only loop is cut at the pin joint marked by  $*$ , and the three joint coordinates are defined for the remaining pin joints. The  $\mathbf{B}$  matrix and the cut-joint Jacobian matrix,  ${}^*\mathbf{D}$ , can be expressed as

$$\mathbf{B} = \begin{bmatrix} \mathbf{R}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{2,1} & \mathbf{R}_{2,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{3,3} \end{bmatrix} = \begin{bmatrix} \check{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 \\ \check{\mathbf{d}}_{2,1} & \check{\mathbf{d}}_{2,2} & \mathbf{0} \\ 1 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \check{\mathbf{d}}_{3,3} \\ 0 & 0 & 1 \end{bmatrix} \quad (a.1)$$

**FIGURE 9.18**

(a) Schematic presentation of a four-bar mechanism and (b) the required vectors for a cut pin joint.

$${}^* \mathbf{D} = \left[ \begin{array}{cc|cc|cc} {}^* \mathbf{0} & {}^* \mathbf{0} & \mathbf{I} & \vec{\mathbf{s}}_2^* & -\mathbf{I} & -\vec{\mathbf{s}}_3^* \end{array} \right] \quad (\text{a.2})$$

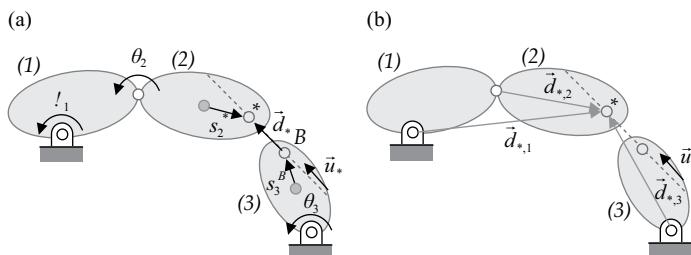
The  $\mathbf{C}$  matrix can be obtained as

$$\begin{aligned} \mathbf{C} = {}^* \mathbf{D} \mathbf{B} &= \left[ \begin{array}{ccc} \check{\mathbf{d}}_{2,1} + \check{\mathbf{s}}_1^* & \check{\mathbf{d}}_{2,2} + \check{\mathbf{s}}_2^* & -\check{\mathbf{d}}_{3,3} - \check{\mathbf{s}}_3^* \end{array} \right] \\ &= \left[ \begin{array}{ccc} \check{\mathbf{d}}_{*,1} & \check{\mathbf{d}}_{*,2} & -\check{\mathbf{d}}_{*,3} \end{array} \right] \end{aligned} \quad (\text{a.3})$$

Vectors  $\vec{d}_{*,1}$ ,  $\vec{d}_{*,2}$ , and  $\vec{d}_{*,3}$  are shown in Figure 9.18b. These vectors connect the *reference* points of each joint coordinate to the *reference* point of the cut-joint. The sign of each vector refers to its sign in the pin joint constraint,  $\vec{r}_2^* - \vec{r}_3^* = \vec{0}$ , that led to the Jacobian matrix  ${}^* \mathbf{D}$ .

As a second example, we consider the system shown in Figure 9.19a where the cut-joint is translational. The  $\mathbf{B}$  matrix for this system is the same as in Eq. (a.1), and the Jacobian matrix for the cut-joint is

$${}^* \mathbf{D} = \left[ \begin{array}{cc|cc|cc} {}^* \mathbf{0} & {}^* \mathbf{0} & \mathbf{0} & 1 & 0 & -1 \\ {}^* \mathbf{0} & {}^* \mathbf{0} & \check{\mathbf{u}}' & \check{\mathbf{u}}' \vec{\mathbf{s}}_2^* & -\check{\mathbf{u}}' & -\check{\mathbf{u}}' (\vec{\mathbf{s}}_3^B + \check{\mathbf{d}}_*) \end{array} \right] \quad (\text{b.1})$$

**FIGURE 9.19**

(a) Schematic presentation of an inverted slider-crank mechanism and (b) the required vectors for a cut sliding joint.

The **C** matrix becomes

$$\begin{aligned} \mathbf{C} = {}^*\mathbf{DB} &= \begin{bmatrix} 1 & 1 & -1 \\ \bar{\mathbf{u}}'(\bar{\mathbf{d}}_{2,1} + \bar{\mathbf{s}}_2^*) & \bar{\mathbf{u}}'(\bar{\mathbf{d}}_{2,2} + \bar{\mathbf{s}}_2^*) & -\bar{\mathbf{u}}'(\bar{\mathbf{d}}_{3,3} + \bar{\mathbf{s}}_3^B + \bar{\mathbf{d}}_*) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & -1 \\ \bar{\mathbf{u}}^*\bar{\mathbf{d}}_{*,1} & \bar{\mathbf{u}}^*\bar{\mathbf{d}}_{*,2} & -\bar{\mathbf{u}}^*\bar{\mathbf{d}}_{*,3} \end{bmatrix} \end{aligned} \quad (\text{b.2})$$

where, similar to the first example, a  $\vec{d}_{*,i}$  vector connects the reference point of its joint coordinate to the reference point of the cut-joint, as depicted in Figure 9.19b.

In the example shown in Figure 9.20a, we cut a pin joint and keep the translational joint in the cut system. The  ${}^*\mathbf{D}$  matrix for the cut-joint is the same as in Eq. (a.2), but the **B** matrix has become

$$\mathbf{B} = \begin{bmatrix} \mathbf{R}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{2,1} & \mathbf{T}_{2,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{3,3} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ \bar{\mathbf{d}}_{2,1} & \mathbf{u}_2 & \mathbf{0} \\ \bar{\mathbf{d}}_{3,3} & \mathbf{0} & \bar{\mathbf{d}}_{3,3} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \quad (\text{c.1})$$

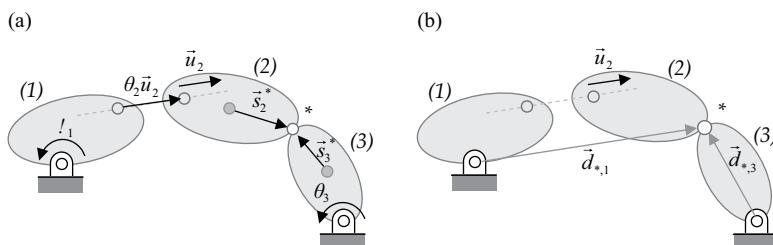
The **C** matrix is found to be

$$\mathbf{C} = {}^*\mathbf{DB} = \begin{bmatrix} (\bar{\mathbf{d}}_{2,1} + \bar{\mathbf{s}}_2^*) & \mathbf{u}_2 & -(\bar{\mathbf{d}}_{3,3} + \bar{\mathbf{s}}_3^*) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{d}}_{*,1} & \mathbf{u}_2 & -\bar{\mathbf{d}}_{*,3} \end{bmatrix} \quad (\text{c.2})$$

As illustrated in Figure 9.20b, there is no  $\vec{d}_{*,2}$  vector associated with the translational joint coordinate  $\theta_2$ .

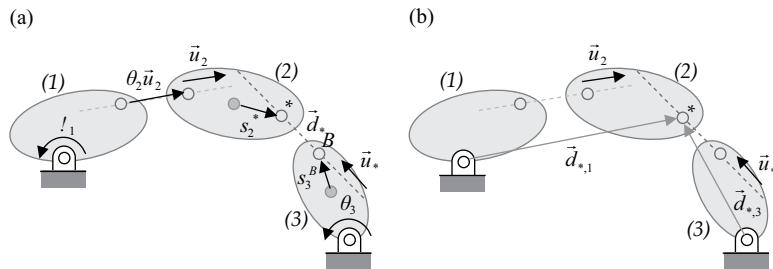
In this final example, there are two sliding joints in the closed chain where one of them is cut, as shown in Figure 9.21. For this system, the **B** matrix is the same as in Eq. (c.1) and the  ${}^*\mathbf{D}$  matrix is the same as in Eq. (b.1). The **C** matrix is determined to be

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & -1 \\ \bar{\mathbf{u}}'(\bar{\mathbf{d}}_{2,1} + \bar{\mathbf{s}}_2^*) & \bar{\mathbf{u}}'\mathbf{u}_2 & -\bar{\mathbf{u}}'(\bar{\mathbf{d}}_{3,3} + \bar{\mathbf{s}}_3^B + \bar{\mathbf{d}}_*) \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ \bar{\mathbf{u}}^*\bar{\mathbf{d}}_{*,1} & \bar{\mathbf{u}}^*\mathbf{u}_2 & -\bar{\mathbf{u}}^*\bar{\mathbf{d}}_{*,3} \end{bmatrix} \quad (\text{d.1})$$



**FIGURE 9.20**

(a) Schematic presentation of an inverted slider-crank mechanism and (b) the required vectors for a cut pin joint.

**FIGURE 9.21**

(a) Schematic presentation of a double slider mechanism and (b) the required vectors for a cut sliding joint.

**TABLE 9.2**

Entries of the C Matrix for Two Commonly Used Joints

Joint coordinate	Cut-Joint	
	Revolute	Translational
Revolute	$[\check{\mathbf{d}}_{*,i}]$	$\begin{bmatrix} 1 \\ \check{\mathbf{u}}' \check{\mathbf{d}}_{*,i} \end{bmatrix}$
Translational	$[\mathbf{u}_i]$	$\begin{bmatrix} 0 \\ \check{\mathbf{u}}' \mathbf{u}_i \end{bmatrix}$

An observation of the **C** matrices for these four examples reveals a clear pattern in how the **C** matrix could be constructed directly based on the type of the cut-joint and the joint coordinates of the loop. We summarize our observation in Table 9.2 for the entries of a **C** matrix. The “+” or “−” sign for an entry refers to the original body-coordinate constraint equation for the cut-joint.

If we construct the **C** matrix directly, we can also directly determine the term  $\dot{\mathbf{C}}\dot{\theta}$ . Direct construction and evaluation of the **C** matrix and  $\dot{\mathbf{C}}\dot{\theta}$  yield computational efficiency compared to the numerical evaluation of Eqs. (9.28) and (9.30).

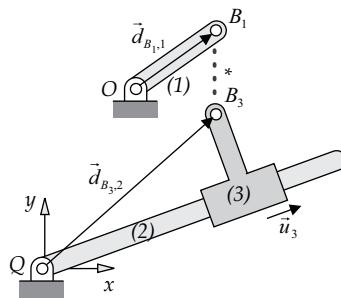
### Example 9.6

In this exercise, we construct the **C** matrix for the inverted slider–crank mechanism of Example 9.5 using the process that is summarized in Table 9.2.

As shown in Figure 9.22, since the cut-joint at  $B$  is a revolute joint, we need to define two vectors from the other two revolute joints to points  $B_1$  and  $B_3$  and. There is no need to define any such vector for the translational joint. With these vectors, the **C** matrix can be constructed as

$$\mathbf{C} = \begin{bmatrix} \check{\mathbf{d}}_{B_1,1} & -\check{\mathbf{d}}_{B_3,2} & -\mathbf{u}_3 \end{bmatrix}$$

We note that the positive and negative signs refer to the two branches of the loop that form the revolute joint constraint,  $\vec{r}_1^B - \vec{r}_3^B = \vec{0}$ . Having the **C** matrix in analytical form, we can construct the right-hand side of the acceleration constraints as



**FIGURE 9.22**  
Defining  $\vec{d}_{B_j,j}$  vectors for matrix  $C$ .

$$-\dot{C}\dot{\theta} = - \left[ \begin{array}{ccc} \dot{\mathbf{d}}_{B_1,1} & -\dot{\mathbf{d}}_{B_3,2} & -\dot{\mathbf{u}}_3 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{array} \right\}$$

### 9.3.4 Initial Conditions

The values of the joint coordinates within a closed chain must satisfy the vector-loop constraints at any given time, including the initial time. The same is true for the joint velocities within a loop that must satisfy the velocity constraints associated with that loop. Assuring that the initial conditions for the joint coordinates and velocities satisfy the constraints is crucial before beginning to integrate the equations of motion forward in time.

If a correct set of initial conditions is not available, we must solve the constraints first for the coordinates and then for the velocities. For this purpose, we need to start with known initial conditions for as many joint coordinates and velocities as the number of DoFs within that loop. This process requires solving a set of nonlinear algebraic equations for the coordinates and linear algebraic equations for the velocities.

#### Example 9.7

In Example 9.5, the initial conditions for only one of the three joint coordinates were given as

$$\theta_1 = 345^\circ \text{ (or } -15^\circ\text{)}, \dot{\theta}_1 = \pi \text{ rad/s}$$

We assumed that the initial conditions for the remaining two joint coordinates were

$$\theta_2 = 0.2149 \text{ rad}, \dot{\theta}_2 = 1.4453 \text{ rad/s}, \theta_3 = 0.4017 \text{ m}, \dot{\theta}_3 = 0.5816 \text{ m/s}$$

How did we find these initial conditions?

#### Solution

To determine  $\theta_2$  and  $\theta_3$ , we must solve the following vector-loop constraints knowing the value of  $\theta_1$ :

$$\mathbf{r}^O + L_1 \mathbf{u}_1 - L_3 \tilde{\mathbf{u}}_3 - \theta_3 \mathbf{u}_3 = \mathbf{0}$$

Then we solve the velocity constraint for  $\dot{\theta}_2$  and  $\dot{\theta}_3$ , knowing  $\dot{\theta}_1$ :

$$\left[ \begin{array}{ccc} L_1 \ddot{\mathbf{u}}_1 & (L_3 \mathbf{u}_3 - \theta_3 \ddot{\mathbf{u}}_3) & -\mathbf{u}_3 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{array} \right\} = \mathbf{0}$$

The following MATLAB program is similar to the programs developed in Chapter 5.

#### MATLAB/Chapter 9/Example \_ 9 \_ 7

We state the constant data, the initial conditions, and the estimates for  $\theta_2$  and  $\theta_3$ :

```
global r_O L1 L3
r_O = [0.15; 0.33]; L1 = 0.208; L3 = 0.195;
theta1 = 345*pi/180; theta1_d = pi; theta23 = [0.2; 0.4];
```

We use `fsolve` to solve the nonlinear constraint equations. We direct `fsolve` to find the equations in the M-file function `s_c`.

```
options = optimset('display', 'off');
coords = fsolve(@s_c, theta23, options, theta1);
angles_deg = coords*180/pi;
theta2 = coords(1) ..... theta2 =
theta3 = coords(2) ..... theta3 =
0.2149
0.4017
```

Next we compute the joint velocities. We treat the problem as three equations in three unknowns.

```
u1 = [cos(theta1); sin(theta1)];
u3 = [cos(theta2); sin(theta2)]; u3r = s_rot(u3);
C = [L1*s_rot(u1) (L3*u3 - theta3*u3r) -u3
      1 0 0];
rhs_v = [0; 0; theta1_d];
velocities = C\rhs_v;
theta2_d = velocities(2) ..... theta2_d =
theta3_d = velocities(3) ..... theta3_d =
1.4453
0.5816
```

The following is a listing of the M-file function `s_c`:

```
function Phi = s_c(x, theta1)
global r_O L1 L3
u1 = [cos(theta1); sin(theta1)];
u3 = [cos(x(1)); sin(x(1))];
Phi = r_O + L1*u1 - L3*s_rot(u3) - x(2)*u3;
```

### 9.3.5 Reaction Forces

The solution of Eq. (9.34) provides values of the joint accelerations and Lagrange multipliers associated with the cut-joints. We can relate the obtained values of the

Lagrange multipliers to the reaction forces and/or torques at the cut-joints in two different ways. If we consider the body-coordinate equations of motion from Eq. (9.32),  $\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + \mathbf{D}'^*\boldsymbol{\lambda}$ , the product  ${}^*\mathbf{D}'^*\boldsymbol{\lambda}$  provides the reaction forces or torques for the cut-joints, exactly as we depict them on a free-body diagram. However, if we consider the joint-coordinate equations of motion from Eq. (9.33),  $\mathbf{M}\ddot{\boldsymbol{\theta}} = {}^{(a)}\mathbf{h} + \mathbf{C}'^*\boldsymbol{\lambda}$ , the product  $\mathbf{C}'^*\boldsymbol{\lambda}$  provides the reaction forces and/or torques as they act on every joint coordinate. The following example should clarify the difference between the two interpretations.

### Example 9.5 (cont.)

In Example 9.5, we constructed and solved the equations of motion for the joint accelerations and Lagrange multipliers, where the Lagrange multipliers at the cut-joint  $B$  were found to be

$${}^*\boldsymbol{\lambda} = \begin{Bmatrix} 2.6085 & -0.5418 \end{Bmatrix}'$$

Let us investigate how these multipliers lead to determining the reaction forces and torques in the equations of motion as  $\mathbf{C}'^*\boldsymbol{\lambda}$  or  $\mathbf{B}'^*\mathbf{D}'^*\boldsymbol{\lambda}$ . Let us examine the product  ${}^*\mathbf{D}'^*\boldsymbol{\lambda}$  first.

**MATLAB/Chapter 9/Example \_ 9 \_ 5 \_ c** (An extension to Example \_ 9 \_ 5 \_ a)

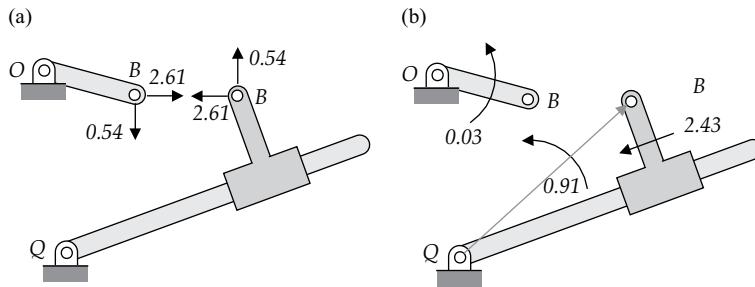
<pre>fReact = Dstar'*LagMult .....</pre>	<pre>fReact =  2.6085  -0.5418  0.0158  0  0  -2.6085  0.5418  0.4744</pre>
--	---

Here we notice that the reaction forces are applied on bodies (1) and (3) as shown in Figure 9.23a.

Next, we examine the product  $\mathbf{C}'^*\boldsymbol{\lambda}$  or  $\mathbf{B}'^*\mathbf{D}'^*\boldsymbol{\lambda}$ . For this purpose, we transform the computed array of forces  $f_{\text{react}}$  to the joint-coordinate space through matrix  $\mathbf{B}$ :

<pre>fReactJoint = B'*fReact .....</pre>	<pre>fReactJoint =  0.0316  0.9105  -2.4330</pre>
--	---

The first component of this array is the moment of the reaction force (1) about point  $O$ ; the second component is the moment of the reaction force on the combined bodies (2) and (3) about point  $Q$ ; and the third component is the projection of the reaction force along the axis of the sliding joint, which is the joint coordinate  $\theta_3$ . This interpretation of the reaction forces is illustrated in Figure 9.23b.

**FIGURE 9.23**

Two interpretations of the reaction force: (a) acting at cut-joint  $B$  and (b) acting as force/torque on bodies.

### 9.3.6 Driver Constraint

Similar to introducing a driver constraint in the body-coordinate equations of motion, we can apply a driver constraint on a joint coordinate directly in the joint-coordinate equations of motion. Assume that the joint coordinate  $\theta_i$  is controlled by a driver constraint as

$${}^{(dr)}\Phi = \theta_i - f(t) = 0 \quad (9.35)$$

where  $f(t)$  is a defined continuous function. This constraint should be appended to the cut-joint constraints of Eq. (9.26) as

$$\left\{ \begin{array}{l} {}^*\Phi(\mathbf{c}) \\ {}^{(dr)}\Phi \end{array} \right\} = \left\{ \begin{array}{l} {}^*\Phi(\mathbf{c}(\theta)) \\ \theta_i - f(t) \end{array} \right\} = \mathbf{0} \quad (9.36)$$

This leads to the following velocity and acceleration constraints:

$$\left\{ \begin{array}{l} {}^*\dot{\Phi}(\mathbf{c}) \\ {}^{(dr)}\dot{\Phi} \end{array} \right\} = \left\{ \begin{array}{l} \mathbf{C}\dot{\theta} \\ \dot{\theta}_i - \dot{f}(t) \end{array} \right\} = \mathbf{0} \Rightarrow \left\{ \begin{array}{l} \mathbf{C}\dot{\theta} \\ \dot{\theta}_i \end{array} \right\} = \left\{ \begin{array}{l} \mathbf{0} \\ \dot{f}(t) \end{array} \right\} \quad (9.37)$$

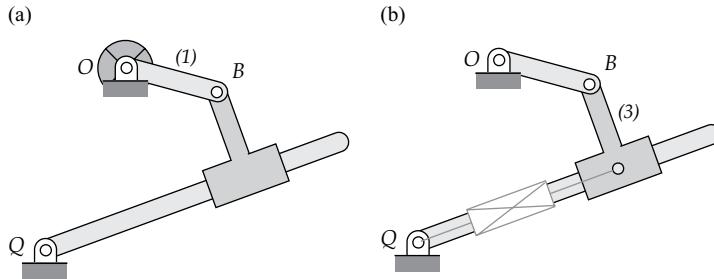
$$\left\{ \begin{array}{l} {}^*\ddot{\Phi} \\ {}^{(dr)}\ddot{\Phi} \end{array} \right\} = \left\{ \begin{array}{l} \mathbf{C}\ddot{\theta} + \dot{\mathbf{C}}\dot{\theta} \\ \ddot{\theta}_i - \ddot{f}(t) \end{array} \right\} = \mathbf{0} \Rightarrow \left\{ \begin{array}{l} \mathbf{C}\ddot{\theta} \\ \ddot{\theta}_i \end{array} \right\} = \left\{ \begin{array}{l} -\dot{\mathbf{C}}\dot{\theta} \\ \ddot{f}(t) \end{array} \right\} \quad (9.38)$$

It should be clear that a driver constraint adds one more row to the Jacobian matrix with a "1" entry in the column associated with the driver joint coordinate. We should note how the terms  $\dot{f}(t)$  and  $\ddot{f}(t)$  appear in the right-hand side arrays of Eqs. (9.37) and (9.38).

#### Example 9.8

For the slider–crank mechanism of Example 9.5, assume the following two cases:

- A driver motor acts about the pin joint  $O$ , as shown in Figure 9.24a, where the driver constraint is stated as  $\theta_1 = {}^{(0)}\theta_1 + \pi t$ .
- A linear driver actuator acts along the axis of the sliding joint between point  $Q$  and the mass center of link (3), as shown in Figure 9.24b, where the driver constraint is  $\theta_3 = {}^{(0)}\theta_3 + 0.1\sin(\pi t)$ .

**FIGURE 9.24**

(a) A rotational driver-motor acts between link (1) and the ground; (b) a linear driver-actuator acts along the axis of the sliding joint.

For each case, determine the necessary changes to the arrays and matrices of the equations of motion.

### *Solution*

- a. The first and second time derivatives of the driver constraint are as follows:

$$\dot{\theta}_1 = \pi \text{ rad/s}, \ddot{\theta}_1 = 0$$

Since the joint coordinates for this system are  $\theta_1$ ,  $\theta_2$ , and  $\theta_3$ , the Jacobian matrix associated with this driver constraint adds one more row to the  $\mathbf{C}$  matrix as

$${}^{(dr)}\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

The right-hand side of the acceleration constraint is zero, and therefore, we need to append a zero to the  $-\dot{\mathbf{C}}\dot{\theta}$  array of the cut-joint. In addition, the initial conditions on the three joint coordinates and three joint velocities must be adjusted to agree with the values that are specified by the driver function.

- b. The first and second time derivatives of the driver constraint are as follows:

$$\dot{\theta}_3 = 0.1\pi \cos(\pi t) \text{ m/s}, \ddot{\theta}_3 = -0.1\pi^2 \sin(\pi t) \text{ m/s}^2$$

The corresponding Jacobian matrix is

$${}^{(dr)}\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

This matrix should be appended to the cut-joint Jacobian matrix  $\mathbf{C}$ . We also need to append the right-hand side of the driver constraints, that is,  $-0.1\pi^2 \sin(\pi t)$ , to the cut-joint  $-\dot{\mathbf{C}}\dot{\theta}$  array. In addition, the initial conditions for the three joint coordinates and three joint velocities must be adjusted to agree with the values that are specified by the driver function.

## 9.4 A MATLAB Program

Based on the joint-coordinate formulation that is discussed in this chapter, a MATLAB program has been developed named DAP\_JC (dynamic analysis program with joint

coordinates). This program and its accompanying user manual (Appendix C) can be downloaded from the textbook's website. Parts of this program have similarities with, and some parts are identical to, the program DAP\_BC. The main difference is that DAP\_JC does not generate all the necessary equations, arrays, and matrices as it is done automatically in DAP\_BC. In DAP\_JC, the user must provide M-files containing most of the equations, arrays, and matrices that describe a model. Therefore, for using this program, the user must first derive all the necessary components of a model by hand before formulating them in the designated M-files.

Whether a system is open chain or closed chain, the user must provide an M-file for

- Recursive forward kinematics of the body coordinates
- Velocity transformation matrix  $\mathbf{B}$
- Array  $\dot{\mathbf{B}}\theta$

For closed-chain systems, the user must provide another M-file for

- Cut-joint constraint(s)
- Cut-joint Jacobian matrix  $\mathbf{C}$
- Array  $\dot{\mathbf{C}}\theta$

For the remaining arrays and matrices, such as the array of applied forces and the mass matrix, the program generates them based on the user's supplied data similar to that in DAP\_BC. Having DAP\_JC to share similar organization and array structure with DAP\_BC allows the two programs to use the same post-processor and animation programs with minor modifications.

Although the computational efficiency of an analysis with the joint-coordinate formulation should be much better (almost an order of magnitude) than that of the body-coordinate formulation, the user may not notice any such differences between the two programs. The reason is that DAP\_JC is mainly developed as a tool to assist in better understanding the concept of the joint-coordinate formulation. However, if the program is revised to automatically generate all the necessary elements for a model, it can be made to be computationally much more efficient.

## 9.5 Problems

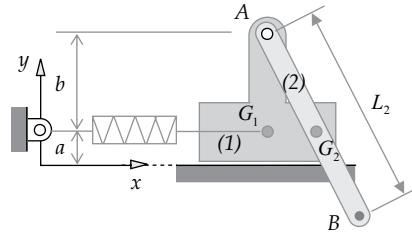
For the open-chain systems in Problems 9.1–9.6, consider the gravity as an applied force.

- a. Define the necessary joint coordinates, and determine the coordinate transformation expressions, the  $\mathbf{B}$  matrix, and the  $\dot{\mathbf{B}}\theta$  array. For a pin joint, consider the global joint coordinate  $\theta_i = \phi_i$ .
- b. Determine the array of forces and the mass matrix.
- c. Construct the equations of motion.

Extract any necessary angle by direct measurements from the figure.

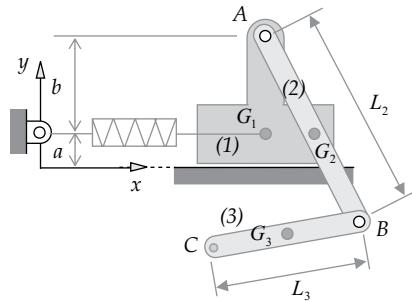
9.1 Consider the following data:

$$a = 0.01, b = 0.03, L_2 = 0.06 \text{ m}; k = 100 \text{ N/m}, {}^0L = 0.08, {}^{def}L = 0.07 \text{ m}; \\ m_1 = 0.3, m_2 = 0.2 \text{ kg}, J_1 = 0.05, J_2 = 0.001 \text{ kg} \cdot \text{m}^2$$



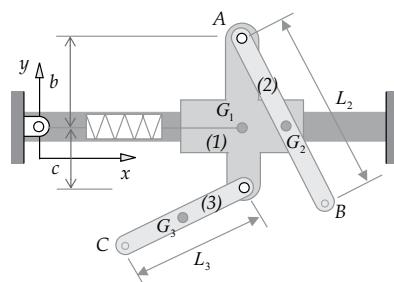
9.2 Consider the following data:

$$a = 0.01, b = 0.03, L_2 = 0.06, L_3 = 0.04 \text{ m}; k = 100 \text{ N/m}, {}^0L = 0.08, {}^{def}L = 0.07 \text{ m}; \\ m_1 = 0.3, m_2 = 0.2, m_3 = 0.15 \text{ kg}, J_1 = 0.05, J_2 = 0.001, J_3 = 0.0008 \text{ kg} \cdot \text{m}^2$$



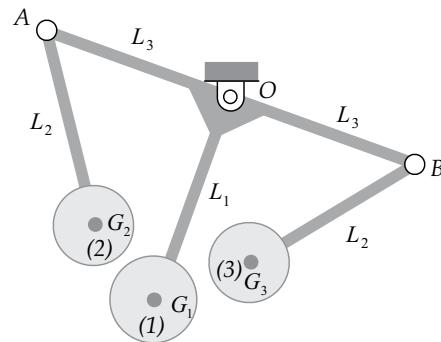
9.3 Consider the following data:

$$b = 0.03, c = 0.02, L_2 = 0.06, L_3 = 0.04 \text{ m}; k = 100 \text{ N/m}, {}^0L = 0.08, {}^{def}L = 0.07 \text{ m}; \\ m_1 = 0.3, m_2 = 0.2, m_3 = 0.15 \text{ kg}, J_1 = 0.05, J_2 = 0.001, J_3 = 0.0008 \text{ kg} \cdot \text{m}^2$$



9.4 Consider the following data:

$$L_1 = 0.2, L_2 = L_3 = 0.15 \text{ m}; m_1 = 0.4, m_2 = m_3 = 0.15 \text{ kg}, J_1 = 0.001, J_2 = J_3 = 0.001 \text{ kg} \cdot \text{m}^2$$

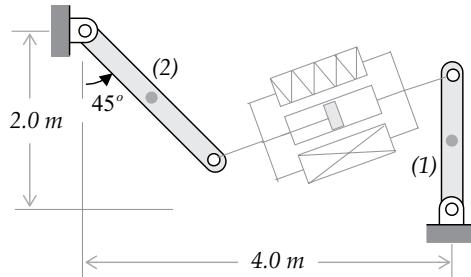


9.5 Consider the following data:

$$L_1 = 1.5, L_2 = 2.0 \text{ m}; k = 100 \text{ N/m}, {}^0L = 2.0 \text{ m}, d_c = 200 \text{ N}\cdot\text{s/m}, {}^{(a)}f = 30 \text{ N};$$

$$m_1 = 0.8, m_2 = 1.2 \text{ kg}, J_1 = 0.15, J_2 = 0.40 \text{ kg}\cdot\text{m}^2;$$

$$\dot{\phi}_1 = 1.0 \text{ rad/s CW}, \dot{\phi}_2 = 2.5 \text{ rad/s CCW}$$



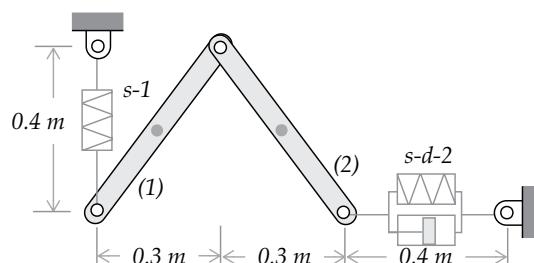
9.6 Consider the following data:

$$L_1 = L_2 = 0.5 \text{ m}; k_1 = 20 \text{ N/m}, {}^0L_1 = 0.5 \text{ m};$$

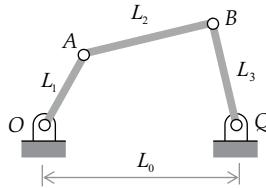
$$k_2 = 30 \text{ N/m}, {}^0L_2 = 0.45 \text{ m}, d_{c2} = 6.0 \text{ N}\cdot\text{s/m};$$

$$m_1 = m_2 = 0.6 \text{ kg}, J_1 = J_2 = 0.125 \text{ kg}\cdot\text{m}^2;$$

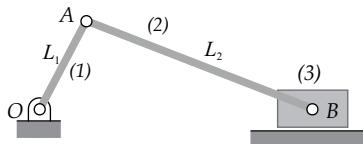
$$\dot{x}_1 = 0.05, \dot{y}_1 = -0.02 \text{ m/s}, \dot{\phi}_1 = 0.06, \dot{\phi}_2 = -0.03 \text{ rad/s}$$



- 9.7 Repeat Problem 9.2, but for the pin joint at  $B$ , consider a relative joint coordinate  $\theta_i = \phi_i - \phi_{i-1}$ .
- 9.8 Repeat Problem 9.4, but for the pin joints at  $A$  and  $B$ , consider relative joint coordinates.
- 9.9 Repeat Problem 9.6, but for the pin joint, consider a relative joint coordinate.
- 9.10 For this four-bar mechanism,  $L_1 = 0.1$ ,  $L_2 = 0.15$ ,  $L_3 = 0.12$ , and  $a = 0.14$  (all in meters). Define the necessary joint coordinates, assuming absolute coordinates  $\theta_i = \phi_i$ . Determine the coordinate transformation expressions, the  $\mathbf{B}$  matrix, the  $\dot{\mathbf{B}}\dot{\theta}$  array, the cut-joint constraints, the Jacobian matrix, and the right-hand side array of the acceleration constraints. Consider the following cases for the choice of the cut-joint:
- Pin joint at  $A$
  - Pin joint at  $B$
  - Pin joint at  $Q$

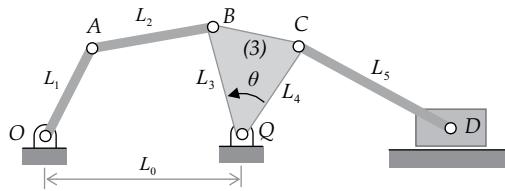


- 9.11 In Problem 9.10, consider relative joint coordinates  $\theta_i = \phi_i - \phi_{i-1}$ .
- 9.12 For this slider–crank mechanism,  $L_1 = 0.1$  m and  $L_2 = 0.2$  m. Define the necessary joint coordinates, assuming absolute coordinates  $\theta_i = \phi_i$ . Determine the coordinate transformation expressions, the  $\mathbf{B}$  matrix, the  $\dot{\mathbf{B}}\dot{\theta}$  array, the cut-joint constraints, the Jacobian matrix, and the right-hand side array of the acceleration constraints. Consider the following cases for the choice of the cut-joint:
- Pin joint at  $A$
  - Sliding joint
  - Revolute-revolute joint between  $A$  and  $B$
  - Revolute-translational joint between  $B$  and the ground (eliminate the block as a body)

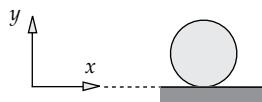


- 9.13 For this six-bar mechanism, select two cut-joints and define the necessary joint coordinates. Determine the coordinate transformation expressions, the  $\mathbf{B}$  matrix, the  $\dot{\mathbf{B}}\dot{\theta}$  array, the cut-joint constraints, the Jacobian matrix, and the right-hand side array of the acceleration constraints.

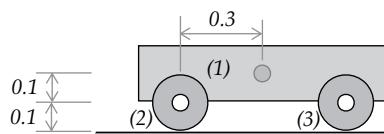
$$L_0 = 2.0, L_1 = 1.0, L_2 = 1.5, L_3 = 1.2, L_4 = 1.0, L_5 = 2.0 \text{ (all in meters); } \theta = 30^\circ$$



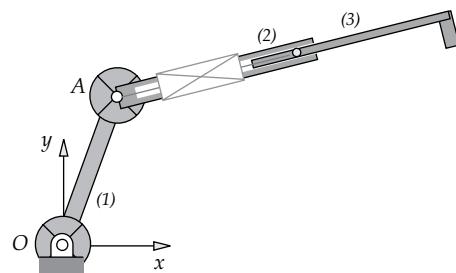
- 9.14 Repeat Problem 9.13 for the six-bar quick-return mechanism from Section 15.3.
- 9.15 Repeat Problem 9.13 for the six-bar quick-return mechanism from Section 15.4.
- 9.16 Consider the double A-arm suspension system from Section 15.6 (and Figure 8.1) for the joint-coordinate formulation. Cut one of the joints and then construct all the necessary arrays and matrices for the equations of motion.
- 9.17 Consider the MacPherson suspension system from Section 15.7 for modeling with the joint-coordinate method. Derive the necessary arrays and matrices for the following two cases:
- The system is modeled with three moving bodies as in the model MP-A of Figure 8.5.
  - The system is modeled with two moving bodies as in the model MP-B of Figure 8.7. What is the best choice for the cut-joint in this case?
- 9.18 Consider the half-car double A-arm suspension system of Section 15.8. Derive the necessary arrays and matrices for modeling this system with the joint-coordinate formulation. *Hint:* Assume a floating joint between the ground and the chassis.
- 9.19 Construct a half-car model for the MacPherson suspension system from Section 15.8. Formulate the necessary arrays and matrices for the joint-coordinate formulation. *Hint:* Assume a floating joint between the ground and the chassis.
- 9.20 Derive recursive formulas for a disc rolling on the horizontal ground without slipping. As the joint coordinate, consider the following:
- The rotational coordinate of the disc
  - The translational coordinate of the disc along the  $x$ -axis



- 9.21 Derive the necessary array and matrices for the joint-coordinate formulation of the cart shown. The discs roll without slipping.



- 9.22 The robotic device shown contains three moving bodies, two revolute and one translational joint. Two rotational actuators act on the pin joints and one linear actuator acts along the translational axis. Derive the necessary array and matrices for the joint-coordinate formulation of this system.



# 10

---

## Point-Coordinate Formulation

---

Although a mechanical system is a collection of interconnected links, it is possible to describe a mechanical system as a collection of interconnected points or particles. In other words, a multibody system can be represented as a multiparticle system. This representation will not be an approximated description of the original system—the kinematics and the kinetics of the system will be fully preserved. This representation, called point-coordinate formulation, similar to the body-coordinate formulation, is suitable for programming and numerical computation, and not for analytical or pencil-and-paper approach.

The equations of motion for a multibody system with the point-coordinate formulation have a similar form or structure as those of the body coordinates. One major difference between the two formulations is the absence of any rotational coordinate in the point-coordinate formulation. The absence of rotational coordinates causes the equations of motion to have a simpler form. However, the number of equations and variables in the point-coordinate formulation are, in general, greater than those in the body coordinates for the same multibody system.

The point-coordinate formulation is a generalization of a similar classical method in analyzing simple mechanisms. Although the classical method could represent the kinematics precisely, it provides only an approximation to the kinetics of a system. In the point-coordinate formulation, as presented in this chapter, the kinematics and the kinetics of a multibody system are fully preserved; that is, no approximations are considered at any level. In our discussions, we will use the terms *points* and *particles* interchangeably.

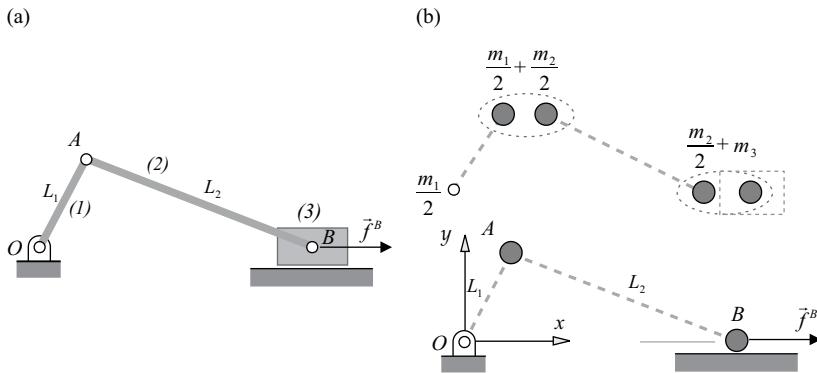
Although this book does not provide a MATLAB® program for modeling and analysis with the point-coordinate formulation, an interested reader should find the structure of such program to some extent be similar to the program that is presented in Chapter 8 for the body-coordinate formulation.

---

### 10.1 Classical Method

This classical method of presenting a mechanism as a set of constrained particles can be found in most textbooks on mechanisms and machine dynamics. The method enabled engineers, before the advent of computers, to perform a quick back-of-the-envelope analysis on a simple mechanism and obtain a general understanding of its response. Due to some simplifications in describing a system, the method provided only an approximation for the kinematic and dynamic behavior of the system.

To provide an overview of this classical method, we consider the simple slider-crank mechanism shown in Figure 10.1a. Assume that the link lengths are  $L_1$  and  $L_2$ , and the masses are  $m_1$ ,  $m_2$ , and  $m_3$ . The mass centers are located at the geometric centers of the bodies, and a force acts on link (3) as shown. In the classical method, we consider two particles at the pin joints A and B, having coordinates  $\mathbf{r}^A = \{x^A \quad y^A\}'$  and  $\mathbf{r}^B = \{x^B \quad 0\}'$  in the

**FIGURE 10.1**

(a) A slider–crank mechanism and (b) its representation as a two-particle system.

nonmoving  $x$ - $y$  frame as shown in Figure 10.1b. Particle  $A$  must keep a distance  $L_1$  from  $O$  and a distance  $L_2$  from  $B$  resulting in two constraints:

$$\begin{aligned} \frac{1}{2} \left( (x^A)^2 + (y^A)^2 - L_1^2 \right) &= 0 \\ \frac{1}{2} \left( (x^A - x^B)^2 + (y^A)^2 - L_2^2 \right) &= 0 \end{aligned} \quad (10.1)$$

The first and second time derivatives of these constraints are as follows:

$$\left[ \begin{array}{ccc} x^A & y^A & 0 \\ x^A - x^B & y^A & x^B - x^A \end{array} \right] \left\{ \begin{array}{c} \dot{x}^A \\ \dot{y}^A \\ \dot{x}^B \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\} \quad (10.2)$$

$$\left[ \begin{array}{ccc} x^A & y^A & 0 \\ x^A - x^B & y^A & x^B - x^A \end{array} \right] \left\{ \begin{array}{c} \ddot{x}^A \\ \ddot{y}^A \\ \ddot{x}^B \end{array} \right\} = \left\{ \begin{array}{c} -(\dot{x}^A)^2 - (\dot{y}^A)^2 \\ -(\dot{x}^A - \dot{x}^B)^2 - (\dot{y}^A)^2 \end{array} \right\} \quad (10.3)$$

These kinematic constraints are exact; no approximation is made for the kinematics of the two-particle system.

To obtain the equations of motion for the particles, the mass of each link must be distributed to the two particles at its ends. Since the mass centers are at the geometric centers, the mass of link (1) is distributed half to particle  $A$  and half to the nonmoving particle at  $O$ . Similarly, half of the mass of link (2) is given to  $A$  and half to  $B$ . Since particle  $B$  is also positioned at the mass center of link (3), it inherits the entire mass of link (3). Therefore, the mass of the two moving particles become

$$m^A = \frac{m_1 + m_2}{2}, \quad m^B = \frac{m_2}{2} + m_3 \quad (10.4)$$

The applied force  $\vec{f}^B$  acts directly on particle  $B$ . We can now write the equations of motion for this system of two constrained particles as

$$\left[ \begin{array}{ccc} \frac{m_1 + m_1}{2} & 0 & 0 \\ 0 & \frac{m_1 + m_1}{2} & 0 \\ 0 & 0 & \frac{m_2}{2} + m_3 \end{array} \right] \left\{ \begin{array}{c} \ddot{x}^A \\ \ddot{y}^A \\ x^B \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \\ f^B \end{array} \right\} + \left[ \begin{array}{cc} x^A & x^A - x^B \\ y^A & y^A \\ 0 & x^B - x^A \end{array} \right] \left\{ \begin{array}{c} \lambda_1 \\ \lambda_1 \end{array} \right\} \quad (10.5)$$

Here we have used the Jacobian matrix of the constraints and two Lagrange multipliers to represent the reaction forces along the axes of the two links.

At any given time, knowing the coordinates, the velocities, and the applied force, Eqs. (10.3) and (10.5) can be solved simultaneously to find the accelerations and the reaction forces. Now the question is: why the results should be only an approximation? The reason is that in the process of distributing the mass of a link to its two corresponding end particles, we only considered the mass of the link and the position of the mass center—we did not include the moment of inertia of the link. As the slider translates, links (1) and (2) translate and *rotate*, and therefore, their moments of inertia could become major contributors to the dynamics of the system. The method of point-coordinate formulation that is discussed in this chapter considers the moment of inertia of a link in the process of mass distribution, and therefore, the method does not have the shortcomings of the classical method.

## 10.2 Primary and Stationary Points

The process of describing a multibody system as a collection of interconnected particles begins with defining points (particles) at proper locations to replace bodies. Most of the points are defined based on the kinematic joints of the system. We can consider the following observations as rule of thumb to satisfy the kinematics of the system:

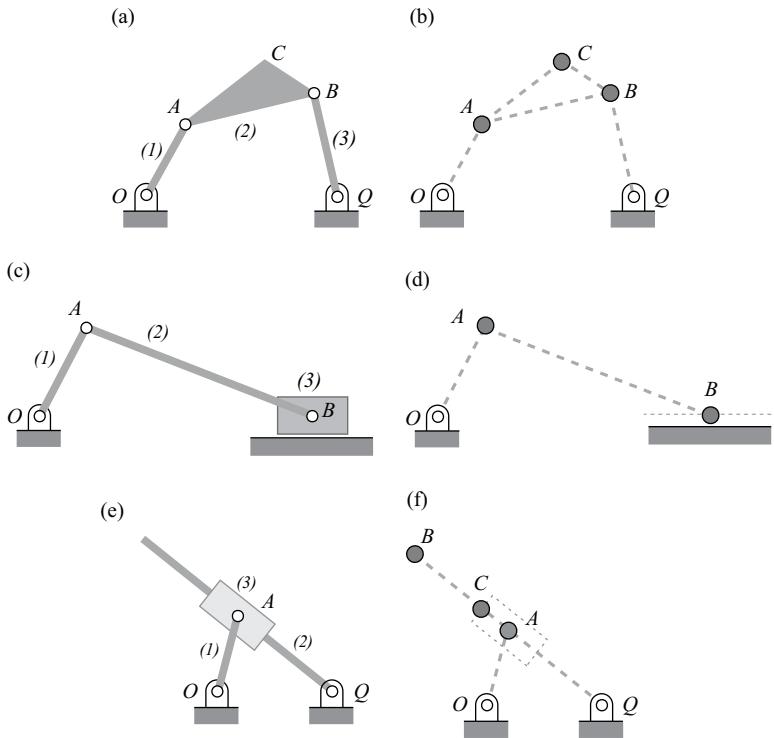
- A point must be defined at the center of a pin joint connecting two links.
- On the axis of a sliding joint between two bodies, two points must be defined on each body; however, there are *exceptions*.

To satisfy the inertial (mass and moment of inertia) characteristics of bodies, consider the following:

- For a slender rod, define at least two points.
- For a plate (a planar body with a surface area), define at least three points.

We explain these observations through several simple examples.

Consider the four-bar mechanism of Figure 10.2a. Link (1) is a slender rod; therefore, its point representation, as shown in Figure 10.2b, contains two points:  $O$  and  $A$ . Similarly,

**FIGURE 10.2**

Point representation for three mechanisms: (a, b) a four-bar, (c, d) a slider–crank, and (e, f) an inverted slider–crank.

for link (3), there are two points at pin joints  $Q$  and  $B$ . For link (2), two points at  $A$  and  $B$  would be sufficient to represent the link kinematically. However, since this link is a plate, to satisfy the inertial conditions, we also need to define a third point such as  $C$ .

Another example is the slider–crank shown in Figure 10.2c and d. Three points at the pin joints  $O$ ,  $A$ , and  $B$  can fully represent the system. However, according to one of our rules of thumb, since link (3) and the ground form a sliding joint, we must define two points on link (3) and two points on the ground; however, we have defined link (3) by only one point at  $B$ . Why? The reason is that link (3) does not rotate; therefore, we do not need at least two points on this link to follow its rotation—this is one of those exceptions.

For the inverted slider–crank shown in Figure 10.2e and f, points  $O$  and  $A$  represent link (1), and points  $Q$  and  $B$  represent link (2). Kinematically, these four points would be sufficient to represent the system. However, to fully represent the rotation of link (3), we need an additional point, such as  $C$ , on link (3).

In these examples, we observed that in describing the kinematics of a multibody system, a minimum number of points must be defined. However, to fully describe the inertia of the bodies, we may need to define additional points. The defined points could be either moving or nonmoving.

A point that is defined with two constant coordinates, such as points  $O$  and  $Q$  in the preceding examples, is referred to as a *stationary* (fixed or nonmoving) point. We can define as many stationary points in a problem without increasing the problem size for computation. A *primary* point, on the other hand, is a moving point, such as  $A$ ,  $B$ , and  $C$  in the preceding

examples. A primary point, in general, has two variable coordinates, such as point  $A$  in our examples having coordinates  $\mathbf{r}^A = \begin{Bmatrix} x^A & y^A \end{Bmatrix}'$ . In special cases, a primary point may have one variable and one constant coordinate, such as point  $B$  in the slider–crank example of Figure 10.2c and d, where  $\mathbf{r}^B = \begin{Bmatrix} x^B & 0 \end{Bmatrix}'$ .

For a mechanical system in motion, the coordinates of the primary points are variables, and their time derivatives will appear in the equations of motion. Increasing the number of primary points in a problem increases the computational effort for an analysis. Therefore, we should try to keep the number of defined primary points to a minimum.

The vector of coordinates for a typical primary point  $i$  is denoted as

$$\mathbf{r}^i = \begin{Bmatrix} x^i \\ y^i \end{Bmatrix} \quad (10.6)$$

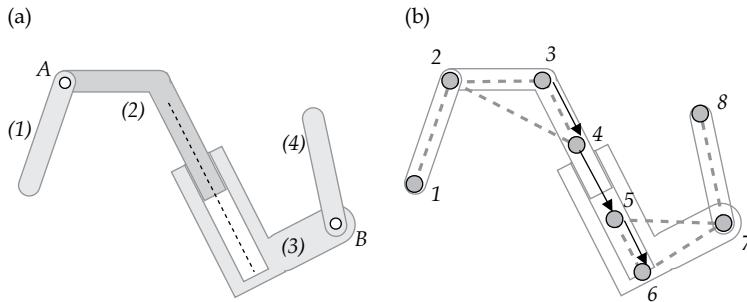
If we define  $n_p$  (number of points) primary points for a system, then an *array of coordinates* is constructed as

$$\mathbf{r} \equiv \begin{Bmatrix} \mathbf{r}^1 \\ \vdots \\ \mathbf{r}^{n_p} \end{Bmatrix} \quad (10.7)$$

The number of elements in the array of coordinates is denoted as  $n_v$  (number of variables). If each of the primary points contains two variable coordinates, then  $n_v = 2 \times n_p$ . However, in general, this may not be true if one of the coordinates of a primary point is a constant.

### 10.3 Constraints

The defined primary and stationary points that describe a multibody system are not completely free particles—they are dependent on one another through algebraic *constraints*. A constraint equation may represent a kinematic joint between two bodies, or the conditions we impose on the primary points that represent a rigid body. As an example, let us consider the schematic representation of the multibody system shown in Figure 10.3a. This system contains four bodies, two pin joints, and one sliding joint. The defined primary points are shown in Figure 10.3b, where points 1 and 2 represent body (1); points 2, 3, and 4 define body (2); points 5, 6, and 7 represent body (3); and points 7 and 8 define body (4). We note that the primary point 2 is shared between two bodies as well as the primary point 7. Sharing a primary point between two bodies eliminates the need to define constraints for a revolute joint. To describe a sliding joint between bodies (2) and (3), we have placed two primary points on body (2) and two on body (3) on the joint axis. We can then define three vectors between these four points, as shown, and state the necessary constraints for the vectors to remain collinear. Other constraints are that any two points on a body must keep a constant distance between them (a necessary condition for a nondeformable body). We will review such algebraic equations representing typical constraints in the point-coordinate formulation in Sections 10.3.1–3.

**FIGURE 10.3**

(a) Schematic presentation of a multibody system and (b) its corresponding primary point representation.

A constraint equation is denoted by the *lightface* character  $\Phi$ . If there is more than one algebraic equation in a constraint, we denote it by the *boldface* character  $\Phi$ . The constraint symbol may carry a left superscript in parentheses with two entries, such as  ${}^{(\text{type}, n)}\Phi$ . The first entry states the constraint *type*—for example, *L* for *length*, *p* for *parallel*, and *n* for *normal*. The second entry, *n*, denotes the *number* of algebraic equations in a constraint; in the point-coordinate formulation, it is most likely equal to one.

An algebraic constraint equation removes one degree of freedom (DoF) from a system. The number of constraint equations,  $n_c$ , must be equal to the difference between the number of variable coordinates and the number of DoFs; i.e.,  $n_c = n_v - n_{dof}$ .

If the array of coordinates is defined as  $\mathbf{r}$ , as stated in Eq. (10.7), then the position constraints are expressed in general form as

$$\Phi \equiv \Phi(\mathbf{r}) = 0 \quad (10.8)$$

The first and second time derivatives of the position constraints provide the velocity and acceleration constraints as

$$\dot{\Phi} \equiv \mathbf{D}\dot{\mathbf{r}} = 0 \quad (10.9)$$

$$\ddot{\Phi} \equiv \mathbf{D}\ddot{\mathbf{r}} + \dot{\mathbf{D}}\dot{\mathbf{r}} = 0 \quad (10.10)$$

The coefficient matrix  $\mathbf{D}$  in the velocity and acceleration constraints is the *Jacobian* matrix. The acceleration constraint can also be expressed as

$$\mathbf{D}\ddot{\mathbf{r}} = \boldsymbol{\gamma} \quad (10.11)$$

where

$$\boldsymbol{\gamma} = -\dot{\mathbf{D}}\dot{\mathbf{r}} \quad (10.12)$$

The array  $\boldsymbol{\gamma}$  is called the *right-hand side array of the acceleration constraints*. This array is quadratic in the velocities.

In the point-coordinate formulation, most constraints describe a condition on a vector connecting two points. For two typical points *i* and *j*, with coordinate vectors  $\mathbf{r}^i$  and  $\mathbf{r}^j$ , a position vector is constructed as

$$\mathbf{d}^{i,j} = \mathbf{r}^i - \mathbf{r}^j \quad (10.13)$$

The two points, *i* and *j*, can be both primary points or one primary and one stationary.

### 10.3.1 Length Constraint

In the examples of Section 10.1, we may have noticed that the most common condition to apply between two points is to maintain a constant distance between them. For two primary points  $i$  and  $j$ , a constant *length* constraint can be constructed as

$${}^{(L,1)}\Phi \equiv \frac{1}{2} \left( \mathbf{d}'^{i,j} \mathbf{d}^{i,j} - (L^{i,j})^2 \right) = 0 \quad (10.14)$$

where  $L^{i,j}$  is the constant distance between the two points. In this equation, we have introduced the coefficient “ $\frac{1}{2}$ ” in order to eliminate the appearance of “2” coefficients in the corresponding velocity and acceleration constraints.

The time derivative of Eq. (10.14) provides the velocity *length constraint* between two points as

$${}^{(L,1)}\dot{\Phi} = \mathbf{d}'^{i,j} \dot{\mathbf{d}}^{i,j} = 0 \Rightarrow \left[ \begin{array}{cc} \mathbf{d}'^{i,j} & -\mathbf{d}'^{i,j} \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}^i \\ \dot{\mathbf{r}}^j \end{array} \right\} = 0 \quad (10.15)$$

At this point, it should be clear why we introduced the coefficient “ $\frac{1}{2}$ ” in Eq. (10.14).

The time derivative of the velocity constraints yields the acceleration constraints as

$${}^{(L,1)}\ddot{\Phi} \equiv \mathbf{d}'^{i,j} \ddot{\mathbf{d}}^{i,j} + \dot{\mathbf{d}}'^{i,j} \dot{\mathbf{d}}^{i,j} = 0 \Rightarrow \left[ \begin{array}{ccc} \mathbf{d}'^{i,j} & -\mathbf{d}'^{i,j} & \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \end{array} \right\} = -\dot{\mathbf{d}}'^{i,j} \dot{\mathbf{d}}^{i,j} \quad (10.16)$$

### 10.3.2 Angle Constraints

Two vectors may be required to remain perpendicular or parallel, or to keep a known constant angle between them. The vectors may have been defined between two separate sets of primary points, or they may share one primary point.

Assume two vectors are defined between three points, such as shown in Figure 10.4a. For the two vectors to remain perpendicular (*normal*), the following constraint is required:

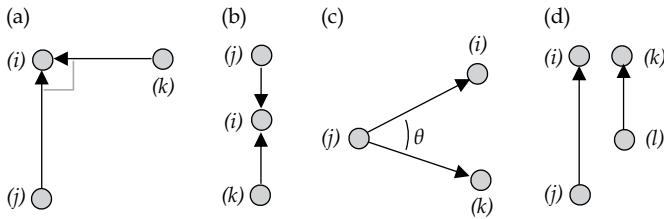
$${}^{(n,1)}\Phi \equiv \mathbf{d}'^{i,j} \mathbf{d}^{i,k} = 0 \quad (10.17)$$

The first and second time derivatives are expressed as

$${}^{(n,1)}\dot{\Phi} = \mathbf{d}'^{i,j} \dot{\mathbf{d}}^{i,k} + \mathbf{d}'^{i,k} \dot{\mathbf{d}}^{i,j} = 0 \Rightarrow \left[ \begin{array}{ccc} \mathbf{d}'^{i,j} + \mathbf{d}'^{i,k} & -\mathbf{d}'^{i,k} & -\mathbf{d}'^{i,j} \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}^i \\ \dot{\mathbf{r}}^j \\ \dot{\mathbf{r}}^k \end{array} \right\} = 0 \quad (10.18)$$

$${}^{(n,1)}\ddot{\Phi} = \mathbf{d}'^{i,j} \ddot{\mathbf{d}}^{i,k} + \mathbf{d}'^{i,k} \ddot{\mathbf{d}}^{i,j} + 2\dot{\mathbf{d}}'^{i,j} \dot{\mathbf{d}}^{i,k} = 0$$

$$\Rightarrow \left[ \begin{array}{ccc} \mathbf{d}'^{i,j} + \mathbf{d}'^{i,k} & -\mathbf{d}'^{i,k} & -\mathbf{d}'^{i,j} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \\ \ddot{\mathbf{r}}^k \end{array} \right\} = -2\dot{\mathbf{d}}'^{i,j} \dot{\mathbf{d}}^{i,k} \quad (10.19)$$

**FIGURE 10.4**

Several angle conditions between two vectors: (a) on perpendicular axes, (b) on the same axis, (c) making an acute angle, and (d) on parallel axes.

Similarly, to enforce two vectors to remain *parallel*, such as the vectors shown in Figure 10.4b, we must enforce

$$(p,1) \Phi = \check{\mathbf{d}}'^{i,j} \mathbf{d}^{i,k} = 0 \quad (10.20)$$

The vectors shown in Figure 10.4c must keep a constant *angle* between them, so we enforce their scalar product to remain as

$$(\theta,1) \Phi = \mathbf{d}'^{i,j} \mathbf{d}^{k,j} - L^{i,j} L^{k,j} \cos \theta = 0 \quad (10.21)$$

where it is assumed that each vector has a constant. The velocity and acceleration constraints associated with Eqs. (10.20) and (10.21) can easily be constructed.

The vectors used in the preceding angle constraints could also be between four points instead of three. For example, for the vectors shown in Figure 10.4d to remain parallel, the constraint becomes

$$(p,1) \Phi = \check{\mathbf{d}}'^{i,j} \mathbf{d}^{k,l} = 0 \quad (10.22)$$

### 10.3.3 Simple Constraints

A simple constraint is a condition that we impose on a single coordinate. For example, to keep the  $x$ - or the  $y$ -coordinate of a typical primary point  $i$  to remain a constant,  $c_x$  or  $c_y$ , we can use one of the following *simple* constraints:

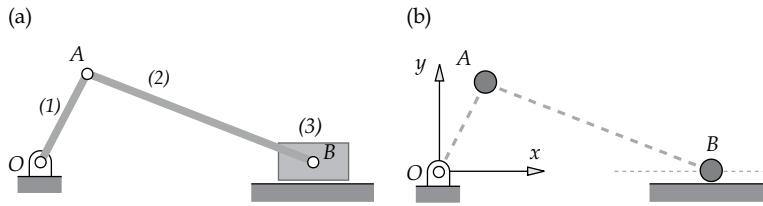
$$x^i - c_x = 0 \quad (10.23)$$

$$y^i - c_y = 0 \quad (10.24)$$

These constraints are usually defined when we want to keep a particular coordinate to appear as a variable in a formulation, but in reality that coordinate is a known constant. This usually simplifies the programming effort with a minor drawback that we unnecessarily increase the number of variables and the number of algebraic constraints in a formulation.

#### Example 10.1

For the slider–crank mechanism shown in Figure 10.5a, the necessary primary points are defined in Figure 10.5b. Write the constraint equations. Assume the link lengths are  $L_1$  and  $L_2$ .

**FIGURE 10.5**

(a) A slider-crank mechanism and (b) its point-coordinate representation.

**Solution 1**

We consider three variable coordinates for the two primary points as  $\mathbf{r}^A = \{x^A \quad y^A\}'$  and  $\mathbf{r}^B = \{x^B \quad 0\}'$ . Then we write one length constraint for each link:

$$\begin{aligned} \frac{1}{2}(\mathbf{d}'^{A,O} \mathbf{d}^{A,O} - L_1^2) &= 0 \\ \frac{1}{2}(\mathbf{d}'^{B,C} \mathbf{d}^{B,C} - L_2^2) &= 0 \end{aligned} \tag{a}$$

With three variable coordinates and two constraints, we have a  $3 - 2 = 1$  DoF system.

**Solution 2**

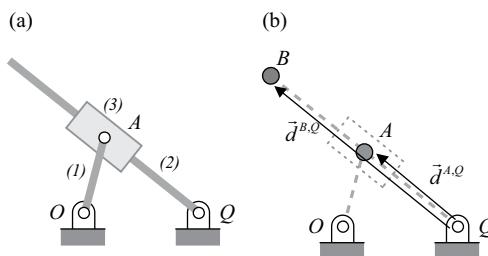
We consider four variable coordinates as  $\mathbf{r}^A = \{x^A \quad y^A\}'$  and  $\mathbf{r}^B = \{x^B \quad y^B\}'$ . We define two length constraints as in Eq. (a), and then we add a third constraint as

$$y^B = 0$$

With four variable coordinates and three constraints, we again have a  $4 - 3 = 1$  DoF system.

**Example 10.2**

For the inverted slider-crank mechanism shown in Figure 10.6a, the necessary primary points are defined in Figure 10.6b. Write the constraint equations. Assume the link lengths are  $L_1$  and  $L_2$ .

**FIGURE 10.6**

(a) An inverted slider-crank mechanism, (b) its point-coordinate representation, and (c) the defined parallel vectors.

### Solution

We provide one length constraint for each link:

$$\frac{1}{2}(\mathbf{d}'^{A,O} \mathbf{d}^{A,O} - L_1^2) = 0$$

$$\frac{1}{2}(\mathbf{d}'^{B,Q} \mathbf{d}^{B,Q} - L_2^2) = 0$$

Vectors  $\vec{d}^{B,Q}$  and  $\vec{d}^{A,Q}$ , as shown in Figure 10.6b, must remain parallel; therefore, we construct the following parallel constraint:

$$\check{\mathbf{d}}'^{B,Q} \mathbf{d}^{A,Q} = 0$$

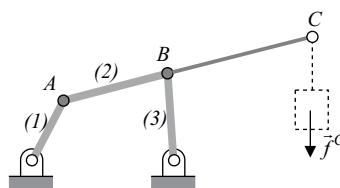
We have constructed three constraints and we have four coordinates for the two primary points. This yields  $4 - 3 = 1$  DoF, which is what we expect for a slider–crank. We should note that in this presentation the slider–block is only a point, which is sufficient for a kinematic analysis. However, for a dynamic analysis, we need to define additional points on the slider–block for correct distribution of mass and moment of inertia.

## 10.4 Secondary Points

Another type of point that we can define in representing a multibody system is a *secondary* point. In contrast to a primary point, the coordinates of a secondary point do not appear in any constraint equations, and therefore, these coordinates are not included in the array of coordinates. The coordinates of a secondary point can be determined from the coordinates of the primary points that are defined on the same body.

Consider the four-bar mechanism shown in Figure 10.7. A load applies a known force at point C on the coupler. In addition to the primary points at A and B, we can define a third primary point at C and apply the force directly on that point. This, however, results in introducing two extra coordinates and two extra constraints into the formulations. Or we can stay with the original two primary points at A and B, and define point C as a secondary point. The coordinates of point C can be determined as a function of the coordinates of A and B without introducing additional variable coordinates and constraints in the formulation.

In planar kinematics, the coordinates of a secondary point can be determined as a function of the coordinates of at least two primary points that are positioned on the same body. Consider the link shown in Figure 10.8a. Points A and B are the primary points with



**FIGURE 10.7**

Point C of the coupler link can be a primary or a secondary point.

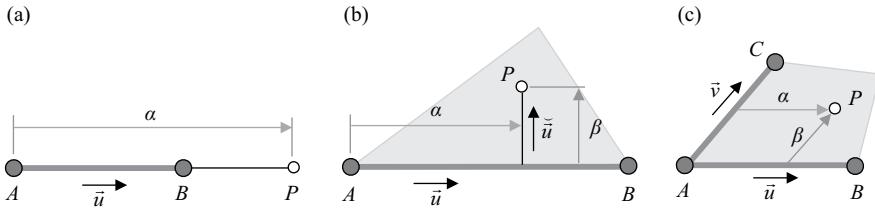


FIGURE 10.8

A secondary point  $P$  defined on a body can be positioned by (a) a single parameter, (b) two orthogonal vectors and two parameters, and (c) two nonorthogonal vectors and two parameters.

coordinates  $\mathbf{r}^A$  and  $\mathbf{r}^B$ . Point  $P$  is defined on the axis of the link with a known distance  $\alpha$  from  $A$  as shown. If the coordinates of  $A$  and  $B$  are known, then the coordinates of  $P$  can be computed. To derive an *expression* for this purpose, we first define a positive direction along the axis of the link; for example, we construct a unit vector  $\bar{u}$  from  $A$  to  $B$ :

$$\mathbf{u} = \frac{1}{L_{B,A}} (\mathbf{r}^B - \mathbf{r}^A) \quad (10.25)$$

Then the coordinates of  $P$  can be computed as

$$\mathbf{r}^P = \mathbf{r}^A + \alpha \mathbf{u} \Rightarrow \mathbf{r}^P = \left(1 - \frac{\alpha}{L_{B,A}}\right) \mathbf{r}^A + \frac{\alpha}{L_{B,A}} \mathbf{r}^B \quad (10.26)$$

It should be noted that  $\alpha$  is a directional magnitude; that is, it could be either a positive or a negative quantity depending on the position of  $P$  on the link relative to  $A$  and the direction of the unit vector. The velocity and acceleration of  $P$  are computed from the first and second time derivatives of Eq. (10.26):

$$\dot{\mathbf{r}}^P = \left(1 - \frac{\alpha}{L_{B,A}}\right) \dot{\mathbf{r}}^A + \frac{\alpha}{L_{B,A}} \dot{\mathbf{r}}^B, \ddot{\mathbf{r}}^P = \left(1 - \frac{\alpha}{L_{B,A}}\right) \ddot{\mathbf{r}}^A + \frac{\alpha}{L_{B,A}} \ddot{\mathbf{r}}^B \quad (10.27)$$

In a second case, the secondary point  $P$  is not on the axis of the primary points  $A$  and  $B$ , as shown in Figure 10.8b. The position of this secondary point can be defined by two known distances  $\alpha$  and  $\beta$ , along the unit vectors  $\bar{u}$  and  $\bar{u}'$ , respectively. We determine the components of the unit vector  $\bar{u}$  from Eq. (10.25), and then compute the coordinates of point  $P$  as

$$\mathbf{r}^P = \mathbf{r}^A + \alpha \mathbf{u} + \beta \bar{u}' \Rightarrow \mathbf{r}^P = \left(1 - \frac{\alpha}{L_{B,A}}\right) \mathbf{r}^A + \frac{\alpha}{L_{B,A}} \mathbf{r}^B + \frac{\beta}{L_{B,A}} (\mathbf{r}^B - \bar{r}^A) \quad (10.28)$$

The corresponding velocity and acceleration expressions can be derived easily.

In another case, as shown in Figure 10.8c, the secondary point  $P$  is attached to a body that is defined by three primary points. Knowing the coordinates of the primary points and the constant distances  $\alpha$  and  $\beta$  as shown, we first define unit vectors  $\bar{u}$  and  $\bar{v}$ , and then determine the coordinates of point  $P$  as

$$\mathbf{r}^P = \mathbf{r}^A + \alpha \mathbf{u} + \beta \mathbf{v} \Rightarrow \mathbf{r}^P = \left(1 - \frac{\alpha}{L_{B,A}} - \frac{\beta}{L_{C,A}}\right) \mathbf{r}^A + \frac{\alpha}{L_{B,A}} \mathbf{r}^B + \frac{\beta}{L_{C,A}} \mathbf{r}^C \quad (10.29)$$

In Eqs. (10.26), (10.28), and (10.29), it is assumed that the values of  $\alpha$  and  $\beta$  are available. This is normally the case in most problems. However, in some problems, these values may not be known initially, but instead, the coordinates of the primary points and point  $P$  would be available. In such a case, we can perform an inverse operation to determine  $\alpha$  and  $\beta$ .

Assume that for the linkage of Figure 10.8a, the coordinates of points  $A$ ,  $B$ , and  $P$  are known initially. To determine the values of  $\alpha$  and  $\beta$ , we premultiply Eq. (10.26) by  $\mathbf{u}'$  and rearrange it as

$$\mathbf{u}'(\mathbf{r}^P - \mathbf{r}^A) = \alpha \mathbf{u}'\mathbf{u}$$

Since  $\mathbf{u}'\mathbf{u} = 1$ , we obtain an expression to compute  $\alpha$  as

$$\alpha = \mathbf{u}'(\mathbf{r}^P - \mathbf{r}^A) \Rightarrow \alpha = \frac{1}{L_{B,A}}(\mathbf{r}^B - \mathbf{r}^A)'(\mathbf{r}^P - \mathbf{r}^A) \quad (10.30)$$

For the case of Figure 10.8b, premultiplying Eq. (10.28) by  $\mathbf{u}'$  and noting that  $\mathbf{u}'\check{\mathbf{u}} = 0$ , we obtain the same expression for  $\alpha$  as in Eq. (10.30). Premultiplying Eq. (10.28) by  $\check{\mathbf{u}}'$  yields an expression for  $\beta$  as

$$\beta = \frac{1}{L_{B,A}}(\check{\mathbf{r}}^B - \check{\mathbf{r}}^A)'(\mathbf{r}^P - \mathbf{r}^A) \quad (10.31)$$

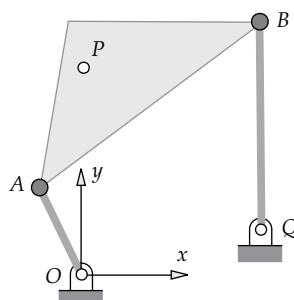
A similar process provides expressions for determining  $\alpha$  and  $\beta$  for the case of Figure 10.8c.

### Example 10.3

For the four-bar mechanism shown in Figure 10.9, an  $x$ - $y$  frame is defined. In this reference frame, the coordinates of the joints and the coupler point are measured accurately to be

$$\mathbf{r}^O = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \mathbf{r}^Q = \begin{Bmatrix} 2.00 \\ 0.50 \end{Bmatrix}, \mathbf{r}^A = \begin{Bmatrix} -0.42 \\ 0.91 \end{Bmatrix}, \mathbf{r}^B = \begin{Bmatrix} 2.00 \\ 2.70 \end{Bmatrix}, \mathbf{r}^P = \begin{Bmatrix} 0.07 \\ 2.27 \end{Bmatrix} \text{ m}$$

Define the primary points, construct the constraint equations, and write the expression for the kinematics of the coupler-point  $P$  only at the position level.



**FIGURE 10.9**

A coupler point  $P$  of a four-bar mechanism.

**Solution**

The primary points are defined at  $A$  and  $B$ . We compute the link lengths as follows:  $L^{A,O} = 1.0\text{ m}$ ,  $L^{B,A} = 3.0\text{ m}$ , and  $L^{B,Q} = 2.2\text{ m}$ . We define the three vectors as

$$\mathbf{d}^{A,O} = \mathbf{r}^A - \mathbf{r}^O = \mathbf{r}^A, \mathbf{d}^{B,A} = \mathbf{r}^B - \mathbf{r}^A, \text{ and } \mathbf{d}^{B,Q} = \mathbf{r}^B - \mathbf{r}^Q$$

We can now construct three length constraints:

$$\frac{1}{2}(\mathbf{d}^{A,O} \cdot \mathbf{d}^{A,O} - 1.0) = 0$$

$$\frac{1}{2}(\mathbf{d}^{B,A} \cdot \mathbf{d}^{B,A} - 3^2) = 0$$

$$\frac{1}{2}(\mathbf{d}^{B,Q} \cdot \mathbf{d}^{B,Q} - 2.2^2) = 0$$

Because the values for  $\alpha$  and  $\beta$  for positioning the coupler point  $P$  are not given, we use Eqs. (10.30) and (10.31) to compute them:

$$\alpha = \frac{1}{3.0}(\mathbf{r}^B - \mathbf{r}^A)'(\mathbf{r}^B - \mathbf{r}^A) = 1.2, \beta = \frac{1}{3.0}(\bar{\mathbf{r}}^B - \bar{\mathbf{r}}^A)'(\mathbf{r}^B - \mathbf{r}^A) = 0.8$$

These values will not change as the four-bar mechanism moves. The expression to determine the coordinates of point  $P$ , based on Eq. (10.28), is

$$\mathbf{r}^P = \left(1 - \frac{1.2}{3.0}\right)\mathbf{r}^A + \frac{1.2}{3.0}\mathbf{r}^B + \frac{0.8}{3.0}(\bar{\mathbf{r}}^B - \bar{\mathbf{r}}^A)$$

## 10.5 Equations of Motion

Assume a multibody system is represented by  $n_p$  primary points. If we consider each primary point as a particle, for this system of  $n_p$  particles, the equations of motion are simply written as

$$\mathbf{M}\ddot{\mathbf{r}} = {}^{(a)}\mathbf{h} + {}^{(r)}\mathbf{h} \quad (10.32)$$

where

$$\ddot{\mathbf{r}} = \begin{Bmatrix} \ddot{\mathbf{r}}^1 \\ \vdots \\ \ddot{\mathbf{r}}^{n_p} \end{Bmatrix}, \mathbf{M} = \begin{bmatrix} \mathbf{M}^1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{M}^{n_p} \end{bmatrix}, {}^{(a)}\mathbf{h} = \begin{Bmatrix} \mathbf{f}^1 \\ \vdots \\ \mathbf{f}^{n_p} \end{Bmatrix}, {}^{(r)}\mathbf{h} = \begin{Bmatrix} {}^{(r)}\mathbf{f}^1 \\ \vdots \\ {}^{(r)}\mathbf{f}^{n_p} \end{Bmatrix} \quad (10.33)$$

The mass matrix  $\mathbf{M}$  is made of  $2 \times 2$  submatrices for each particle:

$$\mathbf{M}^i = \begin{bmatrix} m^i & 0 \\ 0 & m^i \end{bmatrix} = m^i \mathbf{I}, \quad i = 1, \dots, n_p \quad (10.34)$$

where  $m^i$  is the mass of the  $i$ th particle. The array of applied forces,  ${}^{(a)}\mathbf{h}$ , can have contributions from gravitational force, springs, dampers, and other elements. The array  ${}^{(r)}\mathbf{h}$  contains reaction forces acting on the primary points as a result of the constraint equations.

Assume that there are  $n_c$  constraints that are expressed in general form as

$$\Phi(\mathbf{r}) = \mathbf{0} \quad (10.35)$$

The corresponding velocity and acceleration constraints are

$$\dot{\Phi} \equiv \mathbf{D}\dot{\mathbf{r}} = \mathbf{0} \quad (10.36)$$

$$\ddot{\Phi} \equiv \mathbf{D}\ddot{\mathbf{r}} + \dot{\mathbf{D}}\dot{\mathbf{r}} = \mathbf{0} \quad (10.37)$$

According to the method of Lagrange multipliers, the array of reaction forces in Eq. (10.32) can be represented as

$${}^{(r)}\mathbf{h} = \mathbf{D}'\boldsymbol{\lambda} \quad (10.38)$$

where the array  $\boldsymbol{\lambda}$  contains  $n_c$  multipliers. Equation (10.32) can now be rewritten as

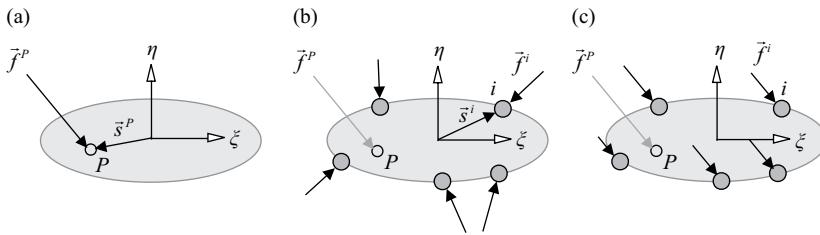
$$\mathbf{M}\ddot{\mathbf{r}} = {}^{(a)}\mathbf{h} + \mathbf{D}'\boldsymbol{\lambda} \quad (10.39)$$

For constructing the complete set of equations of motion for a multibody system in point-coordinate formulation, we have already discussed how to generate the constraint equations, and the corresponding Jacobian matrix and the right-hand side of the acceleration constraints. In the following sections, we will discuss (a) how to distribute forces and torques from bodies to primary points for constructing the array  ${}^{(a)}\mathbf{h}$ ; (b) how to distribute the mass and moment of inertia of bodies to primary points without any approximations.

## 10.6 Force and Torque Distribution

Applied forces on a body must be distributed to the particles that represent the body. If the point of application of a force on a body coincides with a primary point (i.e., one of the particles that represent the body), then the force can be applied directly to that primary point. However, if the point of application of a force does not coincide with a primary point, then that force must be distributed to the primary points that represent that body.

Assume a force  $\vec{f}^P$  acts on a body at point  $P$  as shown in Figure 10.10a. Point  $P$  is positioned on the body from the reference point  $O$  by vector  $\vec{s}^P$ . Furthermore, assume that  $n_p$  primary points represent the body, as shown in Figure 10.10b. We attach a reference frame to the body, which could be either the  $x$ - $y$  or the  $\xi$ - $\eta$  frame. Each primary point is positioned from the origin  $O$  by a vector  $\vec{s}^i$ . To simplify our discussion, we consider the body mass center as the origin.

**FIGURE 10.10**

(a) A force acting on a body and its distributed point representation with (b) arbitrary defined forces, and (c) forces parallel to the original force.

We distribute the force  $\vec{f}^P$  to the primary points as  $\vec{f}^i$ ;  $i = 1, \dots, n_p$ . For the two force systems shown in Figure 10.10a, b to be equivalent, two conditions must be met:

1. The sum of forces must be equal:

$$\sum_{i=1}^{n_p} \mathbf{f}^i = \mathbf{f}^P \quad (10.40)$$

2. The sum of moments about any arbitrary point must be equal:

$$\sum_{i=1}^{n_p} \bar{\mathbf{s}}'^i \mathbf{f}^i = \bar{\mathbf{s}}'^P \mathbf{f}^P \quad (10.41)$$

Equations (10.40) and (10.41) represent three algebraic equations: one for the  $x$ -component of the force, one for the  $y$ -component of the force, and one for the moment about an axis perpendicular to the plane. To obtain a unique solution, these equations should contain only three unknowns. Since we distributed the force to all  $n_p$  points, we have  $2n_p$  unknowns. If we assume that all  $n_p$  forces are parallel to  $\vec{f}^P$ , we reduce the number of unknowns to  $n_p$ . This assumption, as shown in Figure 10.10c, can be stated as

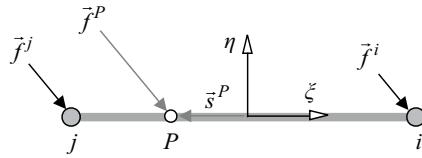
$$\mathbf{f}^i = \alpha^i \mathbf{f}^P; \quad i = 1, \dots, n_p \quad (10.42)$$

where  $\alpha^i$ 's are positive or negative coefficients. Substituting Eq. (10.42) in Eqs. (10.40) and (10.41) yields

$$\sum_{i=1}^{n_p} \alpha^i \mathbf{f}^P = \mathbf{f}^P, \quad \sum_{i=1}^{n_p} \alpha^i \bar{\mathbf{s}}'^i \mathbf{f}^P = \bar{\mathbf{s}}'^P \mathbf{f}^P$$

Because  $\vec{f}^P$  is a vector having an arbitrary magnitude and direction, it can be eliminated from both sides of these equations. This results in the following equations:

$$\sum_{i=1}^{n_p} \alpha^i = 1, \quad \sum_{i=1}^{n_p} \alpha^i \mathbf{s}^i = \mathbf{s}^P \quad (10.43)$$

**FIGURE 10.11**

Force equivalency for a rod represented by two primary points.

Since Eq. (10.43) contains three algebraic equations, we need to distribute the force to only three points in order to have a unique solution. We keep three of the forces and express Eq. (10.43) in expanded form:

$$\begin{bmatrix} 1 & 1 & 1 \\ s_{(\xi)}^i & s_{(\xi)}^j & s_{(\xi)}^k \\ s_{(\eta)}^i & s_{(\eta)}^j & s_{(\eta)}^k \end{bmatrix} \begin{Bmatrix} \alpha^i \\ \alpha^j \\ \alpha^k \end{Bmatrix} = \begin{Bmatrix} 1 \\ s_{(\xi)}^p \\ s_{(\eta)}^p \end{Bmatrix} \quad (10.44)$$

We note that the solution to the three coefficients is independent from the axis of the original force. This means that the coefficients need to be determined only once for each body—they do not need to be updated as the body moves and the original force changes its axis and magnitude.

If a body, such as the rod shown in Figure 10.11, is represented by only two primary points, since all the involved points have zero  $\eta$ -coordinates, Eq. (10.44) can be simplified to

$$\begin{bmatrix} 1 & 1 \\ s_{(\xi)}^i & s_{(\xi)}^j \end{bmatrix} \begin{Bmatrix} \alpha^i \\ \alpha^j \end{Bmatrix} = \begin{Bmatrix} 1 \\ s_{(\xi)}^p \end{Bmatrix} \quad (10.45)$$

This equation leads to the following solution for the two coefficients:

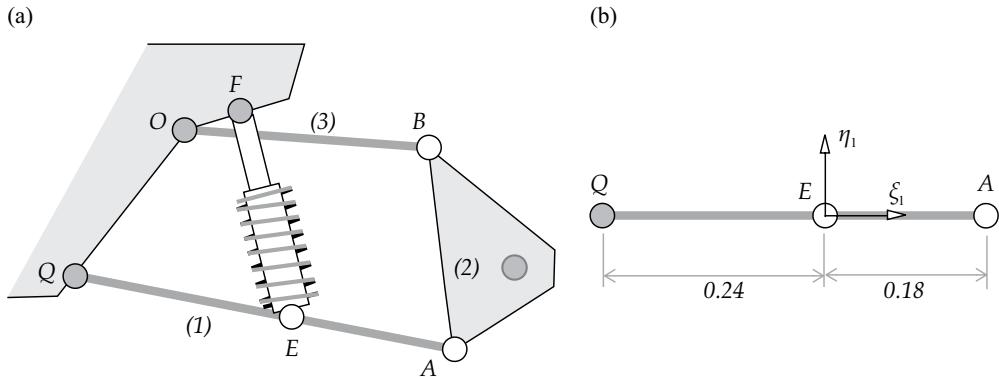
$$\alpha^i = \frac{s_{(\xi)}^p - s_{(\xi)}^j}{s_{(\xi)}^i - s_{(\xi)}^j}, \quad \alpha^j = \frac{s_{(\xi)}^p - s_{(\xi)}^i}{s_{(\xi)}^j - s_{(\xi)}^i}$$

The two forces that act on the primary points are then determined as

$$\mathbf{f}^i = \alpha^i \mathbf{f}^p, \quad \mathbf{f}^j = \alpha^j \mathbf{f}^p \quad (10.46)$$

#### Example 10.4

Consider the double A-arm suspension shown in Figure 10.12a. The force of the spring-damper acts on point  $E$  of the lower arm. Two primary points represent the arm as shown in Figure 10.12b. Distribute the spring-damper force to the two primary points.

**FIGURE 10.12**

(a) A double A-arm suspension; (b) the lower A-arm is represented by three primary points where one is stationary.

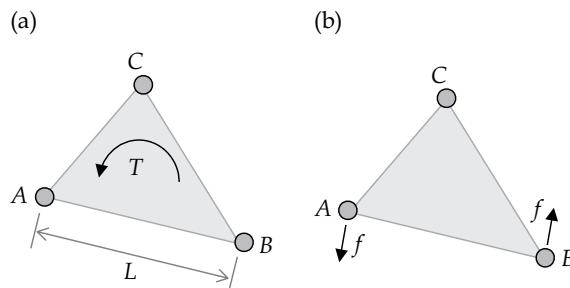
### Solution

For the lower A-arm, we position the origin of a reference frame at  $E$ , which is where the force of the spring-damper is applied. Therefore, Eq. (10.45) becomes

$$\begin{bmatrix} 1 & 1 \\ 0.18 & -0.24 \end{bmatrix} \begin{Bmatrix} \alpha^A \\ \alpha^Q \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}$$

This equation yields  $\alpha^A = 0.571$  and  $\alpha^Q = 0.429$ .

If a torque acts on a body, it must be distributed to the primary points as an equivalent set of forces. Since a torque is a couple, that is, a pair of equal forces but in opposite directions, we only need two primary points for the force distribution. As an example, consider the body shown in Figure 10.13a that is represented by three primary points. A torque  $T$  acts on the body. We can pick two of the primary points and apply a pair of forces as shown in Figure 10.13b. The forces are perpendicular to the axis connecting the two primary points, and the magnitude of each force is  $f = T/L$ .

**FIGURE 10.13**

(a) A torque acts on a body and (b) its equivalent system.

## 10.7 Mass Distribution

Similar to the force distribution, the mass of a body must be distributed to the primary points that represent that body. Assume the body has a mass  $m$  and a mass moment of inertia  $J$ . Furthermore, assume that  $n_p$  primary points represent the body. If the mass that is distributed to each primary point is denoted as  $m^i$ ,  $i = 1, \dots, n_p$ , then the following three conditions must be satisfied in order for this system of mass points to *exactly* represent the inertial characteristics of its corresponding body:

1. The sum of the particle masses must be equal to the mass of the body:

$$\sum_{i=1}^{n_p} m^i = m \quad (10.47)$$

2. The mass center of the system of particles must be at the same position as the mass center of the body:

$$\sum_{i=1}^{n_p} m^i \mathbf{s}^i = \mathbf{0} \quad (10.48)$$

3. The moment of inertia of the system of particles about any point must be equal to the moment of inertia of the body about the same point. To simplify the formulation, we select the reference point to be the mass center:

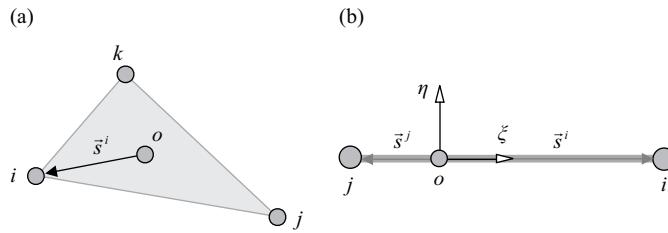
$$\sum_{i=1}^{n_p} m^i (s^i)^2 = J \quad (10.49)$$

where  $\bar{s}^i$  locates primary point  $i$  from the mass center of the body. These three conditions together represent four algebraic equations, meaning that we only need four unknown masses. We append Eqs. (10.47)–(10.49) together in expanded form for four unknown masses:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ s_{(\xi)}^i & s_{(\xi)}^j & s_{(\xi)}^k & s_{(\xi)}^o \\ s_{(\eta)}^i & s_{(\eta)}^j & s_{(\eta)}^k & s_{(\eta)}^o \\ (s^i)^2 & (s^j)^2 & (s^k)^2 & (s^o)^2 \end{bmatrix} \begin{Bmatrix} m^i \\ m^j \\ m^k \\ m^o \end{Bmatrix} = \begin{Bmatrix} m \\ 0 \\ 0 \\ J \end{Bmatrix} \quad (10.50)$$

This equation can slightly be simplified if we place point  $o$  at the mass center, that is,  $\bar{s}^o = \vec{0}$ :

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ s_{(\xi)}^i & s_{(\xi)}^j & s_{(\xi)}^k & 0 \\ s_{(\eta)}^i & s_{(\eta)}^j & s_{(\eta)}^k & 0 \\ (s^i)^2 & (s^j)^2 & (s^k)^2 & 0 \end{bmatrix} \begin{Bmatrix} m^i \\ m^j \\ m^k \\ m^o \end{Bmatrix} = \begin{Bmatrix} m \\ 0 \\ 0 \\ J \end{Bmatrix} \quad (10.51)$$

**FIGURE 10.14**

Mass distribution to (a) four primary points and (b) three primary points.

Such a system of four primary points is depicted in Figure 10.14a. These four equations can be solved to determine the mass of the four primary points.

For a rod, as shown in Figure 10.14b, we only need three primary points, where one of them can be positioned at the rod's mass center. Therefore, Eq. (10.51) can be simplified to

$$\begin{bmatrix} 1 & 1 & 1 \\ s_{(\xi)}^i & s_{(\xi)}^j & 0 \\ (s_{(\xi)}^i)^2 & (s_{(\xi)}^j)^2 & 0 \end{bmatrix} \begin{Bmatrix} m^i \\ m^j \\ m^o \end{Bmatrix} = \begin{Bmatrix} m \\ 0 \\ J \end{Bmatrix} \quad (10.52)$$

In this derivation, we have assumed a nonuniform mass distribution; that is, the mass center of the rod may not be at its geometric center. However, for most cases where the mass center is at the geometric center, where  $\vec{s}^i = -\vec{s}^j$ , the solution of Eq. (10.52) results in  $m^i = m^j$ .

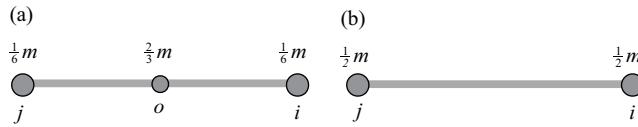
The distribution of the inertia of a planar body requires four primary points, and for a rod, it requires three primary points. These distribution formulas satisfy all the three conditions that were established for the total mass, the position of the mass center, and the mass moment of inertia. In the classical method of mass distribution, the third condition for the moment of inertia was not considered, and therefore, the distribution was an approximation.

### Example 10.5

Consider a slender rod with a mass *m* and a length *L*. The mass center is at the rod's geometric center. The moment of inertia for the rod is  $J = \frac{1}{12}mL^2$ . Applying Eq. (10.52) to the rod results in

$$\begin{bmatrix} 1 & 1 & 1 \\ L/2 & -L/2 & 0 \\ (L/2)^2 & (-L/2)^2 & 0 \end{bmatrix} \begin{Bmatrix} m^i \\ m^j \\ m^o \end{Bmatrix} = \begin{Bmatrix} m \\ 0 \\ \frac{1}{12}mL^2 \end{Bmatrix} \quad (a)$$

Solution of this equation provides the following values for the three masses, which is also illustrated in Figure 10.15a:  $m^i = m^j = \frac{1}{6}m$  and  $m^o = \frac{2}{3}m$ .

**FIGURE 10.15**

(a) Exact mass distribution; (b) approximate mass distribution.

The classical method of distribution only considered the first two conditions: the total mass and the position of the mass center. For the slender rod, these two conditions provide a mass distribution for the two end primary points as  $m^i = m^j = \frac{1}{2}m$ , as illustrated in Figure 10.15b. This mass distribution yields a mass moment of inertia for this system of two particles as

$$J = \frac{m}{2} \left( \frac{L}{2} \right)^2 + \frac{m}{2} \left( \frac{L}{2} \right)^2 = m \frac{L^2}{4}$$

This result is very different than the actual moment of inertia of  $J = \frac{1}{12}mL^2$ .

## 10.8 Mass Condensation

One drawback of the exact mass distribution discussed in Section 10.7 is that we increased the number of primary points for a planar body from three to four and for a slender rod from two to three. This means that for each additional primary point, we have two additional coordinates and two additional constraints in the formulation of the equations of motion. In this section, we learn how to eliminate the additional primary point from the formulation without jeopardizing the exactness of the mass distribution. We show the process for the slender rod formulation first before presenting it for a planar body.

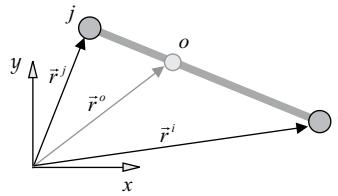
### 10.8.1 Two Primary Points

Let us consider the slender rod shown in Figure 10.16. The process of mass distribution, as provided in Eq. (10.52), added a third primary point at the mass center to the two original primary points,  $i$  and  $j$ . The following length constraint exists between the original primary points:

$$\frac{1}{2} \left( (\mathbf{r}^i - \mathbf{r}^j)' (\mathbf{r}^i - \mathbf{r}^j) - L^2 \right) = 0 \quad (a)$$

Since the primary point  $o$  is at the mass center, according to its definition from Eq. (4.4), we must have

$$m\mathbf{r}^o = m^i\mathbf{r}^i + m^j\mathbf{r}^j + m^o\mathbf{r}^o \Rightarrow \mathbf{r}^o - \mu^i\mathbf{r}^i - \mu^j\mathbf{r}^j = \mathbf{0} \quad (b)$$



**FIGURE 10.16**  
Eliminating a primary point that is positioned at the mass center.

where

$$\mu^i = \frac{m^i}{m^i + m^j}, \mu^j = \frac{m^j}{m^i + m^j}$$

Equations (a) and (b) are the constraints on the coordinates of these three primary points. The time derivative of these constraints leads to the Jacobian matrix of the constraints as

$$\left[ \begin{array}{ccc} (\mathbf{r}^i - \mathbf{r}^j)' & (\mathbf{r}^j - \mathbf{r}^i)' & \mathbf{0}' \\ -\mu^i \mathbf{I} & -\mu^j \mathbf{I} & \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}^i \\ \dot{\mathbf{r}}^j \\ \dot{\mathbf{r}}^0 \end{array} \right\} = \mathbf{0} \Rightarrow \mathbf{D} = \left[ \begin{array}{ccc} (\mathbf{r}^i - \mathbf{r}^j)' & (\mathbf{r}^j - \mathbf{r}^i)' & \mathbf{0}' \\ -\mu^i \mathbf{I} & -\mu^j \mathbf{I} & \mathbf{I} \end{array} \right]$$

We construct the equations of motion for this system of three primary points as

$$\left[ \begin{array}{ccc} m^i \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & m^j \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & m^o \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \\ \ddot{\mathbf{r}}^0 \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{f}^i \\ \mathbf{f}^j \\ \mathbf{0} \end{array} \right\} + \left[ \begin{array}{ccc} \mathbf{r}^i - \mathbf{r}^j & -\mu^i \mathbf{I} & \lambda_1 \\ \mathbf{r}^j - \mathbf{r}^i & -\mu^j \mathbf{I} & \lambda_{2-3} \\ \mathbf{0} & \mathbf{I} & \end{array} \right] \left\{ \begin{array}{c} \lambda_1 \\ \lambda_{2-3} \end{array} \right\} \quad (c)$$

where  $\vec{f}^i$  and  $\vec{f}^j$  are the distributed applied forces that act on the original two points, and we have used the Lagrange multiplier method to represent the reaction forces between the three primary points due to the constraints. Equation (c) contains six equations, two for each point. At this point, we show how to reduce the number of equations to four and, at the same time, eliminate the acceleration of point  $o$  from the equations.

Let us construct the following transformation expressions, using Eq. (b):

$$\left\{ \begin{array}{c} \mathbf{r}^i \\ \mathbf{r}^j \\ \mathbf{r}^0 \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mu^i \mathbf{I} & \mu^j \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \mathbf{r}^i \\ \mathbf{r}^j \\ \mathbf{r}^0 \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \\ \ddot{\mathbf{r}}^0 \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mu^i \mathbf{I} & \mu^j \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \\ \ddot{\mathbf{r}}^0 \end{array} \right\} \quad (d)$$

The coefficient matrix in these expressions is a transformation matrix that we name matrix  $\mathbf{B}$ :

$$\mathbf{B} = \left[ \begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mu^i \mathbf{I} & \mu^j \mathbf{I} \end{array} \right] \quad (e)$$

We substitute the acceleration expression from Eq. (d) in Eq. (c) to get

$$\left[ \begin{array}{cc} m^i\mathbf{I} & \mathbf{0} \\ \mathbf{0} & m^j\mathbf{I} \\ m^o\mu^i\mathbf{I} & m^o\mu^j\mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{f}^i \\ \mathbf{f}^j \\ \mathbf{0} \end{array} \right\} + \left[ \begin{array}{cc} \mathbf{r}^i - \mathbf{r}^j & -\mu^i\mathbf{I} \\ \mathbf{r}^j - \mathbf{r}^i & -\mu^j\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \lambda_1 \\ \lambda_{2-3} \end{array} \right\} \quad (\text{f})$$

We note that  $\ddot{\mathbf{r}}^o$  has been eliminated from the equations, but we still have six equations. We premultiply Eq. (f) by the transpose of matrix  $\mathbf{B}$ :

$$\left[ \begin{array}{ccc} \mathbf{I} & \mathbf{0} & \mu^i\mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mu^j\mathbf{I} \end{array} \right] \left[ \begin{array}{cc} m^i\mathbf{I} & \mathbf{0} \\ \mathbf{0} & m^j\mathbf{I} \\ m^o\mu^i\mathbf{I} & m^o\mu^j\mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{f}^i \\ \mathbf{f}^j \\ \mathbf{0} \end{array} \right\} + \left[ \begin{array}{cc} \mathbf{r}^i - \mathbf{r}^j & -\mu^i\mathbf{I} \\ \mathbf{r}^j - \mathbf{r}^i & -\mu^j\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \lambda_1 \\ \lambda_{2-3} \end{array} \right\} \quad (\text{g})$$

Simplifying this equation results in the transformed equations of motion as

$$\left[ \begin{array}{cc} m^i\mathbf{I} + m^o(\mu^i)^2\mathbf{I} & m^o\mu^i\mu^j\mathbf{I} \\ m^o\mu^i\mu^j\mathbf{I} & m^j\mathbf{I} + m^o(\mu^j)^2\mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{r}}^i \\ \ddot{\mathbf{r}}^j \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{f}^i \\ \mathbf{f}^j \end{array} \right\} + \left[ \begin{array}{c} \mathbf{r}^i - \mathbf{r}^j \\ \mathbf{r}^j - \mathbf{r}^i \end{array} \right] \lambda_1 \quad (10.53)$$

where we have only four equations. In this process, we eliminated particle  $o$  from the system but kept its contribution to the mass matrix, and we transformed a  $6 \times 6$  mass matrix to a  $4 \times 4$  matrix.

Equation (10.53) represents the equations of motion for two primary points,  $i$  and  $j$ , acted upon by applied forces  $\bar{\mathbf{f}}^i$  and  $\bar{\mathbf{f}}^j$ , and also acted upon by the reaction forces introduced by a single length constraint. The only difference between Eq. (10.53) and the equations of motion for two particles connected by a massless rod is in the mass matrix.

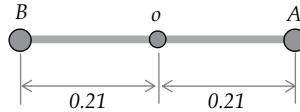
The mass matrix in Eq. (10.53) was constructed as

$$\mathbf{M}^{(rod)} = \mathbf{B}'\mathbf{M}\mathbf{B} = \left[ \begin{array}{cc} \left(m^i + m^o(\mu^i)^2\right)\mathbf{I} & m^o\mu^i\mu^j\mathbf{I} \\ m^o\mu^i\mu^j\mathbf{I} & \left(m^j + m^o(\mu^j)^2\right)\mathbf{I} \end{array} \right] \quad (10.54)$$

We can verify that the sum of the four elements in this matrix is equal to the mass of the rod,  $m$ .

For a rod with a symmetric mass distribution, where the mass center is at the geometric center of the rod, we have  $m^i = m^j$ . This in turn simplifies the mass matrix of Eq. (10.54) to

$$\mathbf{M}^{(rod)} = \left[ \begin{array}{cc} \left(m^i + \frac{m^o}{4}\right)\mathbf{I} & \frac{m^o}{4}\mathbf{I} \\ \frac{m^o}{4}\mathbf{I} & \left(m^j + \frac{m^o}{4}\right)\mathbf{I} \end{array} \right] \quad (10.55)$$

**FIGURE 10.17**Eliminating the primary point  $o$  from a link.**Example 10.6 (Example 10.5 cont.)**

For a rod with symmetric mass distribution, the mass of the three primary points were obtained in Example 10.5 as  $m^i = m^j = \frac{1}{6}m$  and  $m^o = \frac{2}{3}m$ . Construct the mass matrix for the rod.

**Solution**

Substituting the mass values in Eq. (10.55) provides the following mass matrix:

$$\mathbf{M}^{(rod)} = \frac{m}{6} \begin{bmatrix} 2\mathbf{I} & \mathbf{I} \\ \mathbf{I} & 2\mathbf{I} \end{bmatrix}$$

**Example 10.7**

Consider the link shown in Figure 10.17 where the mass center is located at its geometric center. The rod has a mass  $m = 2.0\text{ kg}$  and a moment of inertia  $J = 0.05\text{ kg m}^2$ . Note that the moment of inertia for this link is not  $J = \frac{1}{12}mL^2$  due to its nonuniform mass distribution. Construct the mass matrix for this link.

**Solution**

We use Eq. (10.52) to distribute the mass and the moment of inertia to three points:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0.21 & -0.21 & 0 \\ (0.21)^2 & (-0.21)^2 & 0 \end{bmatrix} \begin{Bmatrix} m^A \\ m^B \\ m^o \end{Bmatrix} = \begin{Bmatrix} 2.0 \\ 0 \\ 0.05 \end{Bmatrix}$$

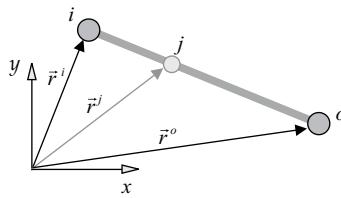
which yields  $m^A = m^B = 0.5669\text{ kg}$  and  $m^o = 0.8662\text{ kg}$ . Substituting these values in the mass matrix of Eq. (10.55) yields the following mass matrix:

$$\mathbf{M} = \begin{bmatrix} 0.7834\mathbf{I} & 0.2166\mathbf{I} \\ 0.2166\mathbf{I} & 0.7834\mathbf{I} \end{bmatrix}$$

In the mass distribution process that led to the mass matrix in Eq. (10.54), the primary point that was eliminated was located at the mass center. Now let us consider the case where we eliminate a primary point that is not at the mass center. Assume that for the link shown in Figure 10.18, its mass properties have been distributed to three primary points to obtain  $m^i$ ,  $m^j$ , and  $m^o$ . The primary point  $j$  is positioned at the mass center, but we want to eliminate point  $o$ .

We write the following expression for the mass center, that is, point  $j$ :

$$m\mathbf{r}^j = m^i\mathbf{r}^i + m^j\mathbf{r}^j + m^o\mathbf{r}^o \Rightarrow \mathbf{r}^o - \mu^i\mathbf{r}^i + \mu^j\mathbf{r}^j = \mathbf{0}$$



**FIGURE 10.18**  
Eliminating the primary point  $o$  that is not positioned at the mass center.

$$\mu^i = \frac{m^i}{m^o}, \mu^j = \frac{m^i + m^o}{m^o}$$

Then, the following transformation expression is constructed:

$$\left\{ \begin{array}{c} \mathbf{r}^i \\ \mathbf{r}^j \\ \mathbf{r}^o \end{array} \right\} = \left[ \begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mu^i \mathbf{I} & -\mu^j \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \mathbf{r}^i \\ \mathbf{r}^j \end{array} \right\} \Rightarrow \mathbf{B} = \left[ \begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mu^i \mathbf{I} & -\mu^j \mathbf{I} \end{array} \right]$$

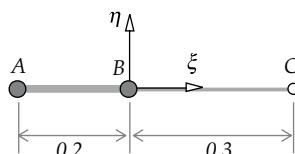
Matrix  $\mathbf{B}$  can now be used to condense the mass matrix of three primary points:

$$\mathbf{M}^{(rod)} = \mathbf{B}' \mathbf{M} \mathbf{B} = \mathbf{B}' \left[ \begin{array}{ccc} m^i \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & m^j \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & m^o \mathbf{I} \end{array} \right] \mathbf{B} = \left[ \begin{array}{cc} \left( m^i + m^o (\mu^i)^2 \right) \mathbf{I} & -m^o \mu^i \mu^j \mathbf{I} \\ -m^o \mu^i \mu^j \mathbf{I} & \left( m^j + m^o (\mu^j)^2 \right) \mathbf{I} \end{array} \right] \quad (10.56)$$

The equations of motion for this system of two primary points will look exactly the same as in Eq. (10.53) except for the mass matrix.

### Example 10.8

Consider the four-bar mechanism of Figure 10.9, where its link (2) is shown in Figure 10.19. The link has a mass  $m = 2.0 \text{ kg}$ , a moment of inertia  $J = 0.08 \text{ kg m}^2$ , and a mass center at  $B$ . Construct the mass matrix for this link by eliminating point  $C$  as a primary point.



**FIGURE 10.19**  
Eliminating the primary point  $o$  from a link.

**Solution**

We first use Eq. (10.52) as

$$\begin{bmatrix} 1 & 1 & 1 \\ -0.2 & 0 & 0.3 \\ (-0.2)^2 & 0 & 0.3^2 \end{bmatrix} \begin{Bmatrix} m^A \\ m^B \\ m^C \end{Bmatrix} = \begin{Bmatrix} 2.0 \\ 0 \\ 0.08 \end{Bmatrix}$$

which yields  $m^A = 0.80\text{ kg}$ ,  $m^B = 0.67\text{ kg}$ , and  $m^C = 0.53\text{ kg}$ . Using these values in the mass matrix of Eq. (10.56), with  $\mu^i = 1.5$  and  $\mu^j = 2.5$ , provides the following mass matrix:

$$\mathbf{M} = \begin{bmatrix} 2.0\mathbf{I} & -2.0\mathbf{I} \\ -2.0\mathbf{I} & 4.0\mathbf{I} \end{bmatrix}$$

It is interesting to note that one of the elements of this mass matrix is larger than the mass of the link itself, and that the off-diagonal terms are negative.

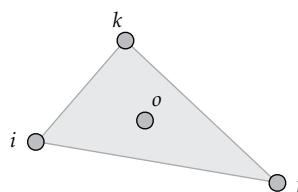
### 10.8.2 Three Primary Points

A planar body is represented by four particles,  $i$ ,  $j$ ,  $k$ , and  $o$ , where  $o$  is positioned at the mass center of the body as shown in Figure 10.20. The mass of the four particles can be computed using Eq. (10.51). Then, through a transformation process similar to that of Section 11.8.1, particle  $o$  can be eliminated to obtain the following mass matrix:

$$\begin{aligned} \mathbf{M}^{(exact)} &= \begin{bmatrix} \mathbf{M}^{i,i} & \mathbf{M}^{i,j} & \mathbf{M}^{i,k} \\ \mathbf{M}^{j,i} & \mathbf{M}^{j,j} & \mathbf{M}^{j,k} \\ \mathbf{M}^{k,i} & \mathbf{M}^{k,j} & \mathbf{M}^{k,k} \end{bmatrix} \\ &= \begin{bmatrix} \left(m^i + m^o(\mu^i)^2\right)\mathbf{I} & m^o\mu^i\mu^j\mathbf{I} & m^o\mu^i\mu^k\mathbf{I} \\ m^o\mu^j\mu^i\mathbf{I} & \left(m^j + m^o(\mu^j)^2\right)\mathbf{I} & m^o\mu^j\mu^k\mathbf{I} \\ m^o\mu^k\mu^i\mathbf{I} & m^o\mu^k\mu^j\mathbf{I} & \left(m^k + m^o(\mu^k)^2\right)\mathbf{I} \end{bmatrix} \quad (10.57) \end{aligned}$$

where

$$\mu^i = \frac{m^i}{m^i + m^j + m^k}, \mu^j = \frac{m^j}{m^i + m^j + m^k}, \mu^k = \frac{m^k}{m^i + m^j + m^k}$$



**FIGURE 10.20**

Mass distribution of a plate to three primary points.

We should note that in the resulting equations of motion for the three particles, in addition to the applied forces, we also have the reaction forces associated with three length constraints.

## 10.9 Force and Mass Addition

A primary point that is shared between two bodies receives its force and mass contributions from both bodies. As an example, the primary point  $C$  that is shared between bodies (1) and (2) shown in Figure 10.21 receives its force and mass contributions from both bodies. The force that acts on body (1),  $\vec{f}_1^P$ , is distributed to the primary points  $A$ ,  $B$ , and  $C$ , based on Eq. (10.42). Similarly, the force  $\vec{f}_2^Q$  that acts on body (2) is distributed to the primary points  $C$  and  $D$ . For these force distributions, the following force arrays can be constructed:

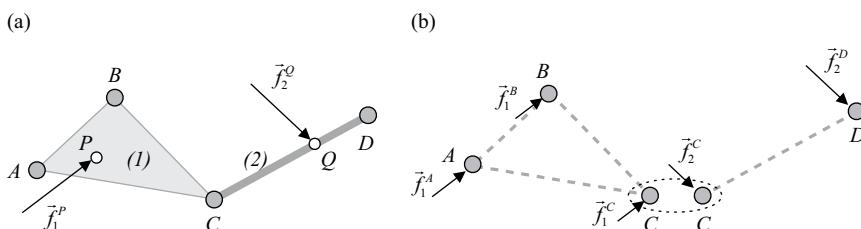
$$\mathbf{f}_1 = \begin{Bmatrix} \mathbf{f}_1^A \\ \mathbf{f}_1^B \\ \mathbf{f}_1^C \end{Bmatrix}, \mathbf{f}_2 = \begin{Bmatrix} \mathbf{f}_2^C \\ \mathbf{f}_2^D \end{Bmatrix} \quad (a)$$

The mass matrix for the primary points  $A$ ,  $B$ , and  $C$ , and that of the primary points  $C$  and  $D$ , can be constructed according to Eqs. (10.57) and (10.54), respectively, as

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{M}_1^{A,A} & \mathbf{M}_1^{A,B} & \mathbf{M}_1^{A,C} \\ \mathbf{M}_1^{B,A} & \mathbf{M}_1^{B,B} & \mathbf{M}_1^{B,C} \\ \mathbf{M}_1^{C,A} & \mathbf{M}_1^{C,B} & \mathbf{M}_1^{C,C} \end{bmatrix}, \mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_2^{C,C} & \mathbf{M}_2^{C,D} \\ \mathbf{M}_2^{D,C} & \mathbf{M}_2^{D,D} \end{bmatrix} \quad (b)$$

Assembling the constructed force arrays and mass matrices yields the equations of motion for this system of four primary points as

$$\begin{bmatrix} \mathbf{M}_1^{A,A} & \mathbf{M}_1^{A,B} & \mathbf{M}_1^{A,C} & \mathbf{0} \\ \mathbf{M}_1^{B,A} & \mathbf{M}_1^{B,B} & \mathbf{M}_1^{B,C} & \mathbf{0} \\ \mathbf{M}_1^{C,A} & \mathbf{M}_1^{C,B} & \mathbf{M}_1^{C,C} + \mathbf{M}_2^{C,C} & \mathbf{M}_2^{C,D} \\ \mathbf{0} & \mathbf{0} & \mathbf{M}_2^{D,C} & \mathbf{M}_2^{D,D} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{r}}^A \\ \ddot{\mathbf{r}}^B \\ \ddot{\mathbf{r}}^C \\ \ddot{\mathbf{r}}^D \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1^A \\ \mathbf{f}_1^B \\ \mathbf{f}_1^C + \mathbf{f}_2^C \\ \mathbf{f}_2^D \end{Bmatrix} + \mathbf{D}'\boldsymbol{\lambda} \quad (10.58)$$



**FIGURE 10.21**

(a) A two-body system represented by four primary points; (b) distribution of forces to the primary points.

where  $\mathbf{D}$  is the Jacobian matrix of four length constraints, and therefore,  $\lambda$  contains four Lagrange multipliers. We should note how the shared primary point  $C$  receives its mass and force contributions from both bodies. A similar process can be applied to other multibody systems that are formulated with the point-coordinate method.

## 10.10 Problems

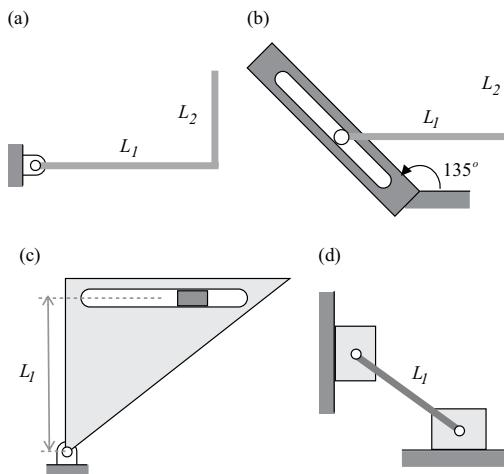
10.1 Consider the simple mechanical systems shown for kinematic modeling with the point-coordinate method. For each system,

Define a nonmoving reference frame at a convenient position and at least two primary points for each system.

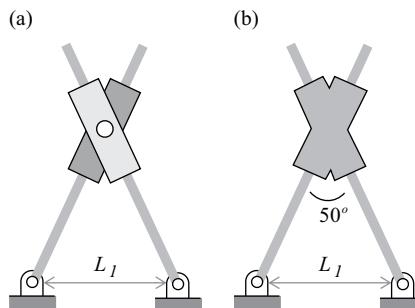
Define an array of coordinates.

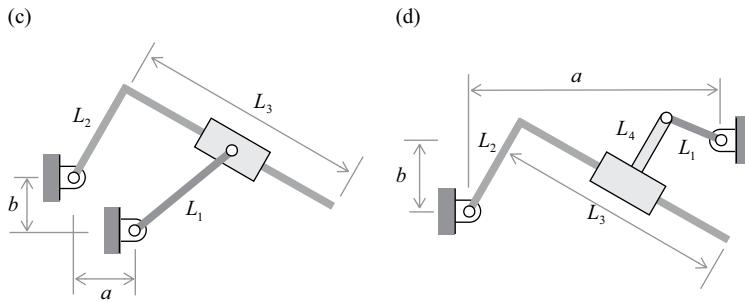
Write the position, velocity, and acceleration constraints in expanded form.

Determine the Jacobian matrix and the right-hand side array of the acceleration constraints.

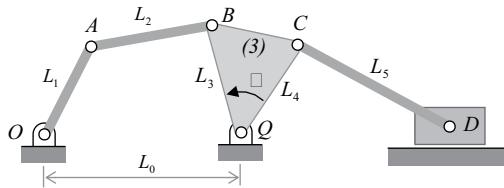


10.2 Use the point-coordinate method to derive the necessary constraints for the systems shown.





- 10.3 If the mechanism shown is considered for modeling with the point-coordinate formulation, determine the minimum number of coordinates and constraints that would represent this system.



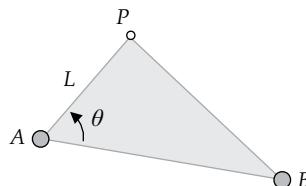
- 10.4 For two parallel vectors, can we use Eq. (10.21) instead of Eq. (10.20) by setting  $\theta = 0$ ? Explain your reasoning.

- 10.5 Derive the velocity and acceleration constraints for the constraint in

- Equation (10.20)
- Equation (10.21)
- Equation (10.22)

- 10.6 Derive the expressions to determine  $\alpha$  and  $\beta$  for the secondary point as shown in Figure 10.8c.

- 10.7 A link is described by two primary points  $A$  and  $B$ . A secondary point  $P$  is positioned on this link by an angle  $\theta$  and a length  $L$  as shown. Derive the expressions similar to those in Eqs. (10.28), (10.30), and (10.31) to compute the position, velocity, and acceleration of  $P$ .



- 10.8 Three points  $A$ ,  $B$ , and  $C$  are defined as the primary points along the axis of a link as shown. Three vectors,  $\mathbf{d}^{A,B} = \mathbf{r}^A - \mathbf{r}^B$ ,  $\mathbf{d}^{A,C} = \mathbf{r}^A - \mathbf{r}^C$ , and  $\mathbf{d}^{B,C} = \mathbf{r}^B - \mathbf{r}^C$ , are used to define three constraints as

$$\mathbf{d}^{A,B'} \mathbf{d}^{A,B} - (L^{A,B})^2 = 0 \quad (a)$$

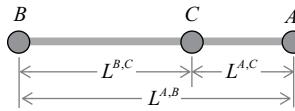
$$\mathbf{d}^{A,C'} \mathbf{d}^{A,C} - (L^{A,C})^2 = 0 \quad (b)$$

$$\check{\mathbf{d}}^{A,C'} \mathbf{d}^{A,B} = 0 \quad (c)$$

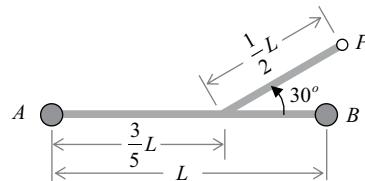
We can also write the following constraint and use it instead of Eq. (c):

$$\mathbf{d}^{B,C'} \mathbf{d}^{B,C} - (L^{B,C})^2 = 0 \quad (d)$$

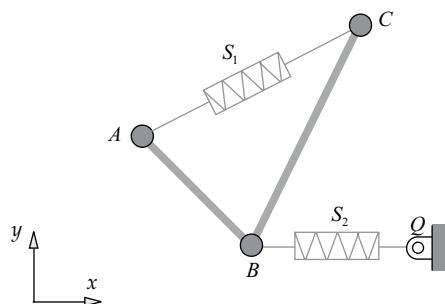
However, using Eq. (d) instead of Eq. (c) may cause numerical difficulties. Explain the reason! Hint: Consider the Jacobian matrix for the three constraints and show that the rows are dependent.



- 10.9 Points  $A$  and  $B$  are the primary points. Find a formulation to express point  $P$  in terms of the coordinates of  $A$  and  $B$ . What are the expressions for the velocity and acceleration of  $P$ ?

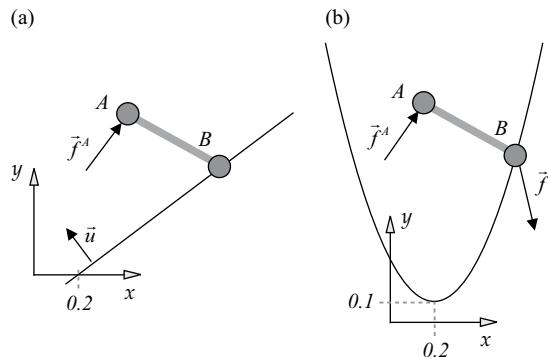


- 10.10 Two rods connected by a pin joint are modeled as three primary points. The mass of the primary points are determined to be  $m^A = 0.5\text{ kg}$ ,  $m^B = 0.8\text{ kg}$ , and  $m^C = 0.6\text{ kg}$ . The rods are  $L^{A,B} = \sqrt{2}\text{ m}$  and  $L^{B,C} = \sqrt{5}\text{ m}$  long. One spring,  $S_1$ , is connected between  $A$  and  $C$ , and one spring,  $S_2$ , connects  $B$  to the ground at  $Q$  having the following data:  $k_1 = 60\text{ N/m}$ ,  $k_2 = 80\text{ N/m}$ ,  ${}^0L_1 = {}^0L_2 = 2.0\text{ m}$ . The spring attachment point to the ground is at  $\mathbf{r}^Q = \{3.5 \quad 0.5\}'\text{ m}$ . In the configuration shown, the coordinates are  $\mathbf{r}^A = \{1.0 \quad 1.5\}'\text{ m}$ ,  $\mathbf{r}^B = \{2.0 \quad 0.5\}'\text{ m}$ , and  $\mathbf{r}^C = \{3.0 \quad 2.5\}'\text{ m}$ , and all the velocities are zero. Construct the equations of motion in numerical form.

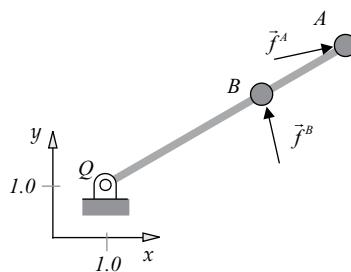


- 10.11 A rod of length  $L = 0.5\text{ m}$  is modeled with two primary points  $A$  and  $B$  having masses  $m^A = 1.0\text{ kg}$  and  $m^B = 2.0\text{ kg}$ . An external force  $\vec{f}^A = \{0.3 \quad 0.4\}'\text{ N}$  acts on point  $A$ . Derive the equations of motion for this system in each of the following conditions:

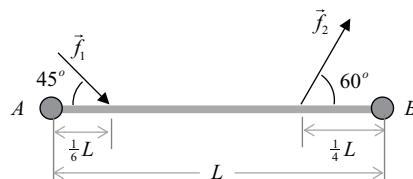
- Point  $B$  is constrained to slide along a straight line as shown. It is assumed that the line intersects the  $x$ -axis at  $x = c = 0.2\text{ m}$  and a unit vector normal to the line is described as  $\vec{u} = \{u_{(x)} \quad u_{(y)}\}' = \{-0.6 \quad 0.8\}'$ .
- Point  $B$  slides on a curve described by  $y = 0.1 + 5(x - 0.2)^2$ . In addition, an external force  $\vec{f}^B = \{0.2 \quad -0.7\}'\text{ N}$  acts on  $B$ .



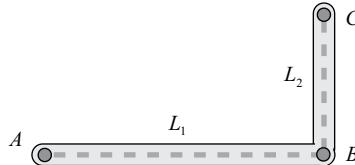
- 10.12 A massless rod of length  $L = 5.0\text{ m}$  is pinned to the ground at point  $Q$ . The rod is modeled with two primary points  $A$  and  $Q$ . The mass of the primary point  $A$  is  $m^A = 1.0\text{ kg}$ . A particle with a mass  $m^B = 2.0\text{ kg}$  can slide along the rod. External forces  $\vec{f}^A = \{0.6 \quad 0.1\}'\text{ N}$  and  $\vec{f}^B = \{-0.1 \quad 0.6\}'\text{ N}$  act on the primary points as shown. Derive the equations of motion for this system.



- 10.13 The mass center of a uniform rod is at its geometric center. Two forces with magnitudes  $f_1 = 100\text{ N}$  and  $f_2 = 200\text{ N}$  act on the rod as shown. The rod is modeled with two primary points  $A$  and  $B$ . Find the force distribution at  $A$  and  $B$ .

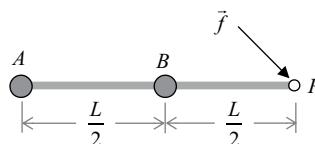


- 10.14 An L-shaped rod with uniform mass distribution is modeled by three primary points as shown. Assume  $L_1 = 2\text{ m}$ ,  $L_2 = 1\text{ m}$ , and  $m = 18\text{ kg}$ . Determine the mass matrix for this system.



- 10.15 A slender rod has a mass  $m$  and a moment of inertia  $J = \frac{1}{12}mL^2$ , where  $L$  is the length of the rod. Two particles may be positioned symmetrically with respect to the mass center, but not at the ends of the rod. Find the distance  $d$  between the two particles for which all the three inertial conditions are satisfied.

- 10.16 The rod shown is described by two primary points  $A$  and  $B$ , where  $B$  is at the mass center of the rod. A force with varying axes and magnitudes acts at point  $P$ . The mass of the rod is  $m$  and its moment of inertia is  $J = \frac{1}{12}mL^2$ .
- Find the coefficients  $\alpha^A$  and  $\alpha^B$  to distribute the force to the two primary points.
  - Find an exact mass matrix for the system of the two particles. *Hint:* Define a third particle at  $P$ , and then follow a process similar to that in Section 10.8.1 to eliminate  $P$ .



- 10.17 In the special case where the three primary points of a planar body form a equilateral triangle, and the mass of the body is distributed symmetrically with respect to the mass center, the masses of the three primary points are equal. For this special case, simplify the mass matrix of Eq. (10.57).

- 10.18 Use the point-coordinate method to construct the equations of motion for the six-bar quick-return mechanism of Section 15.3.
- 10.19 Use the point-coordinate method to construct the equations of motion for the six-bar dwell mechanism of Section 15.4.
- 10.20 Use the point-coordinate method to construct the equations of motion for the double A-arm suspension system of Section 15.6.
- 10.21 Find the exact mass distribution for link (2) of the McPherson suspension in Section 15.7. Consider the three primary points at  $A$ ,  $B$ , and  $C$ .
- 10.22 Use the point-coordinate method to construct the equations of motion for the MacPherson suspension system of Section 15.7.
- 10.23 (Project) Develop a general-purpose forward dynamic analysis program with the point-coordinate formulation similar to the DAP \_ BC in Chapter 8.
- 10.24 (Project) Develop an initial condition correction program similar to the M-file ic \_ correct in DAP \_ BC as an extension of Problem 10.23.

# 11

---

## Contact and Impact

---

Collision between bodies or sudden contact of one body with the ground could occur in some applications of multibody systems. To include a precise and accurate representation of impact or contact in the equations of motion of a system, we must consider the deformation, shape, and possibly other features of the contacting bodies. However, in our discussion of multibody dynamics, we need to combine all these attributes into a very simple and therefore approximate representation. For such simplified representation, we may consider two different approaches. In the first approach, known as the *piecewise* or *intermittent* analysis, we assume that the impact results in an instantaneous change in the velocities. A classical method to determine the change in the velocities considers balancing the system's momenta before and after an impact. In the second approach, known as the *continuous* analysis, we assume that the impact causes the contacting bodies to have local deformation in the contact region. The force of the deformation is represented as a linear or nonlinear spring–damper force element that applies a pair of resistive forces on the two bodies in a continuous form during the period of contact.

Either of the two methods is suitable for computational impact analysis in multibody dynamics. Either method requires accurate determination of the exact time of contact between impacting bodies. Furthermore, implementing the piecewise method in a forward dynamic analysis program requires special attention to several computational issues related to the discontinuities in the velocities. Since most such issues manifest themselves during a forward dynamic analysis, which requires time integration of the equations of motion, the subject is discussed in detail in Chapter 13.

---

### 11.1 Piecewise Analysis

In the piecewise analysis method, it is assumed that when two bodies collide, the contact period is so small that the change in the velocities is instantaneous, and that the position and orientation of the bodies do not change. This assumption leads to discontinuities in the velocities at the moment of contact. Therefore, in a piecewise analysis, we must determine the velocities immediately after an impact based on the velocities before the impact and a parameter describing the characteristic of the impacting bodies. This process is performed using the concept of impulse–momentum. In this section, we first provide a brief description of linear and angular momenta, leading to the necessary formulas to compute the change in the velocities during an impact.

#### 11.1.1 Momentum

For a particle with the mass  $m^i$  and the velocity  $\dot{\mathbf{r}}^i$ , the *linear momentum* is defined as

$$\mathbf{g}^i = m^i \dot{\mathbf{r}}^i \quad (11.1)$$

With this definition, Newton's second law of motion can be restated as

$$\mathbf{f}^i = m^i \ddot{\mathbf{r}}^i = \frac{d}{dt} (m^i \dot{\mathbf{r}}^i) = \dot{\mathbf{g}}^i \quad (11.2)$$

where it is assumed that the mass is a constant. This equation shows that the time rate of change of the linear momentum of a particle is equal to the force that acts on it.

For the rigid body (*i*) with the mass  $m_i$  and the moment of inertia  $J_i$ , the linear momentum and the angular momentum are defined, respectively, as

$$\mathbf{g}_i = m_i \dot{\mathbf{r}}_i \quad (11.3)$$

$$h_i = J_i \dot{\phi}_i \quad (11.4)$$

Appending the angular momentum to the vector of linear momentum forms the array of momenta for the body as

$$\mathbf{m}_i = \begin{Bmatrix} \mathbf{g}_i \\ h_i \end{Bmatrix} = \begin{Bmatrix} m_i \dot{\mathbf{r}}_i \\ J_i \dot{\phi}_i \end{Bmatrix} = \begin{bmatrix} m_i \mathbf{I} & \mathbf{0} \\ \mathbf{0} & J_i \end{bmatrix} \dot{\mathbf{c}}_i \quad (11.5)$$

In the following sections, we apply the definitions of linear and angular momenta to derive the necessary formulas for determining the abrupt change in velocities caused by an impact.

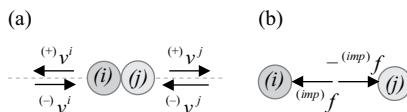
### 11.1.2 Impact of Particles

The impact of two particles that move along a single axis is called *direct central* impact. Assume particles *i* and *j* shown in Figure 11.1a have velocities  $(-) \vec{v}^i$  and  $(-) \vec{v}^j$  at  $t = (-)t$  as they become in contact. As the particles separate at  $t = (+)t$ , let us assume that their velocities have become  $(+) \vec{v}^i$  and  $(+) \vec{v}^j$ . The ratio between the relative velocities at  $t = (+)t$  and  $t = (-)t$  is called the *coefficient of restitution*:

$$e = -\frac{(+) \vec{v}^i - (+) \vec{v}^j}{(-) \vec{v}^i - (-) \vec{v}^j} \quad (11.6)$$

This coefficient is a positive quantity between zero and one. The impact is said to be *elastic* if  $e = 1$  and *inelastic* or *plastic* when  $e = 0$ .

During the period of contact,  $\Delta t = (+)t - (-)t$ , a pair of impact forces acts on the particles as shown in Figure 11.1b. Assuming that the particles have masses  $m^i$  and  $m^j$ , and accelerations  $\vec{a}^i = \ddot{\vec{v}}^i$  and  $\vec{a}^j = \ddot{\vec{v}}^j$ , their equations of motion during the period of contact, along the single axis of motion, can be expressed as



**FIGURE 11.1**

(a) Direct central impact of two particles and (b) the resultant impact forces.

$$m^i a^i = f^i + {}^{(imp)}f, m^j a^j = f^j - {}^{(imp)}f$$

where  $\vec{f}^i$  and  $\vec{f}^j$  are the applied forces that act on the particles (not shown on the figure), and  ${}^{(imp)}\vec{f}$  is the impact force caused by local deformations of the two particles. We integrate the equations of motion for the period of contact,  $\Delta t = {}^{(+)}t - {}^{(-)}t$ :

$$\int_{(-)t}^{(+)} m^i a^i dt = \int_{(-)t}^{(+)} f^i dt + \int_{(-)t}^{(+)} {}^{(imp)}f dt, \int_{(-)t}^{(+)} m^j a^j dt = \int_{(-)t}^{(+)} f^j dt - \int_{(-)t}^{(+)} {}^{(imp)}f dt$$

or,

$$m^i ({}^{(+)}v^i - {}^{(-)}v^i) = \int_{(-)t}^{(+)} {}^{(imp)}f dt, m^j ({}^{(+)}v^j - {}^{(-)}v^j) = - \int_{(-)t}^{(+)} {}^{(imp)}f dt \quad (11.7)$$

If we assume that the period of contact is very short, the applied forces remain unchanged, and therefore,  $\int_{(-)t}^{(+)} f^i dt = 0$  and  $\int_{(-)t}^{(+)} f^j dt = 0$ . The integral of the impact force during the period of contact is defined as

$$\pi = \int_{(-)t}^{(+)} {}^{(imp)}f dt \quad (11.8)$$

where the product of force and time is called *impulse*. We combine the two equations in Eq. (11.7), leading to the following equation known as the *conservation of linear momentum*:

$$m^i ({}^{(+)}v^i - {}^{(-)}v^i) + m^j ({}^{(+)}v^j - {}^{(-)}v^j) = 0 \quad (11.9)$$

or

$$m^i \Delta v^i + m^j \Delta v^j = 0 \quad (11.10)$$

where  $\Delta v^i = {}^{(+)}v^i - {}^{(-)}v^i$  and  $\Delta v^j = {}^{(+)}v^j - {}^{(-)}v^j$ .

### Example 11.1

Two particles having masses  $m^i = 0.2$  kg and  $m^j = 0.4$  kg undergo a direct central impact with velocities  ${}^{(-)}v^i = 3.0$  m/s and  ${}^{(-)}v^j = 3.0$  m/s. Determine the velocities after impact if the coefficient of restitution is  $e = 0.5$ .

#### Solution

The definition of coefficient of restitution from Eq. (11.6) provides the following:

$${}^{(+)}v^i - {}^{(+)}v^j = -0.5(3.0 + 2.0) \text{ m/s}$$

The conservation of linear momentum from Eq. (11.9) yields a second identity as

$$0.2({}^{(+)}v^i - 3.0) + 0.4({}^{(+)}v^j + 2.0) = 0$$

Solving the two equations yields the following velocities:

$${}^{(+)}\dot{v}^i = -2.0 \text{ m/s}, {}^{(+)}\dot{v}^j = 0.5 \text{ m/s}$$

When two particles impact each other in an angle different than zero, as shown in Figure 11.2a, the impact is referred to as *oblique central*. Let us assume that the particles have velocities  $(-) \dot{\vec{r}}^i$  and  $(-) \dot{\vec{r}}^j$  as they impact, and  $(+) \dot{\vec{r}}^i$  and  $(+) \dot{\vec{r}}^j$  as they separate. In such an impact, we may assume that the components of velocities along the tangent axis are not affected by the impact, but the components along the normal axis, denoted by the unit vector  $\vec{u}$ , are affected. Therefore, the definition of the coefficient of restitution from Eq. (11.6) is revised for the relative velocities along the normal axis as

$$e = -\frac{\mathbf{u}'^{(+)} \mathbf{v}^{i,j}}{\mathbf{u}'^{(-)} \mathbf{v}^{i,j}} \quad (11.11)$$

where  $\mathbf{v}^{i,j} = \dot{\mathbf{r}}^i - \dot{\mathbf{r}}^j$  is the relative velocity between the particles. We denote the change in the velocities before and after impact as

$$\Delta \dot{\mathbf{r}}^i = {}^{(+)}\dot{\mathbf{r}}^i - {}^{(+)}\dot{\mathbf{r}}^i, \Delta \dot{\mathbf{r}}^j = {}^{(+)}\dot{\mathbf{r}}^j - {}^{(+)}\dot{\mathbf{r}}^j \quad (11.12)$$

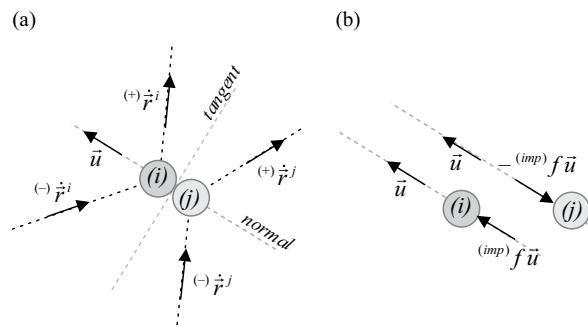
With this description, Eq. (11.11) is expressed in another form as

$$e + 1 = -\frac{\mathbf{u}'^{(+)} \mathbf{v}^{i,j}}{\mathbf{u}'^{(-)} \mathbf{v}^{i,j}} + 1 = \frac{-\mathbf{u}'(\Delta \dot{\mathbf{r}}^i - \Delta \dot{\mathbf{r}}^j)}{\mathbf{u}'^{(-)} \mathbf{v}^{i,j}}$$

or

$$\mathbf{u}'(\Delta \dot{\mathbf{r}}^i - \Delta \dot{\mathbf{r}}^j) = -(e + 1)\mathbf{u}'^{(-)} \mathbf{v}^{i,j} \quad (11.13)$$

This equation expresses the change in the velocities as a function of the coefficient of restitution and the velocities before impact.



**FIGURE 11.2**

(a) Oblique central impact of two particles and (b) the resultant impact forces.

During the period of contact, the particles apply forces on each other in opposite directions along the normal axis, as depicted in Figure 11.2b. Assuming masses  $m^i$  and  $m^j$  for the particles, the equations of motion during the period of contact can be expressed as

$$m^i \ddot{\mathbf{r}}^i = \mathbf{f}^i + {}^{(imp)}\mathbf{f} \mathbf{u}, m^j \ddot{\mathbf{r}}^j = \mathbf{f}^j - {}^{(imp)}\mathbf{f} \mathbf{u}$$

where  $\vec{f}^i$  and  $\vec{f}^j$  are the applied forces (not shown on the figures). We integrate the equations of motion for the period of contact:

$$\int_{(-)_t}^{(+)_t} m^i \ddot{\mathbf{r}}^i dt = \int_{(-)_t}^{(+)_t} \mathbf{f}^i dt + \int_{(-)_t}^{(+)_t} {}^{(imp)}\mathbf{f} \mathbf{u} dt, \int_{(-)_t}^{(+)_t} m^j \ddot{\mathbf{r}}^j dt = \int_{(-)_t}^{(+)_t} \mathbf{f}^j dt - \int_{(-)_t}^{(+)_t} {}^{(imp)}\mathbf{f} \mathbf{u} dt \quad (11.14)$$

Considering that the applied forces do not vary during the short period of contact, it yields  $\int_{(-)_t}^{(+)_t} \mathbf{f}^i dt = 0$  and  $\int_{(-)_t}^{(+)_t} \mathbf{f}^j dt = 0$ . Referring to the definition of linear impulse in Eq. (11.8), Eq. (11.14) becomes

$$m^i \left( {}^{(+)}\dot{\mathbf{r}}^i - {}^{(-)}\dot{\mathbf{r}}^i \right) = \pi \mathbf{u}, m^j \left( {}^{(+)}\dot{\mathbf{r}}^j - {}^{(-)}\dot{\mathbf{r}}^j \right) = -\pi \mathbf{u}$$

or

$$m^i \Delta \dot{\mathbf{r}}^i = \pi \mathbf{u}, m^j \Delta \dot{\mathbf{r}}^j = -\pi \mathbf{u} \quad (11.15)$$

The preceding equation yields the equation of *conservation of linear momentum* for a system of two particles as

$$m^i \Delta \dot{\mathbf{r}}^i + m^j \Delta \dot{\mathbf{r}}^j = \mathbf{0} \quad (11.16)$$

Equation (11.13) can be appended to Eq. (11.15) and arranged in matrix form as

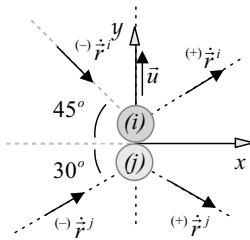
$$\left[ \begin{array}{cc|c} m^i \mathbf{I} & \mathbf{0} & -\mathbf{u} \\ \mathbf{0} & m^j \mathbf{I} & \mathbf{u} \\ \hline \mathbf{u}' & -\mathbf{u}' & 0 \end{array} \right] \left\{ \begin{array}{c} \Delta \dot{\mathbf{r}}^i \\ \Delta \dot{\mathbf{r}}^j \\ \pi \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ -(e+1)\mathbf{u}' {}^{(-)}\mathbf{v}^{i,j} \end{array} \right\} \quad (11.17)$$

Knowing the positions and the velocities before impact, this equation can be solved to obtain the impulse as well as the change in the velocities.

### Example 11.2

Two particles having masses  $m^1 = 0.2$  kg and  $m^2 = 0.3$  kg, and velocities  $\mathbf{v}^1 = \{2.0 - 2.0\}'$  m/s and  $\mathbf{v}^2 = \{1.5 3.0\}'$  m/s undergo an oblique impact as shown in Figure 11.3. For two cases,  $e = 0.6$  and  $e = 1.0$ :

- Determine the velocities after impact.
- Compare the total energy of the system before and after impact.

**FIGURE 11.3**

Oblique central impact of two particles.

***Solution***

We develop a MATLAB® program to perform the necessary computation and to evaluate the kinetic energy of the system before and after impact.

**MATLAB/Chapter 11/Example\_11\_2**

We first provide the constant data and the velocities before the impact.

```
m1 = 0.2; m2 = 0.3; e = 0.6; u = [0; 1];
r1_d_m = [2; -2]; r2_d_m = [1.5; 3];
```

We formulate Eq. (11.17) to determine the change in the velocities. The velocities after impact are  $r1_d_p$  and  $r2_d_p$ . We may also report the computed impulse.

```
DMD = [m1*eye(2) zeros(2) -u
        zeros(2) m2*eye(2) u
        u' -u' 0];
rhs = [zeros(4,1);
       -(e + 1)*u'*(r1_d_m - r2_d_m)];
sol = DMD\rhs;
r1_d_p = r1_d_m + sol(1:2) ..... r1_d_p =
r2_d_p = r2_d_m + sol(3:4) ..... r2_d_p =
impulse = sol(5) ..... impulse =
```

2.0000
2.8000
1.5000
-0.2000
0.9600

Next we compute the kinetic energy of the system before and after impact.

```
energy_m = (m1*r1_d_m'*r1_d_m + ...
            m2*r2_d_m'*r2_d_m)/2 ..... energy_m =
energy_p = (m1*r1_d_p'*r1_d_p + ...
            m2*r2_d_p'*r2_d_p)/2 ..... energy_p =

```

2.4875
1.5275

We note that there is a loss in the energy when  $e = 0.6$ . If we execute the program for  $e = 1.0$ , we will observe that there is no energy loss in an elastic impact.

### 11.1.3 Unconstrained Bodies

Two bodies, whether free or part of a multibody system, may collide with each other as they move. Let us assume that  $P_i$  and  $P_j$  are the contact points on the two bodies, as shown in Figure 11.4. At the instant of contact, the velocity of the bodies undergoes a discontinue change, and therefore, we need to determine the velocities immediately after the impact.

We denote the coordinates, velocities, and accelerations of the two bodies as

$$\mathbf{c}_k = \begin{Bmatrix} \mathbf{r}_k \\ \phi_k \end{Bmatrix}, \dot{\mathbf{c}}_k = \begin{Bmatrix} \dot{\mathbf{r}}_k \\ \dot{\phi}_k \end{Bmatrix}, \ddot{\mathbf{c}}_k = \begin{Bmatrix} \ddot{\mathbf{r}}_k \\ \ddot{\phi}_k \end{Bmatrix} \quad \text{for } k = i \text{ and } j$$

The relative velocity vector between points  $P_i$  and  $P_j$  can be determined as

$$\mathbf{v}_{i,j}^P = \dot{\mathbf{r}}_i^P - \dot{\mathbf{r}}_j^P = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{s}}_i^P \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{r}}_i \\ \dot{\phi}_i \end{Bmatrix} - \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{s}}_j^P \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{r}}_j \\ \dot{\phi}_j \end{Bmatrix} = \mathbf{D}_{i,j} \begin{Bmatrix} \dot{\mathbf{c}}_i \\ \dot{\mathbf{c}}_j \end{Bmatrix} \quad (11.18)$$

where

$$\mathbf{D}_{i,j} = \left[ \begin{array}{cc|cc} \mathbf{I} & \tilde{\mathbf{s}}_i^P & -\mathbf{I} & -\tilde{\mathbf{s}}_j^P \end{array} \right] \quad (11.19)$$

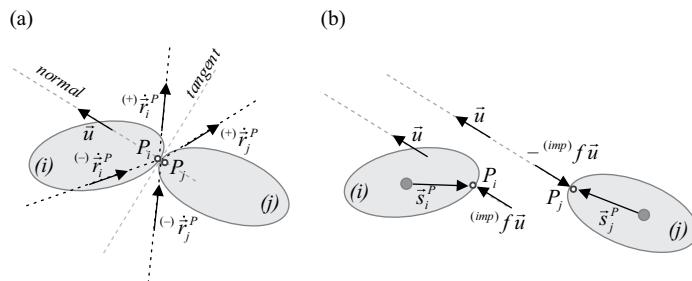
The relative velocity vector,  $\vec{v}_{i,j}^P$ , can be projected on the axis,  $\vec{u}$ , normal to the contact plane (contact line in planar view), and also on the tangential axis. Assuming that only the normal component of the relative velocity is affected by the impact, we restate the definition of coefficient of restitution between two bodies as

$$e = -\frac{\mathbf{u}'^{(+)} \mathbf{v}_{i,j}^P}{\mathbf{u}'^{(-)} \mathbf{v}_{i,j}^P}$$

A slight manipulation of this equation, similar to that of Eq. (11.13), yields

$$\mathbf{u}'(\Delta \dot{\mathbf{r}}_i^P - \Delta \dot{\mathbf{r}}_j^P) = -(e+1) \mathbf{u}'^{(-)} \mathbf{v}_{i,j}^P \quad (11.20)$$

where  $\Delta \dot{\mathbf{r}}_i^P = {}^{(+)}\dot{\mathbf{r}}_i^P - {}^{(-)}\dot{\mathbf{r}}_i^P$  and  $\Delta \dot{\mathbf{r}}_j^P = {}^{(+)}\dot{\mathbf{r}}_j^P - {}^{(-)}\dot{\mathbf{r}}_j^P$ .



**FIGURE 11.4**

(a) Impact of two bodies and (b) the resultant impact forces during contact.

Equations of motion for the two bodies during the period of contact are expressed as

$$\left[ \begin{array}{cc|cc} m_i \mathbf{I} & \mathbf{0} & \ddot{\mathbf{r}}_i & {}^{(a)}\mathbf{f}_i \\ \mathbf{0} & J_i & \ddot{\phi}_i & {}^{(a)}n_i \\ \hline m_j \mathbf{I} & \mathbf{0} & \ddot{\mathbf{r}}_j & {}^{(a)}\mathbf{f}_j \\ \mathbf{0} & J_j & \ddot{\phi}_j & {}^{(a)}n_j \end{array} \right] = \left\{ \begin{array}{l} {}^{(a)}\mathbf{f}_i \\ {}^{(a)}n_i \\ {}^{(a)}\mathbf{f}_j \\ {}^{(a)}n_j \end{array} \right\} + {}^{(imp)}\mathbf{f} \left[ \begin{array}{c} \mathbf{I} \\ \check{\mathbf{s}}_i^P \\ -\mathbf{I} \\ -\check{\mathbf{s}}_j^P \end{array} \right] \mathbf{u}$$

where the moments caused by impact forces are also included in the rotational equations of motion. Noting that the coefficient matrix of the impact force is the transpose of the matrix in Eq. (11.19), the two sets of equations of motion can be expressed in compact form as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + {}^{(imp)}\mathbf{f} \mathbf{D}'_{i,j} \mathbf{u} \quad (11.21)$$

In this equation,  $\mathbf{M}$  is a constant  $6 \times 6$  mass matrix,  $\ddot{\mathbf{c}}$  contains six body accelerations, and  ${}^{(a)}\mathbf{h}$  is a  $6 \times 1$  array of applied loads (the applied forces and torques are not shown in the figure).

We integrate the equations of motion for the period of contact:

$$\int_{(-)_t}^{(+)_t} \mathbf{M}\ddot{\mathbf{c}} dt = \int_{(-)_t}^{(+)_t} {}^{(a)}\mathbf{h} dt + \int_{(-)_t}^{(+)_t} {}^{(imp)}\mathbf{f} \mathbf{D}'_{i,j} \mathbf{u} dt$$

Since the mass matrix  $\mathbf{M}$  is a constant, and based on the assumption that during the short period of contact the position and orientation of bodies do not change, the mass matrix  $\mathbf{M}$  and the product  $\mathbf{D}'_{i,j} \mathbf{u}$  can be taken out of the integrals. Furthermore, noting that the applied loads remain unchanged during contact yields  $\int_{(-)_t}^{(+)_t} {}^{(a)}\mathbf{h} dt = \mathbf{0}$ . Hence, the integral equation becomes

$$\mathbf{M} \int_{(-)_t}^{(+)_t} \ddot{\mathbf{c}} dt = \mathbf{D}'_{i,j} \mathbf{u} \int_{(-)_t}^{(+)_t} {}^{(imp)}\mathbf{f} dt$$

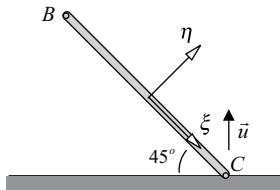
Using the definition of impulse from Eq. (11.8), this equation is expressed as

$$\mathbf{M}\Delta\dot{\mathbf{c}} - \pi \mathbf{D}'_{i,j} \mathbf{u} = \mathbf{0} \quad (11.22)$$

where  $\Delta\dot{\mathbf{c}} = {}^{(+)}\dot{\mathbf{c}} - {}^{(-)}\dot{\mathbf{c}}$ . We append Eq. (11.20) to (11.22) and arrange them in matrix form:

$$\left[ \begin{array}{cc} \mathbf{M} & -\mathbf{D}'_{i,j} \mathbf{u} \\ \mathbf{u}' \mathbf{D}_{i,j} & \mathbf{0} \end{array} \right] \left\{ \begin{array}{l} \Delta\dot{\mathbf{c}} \\ \pi \end{array} \right\} = \left\{ \begin{array}{l} \mathbf{0} \\ -(e+1)\mathbf{u}' \mathbf{D}_{i,j} {}^{(-)}\dot{\mathbf{c}} \end{array} \right\} \quad (11.23)$$

This equation can now be solved to determine the jump in the velocities of the two bodies caused by the impact. We can then determine the velocities after impact as  ${}^{(+)}\dot{\mathbf{c}} = {}^{(-)}\dot{\mathbf{c}} + \Delta\dot{\mathbf{c}}$ .

**FIGURE 11.5**

A slender rod striking the ground.

**Example 11.3**

A slender rod of length 2.0 m, mass  $m = 1.0 \text{ kg}$ , and moment of inertia  $J = 0.01 \text{ kg m}^2$  strikes the ground in the orientation shown in Figure 11.5. At the instant of impact, the rod has a linear velocity  $(\dot{\mathbf{r}}) = \{0 \quad -6.5\}' \text{ m/s}$  and an angular velocity  $(\dot{\phi}) = 0$ . Determine the velocity of the rod immediately after the impact, assuming  $e = 0.95$ .

**Solution**

In the shown orientation, the attached body-fixed frame has an angle  $\phi = 315^\circ$ . The contact point has local coordinates  $\mathbf{s}^C = \{0.5 \quad 0\}' \text{ m}$ . The velocity of point C is computed as  $\dot{\mathbf{r}}^C = [\mathbf{I} \quad \dot{\mathbf{s}}^C] \dot{\mathbf{c}}$ , leading to  $\mathbf{D}_{i,j} = [\mathbf{I} \quad \dot{\mathbf{s}}^C]$ . The unit vector normal to the plane of contact is  $\mathbf{u} = \{0 \quad 1\}'$ .

**MATLAB/Chapter 11/Example\_11\_3**

We provide the constant data and the velocities before impact.

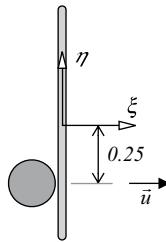
```
L = 2; m = 1; J = 0.01;
e = 0.95; u = [0; 1]; sC_local = [L/2; 0];
phi = 315*pi/180; c_d_m = [0; -6.5; 0];
```

We formulate Eq. (11.23) to determine the change in the velocities and compute the velocity of the contact point after impact.

```
M_array = [m m J]; M = diag(M_array);
A = A_matrix(phi); sC = A*sC_local;
Dij = [eye(2) s_rot(sC)];
DMD = [M -Dij'*u
        u'*Dij      0];
rhs = [zeros(3,1);
       -(e + 1)*u'*Dij*c_d_m];
sol = DMD\rhs;
c_d_p = c_d_m + sol(1:3) .....
```

rC_d_p =	0
-6.2515	
17.5737	
rC_d_p =	12.4265
6.1750	

We note that the velocity of the body in the  $x$ -direction has remained unchanged, but its angular velocity has changed from 0 to 17.5737 rad/s.



**FIGURE 11.6**  
A disc striking a slender rod.

### Example 11.4

A disc of radius  $R = 0.1$  m strikes a slender rod of length  $L = 1.0$  m in the configuration shown in Figure 11.6. The disc has a mass of  $m_1 = 0.5$  kg and its moment of inertia can be computed as  $J_1 = \frac{1}{2}m_1R^2$ . The mass of the rod is  $m_2 = 0.6$  kg and a moment of inertia can be determined as  $J_2 = \frac{1}{12}m_2L^2$ . At the instant of impact, the linear and angular velocities of the disc and the rod are  $(^{\text{-}})\dot{\mathbf{r}}_1 = \{2 \quad 1\}'$  m/s,  $(^{\text{-}})\dot{\phi}_1 = 0.5$  rad/s,  $(^{\text{-}})\dot{\mathbf{r}}_2 = \{0 \quad -1\}'$  m/s, and  $(^{\text{-}})\dot{\phi}_2 = 0$ , respectively. Assume a coefficient of restitution of  $e = 0.8$ . Determine the velocity of the bodies immediately after impact.

#### Solution

The contact points on the two bodies have local coordinates  $\mathbf{s}_1^C = \{0.1 \quad 0\}'$  m and  $\mathbf{s}_2^C = \{0 \quad -0.25\}'$  m. We assume that both body frames are parallel to the  $x$ - $y$  frame, and therefore, both rotational transformation matrices are identities. The velocity of each contact point is computed as  $\dot{\mathbf{r}}_1^C = [\mathbf{I} \quad \mathbf{s}_1^C]\dot{\mathbf{c}}_1$  and  $\dot{\mathbf{r}}_2^C = [\mathbf{I} \quad \mathbf{s}_2^C]\dot{\mathbf{c}}_2$ , leading to  $\mathbf{D}_{i,j} = \left[ \begin{array}{cc|cc} \mathbf{I} & \mathbf{s}_1^C & -\mathbf{I} & -\mathbf{s}_2^C \end{array} \right]$ . The unit vector normal to the plane of contact, in the shown configuration, is  $\mathbf{u} = \{1 \quad 0\}'$ .

#### MATLAB/Chapter 11/Example \_ 11 \_ 4

We provide the constant data and the velocities before impact.

```
R = 0.1; m1 = 0.5; J1 = m1*R^2/2;
L = 1; m2 = 0.6; J2 = m2*L^2/12;
e = 0.8; u = [1; 0];
sC1_local = [R; 0]; sC2_local = [0; L/4];
c1_d_m = [2; 1; 0.5]; c2_d_m = [0; -1; 0];
c_d_m = [c1_d_m; c2_d_m];
```

We formulate Eq. (11.23) to determine the change in the velocities and compute the velocity of the contact points after impact.

```
M1_array = [m1 m1 J1]; M2_array = [m2 m2 J2];
M = diag([M1_array M2_array]);
A1 = eye(2); A2 = eye(2);
sC1 = A1*sC1_local; sC2 = A2*sC2_local;
D1 = [eye(2) s_rot(sC1)];
D2 = [eye(2) s_rot(sC2)];
Dij = [D1 -D2];
```

```

DMD = [M      -Dij'*u
       u'*Dij      0];
rhs = [zeros(6,1)
       -(e + 1)*u'*Dij*c_d_m];
sol = DMD\rhs;
c1_d_p = c1_d_m + sol(1:3) .....
c2_d_p = c2_d_m + sol(4:6) .....
c1_d_p =
0.5356
1.0000
0.5000
c2_d_p =
1.2203
-1.0000
-3.6610

```

We note that the velocities in the tangential direction ( $y$ -direction) have remained unchanged.

#### 11.1.4 Constrained Bodies

In a multibody system, let us assume that bodies ( $i$ ) and ( $j$ ) impact each other at  $t = {}^{(-)}t$ . At this instant, the array of body velocities must satisfy the velocity constraints as

$$\mathbf{D}^{(-)}\dot{\mathbf{c}} = \mathbf{0}$$

Denoting the jump in the velocities due to impact as  $\Delta\dot{\mathbf{c}}$ , the velocities at  $t = {}^{(+)}t$  become

$${}^{(+)}\dot{\mathbf{c}} = {}^{(-)}\dot{\mathbf{c}} + \Delta\dot{\mathbf{c}}$$

With the assumption that during the short period of contact, the coordinates do not change, the array of velocities after impact must also satisfy the velocity constraints:

$$\mathbf{D}\left({}^{(-)}\dot{\mathbf{c}} + \Delta\dot{\mathbf{c}}\right) = \mathbf{0}$$

This equation leads to

$$\mathbf{D}\Delta\dot{\mathbf{c}} = \mathbf{0} \quad (11.24)$$

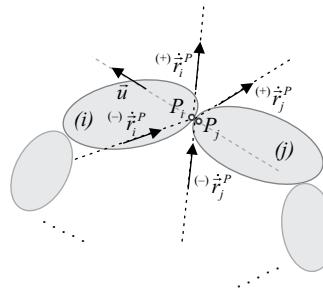
During contact, bodies ( $i$ ) and ( $j$ ) apply impact forces on each other at points  $P_i$  and  $P_j$  along an axis  $\bar{u}$ , normal to the tangent plane as shown in Figure 11.7. The relative velocity between the two contact points can be determined as in Eq. (11.18). The corresponding coefficient matrix  $\mathbf{D}_{i,j}$  of Eq. (11.19) must slightly be modified to include additional columns with zero entries for the remaining bodies of the system:

$$\mathbf{D}_{i,j} = \left[ \begin{array}{c|c|c|c|c}
\cdots & \mathbf{I} & \tilde{\mathbf{s}}_i^P & \cdots & -\mathbf{I} & -\tilde{\mathbf{s}}_j^P & \cdots
\end{array} \right] \quad (11.25)$$

When bodies ( $i$ ) and ( $j$ ) are not in contact, the equations of motion for the system (in body-coordinate) are expressed as

$$\mathbf{M}\ddot{\mathbf{c}} - \mathbf{D}'\lambda = {}^{(a)}\mathbf{h}$$

where  $\mathbf{D}$  represents the Jacobian matrix of all the joints in the system. The equations of motion during the contact period can be expressed as



**FIGURE 11.7**  
Two bodies of a multibody system impacting each other.

$$\mathbf{M}\ddot{\mathbf{c}} - \mathbf{D}'\lambda = {}^{(a)}\mathbf{h} + {}^{(imp)}f \mathbf{D}'_{i,j} \mathbf{u}$$

We integrate the equations of motion for the period of contact:

$$\int_{(-)_t}^{(+)_t} \mathbf{M}\ddot{\mathbf{c}} dt - \int_{(-)_t}^{(+)_t} \mathbf{D}'\lambda dt = \int_{(-)_t}^{(+)_t} {}^{(a)}\mathbf{h} dt + \int_{(-)_t}^{(+)_t} {}^{(imp)}f \mathbf{D}'_{i,j} \mathbf{u} dt$$

Noting that during the short period of contact,  $\mathbf{M}$ ,  $\mathbf{D}'$ ,  ${}^{(a)}\mathbf{h}$ , and  $\mathbf{D}'_{i,j} \mathbf{u}$  do not change, the integral equation can be simplified to

$$\mathbf{M} \int_{(-)_t}^{(+)_t} \ddot{\mathbf{c}} dt - \mathbf{D}' \int_{(-)_t}^{(+)_t} \lambda dt = \mathbf{D}'_{i,j} \mathbf{u} \int_{(-)_t}^{(+)_t} {}^{(imp)}f dt$$

Integration for the short period of contact yields

$$\mathbf{M} \left( {}^{(+)}\dot{\mathbf{c}} - {}^{(-)}\dot{\mathbf{c}} \right) - \mathbf{D}' \left( {}^{(+)}\boldsymbol{\sigma} - {}^{(-)}\boldsymbol{\sigma} \right) = \pi \mathbf{D}'_{i,j} \mathbf{u}$$

or

$$\mathbf{M}\Delta\dot{\mathbf{c}} - \mathbf{D}'\Delta\boldsymbol{\sigma} - \pi\mathbf{D}'_{i,j} \mathbf{u} = \mathbf{0} \quad (11.26)$$

where  $\boldsymbol{\sigma} = \int \lambda dt$  and  $\Delta\boldsymbol{\sigma} = {}^{(+)}\boldsymbol{\sigma} - {}^{(-)}\boldsymbol{\sigma}$ . We append Eqs. (11.20) and (11.24) to Eq. (11.26), and arrange the terms in matrix form to obtain the following equation:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{D}' & -\mathbf{D}'_{i,j} \mathbf{u} \\ \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{u}' \mathbf{D}_{i,j} & \mathbf{0} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\dot{\mathbf{c}} \\ \Delta\boldsymbol{\sigma} \\ \pi \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ -(e+1)\mathbf{u}' \mathbf{D}_{i,j} {}^{(-)}\dot{\mathbf{c}} \end{Bmatrix} \quad (11.27)$$

This equation can be solved to determine the jump in the velocities for all the bodies in the system.

### 11.1.5 Impact with Friction

So far in our discussion of piecewise analysis, we have made the assumption that the tangential component of the relative velocity of the contacting points is not affected by the impact. We can relax this assumption and assume that, in the presence of friction, the tangential component of the relative velocity is affected in an impact. To consider the friction force in the tangential plane, we recall from the discussion on friction in Section 4.5 that the friction force could be determined as a function of the normal force, the relative *tangential* velocity, and friction parameters as provided in Eq. (4.43) or (4.44). Here, for notational simplification purpose, we represent the friction force as  $(^f)\vec{f} = \mu f_N$ , where  $\mu$  must be determined from either of those two or other similar formulas.

The relative velocity of the contact points in the tangential direction can be computed as

$${}^t v_{i,j}^P = \check{\mathbf{u}}' \mathbf{v}_{i,j}^P = \check{\mathbf{u}}' (\dot{\mathbf{r}}_i^P - \dot{\mathbf{r}}_j^P)$$

During the period of contact, the normal contact force that acts along the normal axis is  $(^{imp})\vec{f}\vec{u}$ , and therefore, the friction force can be defined as  $(^f)\vec{f} = \mu (^{imp})\vec{f}\vec{u}$ , as illustrated in Figure 11.8. To include the friction force in the equations of motion of the two bodies, Eq. (11.21) is revised to

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + (^{imp})\vec{f}\mathbf{D}'_{i,j}(\mathbf{u} + \mu\check{\mathbf{u}}) \quad (11.28)$$

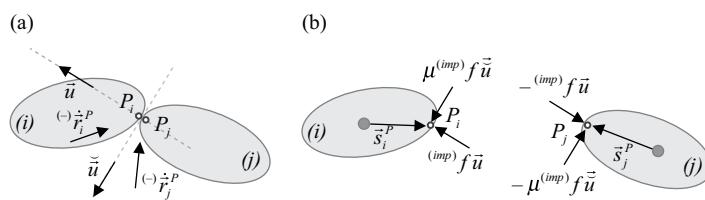
Applying a similar integration process on this equation, as we performed on Eq. (11.21), results in the following equation:

$$\mathbf{M}\Delta\dot{\mathbf{c}} - \pi\mathbf{D}'_{i,j}(\mathbf{u} + \mu\check{\mathbf{u}}) = \mathbf{0} \quad (11.29)$$

Appending Eqs. (11.20)–(11.29) provides the revised form of Eq. (11.23) as

$$\left[ \begin{array}{cc} \mathbf{M} & -\mathbf{D}'_{i,j}(\mathbf{u} + \mu\check{\mathbf{u}}) \\ \mathbf{u}'\mathbf{D}_{i,j} & \mathbf{0} \end{array} \right] \left\{ \begin{array}{c} \Delta\dot{\mathbf{c}} \\ \pi \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{0} \\ -(e+1)\mathbf{u}'\mathbf{D}_{i,j}(^{(-)}\dot{\mathbf{c}}) \end{array} \right\} \quad (11.30)$$

This equation can be solved to determine the jump in the velocities of two impacting bodies in the presence of friction. Similar revision can be implemented in Eq. (11.27) to include contact friction for a multibody system.



**FIGURE 11.8**

(a) Two impacting bodies and (b) the resultant impact and friction forces.

## 11.2 Continuous Analysis

In the continuous analysis method, it is assumed that when two bodies collide, although the contact period is very small, the change in the velocities is not discontinuous—the velocities vary continuously during the period of contact as the contacting bodies undergo local deformations. To represent the local deformations in a simplified manner, it is assumed that the colliding bodies penetrate into each other in the contact region as depicted in the examples of Figure 11.9a. The amount of penetration, which is denoted as  $\delta$ , represents the sum of local deformations of the two contacting bodies. The penetration results in a pair of resistive contact forces acting on the two bodies in opposite directions as shown in the illustrations of Figure 11.9b. The penetration starts in a *compression* phase with  $\delta = 0$  and reaches a maximum value  $\delta = \delta_m$ . During the *restitution* phase, the penetration returns from  $\delta = \delta_m$  back to  $\delta = 0$ . Following an impact, the two bodies may remain in contact or they may separate.

A contact force model can be viewed as a logical point-to-point spring–damper that is only active during the period of contact. The characteristics of the logical spring–damper element must be adjusted based on the material properties of the contacting bodies. In Sections 11.2.1 and 11.2.2, we consider several models for a body contacting a rigid surface and for two bodies contacting each other, respectively.

### 11.2.1 A Body Contacting a Rigid Surface

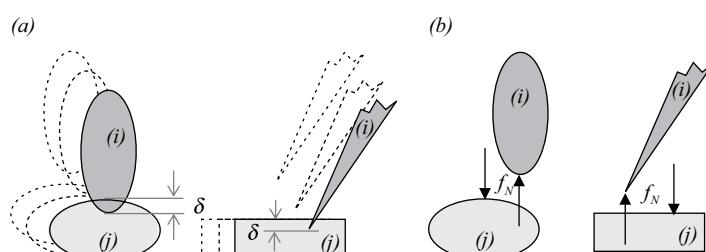
The simplest contact model to consider is the mass–spring system shown in Figure 11.10. The single body with the mass  $m$  moves toward a rigid surface. As the body contacts the surface, it deforms (or penetrates the surface) by an amount  $\delta$ . Assuming the deformation is represented by a spring of stiffness  $k$ , the penetration force can be determined as

$$f_N = k\delta \quad (11.31)$$

If we assume that the deformation also contains damping, the penetration force can be computed as

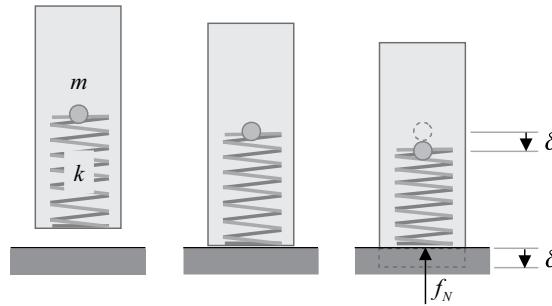
$$f_N = k\delta + d_c\dot{\delta} \quad (11.32)$$

where  $d_c$  is the damping coefficient.



**FIGURE 11.9**

(a) Penetration models for contact–impact and (b) the corresponding free-body diagrams.



**FIGURE 11.10**  
A mass-spring contact model.

In the models of Eqs. (11.31) and (11.32), it is assumed that both the spring and the damper have linear characteristics. We may also consider *nonlinear* characteristics for the spring as

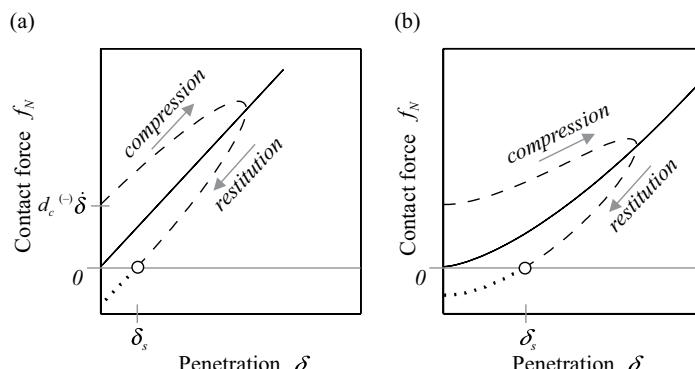
$$f_N = k\delta^n \quad (11.33)$$

where  $n > 1$  is an exponent to be determined based on the material characteristics of the body. With such a nonlinear model for the spring, Eq. (11.32) becomes

$$f_N = k\delta^n + d_c \dot{\delta} \quad (11.34)$$

In this nonlinear model, a reasonable value for the exponent is  $n = 3/2$  as it will be discussed in Section 11.2.2.

In the model of Eq. (11.31), the penetration force varies linearly during both compression and restitution phases shown as a solid line in Figure 11.11a. The force starts from zero at  $\delta = 0$ , reaches a maximum value, and then returns along the same straight line back to zero, where at this point the body separates from the contacting surface. When we add damping to the model, as in Eq. (11.32), the paths of the compression and restitution phases separate



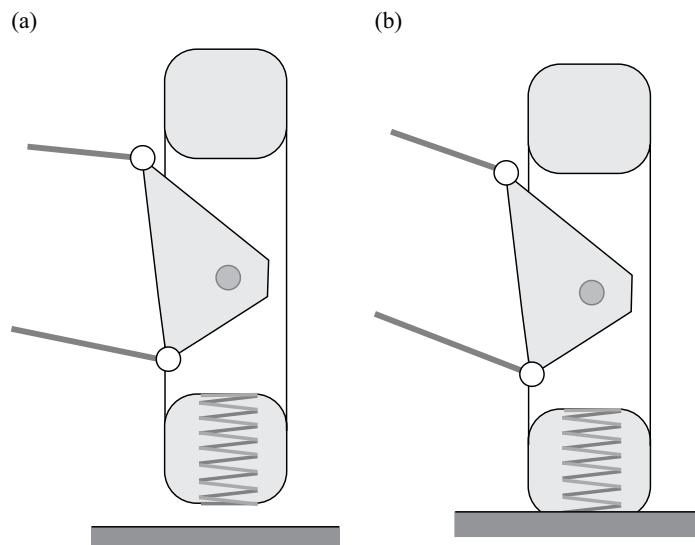
**FIGURE 11.11**  
Compression and restitution phases for the (a) linear and (b) nonlinear models.

as shown in dashed line in Figure 11.11a. This curve shows that as the body contacts the surface, there is a nonzero penetration force due to the velocity of the body at the start of the compression phase. This velocity causes a penetration force  $f_N = d_c^{(-)}\dot{\delta}$ , where  ${}^{(-)}\dot{\delta}$  denotes the initial penetration speed at contact. The figure shows that the path during the restitution phase, in the presence of damping, is different from that of the compression phase (called *hysteresis*). We also observe that near the end of the restitution phase, the penetration force becomes zero, at  $\delta = \delta_s$ , before the restitution phase is completed. This is the instant at which the body separates from the surface while it continues to undeform. At this point, since  $f_N = 0$ , the acceleration of the body is zero, and therefore, the velocity of the body is at a maximum. This means that during the final portion of the restitution phase, the model yields a tensile force, as shown in the figure in dotted line, which should not be applied to the body.

The contact force versus penetration for the model of Eq. (11.33) is shown in solid line in Figure 11.11b. Similar to the linear model, in the absence of damping, this model provides the same paths for both compression and restitution phases. When damping is included, as in Eq. (11.34), the model exhibits hysteresis behavior as shown in dashed line. Similar to the linear model, the penetration force becomes zero at  $\delta = \delta_s$ , which is the instant at which the body leaves the contacting surface.

### Example 11.5

The contact of a wheel/tire with the ground can be represented as a simple spring-damper model as in Eq. (11.32). As shown in Figure 11.12a, when the tire is not in contact with the ground, the spring-damper element should be inactive, but when the tire contacts the ground, as in Figure 11.12b, the spring-damper should apply a force on the wheel. Examples of such models can be found in Chapter 8 for the double A-arm and McPherson suspension systems.



**FIGURE 11.12**

A wheel/tire (a) before and (b) after contacting the ground.

### 11.2.2 Two-Body Contact

The discussion of Section 11.2.1 can be extended to two bodies becoming in contact. Assume that the two bodies shown in Figure 11.13 are in the process of contacting each other. The force models of Eqs. (11.31) through (11.34) can be implemented between the two bodies to determine the contact force. The overall penetration,  $\delta$ , can be considered as the sum of the deformations of the two bodies as  $\delta = \delta_1 + \delta_2$ . The penetration is determined along the axis of contact, which is normal to the contacting surfaces.

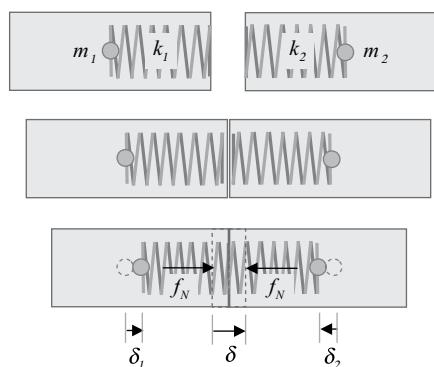
The stiffness  $k$ , for example, in the linear model of Eq. (11.31), can be determined based on the stiffness of the two springs as

$$k = \frac{k_1 k_2}{k_1 + k_2} \quad (11.35)$$

In the presence of damping, as in Eqs. (11.32) and (11.34), the two bodies separate at  $\delta = \delta_s$  before they have completely recovered their deformations, as demonstrated in Figure 11.11.

The formula of Eq. (11.35) can be obtained by realizing the forces that the two bodies apply on each other are equal in magnitude. Therefore,  $f_N = k_1 \delta_1 = k_2 \delta_2$ . An equivalent spring with the stiffness  $k$  undergoing a deformation  $\delta = \delta_1 + \delta_2$  should produce the same force  $f_N = k\delta$ . We can write  $\delta = \frac{f_N}{k} = \delta_1 + \delta_2 = \frac{f_N}{k_1} + \frac{f_N}{k_2}$ , which yields  $\frac{1}{k} = \frac{1}{k_1} + \frac{1}{k_2}$ , resulting in Eq. (11.35).

The spring model with the nonlinear characteristics of Eq. (11.33) is known as the *Hertz* contact force model [1]. The model considers the stiffness to depend on the material property and the local geometry of the contacting bodies. For example, for two colliding spheres with radii  $R_i$  and  $R_j$ , the parameter  $k$  can be determined as



**FIGURE 11.13**

Two bodies becoming in contact.

$$k = \frac{4}{3\pi(h_i + h_j)} \left( \frac{R_i R_j}{R_i + R_j} \right)^{\frac{1}{2}}; \quad h_i = \frac{1 - v_k^2}{\pi E_k}; \quad k = i, j \quad (11.36)$$

where  $v_k$  and  $E_k$  are Poisson's ratio and Young's modulus of each sphere, respectively, and the exponent in Eq. (11.33) is determined to be  $n = 3/2$ . If one of the contacting surfaces is flat (very large radius of  $R_j = \infty$ ), Eq. (11.36) becomes

$$k = \frac{4R_i^{\frac{1}{2}}}{3\pi(h_i + h_j)}; \quad h_i = \frac{1 - v_k^2}{\pi E_k}; \quad k = i, j \quad (11.37)$$

A revised form of damping for Eq. (11.34) considers the damping coefficient to be a function of indentation as [2]

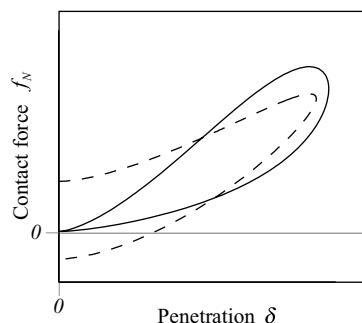
$$d_c = \mu \delta^n \quad (11.38)$$

where  $\mu$  is called the *hysteresis* damping factor. Substituting this description of the damping coefficient in Eq. (11.34) results in the following contact force model:

$$f_N = (k + \mu \dot{\delta}) \delta^n \quad (11.39)$$

Figure 11.14 illustrates a comparison between this contact force model (solid line) and the force model of Eq. (11.34). In the revised model, the contact forces at the start of the compression phase and at the end of restitution phase are zero, which is quite different from the original model of Eq. (11.34). In the revised model, since the contact force does not reach zero until at the end of the restitution phase, the bodies do not separate until the restitution is fully completed.

Although the contact force model of Eq. (11.39) has changed the physical characteristics of the contact, it provides a useful feature. It has been shown that the damping factor in Eq. (11.38) can be expressed as a function of the coefficient of *restitution*, where such a function cannot be found analytically for the damping coefficient in Eq. (11.34). The coefficient of restitution is defined, as in the piecewise analysis, as the ratio between the



**FIGURE 11.14**

Schematic description of contact force versus penetration for Eq. (11.34) (dashed line) and Eq. (11.39) (solid line).

relative penetration speeds at the start of the compression phase,  $(-) \dot{\delta}$ , and at the instant of separation,  $(+) \dot{\delta}$ :

$$e = \frac{(+) \dot{\delta}}{(-) \dot{\delta}} \quad (11.40)$$

One description of the damping factor in terms of the coefficient of restitution has been suggested as [3]

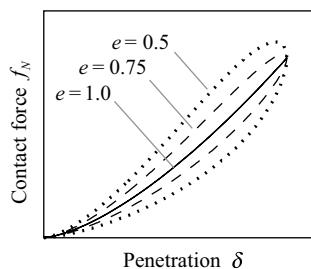
$$\mu = \frac{3k(1-e^2)}{4(-) \dot{\delta}} \quad (11.41)$$

Substitution of this expression in Eq. (11.39) provides the following contact force model:

$$f_N = k\delta^n \left( 1 + \frac{3(1-e^2)}{4} \frac{\dot{\delta}}{(-) \dot{\delta}} \right) \quad (11.42)$$

A graphical description of this force model is shown in Figure 11.15. For values of the coefficient of restitution smaller than 1.0, the force–penetration curves exhibit hysteresis behavior. The area inside each hysteresis curve denotes the loss of energy during impact. For an elastic impact, that is,  $e = 1.0$ , the loss of energy is zero. In other words, the relative speed at the moment of separation is equal to the relative speed at the start of contact. Note that Eq. (11.42) assumes that during both the compression and restitution phases,  $\dot{\delta} > 0$ , where the sign of  $\dot{\delta}$  is positive during compression (in the same direction as that of  $\delta$ ) and negative during restitution.

Typical values for the stiffness parameter  $k$  are  $(1 \rightarrow 10) \times 10^{10}$  N/m<sup>1.5</sup>. The coefficient of restitution can have a value  $0 < e < 1.0$ . For  $e = 1.0$ , Eq. (11.42) becomes identical to Eq. (11.33), that is, no energy loss due to damping. This represents an *elastic* impact causing the separation speed to be equal to the approach speed. By contrast,  $e = 0$ , representing an *inelastic* (or *plastic*) impact, results in the maximum loss of energy due to damping.



**FIGURE 11.15**

Contact force of Eq. (11.42) versus penetration for different values of the coefficient of restitution.

The model of Eq. (11.42) provides a better representation of contact for values of  $e$  closer to "1.0" than the values closer to "0." This shortcoming of Eq. (11.42) has been improved upon in the following force model [4,5]:

$$f_N = k \delta^n \left( 1 + \frac{8(1-e)}{5e} \frac{\dot{\delta}}{(\cdot)\dot{\delta}} \right) \quad (11.43)$$

In this model, the damping factor becomes larger, compared to that of Eq. (11.42), as the coefficient of restitution becomes smaller. We note that for  $e = 0$ , the damping factor becomes infinity, and therefore, for computational purposes, the coefficient of restitution should not be set exactly to zero, but it could be a very small number.

### Example 11.6

A slender rod moves toward the ground vertically as shown in Figure 11.16. The rod has a length of 2.0 m, a mass of 1.0 kg, and a moment of inertia of  $0.01 \text{ kg m}^2$ . The rod is oriented at  $45^\circ$  angle and its velocity at the moment of impact is 6.5 m/s downward with no angular velocity. If the velocity of the contact point drops to 2.5 m/s when the penetration reaches 0.001 m, determine the contact forces using Eqs. (11.42) and (11.43) for  $e = 0.15$  and  $e = 0.95$ , respectively. Assume  $n = 3/2$  and  $K = 10^{11} \text{ N/m}^{-0.5}$ .

#### Solution

Substituting all the known values in Eqs. (11.42) and (11.43) yields the following results for  $e = 0.15$ :

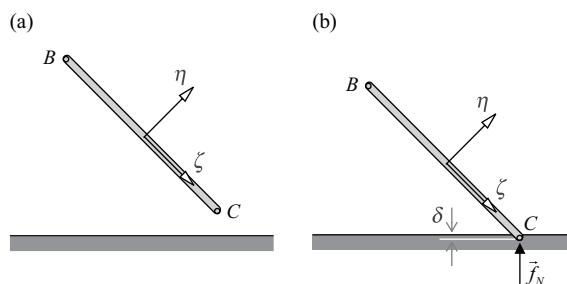
$$\text{Eq. (11.42): } f_N = (10^{11})(0.001^{3/2}) \left( 1 + \frac{3(1-0.15^2)}{4} \frac{2.5}{6.5} \right) = 3.32 \times 10^6 \text{ N}$$

$$\text{Eq. (11.43): } f_N = (10^{11})(0.001^{3/2}) \left( 1 + \frac{8(1-0.15)}{5(0.15)} \frac{2.5}{6.5} \right) = 3.26 \times 10^6 \text{ N}$$

For  $e = 0.95$ , we have

$$\text{Eq. (11.42): } f_N = 4.65 \times 10^6 \text{ N}$$

$$\text{Eq. (11.43): } f_N = 1.42 \times 10^7 \text{ N}$$



**FIGURE 11.16**

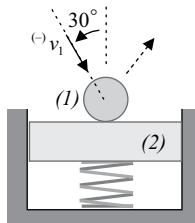
A rod impacting the ground (a) before and (b) during contact.

We note that for  $e = 0.95$  (close to  $e = 1$ ), the results from the two models are very close, but for  $e = 0.15$  (closer to  $e = 0$ ), Eq. (11.43) provides a much larger force than Eq. (11.42). (A simulation of this example can be found in Section 8.1.5.)

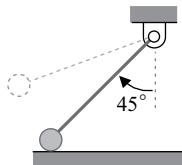
### 11.3 Problems

Problems 11.1–11.6 on piecewise analysis are simple enough to be solved by pencil and paper. However, the intention here is to solve these problems with MATLAB and experiment with some of the equations in Section 11.1.

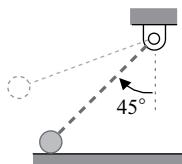
- 11.1 Particle (1) impacts body (2) with a velocity  $(-)v_1 = 10 \text{ m/s}$  in an angle. Determine the velocities of the particle and the body after impact if  $m_1 = 1 \text{ kg}$ ,  $m_2 = 2 \text{ kg}$ , and  $e = 0.6$ .



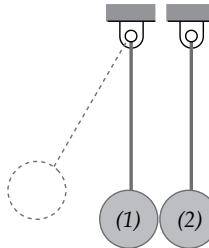
- 11.2 A particle with a mass  $m_1 = 1 \text{ kg}$  is attached to a massless rod forming a pendulum, which impacts the ground at the orientation shown. Consider the effective length of the pendulum to be  $L = 1 \text{ m}$ , a coefficient of restitution to be  $e = 0.9$ , and the angular velocity of the pendulum at the moment of impact to be  $\dot{\phi} = 1 \text{ rad/s}$ . Determine the angular velocity of the pendulum immediately after the impact.



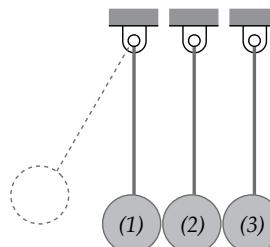
- 11.3 The massless rod of the pendulum in Problem 11.2 is replaced by a massless string; otherwise, all the data remain the same. Determine the velocity of the particle immediately after the impact.



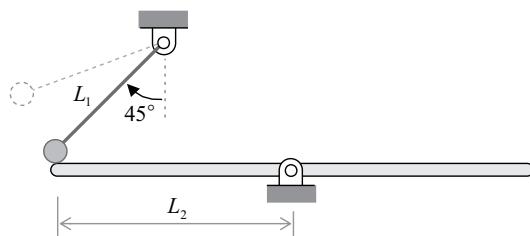
- 11.4 The spheres of two identical pendulums, when at rest, barely touch each other. One pendulum is released from a nonstatic equilibrium, which impacts the second pendulum at a velocity of  $(-)v_1 = 0.5 \text{ m/s}$ . If the effective length, radius, and mass of each pendulum are  $L = 1 \text{ m}$ ,  $R = 0.1 \text{ m}$ , and  $m = 1 \text{ kg}$ , respectively, determine the velocity of both pendulums after the impact if  $e = 0.9$ .



- 11.5 Extend Problem 11.4 to three pendulums as shown; otherwise, all the data remain the same. Determine the velocity of all three pendulums after the impact.



- 11.6 A particle with a mass  $m_1 = 1$  is attached to a massless rod forming a pendulum, which impacts a second body at the orientation shown. Consider  $L_1 = 1 \text{ m}$ ,  $L_2 = 1.5 \text{ m}$ ,  $m_2 = 4 \text{ kg}$ ,  $J_2 = 0.75 \text{ kg m}^2$ , and  $e = 0.9$ . If the angular velocity of the pendulum at the moment of impact is  $\dot{\phi} = 1 \text{ rad/s}$ , determine the angular velocity of the pendulum and body (2) immediately after the impact.



- 11.7 Transform the piecewise formula of Eq. (11.27) from the body-coordinate formulation to the joint-coordinate formulation.

- 11.8 Revise Eq. (11.27) to include friction in the tangential direction to impact.

Problems 11.9–11.13 on continuous analysis are intended for solution with MATLAB using forward dynamic analysis from Chapter 13.

- 11.9 The contact force model of Eq. (11.42) provides an approximated representation of the damping factor as a function of the coefficient of restitution. To test the accuracy of this model, we can perform the following simple simulations. Consider a sphere of mass  $m = 1$  kg and a stiffness parameter  $k = 10^{10}$  N/m<sup>1.5</sup>, impacting a rigid surface with a velocity of  $(-)v = 1$  m/s. For a specified value of the coefficient of restitution,  $e_{in}$ , perform a forward dynamic analysis and determine the velocity of the sphere,  $(+)v$ , at the moment of separation from the ground. The effective value of the coefficient of restitution can be calculated as  $e_{eff} = |(+)v / (-)v|$ . Perform the simulation several times by incrementing the value of the coefficient of restitution by  $\Delta e = 0.1$ . Plot  $e_{eff}$  versus  $e_{in}$ . What do you conclude?
- 11.10 Repeat Problem 11.9 with the contact force model of Eq. (11.43).
- 11.11 Perform a continuous forward dynamic analysis for several seconds for the system of Problem 11.1. Assume the spring is initially undeformed and has a stiffness  $k = 20$  N/m. Gravity is not a factor in this analysis. Apply the continuous contact force model of
- Eq. (11.42)
  - Eq. (11.43)
- 11.12 Consider the pendulum system in Problem 11.2 for a continuous forward dynamic analysis. The pendulum is released from the horizontal orientation, which swings under the force of gravity before striking the ground. Assume the radius of the sphere is negligible compared to the length of the pendulum. Perform an analysis for several seconds and plot the  $y$ -coordinate of the tip of the pendulum versus time. What is the maximum penetration of the tip into the ground?
- 11.13 Consider the double pendulum system of Problem 11.4 for a continuous forward dynamic analysis. Assume that pendulum (1) is released from the horizontal orientation, which swings under the force of gravity before striking pendulum (2). Perform an analysis for several seconds, and plot the  $x$ -coordinates of the two pendulums versus time. Determine the maximum penetration between the two spheres.

## References

1. Hertz, H., *Gesammelte Werke*, Vol. 1. Leipzig, J. A. Barth (1893).
2. Hunt, K.H., and Grossley, F.R.E., Coefficient of Restitution Interpreted as Damping in Vibroimpact, *ASME J. Appl. Mech.*, 42(2), 440–445 (1975).
3. Lankarani, H.M., and Nikravesh, P.E., A Contact Force Model with Hysteresis Damping for Impact Analysis of Multibody Systems, *ASME J. Mech. Des.*, 112(3), 369–376 (1990).
4. Ye, K., Li, L., and Zhu, H., A Note on the Hertz Contact Model with Nonlinear Damping for Pounding Simulation, *Earthquake Eng. Struct. Dyn.*, 38, 1135–1142 (2009).
5. Flores, P., Machado, M., and Silva, M.T., On the Continuous Contact Force Models for Soft Materials in Multibody Dynamics, *J. Multibody Syst. Dyn.*, 25, 357–375 (2011).

# 12

---

## *Kinematics and Inverse Dynamics*

---

Traditionally, kinematic and inverse dynamic analyses refer to the analyses of mechanisms where only algebraic equations are involved. Based on the definition of a mechanism, the system must only have one degree of freedom (DoF), where a driver motor or an actuator (a known time function of a specific coordinate and its corresponding velocity and acceleration) fully controls the motion of the mechanism. The analysis requires solving nonlinear and linear algebraic equations to determine the coordinates, velocities, and accelerations of all the moving links of a mechanism (kinematic analysis). Then, solving another set of linear algebraic equations would determine the required load from the driver motor or actuator, as well as the reaction forces at the joints (inverse dynamic analysis).

In this chapter, we begin the discussion with an algorithm for kinematic analysis. We review numerical procedures for solving linear and nonlinear algebraic equations. Another algorithm for inverse dynamic analysis is also discussed. A MATLAB® program for kinematic and inverse dynamic analyses of the four-bar mechanism is presented that can analyze the response of a four-bar mechanism based on a set of user's supplied data. We must note that in Chapter 13 we will learn that the kinematic and inverse dynamic equations can also be solved as a set of differential equations, as in a forward dynamic analysis.

---

### 12.1 Kinematic Analysis

Kinematic analysis, by definition, is the solution for the coordinates, velocities, and accelerations of closed-chain systems by solving the constraint equations. When kinematic analysis is considered for a mechanism that has one DoF, one driver constraint is included in the formulation. Based on the motion described by the driver, the kinematics of the entire system can be solved for.

Regardless of the choice of the coordinates, for a multibody system with  $n_{dof}$  DoFs and  $n_v$  defined coordinates, if  $n_v > n_{dof}$ , then there must be  $n_c = n_v - n_{dof}$  independent constraint equations. For a kinematic analysis, we define as many driver constraints as the number of system's DoFs. This general statement of kinematic analysis can be applied to any system with one or more DoFs. However, the traditional definition of kinematic analysis is for mechanisms, that is, for systems with one DoF.

In a general formulation for kinematic analysis, if  $\mathbf{q}$  denotes an array of  $n_v$  coordinates, then  $n_c$  position constraints and  $n_{dof}$  driver constraints are formulated as

$$\begin{aligned}\Phi &\equiv \Phi(\mathbf{q}) = \mathbf{0} \\ {}^{(dr)}\Phi &\equiv {}^{(dr)}\Phi(\mathbf{q}) - f(t) = \mathbf{0}\end{aligned}\tag{12.1}$$

As we have already seen in Chapters 7–9, driver constraints are simple functions of one or possibly two coordinates and the parameter *time*. The first and second time derivatives of Eq. (12.1) yield the velocity and acceleration constraints as

$$\begin{aligned}\dot{\Phi} &\equiv \mathbf{D}\dot{\mathbf{q}} = \mathbf{0} \\ {}^{(dr)}\dot{\Phi} &\equiv {}^{(dr)}\mathbf{D}\dot{\mathbf{q}} - \ddot{f}(t) = \mathbf{0}\end{aligned}\Rightarrow \begin{bmatrix} \mathbf{D} \\ {}^{(dr)}\mathbf{D} \end{bmatrix} \dot{\mathbf{q}} = \begin{cases} \mathbf{0} \\ \ddot{f}(t) \end{cases} \quad (12.2)$$

and

$$\begin{aligned}\ddot{\Phi} &\equiv \mathbf{D}\ddot{\mathbf{q}} + \dot{\mathbf{D}}\dot{\mathbf{q}} = \mathbf{0} \\ {}^{(dr)}\ddot{\Phi} &\equiv {}^{(dr)}\mathbf{D}\ddot{\mathbf{q}} + {}^{(dr)}\dot{\mathbf{D}}\dot{\mathbf{q}} - \dddot{f}(t) = \mathbf{0}\end{aligned}\Rightarrow \begin{bmatrix} \mathbf{D} \\ {}^{(dr)}\mathbf{D} \end{bmatrix} \ddot{\mathbf{q}} = \begin{cases} -\dot{\mathbf{D}}\dot{\mathbf{q}} \\ -{}^{(dr)}\dot{\mathbf{D}}\dot{\mathbf{q}} + \ddot{f}(t) \end{cases} \quad (12.3)$$

where the coefficient matrix in the velocity (and acceleration) equation is square.

The objective of kinematic analysis is to solve these algebraic equations for the coordinates, velocities, and accelerations for a specified time-period, at  $\Delta t$  time intervals.

### Algorithm K

We sweep the time parameter from  $t = 0$  to  $t = {}^{(\text{end})}t$  by increments of  $\Delta t$ . At each time step,  $t = {}^i t$ , we perform the following steps:

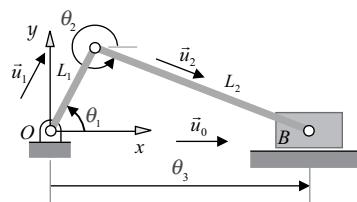
1. Solve Eq. (12.1) for  $\mathbf{q}$ .
2. Solve Eq. (12.2) for  $\dot{\mathbf{q}}$ .
3. Solve Eq. (12.3) for  $\ddot{\mathbf{q}}$ .
4. Compute the coordinates, velocity, and acceleration of any points of interest.

Algorithm K can be applied to any closed-chain system, regardless of whether the constraints are formulated in the body coordinates, vectorial or joint coordinates, or any other formulations. For our continuing discussion in this chapter, we show some examples that are formulated with the classical vectorial method.

### Example 12.1

For the slider–crank mechanism shown in Figure 12.1, assume  $L_1 = 0.12$  m and  $L_2 = 0.26$  m.

Consider a driver constraint as  $\theta_1 = {}^{(0)}\theta_1 + \omega_1 t + \frac{1}{2} \alpha_1 t^2$ , where  ${}^{(0)}\theta_1 = 0$ ,  $\omega = 2\pi$  rad/s, and



**FIGURE 12.1**

A slider–crank mechanism.

$\alpha_1 = 0$ . Solve the position, velocity, and acceleration equations for a complete revolution of the crank at time increments of  $\Delta t = 0.01$  s.

### Solution

This mechanism has already been analyzed in Example 5.1 for one time step. Here, we revise the corresponding MATLAB program, which uses some of the already developed function M-files.

#### MATLAB/Chapter 12/Example \_12 \_1

Following the constant and the driver data, we provide data for the time increment and period.

```
L1 = 0.12; L2 = 0.26;
theta1_0 = 65*pi/180; omega1 = 2*pi; alpha1 = 0; % Driver
del_t = 0.01; t_end = 1.0; % Time parameter data
nsteps = t_end/del_t + 1; % number of time steps
```

We set up arrays to store the results.

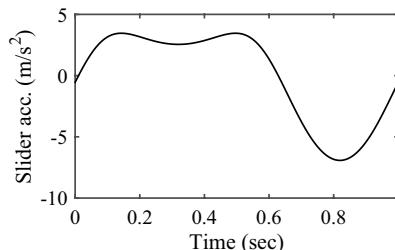
```
T = zeros(nsteps,1); thetas = zeros(nsteps,3);
thetas_d = zeros(nsteps,3); thetas_dd = zeros(nsteps,3);
```

We perform the analysis one step at a time and save the computed results (all the statements are not listed).

```
for i = 1:nsteps
    t = (i - 1)*del_t; T(i) = t;
    ...
    theta1 = theta1_0 + omega1*t + 0.5*alpha1*t^2;
    theta1_d = omega1 + alpha1*t;
    theta1_dd = alpha1;
    ...
    thetas(i,:) = [theta1 theta2 theta3];
    thetas_d(i,:) = [theta1_d theta2_d theta3_d];
    thetas_dd(i,:) = [theta1_dd theta2_dd theta3_dd];
    ...
End
```

The simulation results are saved in the arrays `thetas`, `thetas_d`, and `thetas_dd`, where the time steps are saved in the array `T`. We can plot, for example, the acceleration of the slider versus time, using the following command to obtain the result shown in Figure 12.2:

```
>> plot(T,thetas_dd(:,3))
```



**FIGURE 12.2**

Acceleration of the slider versus time.

### 12.1.1 Solution Procedures

Kinematic analysis requires solving the kinematic and driver constraints at a given time  $t = i t$ , for the coordinates, velocities, and accelerations. The position constraints are, in general, nonlinear algebraic equations, but the velocity and acceleration constraints are linear in the velocities and accelerations, respectively. Whether we solve linear or nonlinear equations, we may consider two possible arrangements for the equations.

Consider the following set of two equations in three unknowns:

$$\begin{aligned} x + 2y - z &= 2 \\ 2x - y + z &= 3 \end{aligned} \tag{a}$$

Assume that the value of one of the unknowns is given as

$$z = 3 \tag{b}$$

Equation (a) may be considered as two kinematic constraints, and Eq. (b) may be viewed as a driver expression without the presence of the time variable.

In the first arrangement, the value of  $z$  is obtained from Eq. (b) and substituted in Eq. (a) to form two equations in two unknowns:

$$\begin{aligned} x + 2y &= 5 \\ 2x - y &= 0 \end{aligned} \Rightarrow \left[ \begin{array}{cc} 1 & 2 \\ 2 & -1 \end{array} \right] \left\{ \begin{array}{c} x \\ y \end{array} \right\} = \left\{ \begin{array}{c} 5 \\ 0 \end{array} \right\} \tag{c}$$

These equations are solved to determine the unknowns:  $x = 1$  and  $y = 2$ . This arrangement is called *coordinate partitioning* (CP) method.

In the second arrangement, Eqs. (a) and (b) are considered simultaneously to form three equations in three unknowns as follows:

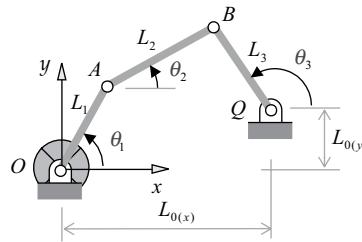
$$\begin{aligned} x + 2y - z &= 2 \\ 2x - y + z &= 3 \\ z &= 3 \end{aligned} \Rightarrow \left[ \begin{array}{ccc} 1 & 2 & -1 \\ 2 & -1 & 1 \\ 0 & 0 & 3 \end{array} \right] \left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\} = \left\{ \begin{array}{c} 2 \\ 3 \\ 3 \end{array} \right\} \tag{d}$$

The solution to this simultaneous set of equations yields  $x = 1$ ,  $y = 2$ , and  $z = 3$ . This arrangement is called *appended constraint* (AC) method. Although the two arrangements are not very different, from the programming and computational points of view, there are advantages and disadvantages associated with each other.

#### Example 12.2

The crank of the four-bar mechanism shown in Figure 12.3 rotates with a constant angular velocity of  $2\pi$  rad/s counterclockwise (CCW). The position of the crank at the initial time is  $\theta_1 = 60^\circ$ . Construct and arrange the kinematic equations for position, velocity, and acceleration analyses suitable for

- a. The AC method
- b. The CP method



**FIGURE 12.3**  
A four-bar mechanism.

### Solution

a. In the AC arrangement, the position constraints are written as

$$\begin{aligned} L_1 \cos \theta_1 + L_2 \cos \theta_2 - L_3 \cos \theta_3 - L_{0x} &= 0 \\ L_1 \sin \theta_1 + L_2 \sin \theta_2 - L_3 \sin \theta_3 - L_{0y} &= 0 \\ \theta_1 - \pi / 3 - 2\pi t &= 0 \end{aligned} \quad (12.4)$$

The velocity and acceleration constraints are expressed as

$$\left[ \begin{array}{ccc} -L_1 \sin \theta_1 & -L_2 \sin \theta_2 & L_3 \sin \theta_3 \\ L_1 \cos \theta_1 & L_2 \cos \theta_2 & -L_3 \cos \theta_3 \\ 1 & 0 & 0 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \\ 2\pi \end{array} \right\} \quad (12.5)$$

and

$$\begin{aligned} \left[ \begin{array}{ccc} -L_1 \sin \theta_1 & -L_2 \sin \theta_2 & L_3 \sin \theta_3 \\ L_1 \cos \theta_1 & L_2 \cos \theta_2 & -L_3 \cos \theta_3 \\ 1 & 0 & 0 \end{array} \right] \left\{ \begin{array}{c} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{array} \right\} &= \left\{ \begin{array}{c} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{array} \right\} \\ &= \left\{ \begin{array}{c} L_1 \cos \theta_1 \dot{\theta}_1^2 + L_2 \cos \theta_2 \dot{\theta}_2^2 - L_3 \cos \theta_3 \dot{\theta}_3^2 \\ L_1 \sin \theta_1 \dot{\theta}_1^2 + L_2 \sin \theta_2 \dot{\theta}_2^2 - L_3 \sin \theta_3 \dot{\theta}_3^2 \\ 0 \end{array} \right\} \end{aligned} \quad (12.6)$$

(b) In the CP method, Eqs. (12.4)–(12.6) are rearranged as

$$\begin{aligned} L_1 \cos(\pi/3 + 2\pi t) + L_2 \cos \theta_2 - L_3 \cos \theta_3 - L_0 &= 0 \\ L_1 \sin(\pi/3 + 2\pi t) + L_2 \sin \theta_2 - L_3 \sin \theta_3 &= 0 \end{aligned} \quad (a)$$

The velocity and acceleration constraints are expressed as

$$\left[ \begin{array}{cc} -L_2 \sin \theta_2 & L_3 \sin \theta_3 \\ L_2 \cos \theta_2 & -L_3 \cos \theta_3 \end{array} \right] \left\{ \begin{array}{c} \dot{\theta}_2 \\ \dot{\theta}_3 \end{array} \right\} = \left\{ \begin{array}{c} 2\pi L_1 \sin \theta_1 \\ -2\pi L_1 \cos \theta_1 \end{array} \right\} \quad (b)$$

and

$$\begin{aligned} & \left[ \begin{array}{cc} -L_2 \sin \theta_2 & L_3 \sin \theta_3 \\ L_2 \cos \theta_2 & -L_3 \cos \theta_3 \end{array} \right] \left\{ \begin{array}{c} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{array} \right\} \\ &= \left\{ \begin{array}{c} (2\pi)^2 L_1 \cos \theta_1 + L_2 \cos \theta_2 \dot{\theta}_2^2 - L_3 \cos \theta_3 \dot{\theta}_3^2 \\ (2\pi)^2 L_1 \sin \theta_1 + L_2 \sin \theta_2 \dot{\theta}_2^2 - L_3 \sin \theta_3 \dot{\theta}_3^2 \end{array} \right\} \end{aligned} \quad (c)$$

It should be obvious that in the CP method, the number of equations, and hence the number of unknowns, is smaller than those in the AC method. Therefore, the *computational effort* with the CP method is less than the AC method. However, for developing a general-purpose kinematic analysis program, the *programming effort* that the AC method requires is much less than that of the CP method. This is due to the fact that in the AC method, the constraint equations keep their original forms—we do not need to split a set of equations into the known and unknown parts and move the known part to the right-hand side. Another important point is that in problems with a large number of equations and coordinates, appending one or two driver equations to the original set will not make much difference in the computation time. Therefore, in most problems, we adopt the AC arrangement.

The velocity and acceleration constraints, whether we arrange them for the CP or the AC method, are, respectively, linear in the velocities and accelerations. Solving these equations for the velocities and accelerations requires solving simultaneous linear algebraic equations as

$$\mathbf{A}\mathbf{x} = \mathbf{c}$$

Solution to a set of simultaneous linear algebraic equations, using the *backslash* operator in MATLAB, was briefly discussed in Section 2.3.2. Another useful method for solving a set of linear algebraic equations is called L–U factorization, which is discussed in Appendix A. This method is also useful in identifying redundant equations, in case the coefficient matrix  $\mathbf{A}$  is singular.

The kinematic position constraints, in contrast to the velocity and acceleration constraints, are nonlinear algebraic equations. As it will be seen in Section 12.1.2, nonlinear algebraic equations can be approximated as linear algebraic equations and solved iteratively for the unknowns.

### 12.1.2 Nonlinear Algebraic Equations

One frequently occurring problem in scientific work is to find the roots of one or a set of nonlinear algebraic equations of the form

$$\Phi(\mathbf{q}) = \mathbf{0} \quad (12.7)$$

Kinematic analysis of mechanical systems is one example for which the solution of position constraints in the form of Eq. (12.7) is required. As we have seen in Example 5.2 chapters, the function `fsoolve` in MATLAB can easily find the roots of such nonlinear functions for us. To learn how the zeros of nonlinear algebraic equations are determined, in this

section we discuss a common iterative process known as the Newton–Raphson method. We describe this process first for one equation in one unknown, and then for a set of  $n$  equations in  $n$  unknowns.

**One Equation:** Consider the following nonlinear equation in one unknown  $q$ :

$$\Phi(q) = 0 \quad (12.8)$$

The Newton–Raphson iteration for solving this single equation is stated in the following algorithm:

### Algorithm NR1

(The left superscript  $i$  represents the iteration number, starting at  $i = 0$ )

1. Assume an estimate for the solution as  ${}^i q = {}^0 q$ .
2. Evaluate the function  ${}^i \Phi = \Phi({}^i q)$ .
3. If  $|{}^i \Phi| \leq \epsilon^*$ , then  ${}^i q$  is the solution to  $\Phi(q) = 0$ ; stop the iteration.
4. Otherwise, evaluate the derivative  ${}^i D = \left( \frac{\partial \Phi}{\partial q} \right)$ .
5. Compute  ${}^i \Delta q = -\frac{{}^i \Phi}{{}^i D}$ .
6. Improve the estimated solution as  ${}^i q + {}^i \Delta q \Rightarrow {}^{i+1} q$ ; return to step 2.

$\epsilon^*$  is a small number that we accept as “zero.”

If the unknown  $q$  is an angle, regardless of whether we work in degrees or radians, the computed value for  $\Delta q$  will be in radians. We must make sure not to add  $\Delta q$  in radians to  $q$  in degrees!

### Example 12.3

Use the Newton–Raphson method to find the root(s) of the equation:

$$f(x) = x^3 - 3x^2 - 10x + 24$$

#### Solution

The derivative of  $f(x)$  with respect to  $x$  is

$$df(x)/dx = 3x^2 - 6x - 10$$

The calculations can be done with the following program:

#### MATLAB/Chapter 12/NR1

```
% Newton-Raphson process
x = input(' estimate for x = ? ')
```

```

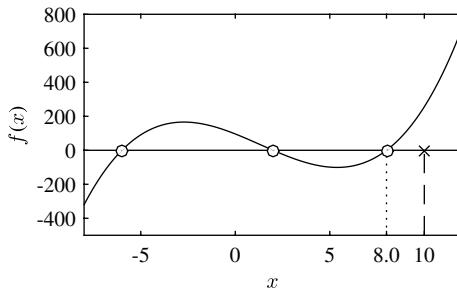
for n = 1:15
    f_x = x^3 - 4*x^2 - 44*x + 96; % Function
    df_dx = 3*x^2 - 8*x - 44;        % Derivative
    if abs(f_x) < 0.0001 % Is the constraint violated?
        break
    end
    d_x = -f_x/df_dx; % Solve for the correction
    x = x + d_x;       % Correct the estimate
end

```

Since this process is iterative, the program asks for an estimate for the unknown  $x$  to start the iteration. Entering a value starts the Newton–Raphson process; for example, if we enter 10, the iteration converges to  $x = 8.0$ . We may report the intermediate values of  $x$  and  $f(x)$  during the process in order to have a better understanding for the convergence of the algorithm. An estimate of 10 for  $x$  takes five iterations to converge as shown below:

n	x	f_x
1	10.0000	256.0000
2	8.5455	51.9309
3	8.0588	5.0089
4	8.0008	0.0677
5	8.0000	0.0000

Figure 12.4 shows that our function has three roots. Since we started with an estimate of  $x = 10$ , we converged to the solution closest to the estimate. If we start with a different initial estimate, we can find the other solutions.



**FIGURE 12.4**  
Plot of  $f(x)$  versus  $x$ .

The Newton–Raphson method, when it works, is very efficient, but it may converge to an unwanted solution if the initial estimate is not close to the desired solution. It is also possible for the iterations to diverge; therefore, it is essential to terminate the process after a finite number of iterations.

**n Equations:** Consider  $n$  nonlinear algebraic equations in  $n$  unknowns:

$$\Phi(\mathbf{q}) = \mathbf{0} \quad (12.9)$$

The Newton–Raphson algorithm for  $n$  equations is the  $n$ -dimensional version of Eq. (12.8). In the algorithm, we need the Jacobian matrix:

$$\mathbf{D} = \frac{\partial \Phi}{\partial \mathbf{q}} \quad (12.10)$$

An algorithm for this process is stated as follows:

### Algorithm NRn

(The left superscript  $i$  represents the iteration number, starting at  $i = 0$ )

1. Assume an estimated solution as  ${}^i\mathbf{q} = {}^0\mathbf{q}$ .
2. Evaluate the function  ${}^i\Phi = \Phi({}^i\mathbf{q})$ .
3. If  $\text{norm}({}^i\Phi) \leq \epsilon^*$ , then  ${}^i\mathbf{q}$  is the solution to  $\Phi(\mathbf{q}) = \mathbf{0}$ ; stop the iteration.
4. Otherwise, evaluate  ${}^i\mathbf{D}$  for  $\mathbf{q} = {}^i\mathbf{q}$ .
5. Solve the set of linear equations  ${}^i\mathbf{D} {}^i\Delta\mathbf{q} = -{}^i\Phi$  for the corrections  ${}^i\Delta\mathbf{q}$ .
6. Improve the estimated solution as  ${}^i\mathbf{q} + {}^i\Delta\mathbf{q} \Rightarrow {}^{i+1}\mathbf{q}$ ; return to step 2.

$\epsilon^*$  is a small number that we accept as “zero”.

The array  ${}^i\Phi$  is known as the array of *residuals*, which corresponds to the *violation* in the equations or the constraint, and the term  $\text{norm}({}^i\Phi) \leq \epsilon$  is the *norm* of the residuals. If this term is smaller than the specified tolerance  $\epsilon$ , then every residual is small enough to stop the iteration.

### Example 12.4

For the four-bar mechanism of Example 12.2, consider the following data:

$$L_1 = 1.0 \text{ m}, L_2 = 3.0 \text{ m}, L_3 = 2.2 \text{ m}, L_{0x} = 2.0 \text{ m}, L_{0y} = 0.5 \text{ m}$$

Develop a general-use Newton–Raphson program to find the solution to Eq. (12.4). Consider starting estimates  $\theta_1 = 1.57$  rad,  $\theta_2 = 0.5$  rad, and  $\theta_3 = 1.2$  rad. Solve for the angles at  $t = 0$ .

#### Solution

We develop the following script and function M-files.

#### MATLAB/Chapter 12/Example \_12 \_ 4

In the main program, we provide the initial estimates for the unknowns, and then we invoke the function NRn. The initial estimates, the desired tolerance, the allowed number of iterations, and a function handle are sent to the general-use Newton–Raphson function named NRn. The function handle (@Ex\_12\_4) will be accessed by NRn to obtain the residuals and the Jacobian matrix.

```
clear all
theta = [1.57; 0.5; 1.2];
[cords, check] = NRn(@Ex_12_4, theta, 0.0001, 10);
if check == 0
    'Convergence failed in NRn (Newton-Raphson)'
else
    coordinates = cords .....co
end
coords =
1.5708
0.5782
1.3357
```

The general-use function NRn is written in a form that can be used in any program. The results are saved in an array named coords that is returned to the main program. This function also returns a variable named check indicating either convergence (check = 1) or no convergence (check = 0).

```
function [coords, check] = NRn(fun, q, tol, iter_max)
% Newton-Raphson iterative process
    check = 0;
for i = 1:iter_max
    [Phi, D] = fun(q);           % constraints and Jacobian
    ff = norm(Phi);             % are the constraints violated?
    if ff < tol
        check = 1;              % a solution found!!!
        break
    end
    delta_q = -D\Phi;           % solve for corrections
    q = q + delta_q;           % correct the estimates
end
coords = q;
```

The following is the function that evaluates the constraints and the corresponding Jacobian matrix for the four-bar mechanism:

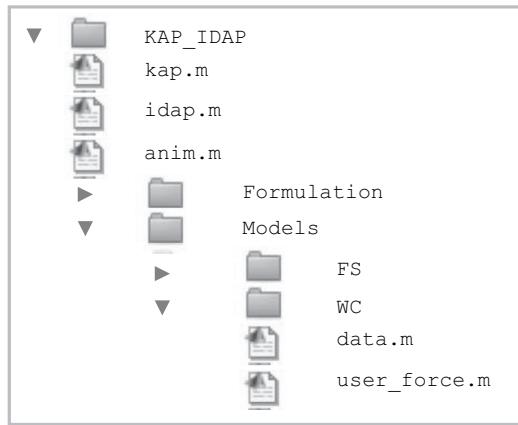
```
function [Phi, D] = Ex_12_4(theta)
% Constraints and Jacobian
    s1 = sin(theta(1)); c1 = cos(theta(1));
    s2 = sin(theta(2)); c2 = cos(theta(2));
    s3 = sin(theta(3)); c3 = cos(theta(3));
    Phi = [(c1 + 3*c2 - 2.2*c3 - 2)
            (s1 + 3*s2 - 2.2*s3 - 0.5)
            (theta(1) - pi/2)];
    D = [-s1 -3*s2 2.2*s3
          C1 3*c2 -2.2*c3
          1 0 0];
```

### 12.1.3 A Program for Four-Bar Kinematics

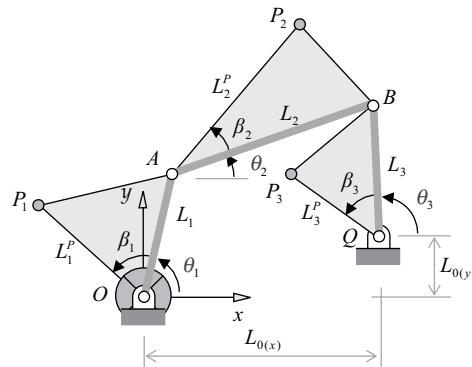
A MATLAB program for kinematic analysis of four-bar mechanisms is presented in this section. The constraints are formulated in vectorial form, and the equations are solved with the AC method. The user must specify the link lengths, some other dimensions if needed, and the data for the driver. The program rotates the crank according to the specified increments and solves for the kinematic response at each increment.

**MATLAB/Chapter 12/KAP \_ IDAP/kap**

The folder KAP \_ IDAP (kinematic analysis program-inverse dynamic analysis program) contains two main programs, as shown in Figure 12.5: kap for kinematic analysis and idap for inverse dynamic analysis, which will be discussed in Section 12.2. This folder also contains the animation program anim. The M-files for the analysis are in the folder



**FIGURE 12.5**  
Folders and script files of KAP.



**FIGURE 12.6**  
A general description of a four-bar mechanism for the program kap.

**Formulation.** The folder Models contains several example folders containing the input files data.m and user\_force.m. The file user\_force.m is used by idap.

The input data for the four-bar must follow the description shown in Figure 12.6. In addition to points A and B, the user can study the kinematics of points  $P_1$ ,  $P_2$ , and  $P_3$ , one on each link. The position of each of these points, with respect to its link, requires an angle and a length as shown.\* The angles  $\beta_1$ ,  $\beta_2$ , and  $\beta_3$  could be either positive (CCW as shown) or negative (clockwise [CW]).

The value for the crank angle,  $\theta_1$ , must be provided as an initial condition. The values for  $\theta_2$  and  $\theta_3$  can be estimated. The driver (motor) assumes the function  $\theta_1 = {}^0\theta_1 + \omega_1 t$ , where  ${}^0\theta_1$  is set equal to the provided initial value for  $\theta_1$ , and  $\omega_1$  is the constant angular velocity. All angles must be stated in degrees—the program converts them to radians.

\* This method of positioning a point on a link of a four-bar can be found in most textbooks on mechanisms, for example, in [1].

The program `kap`, based on the user's supplied data, first determines whether the four-bar is Grashof or non-Grashof.\* Then the program rotates the crank in increments of  $\Delta\theta_1 = 2^\circ$  through one complete rotation (if the four-bar is Grashof). It then computes and saves the coordinates, velocities, and accelerations for the links and the points  $A$ ,  $B$ ,  $P_1$ ,  $P_2$ , and  $P_3$  at every increment. If the four-bar is non-Grashof, or if it is Grashof but link (1) cannot go through a complete revolution, the program reverses the direction of rotation of link (1) as it reaches its limit. Following a simulation by `kap`, we may execute the program `anim` and observe an animation of the simulated response.

The program `kap` saves the simulated responses at every time step in various arrays as listed in Table 12.1. The saved response can be used for postprocessing and animation.

Let us consider the data for the model GEN (general):

**MATLAB/Chapter 12/KAP\_IDAP/Models/GEN/data.m**

```
RQ = [5.0 -1.0];
L = [2.0 6.0 4.0]; % Grashof
% L = [2.0 7.0 3.0]; % Non-Grashof
beta = [20 22.5 -30]; % in degrees
LP = [1.0 5.5 6.5];
theta = [120 30 60]; % in degrees
omegal = 1.0; % in rad/sec
axis_limits = [-3 12 -5 10]; % limits for the plot window
```

The ground link is defined by vector  $\mathbf{R}_Q = \begin{bmatrix} L_{0(x)} & L_{0(y)} \end{bmatrix}'$ , leading to  $L_0 = \sqrt{5^2 + 1^2} = 5.1$  m. With the given link lengths in the array `L` for  $L_1$ ,  $L_2$ , and  $L_3$ , the program will determine that the four-bar is Grashof. (If we switch over to the commented link lengths, the four-bar becomes non-Grashof.) Other arrays contain  $L_1^P$ ,  $L_2^P$ ,  $L_3^P$  in `LP`;  $\beta_1$ ,  $\beta_2$ ,  $\beta_3$  in `beta`;  $\theta_1$ ,  $\theta_2$ ,  $\theta_3$  in `theta`; and  $\omega_1$ . The value for  $\theta_1$  is the crank angle at  $t = 0$ , and the stated values for  $\theta_2$  and  $\theta_3$  are initial estimates. The last line of data provides the limits for the animation plot window.

**TABLE 12.1**

Arrays Used by `kap` for Saving the Simulated Response

Array	Size	Description
T	nsteps × 1	Time increments
thetas	nsteps × 3	$\theta_1, \theta_2, \theta_3$
omegas	nsteps × 3 (each)	$\omega_1, \omega_2, \omega_3$
alphas	nsteps × 3 (each)	$\alpha_1, \alpha_2, \alpha_3$
RA, VA, AA	nsteps × 2 (each)	$\mathbf{r}^A, \dot{\mathbf{r}}^A, \ddot{\mathbf{r}}^A$
RB, VB, AB	nsteps × 2 (each)	$\mathbf{r}^B, \dot{\mathbf{r}}^B, \ddot{\mathbf{r}}^B$
RP1, VP1, AP1	nsteps × 2 (each)	$\mathbf{r}_1^P, \dot{\mathbf{r}}_1^P, \ddot{\mathbf{r}}_1^P$
RP2, VP2, AP2	nsteps × 2 (each)	$\mathbf{r}_2^P, \dot{\mathbf{r}}_2^P, \ddot{\mathbf{r}}_2^P$
RP3, VP3, AP3	nsteps × 2 (each)	$\mathbf{r}_3^P, \dot{\mathbf{r}}_3^P, \ddot{\mathbf{r}}_3^P$
nsteps	Number of time steps (increments) including $t = 0$	

\* We denote the shortest and longest link lengths as ( $S$ ) and ( $L$ ), respectively, and the length of the remaining links as ( $P$ ) and ( $Q$ ). If  $S + L \leq P + Q$ , the four-bar is Grashof; otherwise, it is non-Grashof. In a Grashof four-bar, at least one of the links can rotate a complete  $360^\circ$  revolution.

**Example 12.5**

In this example, we model the film-strip advancer from Section 15.1, which is a four-bar having the following data:

$$L_{0(x)} = -3.0 \text{ m}, L_{0(y)} = -4.5 \text{ m}, L_1 = 1.5 \text{ m}, L_2 = 5.0 \text{ m}, L_3 = 3.0 \text{ m}$$

$$L_2^P = 10.5 \text{ m}, \beta_2 = \left( \tan^{-1}(0.5/10.5) \right) 180/\pi = 2.73^\circ$$

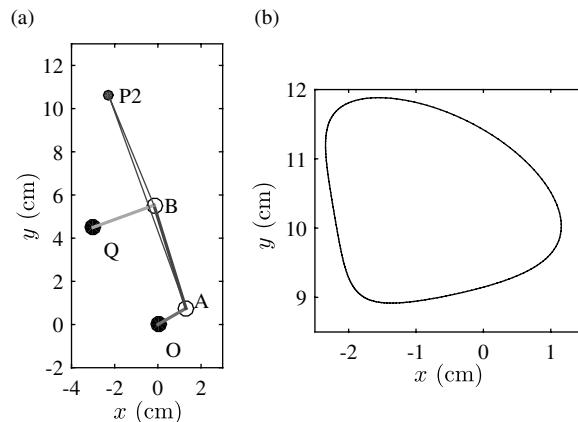
For the driver, we consider  ${}^0\theta_1 = 30^\circ$ ,  $\omega_1 = -60\pi \text{ rad/s}$ , and for the other two angles, we assume  $\theta_2 = 90^\circ$  and  $\theta_3 = 30^\circ$  as initial estimates.

We construct the following input M-file `data.m` for this four-bar mechanism.

```
MATLAB/Chapter 12/KAP _ IDAP/Models/FS/data.m
```

```
RQ = [-3; 4.5];
L = [1.5 5 3];
LP = [0 10.5 0];
beta = [0 2.73 0];
theta = [30 90 30];
omegal = -1.0;
axis_limits = [-4 3 -2 13];
```

We execute `kap` and direct it to the model folder `FS`, and then we run `anim` to produce the stick figure shown in Figure 12.7a. The command `plot(RP2(:,1),RP2(:,2))` provides the path of the coupler point  $P_2$  as shown in Figure 12.7b.



**FIGURE 12.7**

(a) Stick drawing of the film-strip advancer mechanism; (b) path of point  $P_2$ .

## 12.2 Inverse Dynamic Analysis

The equations of motion for a mechanical system represent the relationship between the forces that act on the system and its motion. If a specified motion of a mechanical system is sought and some of the applied forces and moments are known, it could be possible to determine the remaining forces and moments that cause that particular motion. When the

desired motion is known and the objective is to find the unknown forces and torques, the process is referred to as *inverse dynamic analysis*. Inverse dynamics is an extension of kinematic analysis in which the reaction forces and the unknown forces or torques associated with the driver constraints are determined. The fundamental process for inverse dynamic analysis is the same, regardless of the method used to formulate a problem.

Inverse dynamic analysis of a multibody system requires a kinematic analysis to be performed first to determine the coordinates,  $\mathbf{q}$ ; velocities,  $\dot{\mathbf{q}}$ ; and accelerations,  $\ddot{\mathbf{q}}$ . If  $\mathbf{q}$  does not contain the complete kinematics of all the bodies in a system (in the case of a vectorial formulation, for example), the body-coordinate arrays of  $\mathbf{c}$ ,  $\dot{\mathbf{c}}$ , and  $\ddot{\mathbf{c}}$  must be determined prior to performing a force analysis.

The equations of motion for a system, based on either the body coordinates from Eq. (7.55) or the free-body diagram method, can be formed as

$$\mathbf{M}\ddot{\mathbf{c}} = {}^{(a)}\mathbf{h} + \mathbf{D}'\lambda + {}^{(dr)}\mathbf{D}'{}^{(dr)}\lambda$$

where the Jacobian matrix and Lagrange multipliers associated with the driver constraints are included but separated from those of the kinematic joints. These equations can be rearranged as

$$\left[ \mathbf{D}' - {}^{(dr)}\mathbf{D}' \right] \begin{Bmatrix} \lambda \\ {}^{(dr)}\lambda \end{Bmatrix} = \mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h} \quad (12.11)$$

Since the coordinates, velocities, and accelerations are known, the coefficient matrix and all the terms on the right-hand side can numerically be evaluated. Then the equations are solved as a set of linear algebraic equations for the Lagrange multipliers. The results for  $\mathbf{D}'\lambda$  provide the joint reaction forces and torques, and  ${}^{(dr)}\mathbf{D}'{}^{(dr)}\lambda$  yields the force(s) or torque(s) associated with the driver motor(s).

### Algorithm ID1

We sweep the time parameter from  $t = 0$  to  $t = {}^{(end)}t$  in increments of  $\Delta t$ . At each time step,  $t = {}^i t$ , we perform the following steps:

#### Algorithm K

Perform kinematic analysis to obtain  $\mathbf{q}$ ,  $\dot{\mathbf{q}}$ , and  $\ddot{\mathbf{q}}$ .

If  $\mathbf{q} \neq \mathbf{c}$ , determine  $\mathbf{c}$ ,  $\dot{\mathbf{c}}$ ,  $\ddot{\mathbf{c}}$ .

1. Construct the mass matrix and the array of applied forces.
2. Construct the Jacobian matrix for the kinematic joints.
3. Construct the Jacobian matrix for the driver constraints.
4. Solve Eq. (12.11) for the Lagrange multipliers.

The solution to Eq. (12.11) provides the results for all Lagrange multipliers. For mechanisms with one DoF, there is only one Lagrange multiplier  ${}^{(dr)}\lambda$  associated with a single driver motor; therefore, Eq. (12.11) can be transformed to a single algebraic equation with

one unknown—the Lagrange multiplier associated with the driver. To derive this single equation, we premultiply Eq. (12.11) by the transpose of the array of velocities:

$$\dot{\mathbf{c}}' \left[ \mathbf{D}'^{(dr)} \mathbf{D}' \right] \begin{Bmatrix} \lambda \\ {}^{(dr)}\lambda \end{Bmatrix} = \dot{\mathbf{c}}' (\mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h})$$

Using the velocity constraints of Eq. (12.2), that is,  $\mathbf{D}\dot{\mathbf{c}} = \mathbf{0}$  for the body-coordinate formulation, the equation is simplified to

$$[\dot{\mathbf{c}}' {}^{(dr)}\mathbf{D}']^{(dr)} \lambda = \dot{\mathbf{c}}' (\mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h}) \quad (12.12)$$

We note that the coefficient of  ${}^{(dr)}\lambda$  becomes a  $1 \times 1$  matrix—a scalar. This equation can be solved for its single unknown  ${}^{(dr)}\lambda$ . We further note that the Jacobian matrix  $\mathbf{D}$  associated with the kinematic joints is no longer needed for this equation.

Equation (12.12) can further be simplified. The velocity equation for the driver constraint, as stated in Eq. (12.2), is expressed as  ${}^{(dr)}\mathbf{D}\dot{\mathbf{c}} = \dot{\mathbf{f}}(t)$ . If the driver acts on a single coordinate, for example,  $\phi_i = \omega$ , then the product  $\dot{\mathbf{c}}' {}^{(dr)}\mathbf{D}'$  becomes  $\dot{\phi}_i$ , and therefore, we have

$${}^{(dr)}\lambda = \dot{\mathbf{c}}' (\mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h}) / \dot{\phi}_i \quad (12.13)$$

If the driver constraint acts between two coordinates, such as  $\dot{\phi}_i - \dot{\phi}_j = \omega$ , then Eq. (12.12) can be simplified to

$${}^{(dr)}\lambda = \dot{\mathbf{c}}' (\mathbf{M}\ddot{\mathbf{c}} - {}^{(a)}\mathbf{h}) / (\dot{\phi}_i - \dot{\phi}_j) \quad (12.14)$$

### Algorithm ID2

We sweep the time parameter from  $t = 0$  to  $t = {}^{(end)}t$  in increments of  $\Delta t$ . At each time step,  $t = {}^i t$ , we perform the following steps:

#### Algorithm K

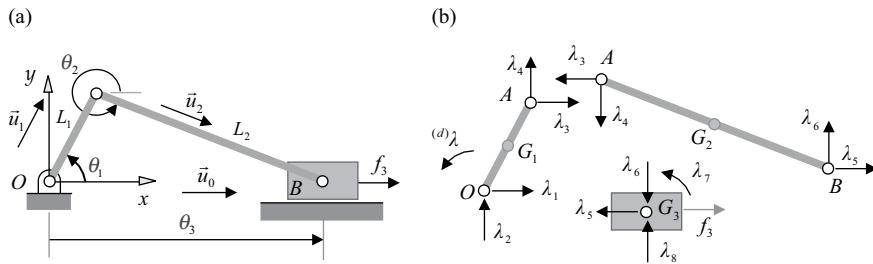
Perform kinematic analysis to obtain  $\mathbf{q}$ ,  $\dot{\mathbf{q}}$ , and  $\ddot{\mathbf{q}}$ .  
If  $\mathbf{q} \neq \mathbf{c}$ , determine  $\mathbf{c}$ ,  $\dot{\mathbf{c}}$ ,  $\ddot{\mathbf{c}}$ .

1. Construct the mass matrix and the array of applied forces.
2. Solve Eq. (12.13) (or Eq. (12.14)) for the Lagrange multiplier.

### Example 12.6

Consider the slider–crank mechanism from Example 12.1 that is shown in Figure 12.8. An applied force  $f_3 = 5.0 \text{ N}$  acts in the horizontal direction on the slider. The mass centers are at the geometric center of each link. The mass and moment of inertia for the links are

$$m_1 = 1.0 \text{ kg}, J_1 = 0.001 \text{ kg m}^2, m_2 = 1.5 \text{ kg}, J_2 = 0.002 \text{ kg m}^2, m_3 = 10.0 \text{ kg}, J_3 = 1.0 \text{ kg m}^2$$

**FIGURE 12.8**

(a) A slider-crank mechanism and (b) its FBD representation.

Based on the kinematic results obtained in Example 12.1, determine the torque that the driver motor applies on the crank during a complete revolution of the crank.

### Solution

We solve this problem by extending the program from Example 12.1.

#### MATLAB/Chapter 12/Example \_12 \_ 6

We first ask the revised program to perform a complete kinematic analysis by executing the program `Example_12_1`. Then we state the additional constant data, followed by initializing several arrays to save the computed torque and reaction force. We then start the inverse dynamic analysis formulations for each time step.

```
Example_12_1; % Perform kinematic analysis
m1 = 1.0; m2 = 1.5; m3 = 10.0;
J1 = 0.001; J2 = 0.002; J3 = 1.0;
f3 = 10; ;
Lag = zeros(nsteps,8);
Torque1 = zeros(nsteps,1); Torque2 = zeros(nsteps,1);
for i = 1:nsteps
    ...
End
```

Within the loop, we first retrieve the kinematic results for that time step:

```
theta      = thetas(i,:);
theta_d   = thetas_d(i,:);
theta_dd = thetas_dd(i,:);
```

We then establish the necessary unit vectors:

```
c1 = cos(theta(1)); s1 = sin(theta(1));
c2 = cos(theta(2)); s2 = sin(theta(2));
u1  = [ c1; s1]; u2  = [ c2; s2]; u0  = [ 1; 0];
u1r = [-s1; c1]; u2r = [-s2; c2]; u0r = [ 0; 1];
```

The velocities of the mass centers are determined as

$$\begin{aligned}\dot{\mathbf{r}}_1^G &= \frac{L_1}{2} \check{\mathbf{u}}_1 \dot{\theta}_1 \\ \dot{\mathbf{r}}_2^G &= L_1 \check{\mathbf{u}}_1 \dot{\theta}_1 + \frac{L_2}{2} \check{\mathbf{u}}_2 \dot{\theta}_2 \\ \dot{\mathbf{r}}_3^G &= \mathbf{u}_0 \dot{\theta}_3\end{aligned}\quad (12.15)$$

```
rG1_d = 0.5*L1*u1r*theta_d(1);
rG2_d = 2*rG1_d + 0.5*L2*u2r*theta_d(2);
rG3_d = u0*theta_d(3);
```

The accelerations of the mass centers are expressed as (knowing that  $\ddot{\theta}_1 = 0$ )

$$\begin{aligned}\ddot{\mathbf{r}}_1^G &= -\frac{L_1}{2} \mathbf{u}_1 \dot{\theta}_1^2 \\ \ddot{\mathbf{r}}_2^G &= -L_1 \mathbf{u}_1 \dot{\theta}_1^2 + \frac{L_2}{2} \left( \ddot{\mathbf{u}}_2 \dot{\theta}_2 - \mathbf{u}_2 \dot{\theta}_2^2 \right) \\ \ddot{\mathbf{r}}_3^G &= \mathbf{u}_0 \ddot{\theta}_3\end{aligned}\quad (12.16)$$

```
rG1_dd = 0.5*L1*(u1r*theta_dd(1) - u1*theta_d(1)^2);
rG2_dd = 2*rG1_dd + 0.5*L2*(u2r*theta_dd(2)
- u2*theta_d(2)^2);
rG3_dd = u0*theta_dd(3);
```

Based on the free-body diagrams (FBDs), or from the body-coordinate formulation, the equations of motion for the system can be written and arranged in matrix form as

$$\left[ \begin{array}{|c|c|c|c|c|} \hline \mathbf{I} & \mathbf{I} & & & | \\ \hline \check{\mathbf{s}}_1'^O & \check{\mathbf{s}}_1'^A & & & | \\ \hline \hline -\mathbf{I} & \mathbf{I} & & & | \\ \hline -\check{\mathbf{s}}_2'^A & \check{\mathbf{s}}_2'^B & & & | \\ \hline \hline -\mathbf{I} & & \mathbf{u}_0 & & | \\ \hline & & 1 & & | \\ \hline \end{array} \right] \left[ \begin{array}{c} \lambda_{1,2} \\ \lambda_{3,4} \\ \lambda_{5,6} \\ \lambda_7 \\ \lambda_8 \\ \hline^{(d)} \lambda \end{array} \right] = \left[ \begin{array}{c} m_1 \ddot{\mathbf{r}}_1^G \\ J_1 \ddot{\phi}_1 \\ m_2 \ddot{\mathbf{r}}_2^G \\ J_2 \ddot{\phi}_2 \\ m_3 \ddot{\mathbf{r}}_3^G - \mathbf{u}_0 f_3 \\ J_3 \ddot{\phi}_3 \end{array} \right] \quad (12.17)$$

We now evaluate the right-hand side array of the equations of motion.

```
rhs = [m1*rG1_dd
       J1*theta_dd(1)
       m2*rG2_dd
       J2*theta_dd(2)
       m3*rG3_dd - f3*u0
       0];
```

**Method 1** (Algorithm ID1): We construct the coefficient matrix of Eq. (12.17) and then solve the equations for all nine Lagrange multipliers. We save the result for the ninth multiplier associated with the motor in the array `Torque1`.

```
s01r = -0.5*L1*u1r; sA1r = -s01r;
sA2r = -0.5*L2*u2r; sB2r = -sA2r;
I22 = eye(2); Z22 = zeros(2); Z12 = zeros(1,2);
Dt = [I22 I22 Z22 Z22 Z12'
      s01r' sA1r' Z12 Z12 1
      Z22 -I22 I22 Z22 Z12'
      Z12 -sA2r' sB2r' Z12 0
      Z22 Z22 -I22 Z12' u0r Z12'
      Z12 Z12 Z12 1 0 0];
sol = Dt\rhs;
Lag(i,:) = sol(1:8)';
Torque1(i) = sol(9);
```

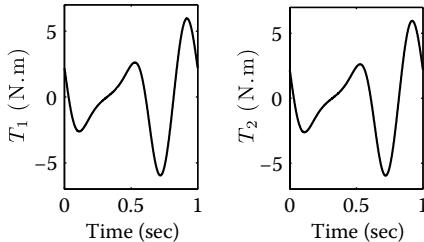
**Method 2 (Algorithm ID2):** We construct the array of body velocities and then solve one single equation, according to Eq. (12.13), for one Lagrange multiplier. We save the result for this multiplier, which is associated with the motor, in the array Torque2.

```
c_d = [rG1_d
       theta_d(1)
       rG2_d
       theta_d(2)
       rG3_d
       0 ]; % Array of body velocities
Torque2(i) = c_d'*rhs/theta_d(1);
```

Finally, we plot the torque of the motor versus time as computed by the two methods of solution.

```
subplot(1,2,1)
plot(T, Torque1)
subplot(1,2,2)
plot(T, Torque2)
```

Executing this program results in two identical plots as shown in Figure 12.9. If interested, we may refer to the results saved in the array Lag to retrieve the reaction forces at the kinematic joints.



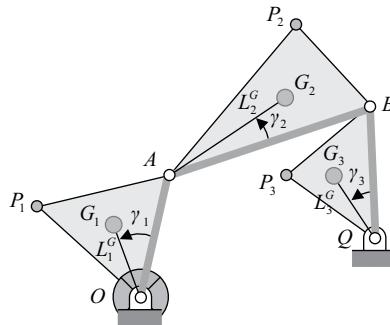
**FIGURE 12.9**  
Torque of the motor versus time.

### 12.2.1 A Program for Four-Bar Inverse Dynamics

**MATLAB/Chapter 12/KAP \_ IDAP/idap**

The program idap, for inverse dynamic analysis of four-bar mechanisms, is an extension of the kinematic analysis program kap. The input data for idap must be provided as an addition to the kinematics data in the script M-file data.m. The additional data for each link are the mass, the moment of inertia, and the position of each mass center with respect to its corresponding link as shown in Figure 12.10. The angles  $\gamma_1$ ,  $\gamma_2$ , and  $\gamma_3$  could be either positive (CCW as shown) or negative (CW).

In the input file, we also need to define the gravitational constant and direction. The gravitational constant that we provide establishes the system of units used for the model. If the gravitational force should not be included in the array of applied forces, we should set the gravitational constant to zero. In the M-file user\_force, we may provide other forces and torques that we want to apply to the system. The forces could be applied to



**FIGURE 12.10**  
A general description of a four-bar for the program idap.

**TABLE 12.2**

Arrays Used by idap for Saving the Moment Arms

Array	Description
sP1, sP1r	$\mathbf{s}_1^P, \dot{\mathbf{s}}_1^P$
sP2, sP2r	$\mathbf{s}_2^P, \dot{\mathbf{s}}_2^P$
sP3, sP3r	$\mathbf{s}_3^P, \dot{\mathbf{s}}_3^P$
sA1, sA1r	$\mathbf{s}_1^A, \dot{\mathbf{s}}_1^A$
sA2, sA2r	$\mathbf{s}_2^A, \dot{\mathbf{s}}_2^A$
sB2, sB2r	$\mathbf{s}_2^B, \dot{\mathbf{s}}_2^B$
sB3, sB3r	$\mathbf{s}_3^B, \dot{\mathbf{s}}_3^B$

points  $P_1$ ,  $P_2$ , and  $P_3$ , as well as points  $A$  and  $B$ . The moment arms for these points with respect to their corresponding mass centers are computed and available to the user, as  $2 \times 1$  arrays, as listed in Table 12.2.

The input file `data.m` can be read by either program `kap` or `idap`. Therefore, `idap` is a revised version of `kap` in which Eqs. (12.11) and (12.13) are constructed and solved; that is, both methods of evaluating the torque of the motor are implemented in the program, where the results are saved in arrays `Torque1` and `Torque2`. The Lagrange multipliers associated with the revolute joints are saved in array `Lag`.

### Example 12.7

In this example, we model the web-cutter mechanism from Section 15.2. We extract the following data from the figure:

$$L_{0(x)} = 0.65 \text{ m}, L_{0(y)} = -0.1 \text{ m}, L_1 = 0.1 \text{ m}, L_2 = 0.7 \text{ m}, L_3 = 1.0 \text{ m}$$

$$L_2^C = 1.0 \text{ m}, L_3^D = 0.82 \text{ m}, \beta_2 = -36.87^\circ, \beta_3 = -37.57^\circ$$

For the driver, we consider  ${}^0\theta_1 = 30^\circ$  and  $\omega_1 = 2\pi \text{ rad/s CCW}$ . The initial estimates for the other two angles are  $\theta_2 = 90^\circ$  and  $\theta_3 = 140^\circ$ .

The mass and moment of inertia for the links are

$$m_1 = 1.0 \text{ kg}, J_1 = 1.0 \text{ kg m}^2, m_2 = m_3 = 10.0 \text{ kg}, J_2 = J_3 = 5.0 \text{ kg m}^2$$

The mass centers are positioned as

$$L_1^G = 0, \gamma_1 = 0, L_2^G = 0.52 \text{ m}, \gamma_2 = -16.7^\circ, L_3^G = 0.68 \text{ m}, \gamma_3 = -17.1^\circ$$

The gravity acts in the negative  $y$ -direction. During the short period when the blades cut the web, resistive forces of 1,000 N are applied vertically to the blades.

We construct the following input M-file `data.m` for this mechanism.

#### MATLAB/Chapter 12/KAP \_ IDAP/Models/WC/data.m

```
RQ = [0.65; -0.1]; L = [0.1 0.7 1.0];
LP = [0 1.0 0.82]; beta = [0 -36.87 -37.57];
theta = [30 90 140]; omega1 = 2*pi;
axis_limits = [-0.2 1 -0.2 1];
m = [1 10 10]; J = [1.0 5.0 5.0];
LG = [0 0.52 0.68]; gamma = [0 -16.7 -17.1];
g = 9.81; ug = [0; -1];
```

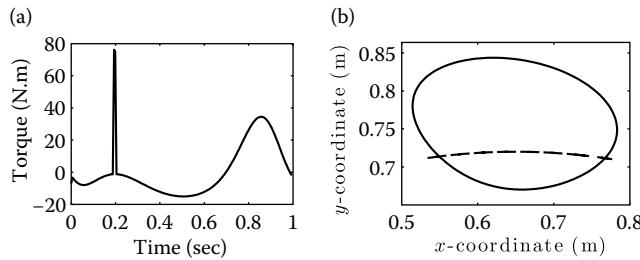
During the cut, it is assumed that a resistive force of 1,000 N is applied on each blade in the vertical direction. In the following user supplied M-file, the  $y$ -coordinates and velocity of points  $C$  and  $D$  are monitored. When the two points approach each other, and the gap is  $0 < \delta < 0.005$  m, the resistive forces are applied to the two points. The sign of the relative velocity is used to assure the forces are applied during the cut and not during the period when the blades are separating.

#### MATLAB/Chapter 12/KAP \_ IDAP/Models/WC/user \_ force

```
fcut = [0; 0];
del = RP2(i,2) - RP3(i,2);
if del > 0 && del < 0.005
    del_d = VP2(i,2) - VP3(i,2);
    if del_d < 0
        fcut = [0; 1000];
    end
end
h_a = h_a + ...
[0
0
0
fcut
sP2r'*fcut
-fcut
-sP3r'*fcut];
```

Executing `idap` and then running `anim` produces an animation of the simulated motion. We can plot the computed torque for the motor, from either `Torque1` or `Torque2`, producing the result shown in Figure 12.11a. We observe an impulsive-type peak in the torque during a cut. We can also plot the paths of points  $C_2$  and  $D_3$  using the following commands to produce the plot shown in Figure 12.11b:

```
plot(RP2(:,1),RP2(:,2))
hold on
plot(RP3(:,1),RP3(:,2),'k')
axis equal
```

**FIGURE 12.11**(a) Torque of the motor versus time; (b) path of points C<sub>2</sub> and D<sub>3</sub>.

### 12.2.2 Application in Robotics

Although the traditional application of inverse dynamics is for mechanisms, which are closed-chain with one DoF, we can apply the process to open-chain systems. One such application is in robotics for determining the required torques/forces of motors/actuators, in order for the robot to follow a prescribed path. This type of inverse dynamic analysis requires defining two different sets of driver constraints: a fictitious set and an actual set.

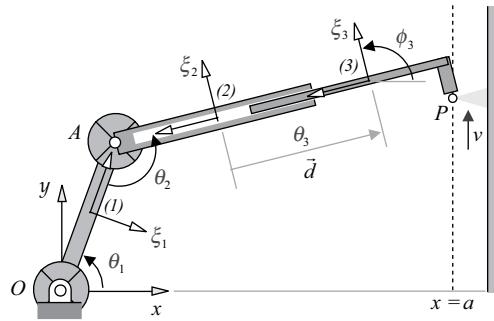
For the kinematic analysis, we define as many fictitious drivers (path generators) as the number of system's DoFs. These are called fictitious because there exist no actual motors or actuators at the corresponding coordinates to directly or explicitly impose the associated constraints. These drivers must specify the desired path or velocity of the robot's end-effector of the robot. With these driver constraints, the algebraic constraints and their first and second time derivatives are obtained at certain  $\Delta t$  time intervals, during a given period of time, or in a specified range of motion. The outcome of this step is known coordinates, velocities, and accelerations for every link of the robotic device.

In the second step of analysis, the fictitious drivers are replaced by as many actual driver constraints as the number of system's DoFs, representing actual motors and actuators. Knowing the kinematics of the system at every time step, the equations of motion are solved for the Lagrange multipliers associated with these actual driver constraints. This type of applications of inverse dynamics can be applied to the equations of motion formulated with any choice of coordinates. The process can be demonstrated with one example.

Let us consider the robotic device shown in Figure 12.12. The end-effector of this robot must follow a specified path, in this case a vertical line, as the end-effector sprays paint on an object along the path. In addition, the end-effector must keep a defined constant angle with respect to the path and move with a constant speed for a uniform spray. The objective of an inverse dynamic analysis for this device is to determine the required torques and the force of the motors and the linear actuator that would provide such motion for the end-effector, in the operating range of the robot.

Since this system has three DoFs, we can impose three conditions on its motion. The conditions are expressed as

$$\begin{aligned} x^P &= a & \dot{x}^P &= 0 & \ddot{x}^P &= 0 \\ y^P &= {}^0y^P + vt , \quad \dot{y}^P = v , \quad \ddot{y}^P = 0 \\ \phi_3 &= c & \dot{\phi}_3 &= c & \ddot{\phi}_3 &= 0 \end{aligned} \quad (12.18)$$

**FIGURE 12.12**

The end-effector of this robot follows a vertical line.

where  $a$ ,  $c$ , and  $v$  are constants and  ${}^0y^P$  is the initial position of point  $P$  along the path. These conditions can be viewed as three fictitious constraints, or path generators, in the kinematic analysis part of the process, resulting in the coordinates, velocities, and accelerations of the three bodies.

To perform a force analysis on this system, we need to define three driver constraints to represent the three actuators. In the vectorial or joint coordinate formulation, the drivers can simply be expressed as

$$\begin{aligned} {}^{(dr)}\Phi_1 &= \theta_1 - f_1(t) = 0 \\ {}^{(dr)}\Phi_2 &= \theta_2 - f_2(t) = 0 \\ {}^{(dr)}\Phi_3 &= \theta_3 - f_3(t) = 0 \end{aligned} \quad (12.19)$$

In the body-coordinate formulation, these equations become

$$\begin{aligned} {}^{(dr)}\Phi_1 &= \phi_1 - f_1(t) = 0 \\ {}^{(dr)}\Phi_2 &= \phi_2 - \phi_1 - f_2(t) = 0 \\ {}^{(dr)}\Phi_3 &= \frac{1}{2}(\mathbf{d}'\mathbf{d} - (f_3(t))^2) = 0 \end{aligned} \quad (12.20)$$

where  $\mathbf{d} = \mathbf{r}_3 - \mathbf{r}_2$ , and  $f_1(t)$ ,  $f_2(t)$ , and  $f_3(t)$  are the time functions that could be determined incrementally from the results obtained in the kinematic analysis step. The corresponding velocity drivers are

$$\left[ \begin{array}{cc|cc|cc} \mathbf{0} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & -1 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & -\mathbf{d}' & 0 & \mathbf{d}' & 0 \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{r}}_1 \\ \dot{\phi}_1 \\ \dot{\mathbf{r}}_2 \\ \dot{\phi}_2 \\ \dot{\mathbf{r}}_3 \\ \dot{\phi}_3 \end{array} \right\} = \left\{ \begin{array}{c} \dot{f}_1(t) \\ \dot{f}_2(t) \\ f_3(t)\dot{f}_3(t) \end{array} \right\} \quad (12.21)$$

The coefficient matrix is the Jacobian,  ${}^{(d)}\mathbf{D}$ , which can be used in Eq. (12.11) to determine the Lagrange multipliers associated with the three drives. We should note that in this process we do not need to determine the three functions in Eq. (12.20) either in analytical or in numerical form, because we have already determined the coordinates, velocities, and accelerations for all the bodies of the system.

### 12.3 Problems

12.1 Use the Newton–Raphson method to find the root(s) of the following equation:

$$x^3 - 10x^2 + 13x + 24 = 0$$

12.2 Use the Newton–Raphson method to find at least one set of answer of the following equations:

$$2x^2 - xy + y^2 - 4 = 0$$

$$3x^3 - 2y^2 + 2xy + 3 = 0$$

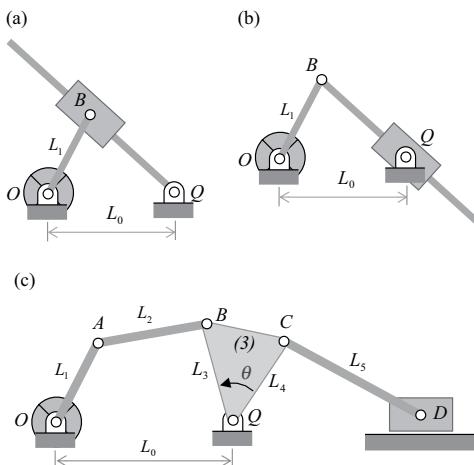
12.3 Consider one of the mechanisms shown for kinematic formulation. Apply the classical vector-loop method by defining the necessary coordinates. Write the position constraint equations and derive the corresponding velocity and acceleration constraints. Assume the crank is the driver, rotating with a constant angular velocity of  $2\pi$  rad/s CCW. Express the constraints, the Jacobian matrix, and the right-hand side of the acceleration constraints for analysis with

I. The CP method

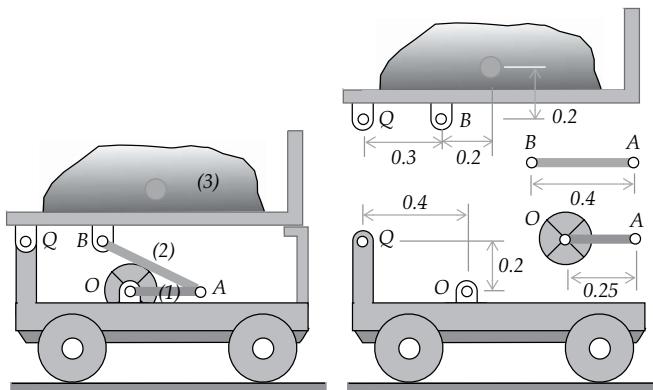
II. The AC method

Assume the following data as applicable:

$$L_0 = 2.0, L_1 = 1.0, L_2 = 1.8, L_3 = 1.5, L_4 = 1.0, L_5 = 2.0 \text{ m}, \theta = 30^\circ$$

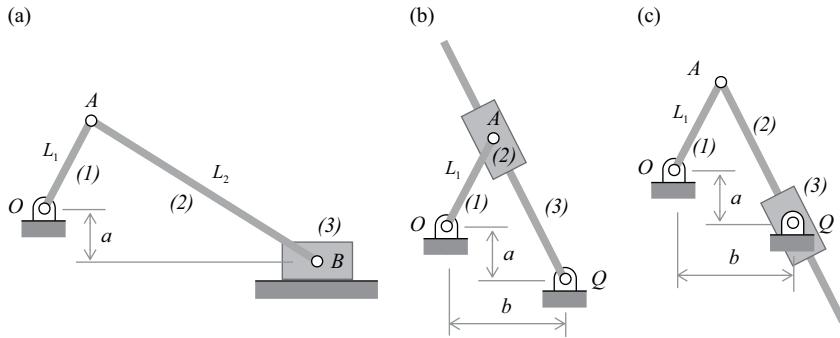


- 12.4 Develop a program to analyze the kinematics of the mechanisms in Problem 12.3. Formulate the equations based on either the CP or the AC method. For the nonlinear position constraints, apply
- MATLAB's `fsove` function
  - Newton-Raphson method
- 12.5 Develop a program to analyze the kinematics of the quick-return mechanism of Section 15.3.
- 12.6 Develop a program to analyze the kinematics of the dwell mechanism of Section 15.4.
- 12.7 Consider the film-strip advancer model from Example 12.5. Based on the simulation results, determine the answers for the following questions:
- How far apart should the sprocket holes be placed on the film strip?
  - What is the mean speed of the film strip during the phase that it is being pulled?
  - What is the mean speed of the film strip during one revolution of the crank?
- 12.8 Consider the web-cutter model from Example 12.7. Based on the simulation results, determine the answers for the following questions:
- What is the angle of the crank at the time of cutting?
  - What should be the velocity of the web at the time of cutting?
- 12.9 Consider the film-strip advancer model from Example 12.5 for inverse dynamic analysis. Assume a resistive force of 1.0 N is applied vertically on point *P* during its engagement with the film. Determine the required torque from the motor during one revolution of the crank.
- 12.10 Consider the small dumpster shown for inverse dynamic analysis. An electric motor rotates the crank of the mechanism with a constant angular velocity of  $\pi/20$  rad/s CCW for unloading. Assume that the load does not leave the dumpster until it is in the upright position. When the container reaches the vertical orientation, the motor is turned off. Use `idap` to determine the required torque from the motor during the unloading process. The figure shows the dimensions in meters. Use the following values for the masses and moments of inertia:  $m_1 = 2 \text{ kg}$ ,  $J_1 = 0.5 \text{ kg m}^2$  (link (1) and motor) with the mass center at *O*;  $m_2 = 1 \text{ kg}$ ,  $J_2 = 0.01 \text{ kg m}^2$ ;  $m_3 = 50 \text{ kg}$ ,  $J_3 = 2.5 \text{ kg m}^2$  (container plus the load).



12.11 Revise the program `kap` for a slider-crank mechanism. The three inversions are shown with an offset ground link in order to make each version more general.

- Inversion 1
- Inversion 2
- Inversion 3



12.12 Revise the program `idap` for any of the three inversions of a slider-crank mechanism shown in Problem 12.11.

## Reference

- Norton, R.L., *Design of Machinery*, 5th ed. New York, McGraw-Hill (2012).

# 13

---

## *Forward Dynamics*

---

The objective of forward dynamic analysis is to predict the motion of a multibody system under given applied loads starting from a set of initial conditions on the coordinates and velocities. Unlike kinematic analysis that the prescribed motion of the driving coordinate(s) determines the motion of the system, in forward dynamic analysis, it is the applied forces and torques that cause the motion. To predict the motion of a multibody system in forward dynamics, the equations of motion that are second-order differential equations are integrated. This process is performed numerically by integrating the accelerations and velocities to obtain the velocities and coordinates at the next time step. Although many numerical integration algorithms exist for this purpose, an algorithm that can dynamically control the integration time step is recommended over a constant step algorithm. Regardless of what algorithm we select for the integration, we must not forget that a numerical integration algorithm only provides an approximation to the correct solution; i.e., numerical error is always present in a solution.

If the equations of motion do not contain any algebraic constraints, a numerical integration algorithm can directly be applied. When the equations of motion contain algebraic constraints, in addition to the standard numerical integration processes, some other issues must be taken into consideration. In this chapter, we first consider the process of forward dynamics for unconstrained and then for constrained formulations. We examine several methods to control constraint violations due to numerical integration errors. We also discuss how to perform forward dynamic analysis on systems that involve impact, constraint addition during motion, or any form of discontinuities in velocities.

---

### 13.1 Unconstrained Formulation

Unconstrained equations of motion can be expressed in the following general form:

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{h} \quad (13.1)$$

This equation may represent the equations of motion for a system of unconstrained particles, a set of unconstrained bodies, or an open-chain system formulated by the joint coordinates. These equations are treated as linear algebraic equations when the unknowns are the accelerations. When the unknowns are the coordinates and velocities, the same equations are treated as second-order ordinary differential equations.

When treating the equations of motion as second-order differential equations, to integrate them numerically, two arrays are defined as

$$\mathbf{u} = \left\{ \begin{array}{c} \mathbf{q} \\ \dot{\mathbf{q}} \end{array} \right\}, \quad \dot{\mathbf{u}} = \left\{ \begin{array}{c} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{array} \right\} \quad (13.2)$$

If we integrate the array  $\dot{\mathbf{u}}$ , containing the velocities and accelerations, we obtain the array  $\mathbf{u}$ , containing the coordinates and velocities at the next time step. Obviously, this requires a set of initial values for the coordinates and velocities to start the integration process.

### Example 13.1

Refer to the system of two unconstrained bodies in Example 7.1. Define the necessary integration arrays to solve the equations of motion for this system using body-coordinate formulation.

#### Solution

For the two bodies, the coordinates were defined as  $\mathbf{r}_1$ ,  $\phi_1$ ,  $\mathbf{r}_2$ , and  $\phi_2$ . Therefore, the integration arrays are constructed as

$$\mathbf{u} = \begin{Bmatrix} \mathbf{r}_1 \\ \phi_1 \\ \mathbf{r}_2 \\ \phi_2 \\ \dot{\mathbf{r}}_1 \\ \dot{\phi}_1 \\ \dot{\mathbf{r}}_2 \\ \dot{\phi}_2 \end{Bmatrix}, \quad \dot{\mathbf{u}} = \begin{Bmatrix} \dot{\mathbf{r}}_1 \\ \dot{\phi}_1 \\ \dot{\mathbf{r}}_2 \\ \dot{\phi}_2 \\ \ddot{\mathbf{r}}_1 \\ \ddot{\phi}_1 \\ \ddot{\mathbf{r}}_2 \\ \ddot{\phi}_2 \end{Bmatrix}$$

### Example 13.2

For the variable-length pendulum of Example 9.3, define integration arrays using joint-coordinate formulation.

#### Solution

For this system, two joint coordinates are defined as  $\theta_1$  and  $\theta_2$ . Therefore, the integration arrays are as follows:

$$\mathbf{u} = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{Bmatrix}, \quad \dot{\mathbf{u}} = \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix}$$

#### 13.1.1 Initial Value Problems

Most general-purpose numerical integration algorithms are developed to solve first-order differential equations. A first-order differential equation is written as

$$\dot{u} = f(u, t) \tag{13.3}$$

where  $t$  is referred to as the *independent* variable and  $u = u(t)$  is considered a *dependent* variable. Equation (13.3) has a family of solutions  $u(t)$ . The choice of an initial value  ${}^0u$  serves to determine one of the solutions of the family. The initial value  ${}^0u$  could be defined for any

value of  ${}^0t$ , although it is often assumed that a transformation has been made so that  ${}^0t = 0$ . This does not affect the solution or methods used to approximate the solution.

In general, if  $n$  dependent variables are involved and  $n$  first-order differential equations are available, Eq. (13.3) is written in vector form as

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}, t) \quad (13.4)$$

with a set of initial conditions  ${}^0\mathbf{u}$ . In dynamics problems,  $t$  represents the time, and it may not appear explicitly in the equations.

Solving initial value problems involves a step-by-step process in which a sequence of discrete points  ${}^0t, {}^1t, {}^2t, \dots$  is generated. The discrete points may have either constant or variable spacing  ${}^i\Delta t = {}^{i+1}t - {}^i t$ , where  ${}^i\Delta t$  is the step size for any discrete point  ${}^i t$ . At each point  ${}^i t$ , the solution  $\mathbf{u}({}^i t)$  is approximated by a number  ${}^i u$ .

Since no numerical algorithm is capable of finding  $u({}^i t)$  exactly, the quantity  ${}^i \varepsilon = |u({}^i t) - {}^i u|$  is defined to represent the *total error* at  $t = {}^i t$ . The total error consists of two components: a *truncation error* and a *roundoff error*. The truncation error depends on the nature of the numerical algorithm used in computing  ${}^i u$ . The roundoff error is due to the finite word length in a computer. In this textbook, when the term *numerical error* is used, we are referring to the truncation error. Although there exist many algorithms for solving initial value problems, most of them are based on either *Taylor series expansion* or *polynomial approximation*.

An algorithm can be categorized either as *explicit* or as *implicit*. An explicit algorithm evaluates  ${}^{i+1}u \equiv u({}^{i+1}t)$  as a function of  ${}^i t$  and  ${}^i u \equiv u({}^i t)$  in one step, whereas implicit algorithms are iterative. In these algorithms, an estimate of  ${}^{i+1}u$  is required to start the iteration of the formula. In order to obtain a relatively good estimate for  ${}^{i+1}u$ , an explicit formula can be used. This step is known as a *predictor* step. Then, the implicit formula is used to correct the predicted value of  ${}^{i+1}u$ . This step is known as a *corrector* step. Normally, an algorithm iterates on the corrector step.

Algorithms that are based on Taylor series expansion require the derivatives of the function with respect to  $t$ . A  $k$ th order algorithm needs the derivatives up to  $k - 1$ . These derivatives, in general, are not readily available. Therefore, the higher order algorithms with better accuracy are not computationally efficient.

A classical theory asserts that any continuous function can be approximated arbitrarily within any closed interval by a polynomial of sufficiently high degree, even if the solution is not a polynomial. In view of this theory, a polynomial of sufficiently high order can, in principle, be used to calculate  $u({}^{i+1}t)$  to any desired accuracy. A  $k$ th-degree polynomial is referred to as an algorithm of order  $k$ . In practice, the amount of computation increases with the order of polynomial. In most polynomial approximation algorithms, information from previous time steps is utilized to compute  ${}^{i+1}u$ . An algorithm that utilizes information from time steps prior to  ${}^i t$  is called a *multistep* algorithm. The accuracy of an algorithm is directly proportional to its *order*. In general, the higher the order, the more accurate the algorithm can predict a solution.

Another commonly used family of explicit algorithms is the Runge–Kutta that can be considered as a revised form of Taylor algorithms. In Section 13.1.2, we review the fourth-order Runge–Kutta algorithm.

### 13.1.2 Runge–Kutta Algorithm

For a relatively large step size and reasonable accuracy, the *fourth-order Runge–Kutta* is a widely used algorithm. This algorithm is stated as

$${}^{i+1}u = {}^i u + \Delta t f \quad (13.5)$$

where

$$f = \frac{1}{6}(f_1 + 2f_2 + 2f_3 + f_4) \quad (13.6)$$

and

$$\begin{aligned} f_1 &= f\left({}^i u, {}^i t\right) \\ f_2 &= f\left({}^i u + \frac{\Delta t}{2} f_1, {}^i t + \frac{\Delta t}{2}\right) \\ f_3 &= f\left({}^i u + \frac{\Delta t}{2} f_2, {}^i t + \frac{\Delta t}{2}\right) \\ f_4 &= f\left({}^i u + \Delta t f_3, {}^i t + \Delta t\right) \end{aligned} \quad (13.7)$$

Because the algorithm is a fourth-order one, the truncation error remains relatively small even for a relatively large step size. The major disadvantage of this algorithm is that the function  $f(u, t)$  must be evaluated four times at each time step. In addition, the values of the function are not used in any subsequent computations. Hence, this algorithm is not computationally as efficient as some of the multistep algorithms.

The following is the function M-file RK4, written in a general form based on the fourth-order Runge–Kutta. The input–output arguments of this function are organized similar to the built-in MATLAB® function, `ode45`, which will be discussed later. The input arguments for RK4 are: the function where the equations of motion are constructed and solved, the time span `Tspan` that should be expressed as `start t : Δt : final t`, and the integration array containing the initial conditions on the variables (coordinates and velocities). The output of RK4 contains two arrays, `T` and `uT`. The column array `T` contains the time marks of the integration. The array `uT` contains the time record of `u'` (coordinates and velocities). The accelerations are not returned from RK4—if the accelerations are needed, they must be recomputed.

#### MATLAB/Chapter 13/RK4

```
function [T, uT] = RK4(fun, Tspan, u)
% 4th order Runge-Kutta algorithm
T = Tspan'; uT = u'; nT = size(Tspan);
for i=1:(nT(2) - 1)
    t = Tspan(i); dt = Tspan(i + 1) - Tspan(i);
    f1 = fun(t, u);
    f2 = fun((t + dt/2), (u + dt*f1/2));
    f3 = fun((t + dt/2), (u + dt*f2/2));
    f4 = fun((t + dt), (u + dt*f3));
    u = u + dt*(f1 + 2*(f2 + f3) + f4)/6;
    uT = [uT; u'];
end
```

**Example 13.3**

Refer to the equations of motion for the variable-length pendulum in Example 9.3. Use the fourth-order Runge–Kutta algorithm to find the response of the system for 5s.

**Solution**

To simulate the dynamics of the variable-length pendulum, we take the already constructed M-file `Ex_9_3` and split it in two parts with some slight modifications.

In the main M-file, we keep the input data, state them in a global statement, and keep the initial conditions for the joint coordinates and velocities. We construct the integration array `u`, provide values for the time increments `Tspan`, and then invoke the `RK4` algorithm. We direct `RK4` to find the equations of motion in function `Ex_13_3`.

**MATLAB/Chapter 13/Example\_13\_3, Ex\_13\_3**

```
...
theta = [pi/3; 0.8]; theta_d = [0; 0]; % Initial conditions
u = [theta; theta_d]; % Integration array
Tspan = 0:0.02:5.0; % Time span
[T, uT] = RK4 (@Ex_13_3, Tspan, u); % 4th order R-K
```

In the function M-file for the equations of motion, we keep most of the statements from the original `Example_9_3` file intact. We pass the constant input data to this function through the global statement. Following the computation of the joint accelerations, we stack the joint velocities and acceleration in the `ud` array to be returned to the integrator.

```
function ud = Ex_13_3(t, u)
...
theta = u(1:2); theta_d = u(3:4); % coord. and velocities
% Recursive transformations
phi1 = theta(1); A1 = A_matrix(phi1);
...
theta_dd = M_joint\h_joint; % Joint accelerations
ud = [theta_d; theta_dd]; % ud array
```

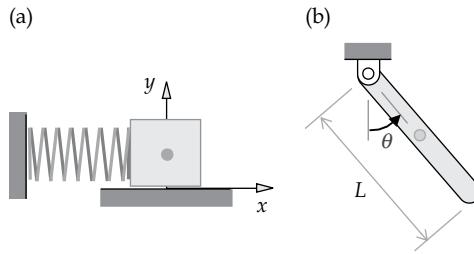
Executing the program `Example_13_3` provides the joint coordinates and velocities at every time step. We may plot, for example,  $\dot{\theta}_1$  versus  $t$  using the following command:

```
plot(T, uT(:, 3))
```

### 13.1.3 Integration Time-Step Size

An important issue in using a constant-step numerical integration algorithm, such as the fourth-order Runge–Kutta, is the selection of a proper step size. A large step size may cause erroneous results. A step size too small may yield accurate results while increasing the computation time too much. Therefore, it is important to choose a reasonably small step size to obtain accurate results without unnecessarily increasing the computation time. A thorough discussion on the subject of time-step selection is outside the scope of this textbook. Interested reader may refer to textbooks on the subject of numerical solution of differential equations.

The size of the integration time step should be adjusted based on of the natural frequencies of a system—the higher the frequency, the smaller the time step. To demonstrate this concept, we consider the one-dimensional mass–spring system shown in Figure 13.1a. The equation of motion for this system is written as

**FIGURE 13.1**

(a) A one-dimensional oscillating mass–spring; (b) a single pendulum.

$$m\ddot{x} + kx = 0 \Rightarrow \ddot{x} + \frac{k}{m}x = 0 \quad (13.8)$$

This system exhibits an oscillatory motion, and therefore, the solution to this differential equation can be assumed to be of the form

$$x = a \cos(\omega t + \phi) \quad (13.9)$$

where  $a$  and  $\phi$  are the *amplitude* and the *phase shift*, respectively, that can be determined based on the initial conditions of  $x$  and  $\dot{x}$ . In this equation,  $\omega$  is the *natural frequency* and is equal to

$$\omega = \sqrt{\frac{k}{m}} \quad (13.10)$$

The period of oscillation associated with  $\omega$  is

$$\tau = \frac{2\pi}{\omega} \quad (13.11)$$

If Eq. (13.8) is solved numerically, the step size  $\Delta t$  must be much smaller than the period  $\tau$ . A reasonable value for  $\Delta t$  could be

$$\Delta t < \tau/10 \quad (13.12)$$

Assuming that  $m = 1 \text{ kg}$  and  $k = 100 \text{ N/m}$ , the time period is found to be  $\tau = 0.628 \text{ s}$ .

As another example, consider the single pendulum shown in Figure 13.1b. For a rod with a length  $L$  and a mass  $m$ , the moment of inertia is  $\frac{1}{3}mL^2$ . If the only external force is the gravity, the equation of motion in the joint-coordinate formulation is

$$\frac{1}{3}mL^2\ddot{\theta} + \frac{1}{2}mgL\sin\theta = 0$$

For oscillations of small amplitudes,  $\sin\theta$  can be replaced by  $\theta$ , and therefore, this equation is linearized to

$$\frac{1}{3}mL^2\ddot{\theta} + \frac{1}{2}mgL\theta = 0 \Rightarrow \ddot{\theta} + \frac{3g}{2L}\theta = 0 \quad (13.13)$$

This equation has the same form as that of Eq. (13.8). Therefore, its natural frequency is

$$\omega = \sqrt{\frac{3g}{2L}} \quad (13.14)$$

Assuming  $L = 1$  and  $g = 9.81$ , the time period is found to be  $\tau = 1.64$ s.

To see how damping affects the natural frequencies and, hence, the selection of the time steps, let us revise Eq. (13.8) to include damping, where  $d_c$  is the *damping coefficient*:

$$m\ddot{x} + d_c\dot{x} + kx = 0 \Rightarrow \ddot{x} + \frac{d_c}{m}\dot{x} + \frac{k}{m}x = 0 \quad (13.15)$$

For convenience, we define a *damping ratio*,  $\zeta$ , as [1,2]

$$\zeta = \frac{d_c}{2\sqrt{km}} \quad (13.16)$$

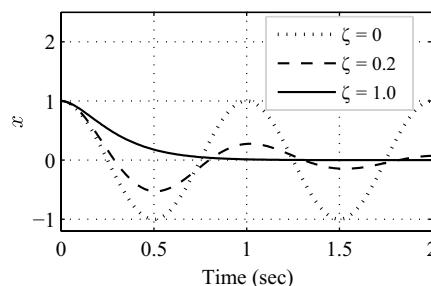
With this definition, Eq. (13.15) is expressed as

$$\ddot{x} + 2\zeta\omega\dot{x} + \omega^2x = 0 \quad (13.17)$$

In this equation,  $\omega = \sqrt{k/m}$  is the *undamped frequency* as defined in Eq. (13.10). The *damped frequency* is expressed as

$$\omega_d = \omega\sqrt{1 - \zeta^2} \quad (13.18)$$

Figure 13.2 shows the solution to Eq. (13.15) for three different values of the damping ratio, where it is assumed that  $\omega = 2\pi$  rad/s, and the initial conditions  $x = 1.0$  and  $\dot{x} = 0$ . For  $\zeta = 0$ , we have an oscillatory response with no damping, with the time period  $\tau = 1.0$ s. For  $\zeta = 0.2$ , the response begins to decay—as long as  $0 < \zeta < 1$ , the system is called *underdamped*. We note that for this response,  $\omega_d$  is slightly smaller than  $2\pi$ . When  $\zeta = 1$ , the system is *critically damped*.



**FIGURE 13.2**

The undamped ( $\zeta = 0$ ), underdamped ( $0 < \zeta < 1$ ), and critically damped ( $\zeta = 1$ ) responses.

damped, for which  $\omega_d = 0$ . In general, since  $\omega_d < \omega$ , for the purpose of the time-step selection, we should use the undamped frequency as our reference.

For the mass-spring example, the integration time step should be  $\Delta t \leq 0.06$ , and for the pendulum example, the time step should be  $\Delta t \leq 0.16$ . These examples show how the time period of the oscillation can be calculated. However, for more complex systems, the calculation of natural frequencies will not be that simple. For multibody systems, the linearization of the equations of motion in explicit form can be rather cumbersome. For systems with more than one degree of freedom (DoF), there will be more than one natural frequency. For such systems, the highest frequency must be found, and a step size much smaller than the period of the highest frequency must be selected. Since the equations of motion are generally nonlinear in terms of the coordinates, linearization of these equations yields a time step, which is valid only for the configuration around which the system has been linearized. In a different configuration, the linearization process may yield a different time period, and consequently a different time-step size.

To avoid the difficulties associated with the selection of a proper size for the time step, it is strongly recommended that a variable-step size algorithm be used. Many well-developed algorithms are available that determine a proper time step and will automatically adjust the time-step size during the course of a simulation. One such algorithm is the `ode45` in MATLAB.\* The function `ode45` can be utilized as

```
[T, zT] = ode45 (@Example, Tspan, z)
```

The input-output arguments for this function are the same as those for function `RK4`. The main difference is that the value we specify for  $\Delta t$  (in `Tspan`) is the reporting time interval and not the time step for the integration. If  $\Delta t$  is not stated in `Tspan`, `ode45` will report the results at every integration time step.

### 13.1.4 General Procedure

Assume that a multibody system is modeled without any constraints, and therefore, the number of defined coordinates,  $n_v$ , is equal to the number of DoFs of the system,  $n_{dof}$ . A process for forward dynamic analysis can be stated as in the following algorithm.

#### Algorithm UC

1. Define the initial conditions for the coordinates and velocities,  ${}^0\mathbf{q}$  and  ${}^0\dot{\mathbf{q}}$ .
2. Initialize an integration array as  $\mathbf{u} = \left\{ \begin{array}{c} {}^0\mathbf{q} \\ {}^0\dot{\mathbf{q}} \end{array} \right\}$ .

---

\* The function `ode45` uses two error tolerances: a relative error tolerance, `RelTol`, and an absolute error tolerance, `AbsTol`. The default settings for these tolerances are `RelTol = 1e-3` and `AbsTol = 1e-6`. These tolerances can be reset by the user as shown in the following statements:

```
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-9);
[T, zT] = ode45(@BC_eqsmo, Tspan, z, options);
```

If we observe a significant numerical error when simulating the response of a multibody system, tightening the error tolerances may remedy the problem. This in turn significantly increases the computational time.

3. Establish the time parameters for integration.
4. Utilize a numerical integrator such as `ode45`; send **u** and other necessary data to the integrator.

Numerical integrator such as `ode45`, RK4, ...

The integrator sends **u**, for  $t = it$ , to “function \_ uc” and asks for  $\dot{u}$ :

```
function _ uc
```

This function constructs and solves the equations of motion for accelerations, and returns  $\dot{u}$  to the integrator.

5. The integrator returns the computed coordinates and velocities for every time step.

Most numerical integrators, such as `ode45` or `RK4`, do not return the computed accelerations to the main program, so the computed accelerations are lost. To recover the accelerations at the completion of the integration, we can recompute them by solving the equations of motion as algebraic equations at every time step.

## 13.2 Constrained Formulation

The constrained equations of motion, including all the kinematic constraints, are expressed as

$$\Phi(\mathbf{q}) = \mathbf{0} \quad (13.19)$$

$$\dot{\Phi} = \mathbf{D}\dot{\mathbf{q}} = \mathbf{0} \quad (13.20)$$

$$\ddot{\Phi} = \mathbf{D}\ddot{\mathbf{q}} - \boldsymbol{\gamma} = \mathbf{0} \quad (13.21)$$

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{h} + \mathbf{D}'\boldsymbol{\lambda} \quad (13.22)$$

These equations could represent a system of constrained particles, a set of bodies formulated in body coordinates, or a closed-chain system represented by joint coordinates. These equations can be treated as linear algebraic equations, if the coordinates and velocities are known and the accelerations and Lagrange multipliers are unknown. When the unknowns are the coordinates and velocities, the same equations must be treated as second-order mixed differential-algebraic equations.

To solve Eq. (13.22) for the accelerations, we append to it the acceleration constraints of Eq. (13.21) to form the following set of equations:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{D}' \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{h} \\ \boldsymbol{\gamma} \end{Bmatrix} \quad (13.23)$$

Knowing the coordinates and velocities, these equations can be solved for  $\ddot{\mathbf{q}}$  and  $\lambda$ .

Considering Eq. (13.22) as differential equations, two integration arrays are defined, similar to the unconstrained formulation, as

$$\mathbf{u} = \begin{Bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{Bmatrix}, \quad \dot{\mathbf{u}} = \begin{Bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{Bmatrix} \quad (13.24)$$

If we integrate  $\dot{\mathbf{u}}$ , which contains velocities and accelerations, we obtain  $\mathbf{u}$  that contains the coordinates and velocities for the next time step. This requires a set of initial values for the coordinates and velocities to start the integration. The initial values for the coordinates and velocities must satisfy the position and velocity constraints, respectively.

#### Example 13.4

The equations of motion for the variable-length pendulum, shown in Figure 13.3, were constructed with the body-coordinate formulation in Example 7.3. These equations are considered for numerical integration by RK4 or `ode45`. Construct the required integration arrays.

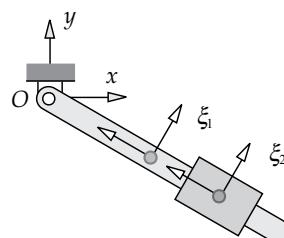
#### Solution

For a general-use integrator such as RK4 or `ode45`, it is not an issue whether the differential equations are constrained. As long as there are two bodies and the equations are obtained with the body coordinates, each integration array will contain  $2(3n_b) = 12$  variables. To simplify the notation, let us define the coordinates and velocities of the bodies as

$$\mathbf{c}_1 = \begin{Bmatrix} \mathbf{r}_1 \\ \phi_1 \end{Bmatrix}, \quad \mathbf{c}_2 = \begin{Bmatrix} \mathbf{r}_2 \\ \phi_2 \end{Bmatrix}, \quad \dot{\mathbf{c}}_1 = \begin{Bmatrix} \dot{\mathbf{r}}_1 \\ \dot{\phi}_1 \end{Bmatrix}, \quad \dot{\mathbf{c}}_2 = \begin{Bmatrix} \dot{\mathbf{r}}_2 \\ \dot{\phi}_2 \end{Bmatrix}$$

Then the integration arrays are constructed as

$$\mathbf{u} = \begin{Bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \dot{\mathbf{c}}_1 \\ \dot{\mathbf{c}}_2 \end{Bmatrix}, \quad \dot{\mathbf{u}} = \begin{Bmatrix} \dot{\mathbf{c}}_1 \\ \dot{\mathbf{c}}_2 \\ \ddot{\mathbf{c}}_1 \\ \ddot{\mathbf{c}}_2 \end{Bmatrix}$$



**FIGURE 13.3**

A variable length pendulum.

The coefficient matrix in Eq. (13.23) contains numerous zero entries— $\mathbf{M}$  is diagonal in body-coordinate formulation,  $\mathbf{D}$  contains many zeros, and we also have a zero  $n_c \times n_c$  matrix at the lower right corner. Any operation on such a matrix in the process of determining the accelerations and Lagrange multipliers may computationally be inefficient. To improve the computational efficiency, we can obtain the solution in a two-step process. For that purpose, we rewrite Eq. (13.22) as

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{h} + \mathbf{D}'\lambda) \quad (13.25)$$

Then we substitute the preceding equation in Eq. (13.21) and rearrange the terms to get

$$\mathbf{D}\mathbf{M}^{-1}\mathbf{D}'\lambda = \gamma - \mathbf{D}\mathbf{M}^{-1}\mathbf{h} \quad (13.26)$$

Since the product  $\mathbf{D}\mathbf{M}^{-1}\mathbf{D}'$  is a square matrix, we first solve Eq. (13.26) for  $\lambda$  and then solve Eq. (13.25) for  $\ddot{\mathbf{q}}$ . We note that in the body-coordinate formulation,  $\mathbf{M}$  is a constant diagonal matrix; therefore,  $\mathbf{M}^{-1}$  is computed easily and only once.

### 13.2.1 Initial Conditions

Solving second-order differential equations requires initial conditions on the coordinates and velocities. Although these initial conditions must represent the state of the system at the initial time, in general, for unconstrained equations, the initial conditions may be assigned arbitrary values. But, when the system is constrained, the initial conditions must satisfy the constraints at the position and velocity levels:

$$\begin{aligned} \Phi({}^0\mathbf{q}) &= \mathbf{0} \\ {}^{(dr)}\Phi({}^0\mathbf{q}) - \mathbf{f}(t) &= \mathbf{0} \end{aligned} \quad (13.27)$$

$$\begin{aligned} \dot{\Phi} = \mathbf{D}{}^0\dot{\mathbf{q}} &= \mathbf{0} \\ {}^{(dr)}\dot{\Phi} = {}^{(dr)}\mathbf{D}{}^0\dot{\mathbf{q}} - \dot{\mathbf{f}}(t) &= \mathbf{0} \end{aligned} \quad (13.28)$$

In these equations, we have considered the possibility of driver-type constraints that might be present. If we start numerical integration with a set of initial conditions that do not satisfy the constraints, most likely the integrated response will be erroneous. In Chapter 14, we will discuss several methods to assure the initial conditions do not violate the constraints.

### 13.2.2 General Integration Procedure

Assume that a multibody system with  $n_{dof}$  DoFs is represented by a set of  $n_v$  coordinates and  $n_c = n_v - n_{dof}$  independent kinematic constraints. A process for forward dynamic analysis can be stated as in the following algorithm.

### Algorithm C

1. Define the initial conditions for the coordinates and velocities, i.e.,  ${}^0\mathbf{q}$  and  ${}^0\dot{\mathbf{q}}$ . The initial conditions must satisfy the constraints at the coordinate and velocity levels.
2. Initialize an integration array as  $\mathbf{u} = \begin{Bmatrix} {}^0\mathbf{q} \\ {}^0\dot{\mathbf{q}} \end{Bmatrix}$ .
3. Establish the time parameters for integration.
4. Utilize a numerical integrator such as `ode45`; send  $\mathbf{u}$  and other necessary information to the integrator.

Numerical integrator such as `ode45`, `RK4`,...

The integrator sends  $\mathbf{u}$ , for  $t = it$ , to “function\_c” and asks for  $\dot{\mathbf{u}}$ :

```
function _c
```

This function constructs and solves the equations of motion for accelerations and Lagrange multipliers, and returns  $\dot{\mathbf{u}}$  to the integrator.

5. The integrator returns the computed coordinates and velocities for every reporting time step.

### Example 13.5

In Example 13.3, we considered the joint-coordinate formulation of the equations of motion for the variable-length pendulum as a set of unconstrained differential equations. In this example, we consider the body-coordinate formulation of the equations of motion for the same system. We can use either `RK4` or `ode45` to find the response of the system.

#### *Solution*

The body-coordinate formulation of the equations of motion for this pendulum has been derived in Example 7.3 as a set of constrained differential equations. For a dynamic simulation, similar to the program in Example 13.3, we construct the following two M-files:

**MATLAB/Chapter 13/ Example\_13\_5, Ex\_13\_5**

In the M-file `Example_13_5`, we provide the constant data and make it available to the function M-file `Ex_13_5` through a global statement.

```

...
c = [0.4330; -0.2500; pi/3; 0.6928; -0.4000; pi/3]; %
coordinates
c_d = zeros(6,1)      % velocities
u = [c; c_d];         % Integration array
Tspan = 0:0.02:5.0;   % Time span
[T, uT] = ode45(@Ex_13_5, Tspan, u);

```

In the function M-file `Ex_13_5`, we solve the equations of motion and return the time derivative of the integration variables, such as velocities and accelerations, to the integrator.

```
function ud = Ex_13_5(t, u)
    ...
    c = u(1:6); c_d = u(7:12);
    % Extract coordinates and velocities
    r1 = c(1:2); phi1 = c(3); A1 = A_matrix(phi1);
    r2 = c(4:5); phi2 = c(6); A2 = A1;
    s_O1 = A1*s_O1_p;
    u1 = A1*u1_p; u2 = A2*u2_p;
    r1_d = c_d(1:2); phi1_d = c_d(3);
    r2_d = c_d(4:5); phi2_d = c_d(6);
    ...
    solution = DMD\rhs;    % Accelerations and Lag. multipliers
    c_dd = solution(1:6); % Extract accelerations
    ud = [c_d; c_dd];     % Construct ud array
```

If we compare the simulation results from Example 13.3 against those from Example 13.5, we find that they are the same. For example, the command `plot(T, uT(:, 9))` provides the angular velocity of body (1) versus time. The main difference between the two simulations is that in Example 13.3 we integrated 4 variables, whereas in Example 13.5 we integrated 12 variables. Furthermore, in the body-coordinate formulation, there is a possibility of constraint violations, as it will be discussed in Section 13.3. There are no constraints in the joint-coordinate formulation of open chains and, therefore, no violations to be concerned about.

### 13.3 Constraint Violation

The procedure for forward dynamic analysis discussed in Section 13.2.2 does not guarantee that the coordinate and velocity constraints are not violated as the integration process progresses. The violation is due to the presence of numerical error that is inherent in any numerical integration routine. In the procedure of Algorithm C, only the constraints at the acceleration level are used—the constraints at the coordinate and velocity levels are not considered anywhere in the process. When the integration algorithm predicts values for the coordinates and velocities for the next time step, the predicted (approximated) values may not satisfy their corresponding constraints with adequate accuracy.

Violation of constraints may be negligible in some problems but significant in others. In Sections 13.3.1–13.3.5, we discuss several techniques for controlling the growth of the constraint violation or eliminating it completely.

#### 13.3.1 Constraint Violation Stabilization Method

The *constraint violation stabilization method* [3] is an extension of the feedback control theory applied to the equations of motion. One of the goals in designing a feedback controller is to suppress the growth of error and achieve a stable response. This stabilization method requires the coordinate and velocity constraints to be evaluated before we determine the accelerations:

$$\Phi = \Phi(\mathbf{q}) \quad (13.29)$$

$$\dot{\Phi} = \mathbf{D}\dot{\mathbf{q}} \quad (13.30)$$

If the coordinates and velocities satisfy the constraints, then  $\Phi = 0$  and  $\dot{\Phi} = 0$ . However, numerical error causes the constraint evaluation to yield  $\Phi \neq 0$  and  $\dot{\Phi} \neq 0$ . The stabilization method modifies the acceleration constraints,  $\ddot{\Phi} = 0$ , by adding two feedback terms as

$$\ddot{\Phi} + 2\alpha\dot{\Phi} + \beta^2\Phi = 0$$

or

$$\ddot{\Phi} = \mathbf{D}\ddot{\mathbf{q}} - \gamma + 2\alpha\dot{\Phi} + \beta^2\Phi = 0 \quad (13.31)$$

where  $\alpha$  and  $\beta$  are two positive parameters. Hence, Eq. (13.23) becomes

$$\begin{Bmatrix} \mathbf{M} & -\mathbf{D}^T \\ \mathbf{D} & \mathbf{0} \end{Bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \mathbf{h} \\ \gamma - 2\alpha\dot{\Phi} - \beta^2\Phi \end{Bmatrix} \quad (13.32)$$

Then, in the procedure of Section 13.2.2, Eq. (13.32) replaces Eq. (13.23).

We should be reminded that the feedback terms stabilize a *linear* system of second-order differential equations, whereas the equations of motion for multibody systems are nonlinear. Therefore, the feedback terms may not always be effective in controlling the error. In some problems, the feedback terms may actually cause the constraint violations to grow more rapidly. In general, for nonzero values of  $\alpha$  and  $\beta$ , the solution oscillates about the *approximated* solution. The amplitude and frequency of the oscillation, caused by the stabilization terms, depend on the values of  $\alpha$  and  $\beta$ . In most practical problems, a range of values between 1 and 10 for these parameters might be adequate. When  $\alpha = \beta$ , critical damping is achieved that usually stabilizes the response more quickly.\*

### Example 13.6

Consider the body-coordinate formulation of the equations of motion for the variable-length pendulum from Example 13.5. Revise the equations of motion in the program from that example by implementing the constraint stabilization terms, and set  $\alpha = \beta$ .

#### Solution

We duplicate the M-files from Example 13.5 and rename them `Example_13_6` and `Ex_13_6`.

#### MATLAB/Chapter 13/Example\_13\_6, Ex\_13\_6

In the M-file `Example_13_6`, we provide additional data for the stabilization parameters, include the parameters in a global statement, and revise the function handle in the argument list of `ode45` (or `RK4`) to `Ex_13_6`.

---

\* Comparing Eqs. (13.31) and (13.17), we note that  $\beta$  is the natural frequency of the undamped system, and setting  $\alpha = \beta$  is equivalent to defining a damping ratio  $\zeta = 1$ , that is, a critically damped system.

```

    ...
alpha = 5; beta = alpha;
...
[T, uT] = ode45(@Ex_13_6, Tspan, u);
% Recover constraint violations
nt = length(T);
Phi_norm = zeros(nt,1); Phi_d_norm = zeros(nt,1);
for i = 1:nt
    [P P_d] = Ex_13_6(T(i), u = uT(i,:)');
    Phi_norm(i) = norm(Phi); Phi_d_norm(i) = norm(Phi_d);
end

```

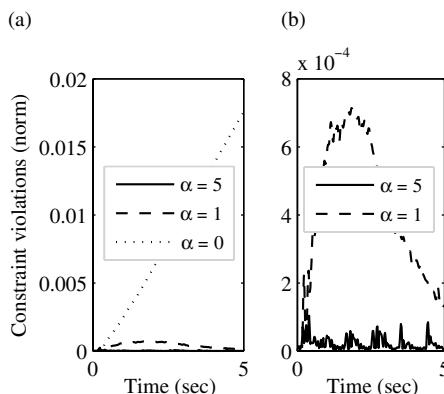
In the main program, following the integration, we include statements to recompute the constraint violations at every reported time step. The norms of the position and velocity constraints are computed and saved in separate arrays for further processing. In the function M-file `Ex_13_6`, we include an additional global statement. We add statements for evaluating the position and velocity constraints. And finally we include the stabilization terms in the statement for determining the accelerations:

```

function ud = Ex_13_6(t, u)
    ...
Phi = [r_O1
       ulr'*u2
       ulr'*d]; % Evaluate position constraints
Phi_d = D*c_d; % Evaluate velocity constraints
gamma = gamma - 2*alpha*Phi_d - beta^2*Phi; % Stab. terms
    ...

```

To observe how the feedback terms control the accumulation of the constraint error, we perform three simulations with  $\alpha = 0$  (no feedback),  $\alpha = 1$ , and  $\alpha = 5$ . Figure 13.4a shows the norm of the four position constraint violations versus time. For  $\alpha = 0$ , the integration causes the largest violation, and as the value of  $\alpha$  is increased, the stabilization terms provide some degree of control over the violations. It may appear that  $\alpha = 5$  leads to zero violation. However, if we remove the results for  $\alpha = 0$  from the plot for a higher resolution, as shown in Figure 13.4b, we observe that the error associated with  $\alpha = 5$  is small, but not zero.



**FIGURE 13.4**

Norm of the violations in the position constraint versus time for (a)  $\alpha = 0, 1, 5$ ; (b)  $\alpha = 1, 5$ .

Based on the results from Example 13.6, we should not get the impression that by increasing the value of  $\alpha$  we will always reduce violations. Depending on the system being simulated, large values of  $\alpha$  may become the dominant terms in the acceleration constraints and cause the integration process to become unstable.

### 13.3.2 Coordinate Partitioning Method

The *coordinate partitioning method* [4] controls the accumulation of the numerical error quite differently than the constraint violation stabilization method. The method makes use of the fact that when constraints are present, the coordinates are not independent. The coordinates, and their first and second time derivatives, are partitioned into *independent* and *dependent* sets as follows:

$$\mathbf{q} = \begin{Bmatrix} {}^{(i)}\mathbf{q} \\ {}^{(d)}\mathbf{q} \end{Bmatrix}, \dot{\mathbf{q}} = \begin{Bmatrix} {}^{(i)}\dot{\mathbf{q}} \\ {}^{(d)}\dot{\mathbf{q}} \end{Bmatrix}, \ddot{\mathbf{q}} = \begin{Bmatrix} {}^{(i)}\ddot{\mathbf{q}} \\ {}^{(d)}\ddot{\mathbf{q}} \end{Bmatrix} \quad (13.33)$$

We are reminded that there are as many independent coordinates as the number of DoFs, and there are many dependent coordinates as the number of constraints. Based on this partitioning, the integration arrays of Eq. (13.24) are redefined in smaller  $2n_{dof}$  arrays as

$$\mathbf{u} = \begin{Bmatrix} {}^{(i)}\mathbf{q} \\ {}^{(i)}\dot{\mathbf{q}} \end{Bmatrix}, \dot{\mathbf{u}} = \begin{Bmatrix} {}^{(i)}\dot{\mathbf{q}} \\ {}^{(i)}\ddot{\mathbf{q}} \end{Bmatrix} \quad (13.34)$$

The integration algorithm predicts the contents of the  $\mathbf{u}$  array for the next time step. Having a solution for the independent coordinates  ${}^{(i)}\mathbf{q}$ , we can solve the constraints of Eq. (13.19) for  ${}^{(d)}\mathbf{q}$ :

$${}^{(i)}\mathbf{q} \Rightarrow \Phi({}^{(d)}\mathbf{q}, {}^{(i)}\mathbf{q}) = \mathbf{0} \Rightarrow {}^{(d)}\mathbf{q} \quad (13.35)$$

Having the solution for all of the coordinates and the independent velocities,  ${}^{(i)}\dot{\mathbf{q}}$ , Eq. (13.20) can be solved for the dependent velocities,  ${}^{(d)}\dot{\mathbf{q}}$ :

$${}^{(i)}\dot{\mathbf{q}} \Rightarrow {}^{(d)}\mathbf{D}({}^{(d)}\dot{\mathbf{q}} + {}^{(i)}\mathbf{D}({}^{(i)}\dot{\mathbf{q}}) = \mathbf{0} \Rightarrow {}^{(d)}\dot{\mathbf{q}} \quad (13.36)$$

This process ensures that at every solution time step, all of the constraints are satisfied.

Although in principle the coordinate partitioning method can completely eliminate the constraint violation, the actual process may encounter difficulties. One troublesome part of this process is the step where position constraints are solved for the dependent coordinates. Since the position constraints are solved by iterative methods, such as Newton-Raphson, the process requires relatively accurate estimates for the dependent coordinates in every time step. Finding a good estimate for the coordinates is not a problem since the values of  ${}^{(d)}\mathbf{q}$  at  $i-1$  can be used as estimates for  ${}^{(d)}\mathbf{q}$  at  $it$ . In most cases, this estimate will yield a solution; however, in cases where the independent coordinates are poorly selected, the iterative process may fail to converge.

Proper partitioning of the coordinates into independent and dependent sets is critical in controlling the accumulation of numerical error. To clarify this point, let us consider the

single pendulum shown in Figure 13.5a. The body coordinates of this pendulum are  $x$ ,  $y$ , and  $\phi$ . The constraint equations for the revolute joint are

$$x - L \sin \phi = 0, y + L \cos \phi = 0 \quad (\text{a})$$

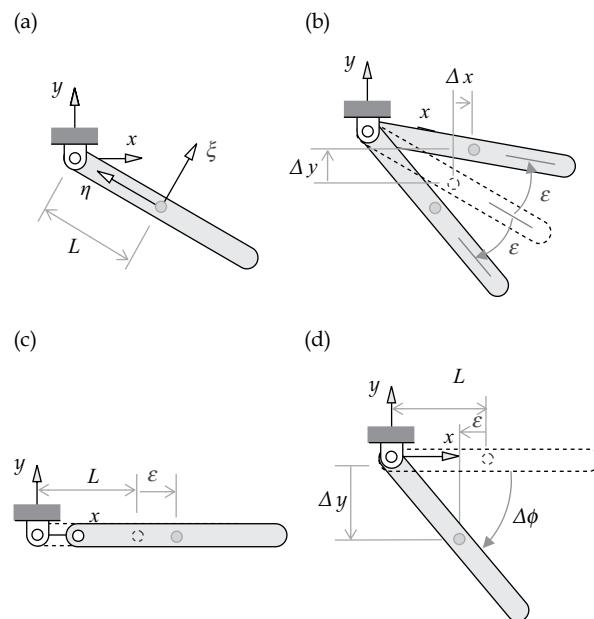
Assume that the values of these coordinates are the result of an integration process and the corresponding numerical errors are  $\Delta x$ ,  $\Delta y$ , and  $\Delta \phi$  that are dependent through the constraints as

$$\Delta x - L \cos \phi \Delta \phi = 0, \Delta y - L \sin \phi \Delta \phi = 0 \quad (\text{b})$$

This is a one DoF system, and therefore, any of the three coordinates can be a candidate to be the independent one.

If we select  $\phi$  as the independent coordinate, an error of  $\Delta \phi = \varepsilon$  will result in errors in the dependent coordinates such as  $\Delta x = \varepsilon L \cos \phi$  and  $\Delta y = \varepsilon L \sin \phi$ . Such a case is illustrated in Figure 13.5b for an exaggerated large positive  $\varepsilon$ . The error,  $\varepsilon$ , could be either a positive or a negative that will not cause any particular difficulties at any orientation of the pendulum; that is, the constraint equations can be solved for the dependent coordinates, and the integration can be continued.

Next let us assume that  $x$  is selected to be the independent coordinate. An integration error  $\Delta x = \varepsilon$  in predicting the value of  $x$  yields errors in the computed values of  $\phi$  and  $y$  as  $\Delta \phi = \varepsilon / (L \cos \phi)$  and  $\Delta y = \varepsilon \sin \phi / \cos \phi$ . As long as the pendulum is not close to a horizontal orientation, we will not have any problem in the process. But when the pendulum is close to be horizontal, small error in predicting  $x$  yields either no solution or a large error in the computed values of  $\phi$  and  $y$  as demonstrated graphically in Figure 13.5c and d. If the pendulum is exactly in the horizontal configuration on the right of the pin joint, the predicted



**FIGURE 13.5**

(a) A single pendulum; (b-d) various scenarios due to integration error.

value of  $x$  should be exactly  $L$ , i.e.,  $\Delta x = \varepsilon = 0$ . But if the predicted value is  $x = L + \varepsilon$ , for  $\varepsilon > 0$ , the constraints cannot be enforced. If  $\varepsilon < 0$ , the solution for  $\phi$  and  $y$  will contain large errors as illustrated.

Similar phenomenon exists if  $y$  is selected as the independent coordinate, and the pendulum approaches the vertical position. Therefore, we can conclude that for this simple system,  $\phi$  is the best choice for the independent coordinate. For more complicated systems with several DoFs, it may not be an easy matter to identify the best set of independent coordinates a priori. Furthermore, as the bodies rotate, it may become necessary to switch from one set of independent coordinates to another. Although there are numerical procedures to identify the best or a reasonable choice of independent coordinates automatically (such as the L-U factorization with pivoting as briefly discussed in Appendix A), the overall process may become computationally too time consuming.

### 13.3.3 Penalty Method

The *penalty method* [5] has similarities to the constraint violation stabilization technique but with a different implementation. The method removes the constraints from the equations of motion and replaces the reaction forces, or the Lagrange multipliers, with resistive penalty forces caused by constraint violations. The method considers our general form of the constrained equations of motion:

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{h} + \mathbf{D}'\lambda$$

The Lagrange multipliers are expressed as a function of constraint violations as

$$\lambda = -\left( {}^{(c)}m\ddot{\Phi} + {}^{(c)}d_c\dot{\Phi} + {}^{(c)}k\Phi \right) \quad (13.37)$$

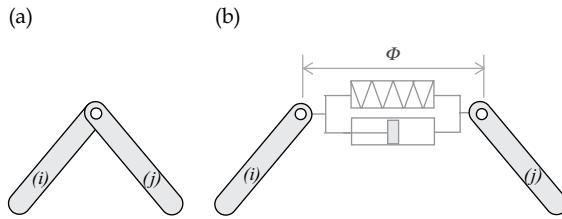
where the negative sign indicates a resistive force (or torque). The violation in the position constraints is assigned a coefficient  ${}^{(c)}k$  as stiffness, and therefore,  ${}^{(c)}k\Phi$  is a force. Similarly, the coefficient for the velocity constraints is a damping coefficient, and for the acceleration constraints, it is a mass. With this representation of Lagrange multipliers, the equations of motion become

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{h} - \mathbf{D}'\left( {}^{(c)}m\ddot{\Phi} + {}^{(c)}d_c\dot{\Phi} + {}^{(c)}k\Phi \right)$$

Expressing the acceleration constraints as  $\ddot{\Phi} = \mathbf{D}\ddot{\mathbf{q}} + \dot{\mathbf{D}}\dot{\mathbf{q}}$ , the equations of motion can be rearranged as

$$\left( \mathbf{M} + {}^{(c)}m\mathbf{D}'\mathbf{D} \right)\ddot{\mathbf{q}} = \mathbf{h} - \mathbf{D}'\left( {}^{(c)}m\dot{\mathbf{D}}\dot{\mathbf{q}} + {}^{(c)}d_c\dot{\Phi} + {}^{(c)}k\Phi \right) \quad (13.38)$$

The penalty forces can be viewed as fictitious force elements replacing the constraints. To visualize this concept, as an example, consider the pin joint shown in Figure 13.6a being replaced by a force element, such as the spring–damper shown in Figure 13.6b, where the element also has an associated mass. To keep the violation as small as possible, the spring should have a very large stiff,  ${}^{(c)}k$ . A stiff spring causes an artificial high frequency in the

**FIGURE 13.6**

(a) A kinematic joint; (b) its representation as a fictitious penalty force element.

response, resulting in very small integration time steps. To lower the frequency, the fictitious mass  ${}^{(c)}m$  must be assigned a large value. We can rewrite Eq. (13.38) as

$$(\mathbf{M} + {}^{(c)}m\mathbf{D}'\mathbf{D})\ddot{\mathbf{q}} = \mathbf{h} - {}^{(c)}m\mathbf{D}'\left(\dot{\mathbf{D}}\dot{\mathbf{q}} + \frac{{}^{(c)}d_c}{{}^{(c)}m}\dot{\Phi} + \frac{{}^{(c)}k}{{}^{(c)}m}\Phi\right) \quad (13.39)$$

According to Eq. (13.10), the corresponding natural frequency is  $\omega = \sqrt{{}^{(c)}k/{}^{(c)}m}$ . If we introduce a damping ratio  $\zeta = {}^{(c)}d_c/(2\sqrt{{}^{(c)}k{}^{(c)}m})$ , as defined in Eq. (13.16), Eq. (13.39) can be expressed as

$$(\mathbf{M} + {}^{(c)}m\mathbf{D}'\mathbf{D})\ddot{\mathbf{q}} = \mathbf{h} - {}^{(c)}m\mathbf{D}'(\dot{\mathbf{D}}\dot{\mathbf{q}} + 2\zeta\omega\dot{\Phi} + \omega^2\Phi) \quad (13.40)$$

Equation (13.40) is the final form of the equations of motion with the penalty method. Selection of values for the parameters  ${}^{(c)}m$ ,  $\zeta$ , and  $\omega$  is totally up to us. We need to select a large value for  ${}^{(c)}m$ . For example, this mass could be  $10^3$  times the largest mass in a multibody system. For the critical damping, it is safe to set  $\zeta = 1$ . For a “reasonable” value of  $\omega$ , we may consider a value in the range of (or larger than) the largest natural frequency of the system. If we have no knowledge of approximate range of a system’s natural frequencies, we can experiment with values that result in a time period of, for example,  $\tau = 0.1\text{--}0.5\text{ s}$ .

We should note two interesting and useful features of this method: (1) Although the method deals with constrained equations of motion, the number of equations in Eq. (13.40) is the same as the number of unknown accelerations—no acceleration constraints are appended to these equations. (2) The method does not care whether we have redundant constraints.

### Example 13.7

Consider the body-coordinate formulation of the equations of motion for the variable-length pendulum from Example 13.5. Revise the program from that example by implementing the penalty method. Assume the following values for the penalty parameters:

$${}^{(c)}m = 1,000 \text{ kg}, \zeta = 1, \omega = 20\pi \text{ rad/s}$$

### Solution

We duplicate the M-files from Example 13.5 and rename them `Example_13_7` and `Ex_13_7`.

#### MATLAB/Chapter 13/Example\_13\_7, Ex\_13\_7

In the M-file `Example_13_7`, we provide additional data for the penalty parameters, include them in a global statement, and revise the function handle for `ode45` (or `RK4`) to `Ex_13_7`.

In the function M-file `Ex_13_7`, we include an additional global statement and add statements for computing the constraints at both the coordinate and velocity levels.

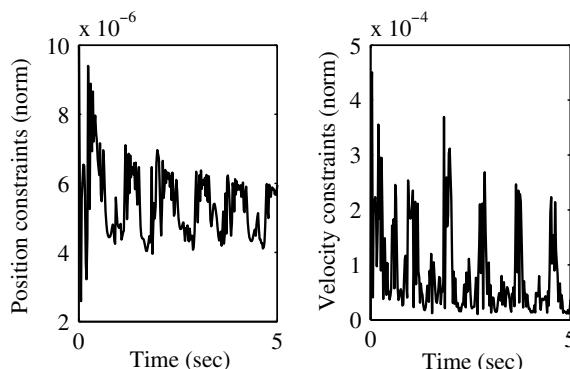
```
...
Phi = [r_O1
       u1r'*u2
       u1r'*d]; % position constraints
Phi_d = D*c_d; % velocity constraints
```

Finally, we replace the statements for determining accelerations with the penalty equations of motion.

```
...
Mc = diag(M) + mc*D'*D;
rhs = f_a - mc*D'*(-gamma + 2*zeta*omega*Phi_d + ...
                      omega^2*Phi);
c_dd = Mc\rhs;
...
```

If we recover the constraint violations, we can plot the norms versus time as shown in Figure 13.7. We note that the violations are very small, and therefore, if we compare the results from this simulation against those from either Example 13.3 or Example 13.5, we find that they are identical.

We can also compute the Lagrange multipliers associated with the four constraints based on Eq. (13.37). We can verify that these results are the same as those obtained from the standard method in Example 13.5 or the constraint violation stabilization method in Example 13.6.



**FIGURE 13.7**

Norm of violations for the position and velocity constraints.

### 13.3.4 Joint-Coordinate Method

The most effective method to eliminate any possibilities of constraint violation is to remove the constraints from the formulation of equations of motion. In Chapter 9, we learned that the joint-coordinate method completely eliminates the kinematic constraints from the equations of motion for open-chain systems, and only constraints associated with the cut joints remain in the formulation for closed-chain systems. If constraints are present in a joint-coordinate formulation, any of the methods from Sections 13.3.1–13.3 can be implemented to control or eliminate the violations.

### 13.3.5 Momentum Method

We have observed that the integration procedure of Section 13.2.2 may cause violation of the coordinate and velocity constraints due to numerical integration error. In this section, we show how to eliminate the constraint violations at the velocity level by using the array of momenta in the integration process.

In Section 11.1.1, we defined the linear and angular momenta for rigid body ( $i$ ) as

$$\mathbf{g}_i = m_i \dot{\mathbf{r}}_i$$

$$h_i = J_i \dot{\phi}_i$$

The combined array of momenta for the body was then defined as

$$\mathbf{m}_i = \begin{Bmatrix} \mathbf{g}_i \\ h_i \end{Bmatrix} = \begin{Bmatrix} m_i \dot{\mathbf{r}}_i \\ J_i \dot{\phi}_i \end{Bmatrix} = \begin{bmatrix} m_i \mathbf{I} & \mathbf{0} \\ \mathbf{0} & J_i \end{bmatrix} \mathbf{c}_i$$

For a multibody system containing  $n_b$  bodies, we define the array of momenta as

$$\mathbf{m} = \begin{Bmatrix} \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_{n_b} \end{Bmatrix} \quad (13.41)$$

For the numerical integration of the equations of motion in the body-coordinate formulation, we revise the integration array  $\mathbf{u}$  of Eq. (13.24) to contain the coordinates and the momenta (instead of velocities) as

$$\mathbf{u} = \begin{Bmatrix} \mathbf{c} \\ \mathbf{m} \end{Bmatrix} \quad (13.42)$$

The numerical integration algorithm predicts the contents of  $\mathbf{u}$  by integrating the array  $\dot{\mathbf{u}}$ . For this purpose, we can compute the array of velocities as

$$\dot{\mathbf{c}} = \mathbf{M}^{-1} \mathbf{m} \quad (13.43)$$

However, since the predicted values of  $\mathbf{m}$  may contain numerical error, and therefore, the constraints  $\dot{\Phi} = \mathbf{D}\dot{\mathbf{c}} = \mathbf{0}$  may be violated, we evaluate the velocities by enforcing the velocity constraints as [6]

$$\left[ \begin{array}{cc} \mathbf{M} & -\mathbf{D}' \\ \mathbf{D} & \mathbf{0} \end{array} \right] \left\{ \begin{array}{c} \dot{\mathbf{c}} \\ \boldsymbol{\sigma} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{m} \\ \mathbf{0} \end{array} \right\} \quad (13.44)$$

This process ensures  $\dot{\Phi} = \mathbf{D}\dot{\mathbf{c}} = \mathbf{0}$ . Now we continue with solving the equations of motion as

$$\left[ \begin{array}{cc} \mathbf{M} & -\mathbf{D}' \\ \mathbf{D} & \mathbf{0} \end{array} \right] \left\{ \begin{array}{c} \ddot{\mathbf{c}} \\ \boldsymbol{\lambda} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{h} \\ \boldsymbol{\gamma} \end{array} \right\} \quad (13.45)$$

We determine the time derivative of the array of momenta as

$$\dot{\mathbf{m}} = \mathbf{M}\dot{\mathbf{c}} \quad (13.46)$$

where the mass matrix  $\mathbf{M}$  is a constant in the body-coordinate formulation. Finally, we construct the integration array  $\dot{\mathbf{u}}$  as

$$\dot{\mathbf{u}} = \left\{ \begin{array}{c} \dot{\mathbf{c}} \\ \dot{\mathbf{m}} \end{array} \right\} \quad (13.47)$$

This array is then returned to the integrator. With this process, we have enforced the constraints at the velocity and acceleration levels. The only possibility of constraint violations remains to be at the coordinate level.

### Example 13.8

Revise the integration process of Example 13.5 (variable-length pendulum, body-coordinate formulation) by integrating the system momenta instead of velocities.

#### Solution

We revise the M-files `Example_13_5` and `Ex_13_5` by implementing the process described in Eqs. (13.42)–(13.47).

#### MATLAB/Chapter 13/Example\_13\_8, Ex\_13\_8

In the M-file `Example_13_8`, we make the following revisions:

```

...
c = [0.4330; -0.2500; pi/3; 0.6928; -0.4000; pi/3];
c_d = zeros(6,1); mom = M.*c_d;
u = [c; mom];
...

```

In the function M-file `Ex_13_8`, we first update the coordinates and then solve for the velocities and accelerations. Finally, we compute the time derivative of the array of momenta, and then return the velocities and the time derivative of the momenta to the integrator.

```

function ud = Ex_13_8(t, u)
...
c = u(1:6); mom = u(7:12); % coordinates and momenta
% Extract coordinates
...
r1_d = c_d(1:2); phi1_d = c_d(3);
r2_d = c_d(4:5); phi2_d = c_d(6);
...

```

```
% Solve for velocities
DMD = [diag(M) -D'
        D zeros(4)];
rhs = [mom; zeros(4,1)];
solution = DMD\rhs; c_d = solution(1:6);
...
mom_d = M.*c_dd;
ud = [c_d; mom_d];
```

This example shows that implementing the array of momenta in the integration process is very simple. Although we have discussed this method for the body-coordinate formulation, we can apply the process to other formulations. For example, in the integration process of the joint-coordinate formulation of closed-chain systems, we can apply a similar process and integrate an array of system momenta. However, in such a process, we must consider that the mass matrix in the joint-coordinate equations of motion (Eq. 9.23) is no longer a constant, and therefore, its time derivative must be included in computation of the time derivative of the array of momenta.

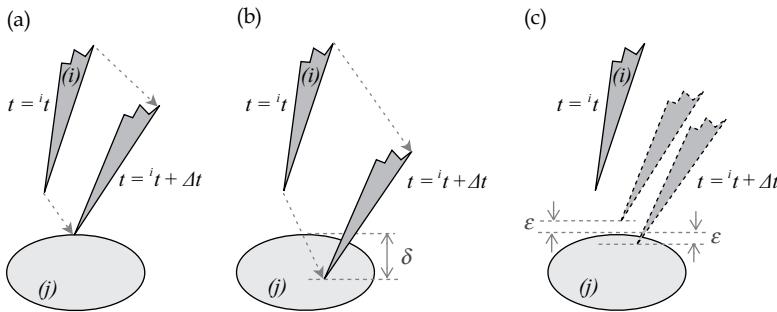
### 13.4 Contact and Impact

The concepts of contact and impact between the bodies of a multibody system have been discussed in Chapter 11, where two methods of analyses have been proposed: piecewise analysis and continuous analysis. When either of these two methods is implemented in a forward dynamic analysis, some important computational issues must be considered. In this section, we discuss two such issues: exact time of contact and discontinuity in the velocities.

In forward dynamic analysis, a numerical integration algorithm predicts the coordinates and velocities at  $t = i^t + \Delta t$  based on the values of the velocities and accelerations at  $t = i^t$ . As two bodies of a multibody system are about to contact each other, the ideal size of the time step  $\Delta t$  would be to bring the bodies in contact with zero penetration, that is,  $\delta = 0$ , as shown in Figure 13.8a. If the integrator picks a time step that is too large, it results in a situation depicted in Figure 13.8b, where the amount of initial penetration at  $t = i^t + \Delta t$  is  $\delta \gg 0$ . Whether we perform a continuous or a piecewise analysis, for achieving better accuracy in the simulated results, the initial contact between the two bodies should result in a very small penetration, or ideally  $\delta = 0$ . Therefore, the predicted results at  $t = i^t + \Delta t$  are not acceptable, and the integrator should be directed to make the time step  $\Delta t$  smaller and to repeat the prediction for  $t = i^t + \Delta t$ . Since computationally it may be too time consuming to achieve a *perfect* zero penetration, we should provide a small parameter  $\epsilon$  as shown in Figure 13.8c, and if  $-\epsilon < \delta < \epsilon$ , we accept the corresponding time step as the initial time of contact.

Determination of the exact time of contact is important in both continuous and piecewise analyses. In the continuous analysis, whether we apply Eq. (11.9) or Eq. (11.10), we need to know the relative velocity  $(^-)\dot{\delta}$  of the contacting points, at the exact initial time of contact; otherwise, we can introduce too much error in determining the contact force.

In the piecewise analysis, based on the velocities before impact and an assumed value for the coefficient of restitution, the velocities after impact are determined using Eq. (11.27)

**FIGURE 13.8**

(a) Two bodies before contact; (b) unacceptable penetration after contact; (c) the acceptable range for contact.

or other similar equations. Due to the discontinuity in the velocities, we must stop the numerical integration of the equations of motion, and then restart it with the new set of velocities. This is an important issue because most numerical integration algorithms, especially those based on polynomial approximation, use the history of the response from previous time steps to predict the response for the next time step. When there is discontinuity in the response—in our case in the velocities—the saved history is no longer useful. Therefore, the integration process should restart, as in  $t = 0$ , when there is no history to rely on for the first few time steps.

To halt an integration process before reaching the end of a specified time span, MATLAB provides a feature called the *event* function. In this function, we can state the required condition for stopping an integration process precisely at the time when the condition is met. MATLAB adjusts the size of the integration time steps to smaller values, as the simulation gets closer to meeting the condition.

To use the event function, we need to provide the following *option* statement:

```
options = odeset('Event', @event_function, ...);
[T, uT] = ode45(..., options);
```

where in the function M-file *event\_function* we state the condition(s). The output arrays *T* and *uT* contain the result of integration up to or just before the time of event, depending on whether we have requested the results to be reported at every integration time step or only at specified reporting time steps.

We may also set the output arguments of the integrator as

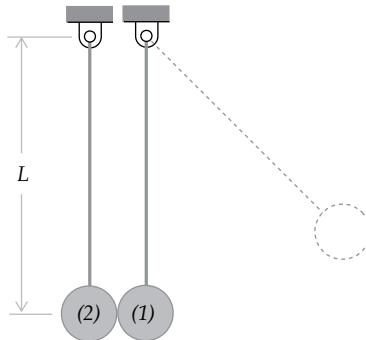
```
[T, uT, Te, uTe, ie] = ode45(..., options);
```

Here, in addition to the standard output arrays *T* and *uT*, we also get the exact time of the event in *Te* and the result of the integration at the time of event in *uTe*. The output *ie* reports the index of the met condition (in the event function, we may state more than one condition). Refer to MATLAB's *help* for more explanation of *odeset* options.

### Example 13.9

Two rigid balls of radius  $R = 0.01$  m hang by rigid slender rods from the ground forming two pendulums, as shown in Figure 13.9. The effective length of each rod is  $L = 0.1$  m. One of the balls is released from an orientation making a  $45^\circ$  angle with the vertical axis. The impact between the balls is considered having a coefficient of restitution  $e = 1$ .

The balls have identical mass and moment of inertia of  $m = 0.1$  kg and  $J = \frac{1}{2}mR^2$ , respectively. Simulate the response of the system for several seconds.

**FIGURE 13.9**

Two pendulums impacting each other.

### *Solution*

We construct the M-files `Example_13_9`, `Ex_13_9_1`, `Ex_13_9_2`, `Ex_13_9_3`, and `Ex_13_9_4` by implementing the event function and Eq. (11.27).

**MATLAB/Chapter 13/Example\_13\_9, Ex\_13\_9\_1, Ex\_13\_9\_2, Ex\_13\_9\_3**

(M-file `Ex_13_9_4` provides an animation of the response)

In the M-file `Example_13_9`, we provide the constant data and the initial conditions. We set the time span without specifying a reporting time step; therefore, the integration results will be reported at every integration time step.

```
% Data
L = 0.1; R = 0.01; R2 = 2*R; e = 1; g = 9.81;
m1 = 0.1; m2 = m1; J1 = 0.5*m1*R^2; J2 = J1;
M_diag = [m1 m1 J1 m2 m2 J2]; M = diag(M_diag);
h_a = [0 -m1*g 0 0 -m2*g 0]';
% Initial coordinates
p1 = pi/3; r1 = [R2; 0] + L*[sin(p1); -cos(p1)];
p2 = 0; r2 = [0; -L];
u = [r1' p1 r2' p2 zeros(1,6)']; % Integration array
Tstart = 0; Tfinal = 2; % Time span
...
```

Next we activate the event function as an option in `odeset`, in addition to tightening the integration error tolerances. We guide the event function to find the condition for terminating the integration in the function `Ex_13_9_2`.

```
...
% Integration options
options = odeset('Event', @Ex_13_9_2, ...
    'RelTol', 1e-6, 'AbsTol', 1e-9);
...
```

In the event function `Ex_13_9_2`, we check whether the distance between the two balls has reached zero. We construct vector  $\vec{d} = \vec{r}_1 - \vec{r}_2$  between the two mass centers, and then compute the gap between the balls as  $2R - d$ . The integration will be halted when the gap reaches zero.

```

function[value, isterminal, direction] = Ex_13_9_2(t, u)
global R2
    value = R2 - norm(u(1:2) - u(4:5));
    isterminal = 1; direction = 1;

```

In the main program, we directed `ode45` to find the equations of motion in M-file `Ex_13_9_1`. During the integration, if the event function detects an impact, it stops the integration and returns the results up to the moment of impact back to the main program. We save the returned results, compute the jump in the velocities, and then restart the integration. Computing the jump in the velocities is done in the function `Ex_13_9_3`, which implements the momentum-impulse process from Eq. (11.27). The results from the piecewise integrations are saved in the arrays `TT` (for time) and `uTT` (for coordinates and velocities).

```

...
TT = [] ; uTT = [] ; % storage arrays for segmented results
Tspan = [Tstart Tfinal];
[T, uT, Te, uTe, ie] = ode45(@Ex_13_9_1, Tspan, u,
options);
TT = [TT; T]; uTT = [uTT; uT];
while ~isempty(Te)
    vp = Ex_13_9_3(uTe(end,:)', M, R, L, e);
    u = [uTe(end,1:6)'; vp];
    Tspan = [Te Tfinal];
    [T, uT, Te, uTe, ie] = ode45(@Ex_13_9_1, Tspan, u,
options);
    TT = [TT; T]; uTT = [uTT; uT];
End

```

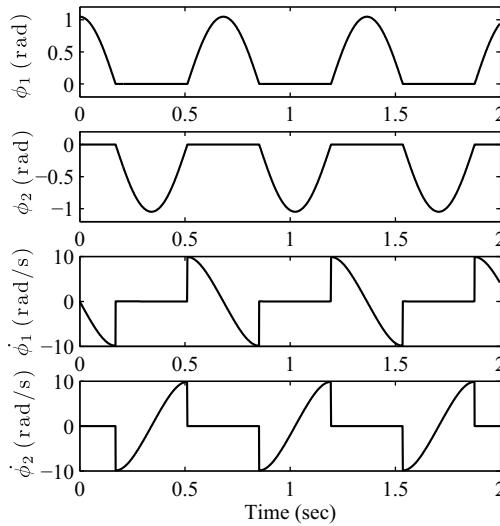
The followings are M-files `Ex_13_9_1` for the equations of motion and `Ex_13_9_3` for the momentum-impulse equations:

```

function ud = Ex_13_9_1(t, u)
global M h_a L
    p1 = u(3); u1 = [cos(p1); sin(p1)]; u1r = s_rot(u1);
    p2 = u(6); u2 = [cos(p2); sin(p2)]; u2r = s_rot(u2);
    D = [eye(2) -L*u1 zeros(2,3)
          zeros(2,3) -eye(2) L*u2]; % Jacobian
    gamma1 = L*u1r*u(9)^2; gamma2 = -L*u2r*u(12)^2;
    gamma = [gamma1; gamma2];
% Equations of motion
    DMD = [M -D'
            D zeros(4)];
    rhs = [h_a; gamma];
    sol = DMD\rhs; % accelerations and multipliers
    ud = [u(7:12); sol(1:6)]; % [velocities and accelerations]

function vp = Ex_13_9_3(u, M, R, L, e)
    vm = u(7:12); % velocities before impact
    p1 = u(3); u1 = [cos(p1); sin(p1)]; u1r = s_rot(u1);
    p2 = u(6); u2 = [cos(p2); sin(p2)]; u2r = s_rot(u2);
    D = [eye(2) -L*u1 zeros(2,3)
          zeros(2,3) -eye(2) L*u2]; % Jacobian of constraints
    d12 = u(1:2) - u(4:5); u12 = d12/norm(d12);
    % Jacobian of contact
    Du12 = u12'*[eye(2) -R*u1r -eye(2) -R*u2r];

```

**FIGURE 13.10**

Angles and angular velocity of the two pendulums versus time.

```

DD = [D; Du12];
DMD = [M -DD'
        DD zeros(5,5)];
rhs = [zeros(10,1); -(e + 1)*Du12*vm];
sol = DMD\rhs;
vp = vm + sol(1:6); % velocities after impact

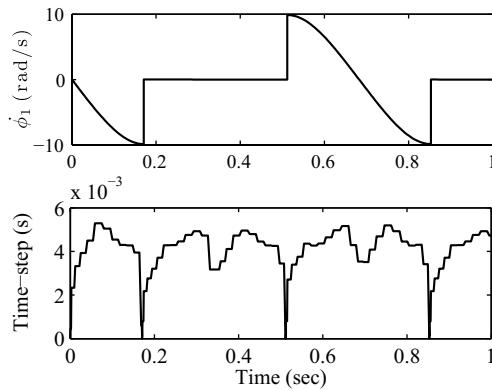
```

Simulating the response for 2.0s provides the results shown in Figure 13.10. The top two plots show the angles, and the bottom two plots show the angular velocity of the two pendulums. We note that when pendulum (1) comes to rest as it hits pendulum (2), pendulum (2) begins to swing in the other direction, and then the process reverses. Executing the program, in addition to the plots, also provides an animation of the pendulums (not shown here).

The *event* function causes an integrator, such as *ode45*, to reduce the size of the time steps just before the event takes place. This is illustrated in Figure 13.11 for the impact between the pendulums of Example 13.9. When the pendulums are not close to contact, the integration time steps are on the order of  $5 \times 10^{-3}$ s, where in the vicinity of an impact the time steps become very small (close to  $5 \times 10^{-6}$ s). This is a very useful feature for modeling impact, whether it is the piecewise or the continuous method.

### 13.5 Adding or Deleting Constraints

In some applications of multibody dynamics, we may need to add or delete a constraint as the system is in motion. Deleting or adding a constraint from or to the equations of motion,



**FIGURE 13.11**  
Reduction of the size of time steps by the integrator in the vicinity of an *event*.

as the equations are being integrated, is not a difficult task. However, when a constraint is added, we must assure that the principle of conservation of momenta is not violated.

As an example for adding a constraint, assume that in a multibody system, bodies (*i*) and (*j*) are connected by a revolute joint, and therefore, there is one relative DoF between those two bodies. Furthermore, let us assume that as the bodies move, at a certain point in time, we need to lock that joint by eliminating that relative DoF. If the joint is locked, the relative velocity and acceleration between bodies (*i*) and (*j*) become zero, and must remain zero as long as the joint is locked.

When we add a constraint, at that instant we must assure that the velocities satisfy not only the original constraints but also the newly added constraint, before we can continue with the integration. Adding a new constraint to a multibody system in motion is equivalent to an impact between bodies, as it has been discussed in Chapter 11.

Let us assume that a multibody system is represented, in the body-coordinate formulation, by the following constraints and equations of motion:

$$\Phi(\mathbf{c}) = \mathbf{0} \quad (13.48)$$

$$\dot{\Phi} = \mathbf{D}\dot{\mathbf{c}} = \mathbf{0} \quad (13.49)$$

$$\ddot{\Phi} = \mathbf{D}\ddot{\mathbf{c}} + \dot{\mathbf{D}}\dot{\mathbf{c}} = \mathbf{0} \quad (13.50)$$

$$\mathbf{M}\ddot{\mathbf{c}} - \mathbf{D}'\boldsymbol{\lambda} = {}^{(a)}\mathbf{h} \quad (13.51)$$

At a given time, let us assume that a new set of constraints (or a single constraint) must be added to the system as

$$\Phi_{new}(\mathbf{c}) = \mathbf{0} \quad (13.52)$$

The first and second time derivatives of this new set of constraints are

$$\dot{\Phi}_{new} = \mathbf{D}_{new}\dot{\mathbf{c}} = \mathbf{0} \quad (13.53)$$

$$\ddot{\Phi}_{new} = \mathbf{D}_{new}\ddot{\mathbf{c}} + \dot{\mathbf{D}}_{new}\dot{\mathbf{c}} = \mathbf{0} \quad (13.54)$$

The additional constraints result in additional reaction forces (or torques) that must be included in Eq. (13.51) as

$$\mathbf{M}\ddot{\mathbf{c}} - \mathbf{D}'\lambda - \mathbf{D}'_{new}\lambda_{new} = {}^{(a)}\mathbf{h} \quad (13.55)$$

The term  $\mathbf{D}'_{new}\lambda_{new}$  is equivalent to impulsive forces or torques similar to that of impact between two bodies. If we assume that the addition of the new constraints occurs during a very short period,  $(-)t$  to  $(+)t$ , we can integrate Eq. (13.55) for this period of time:

$$\int_{(-)t}^{(+)} \mathbf{M}\ddot{\mathbf{c}} dt - \int_{(-)t}^{(+)} \mathbf{D}'\lambda dt - \int_{(-)t}^{(+)} \mathbf{D}'_{new}\lambda_{new} dt = \int_{(-)t}^{(+)} {}^{(a)}\mathbf{h} dt$$

Noting that during this short period,  $\mathbf{M}$ ,  $\mathbf{D}'$ ,  ${}^{(a)}\mathbf{h}$ , and  $\mathbf{D}'_{new}$  do not vary, the integral equation can be simplified to

$$\mathbf{M} \int_{(-)t}^{(+)} \ddot{\mathbf{c}} dt - \mathbf{D}' \int_{(-)t}^{(+)} \lambda dt - \mathbf{D}'_{new} \int_{(-)t}^{(+)} \lambda_{new} dt = \mathbf{0}$$

Denoting  $\boldsymbol{\sigma} = \int_{(-)t}^{(+)} \lambda dt$  and  $\boldsymbol{\sigma}_{new} = \int_{(-)t}^{(+)} \lambda_{new} dt$ , we get

$$\mathbf{M} \left( {}^{(+)}\dot{\mathbf{c}} - {}^{(-)}\dot{\mathbf{c}} \right) - \mathbf{D}' \left( {}^{(+)}\boldsymbol{\sigma} - {}^{(-)}\boldsymbol{\sigma} \right) - \mathbf{D}'_{i,j} \left( {}^{(+)}\boldsymbol{\sigma}_{new} - {}^{(-)}\boldsymbol{\sigma}_{new} \right) = \mathbf{0}$$

Furthermore, denoting  $\Delta\dot{\mathbf{c}} = {}^{(+)}\dot{\mathbf{c}} - {}^{(-)}\dot{\mathbf{c}}$ ,  $\Delta\boldsymbol{\sigma} = {}^{(+)}\boldsymbol{\sigma} - {}^{(-)}\boldsymbol{\sigma}$ , and  $\Delta\boldsymbol{\sigma}_{new} = {}^{(+)}\boldsymbol{\sigma}_{new} - {}^{(-)}\boldsymbol{\sigma}_{new}$ , we have

$$\mathbf{M}\Delta\dot{\mathbf{c}} - \mathbf{D}'\Delta\boldsymbol{\sigma} - \mathbf{D}'_{new}\Delta\boldsymbol{\sigma}_{new} = \mathbf{0} \quad (13.56)$$

The original velocity constraints of Eq. (13.49) at  $(-)t$  and  $(+)t$  yield

$$\mathbf{D}\Delta\dot{\mathbf{c}} = \mathbf{0} \quad (13.57)$$

The added velocity constraints of Eq. (13.53) can be expressed as

$$\mathbf{D}_{new}\Delta\dot{\mathbf{c}} = -\mathbf{D}_{new}{}^{(-)}\dot{\mathbf{c}} \quad (13.58)$$

Appending Eqs. (13.57) and (13.58) to Eq. (13.56), and arranging the terms in matrix form yield

$$\begin{bmatrix} \mathbf{M} & -\mathbf{D}' & -\mathbf{D}'_{new} \\ \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}_{new} & \mathbf{0} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\dot{\mathbf{c}} \\ \Delta\boldsymbol{\sigma} \\ \Delta\boldsymbol{\sigma}_{new} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ -\mathbf{D}_{new}{}^{(-)}\dot{\mathbf{c}} \end{Bmatrix} \quad (13.59)$$

Knowing the coordinates and velocities at  $(-)t$ , Eq. (13.59) can be solved to determine the jump in the velocities, and then the velocities  ${}^{(+)}\dot{\mathbf{c}} = {}^{(-)}\dot{\mathbf{c}} + \Delta\dot{\mathbf{c}}$ . At this point, we must restart the integration process with a new set of initial conditions.

An example for deleting a constraint is to unlock a locked revolute joint and allow the two connected bodies to rotate relative to each other. Another example could be to remove a kinematic joint completely due to structural failure. If the failed joint is revolute, we remove two constraint equations. Deleting a constraint does not require any adjustments to the velocities. If the velocities satisfy all of the original constraints, then the velocities definitely satisfy the remaining constraints after the elimination of one or more constraints.

---

### 13.6 Combined Analyses

In the kinematic and inverse dynamic analyses discussed in Chapter 12, we only dealt with solving algebraic equations. For position constraints, we solved nonlinear algebraic equations, and for velocity and acceleration constraints, and for force equilibrium equations, we solved linear algebraic equations. The solution of the algebraic equations provided the coordinates, velocities, accelerations, reaction forces, and an unknown applied force or torque at incremental time steps. In this section, we discuss how to perform kinematic and inverse dynamic analyses by integration as presented for forward dynamic analysis in Algorithm C. In other words, we make kinematic and inverse dynamic analyses *to appear* as forward dynamics. This is the process implemented in the program DAP \_ BC of Chapter 8.

In a forward dynamic analysis, whether the equations of motion are unconstrained or constrained, as stated in Eqs. (13.2) and (13.24), an integration array and its time derivative are defined as

$$\mathbf{u} = \begin{Bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{Bmatrix}, \quad \dot{\mathbf{u}} = \begin{Bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{Bmatrix} \quad (13.60)$$

Assuming that there are  $n_v$  coordinates and  $n_c$  independent constraints, where  $n_{dof} = n_v - n_c$ , the  $n_v$  equations of motion and  $n_c$  acceleration constraints, based on Eq. (13.23), are

$$\begin{bmatrix} \mathbf{M} & -\mathbf{D}' \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{h} \\ \boldsymbol{\gamma} \end{Bmatrix} \quad (13.61)$$

Let us assume that we include  $n_d$  driver constraints where their second time derivatives are expressed as

$${}^{(dr)}\mathbf{D}\ddot{\mathbf{q}} = -{}^{(dr)}\dot{\mathbf{D}}\dot{\mathbf{q}} + \ddot{\mathbf{f}}(t) \quad (13.62)$$

With the inclusion of these driver constraints, Eq. (13.61) becomes

$$\begin{bmatrix} \mathbf{M} & -\mathbf{D}' & -{}^{(dr)}\mathbf{D}' \\ \mathbf{D} & \mathbf{0} & \mathbf{0} \\ {}^{(dr)}\mathbf{D} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \\ {}^{(dr)}\boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{h} \\ \boldsymbol{\gamma} \\ -{}^{(dr)}\dot{\mathbf{D}}\dot{\mathbf{q}} + \ddot{\mathbf{f}}(t) \end{Bmatrix} \quad (13.63)$$

Depending on the number of driver constraints, there are three possible cases:

- **Case 1,  $n_d < n_{dof}$ :** This is a *forward dynamic* analysis since  $n_c + n_d < n_v$ —some of the DoFs are not controlled by the driver constraints. Although driver constraints are not specifically considered in the forward dynamic discussion of Section 13.2, we can include them in a formulation like any other constraints.
- **Case 2,  $n_d = n_{dof}$ :** This is a *kinematic/inverse dynamic* analysis since  $n_c + n_d = n_v$ —all DoFs are controlled by the driver constraints. Equation (13.63) can be split into two sets as

$$\begin{bmatrix} \mathbf{D} \\ {}^{(dr)}\mathbf{D} \end{bmatrix} \ddot{\mathbf{q}} = \left\{ \begin{array}{l} \gamma \\ -{}^{(dr)}\dot{\mathbf{D}}\dot{\mathbf{q}} + \ddot{\mathbf{f}}(t) \end{array} \right\} \quad (13.64)$$

$$\begin{bmatrix} \mathbf{D}' & {}^{(dr)}\mathbf{D}' \end{bmatrix} \left\{ \begin{array}{l} \lambda \\ {}^{(dr)}\lambda \end{array} \right\} = \mathbf{M}\ddot{\mathbf{q}} - \mathbf{h} \quad (13.65)$$

The coefficient matrix in Eq. (13.64) is square. With known arrays  $\mathbf{q}$  and  $\dot{\mathbf{q}}$ , the equation is solved for  $\ddot{\mathbf{q}}$ , and then Eq. (13.65) is solved for the Lagrange multipliers. These are exactly the equations we considered in Chapter 12 for kinematics and inverse dynamics. The main difference between this process (solution by integration) and the classical solution of algebraic equations is that, at every time step, the contents of  $\mathbf{q}$  and  $\dot{\mathbf{q}}$  come from the integration array and not from the driver position and velocity constraints. Therefore, not having to solve nonlinear algebraic constraints by iterative methods may be considered as an advantage of the solution by integration. However, the disadvantage is the possibility of constraint violation due to the integration error. We should note that in the integration method, we must have initial conditions for the coordinates and velocities that satisfy the constraints.

- **Case 3,  $n_d > n_{dof}$ :** In this case, the system is a structure (static problem) or there are redundant constraints.

## 13.7 Problems

The following problems are simple systems that we can consider for forward dynamic analysis. In these exercises we experiment with various numerical integration issues such as the step size, constraint violation, etc. We will use MATLAB's `ode45` as a benchmark for comparison. It is recommended to tighten the error tolerances in `ode45` by setting the options as follows:

```
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-9)
```

```
[T, uT] = ode45(@Problem_13_x, Tspan, u, options)
```

- 13.1 Two particles that can only move along the  $x$ -axis are connected by a spring.  
Assume the following constants and initial conditions:

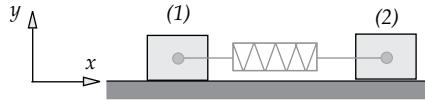
$$m_1 = 2.0, m_2 = 3.0 \text{ kg}, k = 50 \text{ N/m}, {}^0L = 0.3 \text{ m}$$

$${}^0x_1 = 0.2, {}^0x_2 = 0.6 \text{ m}, {}^0\dot{x}_1 = -0.1, {}^0\dot{x}_2 = 0.1 \text{ m/s}$$

Construct the equations of motion for this system with two coordinates and no constraints. Solve the equations for 5.0s using the following integrators and time steps:

Integrator	$\Delta t$
ode45	0.02
RK4	0.02
RK4	0.10
RK4	0.20

Plot the coordinate of one of the particles versus time in every simulation. What can you conclude from the results?



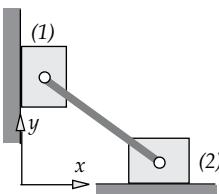
- 13.2 Two particles are connected to each other by a massless rod of length 0.5 m. One particle slides along the  $y$ -axis, and the other slides along the  $x$ -axis. There are no external forces acting on the particles. Assume the following constants and initial conditions:

$$m_1 = 2.0, m_2 = 3.0 \text{ kg}, {}^0y_1 = 0.3, {}^0x_2 = 0.4 \text{ m}, {}^0\dot{y}_1 = -2.0, {}^0\dot{x}_2 = 1.5 \text{ m/s}$$

Construct the equations of motion for this system with two coordinates and one constraint. Solve the equations for 5.0s using the following integrators and time steps. In the last simulation, experiment with the constraint violation stabilization method of Section 13.3.1.

Integrator	$\Delta t$	$\alpha, \beta$
ode45	0.02	
RK4	0.02	
RK4	0.10	
RK4	0.14	
RK4	0.14	$\alpha = \beta = 2.0$

Plot one of the coordinates versus time in every simulation. What can you conclude from the results?



- 13.3 Two particles are connected to each other by a massless rod of length 0.25 m. Particle  $B$  is constrained to slide along a straight line as shown. It is assumed that the line intersects the  $x$ -axis at  $x = 0.2$  m, and a unit vector normal to the line is described as  $\mathbf{u} = \{-0.6 \quad 0.8\}'$ . A spring is connected between particle  $A$  and the ground as shown. Assume the following constants and initial conditions (with zero initial velocities):

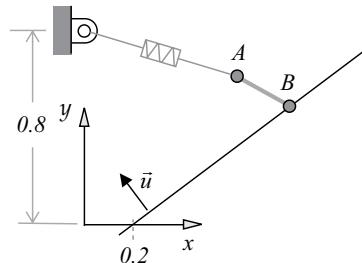
$$m^A = 1.0, m^B = 2.0 \text{ kg}, k = 100 \text{ N/m}, {}^0L = 0.3 \text{ m}$$

$${}^0x^A = 0.57, {}^0y^A = 0.56, {}^0x^B = 0.80, {}^0y^B = 0.45 \text{ m}$$

Construct the equations of motion for this system with four coordinates and two constraints. Solve the equations for 5 s using the following integrators and time steps. In the last simulation, apply the constraint violation stabilization method.

Integrator	$\Delta t$	$\alpha, \beta$
ode45	0.02	
RK4	0.02	
RK4	0.10	
RK4	0.15	
RK4	0.15	$\alpha = \beta = 2.0$

Plot the paths for the two particles for every simulation. What do you conclude from the results?



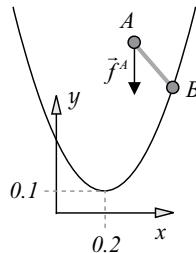
- 13.4 Two particles are connected to each other by a massless rod of length 0.25 m. Particle  $B$  is constrained to slide along a curve described by  $y = 0.1 + 5(x - 0.2)^2$ . A constant force of 1.0 N acts on particle  $A$  in the negative  $y$ -direction. Assume the following constants and initial conditions (with zero initial velocities):

$$m^A = 1.0, m^B = 2.0 \text{ kg}, {}^0x^A = 0.40, {}^0y^A = 0.775, {}^0x^B = 0.50, {}^0y^B = 0.55 \text{ m}$$

Construct the equations of motion for this system with four coordinates and two constraints. Solve the equations for 5.0 s using the following integrators and time steps. In the last two simulations, apply the constraint violation stabilization method.

Integrator	$\Delta t$	$\alpha, \beta$
ode45	0.02	
RK4	0.02	
RK4	0.10	
RK4	0.15	
RK4	0.15	$\alpha = \beta = 2.0$
RK4	0.15	$\alpha = \beta = 5.0$

Plot the paths for the two particles in every simulation. What can you conclude from the results?

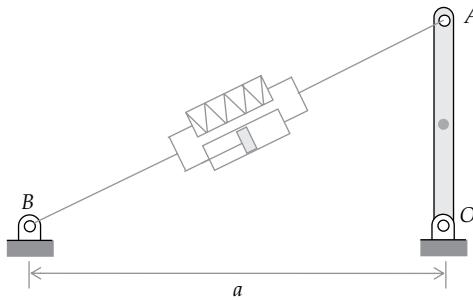


- 13.5 The rod shown is 0.3 m long, and it is pinned to the ground at one end. The rod has a mass  $m = 4.0 \text{ kg}$  and a moment of inertia  $J = 0.03 \text{ N m}^2$ . A spring-damper connects the other end of the rod to the ground as shown, where  $a = 0.6 \text{ m}$ . The characteristics of this force element are  $k = 100 \text{ N/m}$ ,  ${}^0L = 0.8 \text{ m}$ , and  $d_c = 20 \text{ s/m}$ . In the shown configuration, the rod has an angular velocity of  $1.0 \text{ rad/s}$  clockwise.

Construct the equations of motion for this system with the body-coordinate formulation. Solve the equations for 5.0 s using the following integrators and time steps. In the last two simulations, apply the constraint violation stabilization method.

Integrator	$\Delta t$	$\alpha, \beta$
ode45	0.02	
RK4	0.02	
RK4	0.10	
RK4	0.20	
RK4	0.20	$\alpha = \beta = 2.0$
ode45	0.02	$\alpha = \beta = 2.0$

Plot the angular orientation of the rod versus time in every simulation. What can you conclude from the results?

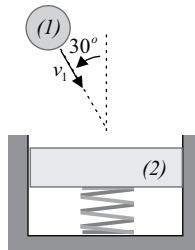


- 13.6 Construct the equations of motion for the system in Problem 13.5 with the joint-coordinate formulation. To decide what should be the largest time step for an integrator such as RK4, we need to know an approximate value for the natural frequency of the system. For this purpose, solve the equations for 5.0 s using `ode45` and plot the angular orientation of the rod versus time. From the plot, determine an approximate value for the natural frequency of the system and its corresponding time period. Refer to Eq. (13.12) to determine a reasonable largest value for the time step. Experiment with RK4 and try several simulations using  $\Delta t$ ,  $2\Delta t$ , and  $3\Delta t$ .
- 13.7 Refer to the equations of motion for the example in Problem 13.2. Solve the equations of motion for 5.0 s with the penalty method using parameters  $(^c)m = 1,000$ ,  $\zeta = 1.0$ , and  $\omega = 20\pi$ . Use `ode45` and compare the results against the standard method of solution as in Problem 13.2. Repeat the solution by the penalty method with  $(^c)m = 100$ ,  $(^c)m = 10$ , and  $(^c)m = 1$ . What can you conclude from the results?
- 13.8 Repeat Problem 13.7 for the system in
- Problem 13.3
  - Problem 13.4
  - Problem 13.5
- 13.9 Solve the equations of motion for the following systems using the momentum method of integration described in Section 13.3.5:
- Problem 13.2
  - Problem 13.3
  - Problem 13.4
  - Problem 13.5

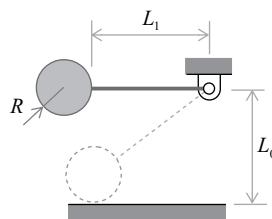
In the following problems that involve contact and impact between bodies, it is important to know the exact time of contact. For this purpose, use the event function of MATLAB. In a continuous analysis, we may specify two events for a contact: one for the start and the other for the end of penetration.

- 13.10 A sphere, body (1), impacts body (2) with a velocity  $(^1)v_1 = 10 \text{ m/s}$  in an angle as shown. Body (2) can only translate vertically. Develop a program to perform a forward dynamic analysis and apply the piecewise method to model

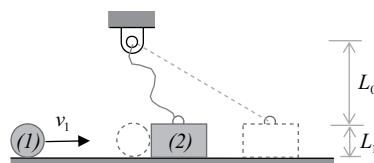
the impact. Consider the following data:  $m_1 = 1\text{ kg}$ ,  $J_1 = 0.004\text{ kg m}^2$ ,  $m_1 = 2\text{ kg}$ ,  $k = 20\text{ N/m}$ , and  $e = 0.6$ . The spring is initially in equilibrium, and gravity is not a factor in this problem.



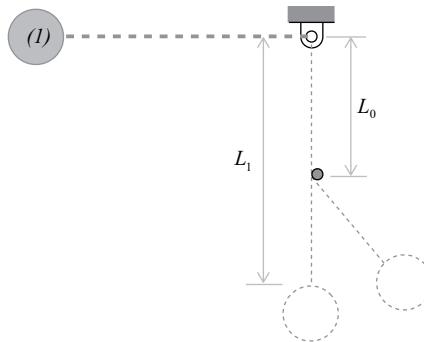
- 13.11 A sphere is attached to a massless rod forming a pendulum. The pendulum is released from the horizontal configuration as shown. The pendulum moves downward under the force of gravity before impacting the ground. Develop a program to perform a forward dynamic analysis and apply the piecewise method to model the impact. Consider the following parameters:  $L_0 = 0.5\text{ m}$ ,  $L_1 = 1\text{ m}$ ,  $R = 0.1\text{ m}$ ,  $m = 1\text{ kg}$ ,  $J = 0.004\text{ kg m}^2$ , and  $e = 0.8$ .



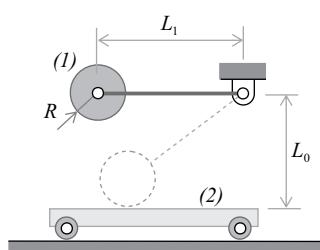
- 13.12 Repeat Problem 13.11 with the continuous analysis method. Apply any of the contact models from Section 11.2 that you find appropriate.
- 13.13 Replace the rod in the pendulum of Problem 13.11 with a massless string. Perform a forward dynamic analysis with the piecewise method for a long enough period in order for the sphere to impact the ground several times.
- 13.14 Repeat Problem 13.13 with the continuous analysis method. Apply any of the contact models from Section 11.2 that you find appropriate.
- 13.15 A sphere moves with a constant speed toward a block that can slide without friction. The block is attached to a massless string of length  $L_2 = 1\text{ m}$ . Develop a program to perform a forward dynamic analysis and apply the piecewise method to model: (a) the impact between the sphere and the block, and (b) when the block reaches the point where the string is taut (adding a constraint). Consider the following parameters:  $L_0 = 0.4\text{ m}$ ,  $L_1 = 0.1\text{ m}$ ,  $m_1 = 1\text{ kg}$ ,  $J_1 = 0.001\text{ kg m}^2$ ,  $m_2 = 2\text{ kg}$ ,  $J_2 = 0.01\text{ kg m}^2$ , and  $e = 0.9$ .



- 13.16 Simulate the response of the impacting pendulums in Example 13.9 with the continuous analysis method (instead of the piecewise method). Use the formula from Eq. (11.33) to compute the contact force, with  $n = 3/2$  and  $k = 10^9 \text{ N/m}^{1.5}$ .
- 13.17 A sphere and a massless string form a pendulum that is released from the horizontal orientation as shown. The pendulum falls under the force of gravity. When the pendulum reached the vertical orientation, the string comes in contact with a pointed obstacle attached to the wall, but it continues to swing as a pendulum with a shorter string. Perform a piecewise forward dynamic simulation of this system for several seconds. Consider the following parameters:  $L_0 = 0.5 \text{ m}$ ,  $L_1 = 1 \text{ m}$ ,  $R = 0.1 \text{ m}$ ,  $m = 1 \text{ kg}$ , and  $J = 0.004 \text{ kg m}^2$ . Hint: The process requires switching between constraints.



- 13.18 Repeat the simulation of Problem 13.17, but this time assume that the string breaks as it hits the pointed obstacle.
- 13.19 The pendulum shown consists of a rotating disc and a massless rod. As the disc rotates about its axis with an angular velocity of  $2\pi \text{ rad/s}$ , the pendulum is released from the horizontal orientation. As the pendulum drops under the force of gravity, it hits body (2) that can slide horizontally with respect to the ground without friction. Due to the roughness of the contacting surfaces, body (2) begins to slide immediately after impact. Perform a piecewise forward dynamic analysis of this system for several seconds. Consider the following data:  $L_0 = 0.5 \text{ m}$ ,  $L_1 = 1 \text{ m}$ ,  $R = 0.2 \text{ m}$ ,  $m_1 = 1 \text{ kg}$ ,  $J = 0.004 \text{ kg m}^2$ ,  $m_2 = 2 \text{ kg}$ ,  $\mu^s = 0.4$ , and  $\mu^d = 0.3$ .



## References

1. Meirovitch, L., *Elements of Vibration Analysis*, 2nd ed. New York: McGraw-Hill (1986).
2. Inman, D.J., *Vibration: With Control, Measurement, and Stability*. Englewood Cliffs, NJ: Prentice Hall (1989).
3. Baumgarte, J., Stabilization of Constraints and Integral of Motion, *Computer Methods Appl. Mech. Eng.*, 1, 1–16 (1972).
4. Wehage, R.A., and Haug, E.J., Generalized Coordinate Partitioning of Dimension Reduction in Analysis of Constrained Dynamic Systems, *ASME J. Mech. Design*, 104, 247–255 (1982).
5. García de Jalón, J., and Bayo, E., *Kinematic and Dynamic Simulations of Multibody Systems*. New York, Springer (1994).
6. Baumgarte, J., Stabilisierung Von Bindungen im Lagraneshen, *ZAMM*, 58, T360–361 (1978).

# 14

## Complementary Analyses

### 14.1 Static Analysis

A mechanical system becomes a structure (a nonmoving system) when the number of degrees of freedom (DoF) is zero. In the formulation of the constraint equations for such a system, the number of independent constraints becomes equal to the number of defined coordinates, i.e.,  $n_c = n_v$ . As an example, consider the structure shown in Figure 14.1. Assume that this system is modeled with the body-coordinate formulation. There are eight links and 12 pin joints (when  $n$  links are pinned together at one point, we must count/model that as  $n - 1$  joints),  $n_v = 3 \times 8 = 24$  coordinates and  $n_c = 2 \times 12 = 24$  constraints.

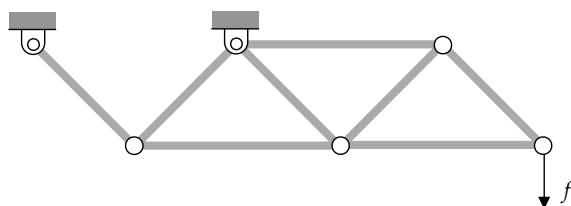
Assume that the constraint equations for a structure are expressed as  $n_c$  equations:

$$\Phi(\mathbf{q}) = \mathbf{0} \quad (14.1)$$

where  $\mathbf{q}$  contains  $n_v = n_c$  coordinates. Since there are as many coordinates as constraints, the equations can be solved for the coordinates. The equations of motion for any constrained mechanical system are expressed as  $\mathbf{M}\ddot{\mathbf{q}} - \mathbf{D}'\boldsymbol{\lambda} = {}^{(a)}\mathbf{h}$ . For a structure  $\dot{\mathbf{q}} = \ddot{\mathbf{q}} = \mathbf{0}$ , the equations of motion are simplified to

$$\mathbf{D}'\boldsymbol{\lambda} = -{}^{(a)}\mathbf{h} \quad (14.2)$$

where  ${}^{(a)}\mathbf{h}$  contains the known applied loads. The Jacobian in this equation is a square matrix, and hence, Eq. (14.2) can be solved for the Lagrange multipliers yielding the reaction forces at each joint.



**FIGURE 14.1**  
A planar structure subject to external forces.

**Example 14.1**

Consider the structure shown in Figure 14.2. The constant lengths are given as  $L_1 = 5.0$  m,  $L_2 = 4.0$  m, and  $L_0 = 3.0$  m. The link masses are  $m_1 = 2.5$  kg and  $m_2 = 2.0$  kg. An external load of  $f = 50$  N acts at point A in the positive  $x$ -direction on body (2), and the gravity acts in the negative  $y$ -direction. Formulate the static equations for this structure with the body coordinates.

**Solution**

Due to the simple structure and geometry, we can determine the following algebraic vectors from the figure:

$$\mathbf{s}_1^O = \begin{Bmatrix} -1.5 \\ -2.0 \end{Bmatrix} \text{ m}, \mathbf{s}_1^A = \begin{Bmatrix} 1.5 \\ 2.0 \end{Bmatrix} \text{ m}, \mathbf{s}_2^Q = \begin{Bmatrix} 0 \\ -2.0 \end{Bmatrix} \text{ m}, \mathbf{s}_2^A = \begin{Bmatrix} 0 \\ 2.0 \end{Bmatrix} \text{ m}$$

The forces that act on the system are

$$\mathbf{w}_1 = m_1 g \mathbf{u}_{(y)}, \mathbf{w}_2 = m_2 g \mathbf{u}_{(y)}, \mathbf{f}_2^A = \begin{Bmatrix} 50 \\ 0 \end{Bmatrix} \text{ N}$$

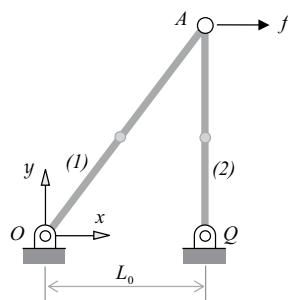
where  $\mathbf{u}_{(y)} = \{0 \ -1\}'$ .

The Jacobian matrix and the array of applied forces are constructed as

$$\mathbf{D} = \left[ \begin{array}{c|cc} \mathbf{I} & \bar{\mathbf{s}}_1^O & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} & \bar{\mathbf{s}}_2^Q \\ \hline \mathbf{I} & \bar{\mathbf{s}}_1^A & -\mathbf{I} & -\bar{\mathbf{s}}_2^A \end{array} \right], \quad {}^{(a)}\mathbf{h} = \left\{ \begin{array}{l} \mathbf{w}_1 \\ 0 \\ \mathbf{w}_2 + \mathbf{f}_2^A \\ \bar{\mathbf{s}}_2^A \mathbf{f}_2^A \end{array} \right\}$$

Substituting numerical values, the static equations become

$$\left[ \begin{array}{c|cc|cc} 1 & 0 & & 1 & 0 \\ 0 & 1 & & 0 & 1 \\ \hline 2.0 & -1.5 & & -2.0 & 1.5 \\ & & 1 & 0 & -1 \\ & & 0 & 1 & 0 \\ & & 2.0 & 0.0 & 2.0 \\ \hline \end{array} \right] \left\{ \begin{array}{l} \lambda_x^O \\ \lambda_y^O \\ \lambda_x^Q \\ \lambda_y^Q \\ \lambda_x^A \\ \lambda_y^A \end{array} \right\} = \left\{ \begin{array}{l} 0 \\ -24.5 \\ 0 \\ 50.0 \\ -19.6 \\ -100.0 \end{array} \right\}$$

**FIGURE 14.2**

A force acting on a structure.

The program DAP \_ BC in Chapter 8 can be used to model and analyze structures that are represented in the body-coordinate formulation as multibody systems with zero DoFs.

## 14.2 Static Equilibrium

A multibody system is said to be in a state of static equilibrium when all the accelerations are zeros; however, the velocities could be either zeros or nonzeros but constants. In such a state, the applied and reaction forces are in balance. As an example, consider the single pendulum shown in Figure 14.3, where the gravity is the only applied force. In orientation (a), the pendulum is not in a static equilibrium state. In orientation (b), the pendulum is in a *stable* static equilibrium state, and in orientation (c), the pendulum is also in an equilibrium state but *unstable*. In the stable state, the *potential energy* of a system is at its minimum, whereas in the unstable state, the potential energy is at its maximum. There are a variety of methods to determine the state of static equilibrium of a system.

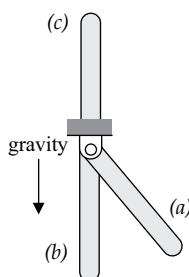
One method for determining the state of static equilibrium is to construct a *potential energy function* for the system and minimize it. This approach is outside the scope of the methods that have been discussed in this textbook.

Another method is to set all of the velocities and accelerations in the equations of motion to zero. If the equations of motion are unconstrained and in the form  $\mathbf{M}\ddot{\mathbf{q}} = {}^{(a)}\mathbf{h}$ , then  $\dot{\mathbf{q}} = \ddot{\mathbf{q}} = \mathbf{0}$  yields

$${}^{(a)}\mathbf{h} = {}^{(a)}\mathbf{h}(\mathbf{q}) = \mathbf{0} \quad (14.3)$$

Treating this equation as a set of  $n_v$  nonlinear algebraic equations in  $n_v$  unknown coordinates  $\mathbf{q}$  provides the solution at the equilibrium state.

If the equations of motion are constrained and in the form  $\mathbf{M}\ddot{\mathbf{q}} - \mathbf{D}'\boldsymbol{\lambda} = {}^{(a)}\mathbf{h}$ , then for  $\dot{\mathbf{q}} = \ddot{\mathbf{q}} = \mathbf{0}$ , we have  $n_v + n_c$  nonlinear algebraic equations, including the position constraints:



**FIGURE 14.3**

A single pendulum in (a) nonequilibrium, (b) stable equilibrium, and (c) unstable static equilibrium states.

$$\begin{aligned} \mathbf{D}'\lambda + {}^{(a)}\mathbf{h} &= \mathbf{0} \\ \Phi(\mathbf{q}) &= \mathbf{0} \end{aligned} \quad (14.4)$$

where there are  $n_v + n_c$  unknowns,  $\mathbf{q}$  and  $\lambda$ .

Solving Eq. (14.3) or (14.4) with a method such as Newton–Raphson requires the partial derivative of the equations with respect to the unknowns. Since computing these derivatives is not a simple task, although possible, we consider a third method for finding a stable static equilibrium state. However, we can solve Eq. (14.3) or Eq. (14.4) with MATLAB® function `fsoolve`, which does not need the partial derivatives to be provided explicitly.

The third method is called the *fictitious damping* method. This method is based on the fact that a mechanical system with no damping elements can oscillate about its stable static equilibrium state. If dampers are added to a system, as time progresses, the kinetic energy of the system dissipates and the potential (including strain) energy reaches its minimum. If a system does not have adequate damping, we can add *fictitious* damping in the equations of motion. Adding fictitious damping will not affect the static equilibrium state of a system.

The fictitious damping term does not have to represent any actual damping elements. The fictitious damping can be added to the array of the applied forces in analytical form, whether the equations of motion are constrained or unconstrained:

$$\mathbf{M}\ddot{\mathbf{q}} - \mathbf{D}'\lambda = {}^{(a)}\mathbf{h} - d_c\dot{\mathbf{q}} \quad (14.5)$$

where  $d_c$  is a single damping coefficient. The value of this coefficient does not change the static equilibrium state, but it influences the speed of reaching that state.

### Example 14.2

Consider the variable-length pendulum and the simulation program from Example 13.3. Include fictitious damping in the system and find the static equilibrium of both bodies.

#### *Solution*

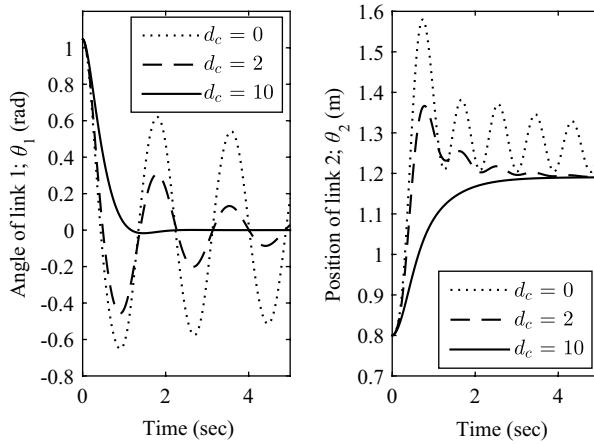
We perform three simulations with fictitious damping coefficients of 0, 2.0, and 10.0 N s/m. We copy programs `Example_13_3` and `Ex_13_3`, and rename them for this example.

#### MATLAB/Chapter 14/Example\_14\_2, Ex\_14\_2

There is no need to make any changes in the main program. In the program `Ex_14_2_eqm`, we revise the statement that constructs the array of applied forces in the joint-coordinate formulation.

```
% Add fictitious damping
h_joint = B'* (f_a - diag(M)*Bdtd) - 10*theta_d;
```

We repeat the simulations for three coefficients. The results for  $\theta_1$  and  $\theta_2$  versus time are shown in Figure 14.4. We note that with a larger damping coefficient, we can reach to the equilibrium state more quickly without changing the outcome.

**FIGURE 14.4**

The joint coordinates versus time for three different values of the fictitious damping coefficient.

### 14.3 Initial Condition Correction

Forward dynamic analysis of constrained equations of motion requires a set of initial conditions on coordinates and velocities that satisfy the corresponding constraints. Failure in providing such initial conditions, most likely, results in erroneous responses. For highly simple multibody systems, we might be able to come up with correct initial conditions for both the coordinates and the velocities by inspection or hand calculation. However, for more complex systems, determining a correct set of initial conditions may not be simple. In this section, we discuss two computational procedures for correcting the initial conditions.

To determine a set of initial conditions that satisfy the constraints, we must solve the constraints at the coordinate and the velocity level. The solution methods that are presented in this section can be illustrated through a simple example, using a set of linear algebraic equations. Assume that a set of initial conditions (estimates) on three variables are given as  $x^e = 0.9$ ,  $y^e = 2.1$ , and  $z^e = 3.2$ . These estimates must satisfy the equations

$$\begin{aligned}\Phi_1 &= x + 2y - z - 2 = 0 \\ \Phi_2 &= 2x - y + z - 3 = 0\end{aligned}\tag{a}$$

Evaluating the constraints reveals that  $\Phi_1 = 3.9$  and  $\Phi_2 = -0.5$ ; that is, the constraints are not satisfied, and therefore, the initial conditions must be corrected.

We can correct the initial conditions by keeping the value of one of the variables unchanged and solve for the other two variables. For example, we can keep  $x = x^e = 0.9$  and solve the equations for the other two variables to find  $y = 1.1333$  and  $z = 1.1667$ . Obviously, if we select  $y$  or  $z$  to keep its given value, the result would be different. This method is exactly what we do for kinematic analysis by solving algebraic equations, as it was discussed in Chapter 12. Here, instead of introducing driver constraints, we specify as many as  $n_{dof}$  variables to keep their estimated values and solve for the remaining variables. Of course,

this must be done for the coordinates first that requires the solution of nonlinear constraint equations, and then for the velocity that requires the solution of linear velocity constraints.

In the second method, we adjust all the unknowns in such a way that the sum of squares of the corrections is at its minimum. To correct the coordinates, if their estimated initial conditions are  $\mathbf{q}^e$ , they must satisfy the constraints

$$\Phi(\mathbf{q}) = \mathbf{0} \quad (14.6)$$

Otherwise, the constraints are solved by the Newton–Raphson iterations, using the following correction steps until  $\text{norm}({}^i\Phi) < \epsilon$ :

$$\begin{aligned} {}^i\Delta\mathbf{q} &= -{}^i\mathbf{D}'({}^i\mathbf{D} {}^i\mathbf{D}')^{-1} {}^i\Phi \\ {}^{i+1}\mathbf{q} &= {}^i\mathbf{q} + {}^i\Delta\mathbf{q} \end{aligned} \quad (14.7)$$

After the correction of the coordinates, the estimated velocities,  $\dot{\mathbf{q}}^e$ , are corrected in order to satisfy the velocity constraints

$$\mathbf{D}\dot{\mathbf{q}} = \mathbf{0} \quad (14.8)$$

Since the velocity constraints are linear in velocities, the estimated velocities are corrected in one step as

$$\begin{aligned} \Delta\dot{\mathbf{q}} &= -\mathbf{D}'(\mathbf{D}\mathbf{D}')^{-1}\mathbf{D}\dot{\mathbf{q}}^e \\ \dot{\mathbf{q}} &= \dot{\mathbf{q}}^e + \Delta\dot{\mathbf{q}} \end{aligned} \quad (14.9)$$

In problems where all the initial velocities are zero, regardless of the complexity of the system, the velocity constraints are automatically satisfied.

To have a better understanding of the second method, let us return to our numerical example of Eq. (a). In the second method, we do not have to specify any of the coordinates to keep its estimated value. Since our example contains a set of linear equations, the method corrects the estimated values in one step:  $x = -0.058$ ,  $y = 2.256$ , and  $z = 2.570$ , where all three estimates have been adjusted.

In the first method, the three variables were corrected that lead to the following sum of squares:

$$\Delta x = 0, \Delta y = -0.9867, \Delta z = -2.0333 \Rightarrow \Delta x^2 + \Delta y^2 + \Delta z^2 = 5.1078$$

The sum of squares of the adjustments from the second method is

$$\Delta x = -0.9580, \Delta y = 0.1560, \Delta z = -0.6300 \Rightarrow \Delta x^2 + \Delta y^2 + \Delta z^2 = 1.3390$$

It is clear that the second method made a smaller overall adjustment to the estimates. We should note that the initial condition correction algorithms that are implemented in the programs of Chapters 8 and 9 are based on the second method.

The following is a proof for the correction formulas based on minimizing the sum of squares of the corrections. The proof is shown for a set of linear algebraic equations, such as the velocity constraints, but it can also be applied iteratively to nonlinear algebraic equations.

**Proof 14.1**

To minimize the sum of squares of the corrections in the array of variables,  $\mathbf{x}$ , that satisfies a set of algebraic equations  $\mathbf{D}\mathbf{x} = 0$ , we assume that an estimate for the array of variables is given as  $\mathbf{x}^e$ . This estimated array results in  $\mathbf{D}\mathbf{x}^e = \boldsymbol{\varepsilon}$ . The correction in the array of variables is denoted as  $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^e$ . A minimization problem can be stated as

$$\text{Minimize } f = \Delta\mathbf{x}'\Delta\mathbf{x} \text{ subject to constraints } \mathbf{D}\mathbf{x} = 0$$

This constrained minimization problem can be restated as

$$\text{Minimize } f = \Delta\mathbf{x}'\Delta\mathbf{x} - \Delta\mathbf{x}'\mathbf{D}'\boldsymbol{\lambda} \quad (\text{a})$$

where  $\boldsymbol{\lambda}$  is an array of Lagrange multipliers.\* The function in Eq. (a) is minimized when

$$\frac{\partial f}{\partial \Delta\mathbf{x}} = 2\Delta\mathbf{x} - \mathbf{D}'\boldsymbol{\lambda} = 0 \quad (\text{b})$$

Premultiplying this equation by  $\mathbf{D}$  results in  $2\mathbf{D}\Delta\mathbf{x} = \mathbf{D}\mathbf{D}'\boldsymbol{\lambda}$ , or  $\boldsymbol{\lambda} = -2(\mathbf{D}\mathbf{D}')^{-1}\boldsymbol{\varepsilon}$ . Substituting this result in Eq. (b) yields

$$\Delta\mathbf{x} = -\mathbf{D}'(\mathbf{D}\mathbf{D}')^{-1}\boldsymbol{\varepsilon}$$

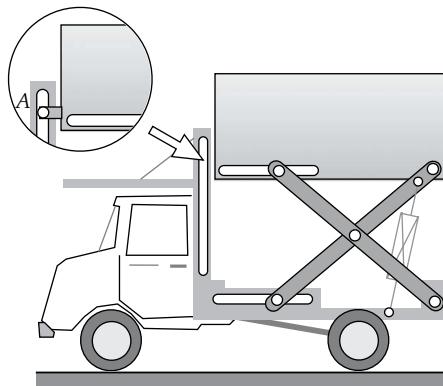
$$\mathbf{x} = \mathbf{x}^e + \Delta\mathbf{x}$$

## 14.4 Redundant Constraints

In all analyses that involve the solution of equations containing constraints, the assumption is that the constraints are independent. However, this may not be the case in every problem. In some multibody systems, inclusion of redundant links and kinematic joints may be part of the design. This could be for distributing of reaction forces more evenly, lowering the amount of structural deformations, safety reasons, or not allowing a mechanism to end up in an unwanted configuration. Regardless of the actual reason for the redundancy in the design, we must be aware of the computational consequences in modeling such systems.

The following example is provided to show how redundancy may be part of a design. Consider the lift mechanism of a delivery truck shown in Figure 14.5. A hydraulic actuator lifts the container to the desired height for goods to be delivered. There are no redundancies in the design of this lift mechanism. Assuming that this system is modeled with the body-coordinate formulation, for the three moving bodies, we need  $n_v = 9$  coordinates, and for the three pin joints and two revolute-translational joints, we have  $n_c = 8$  constraints. This leaves the formulation with one DoF that is controlled by the hydraulic actuator.

\* Interested reader should refer to any textbook on the subject of optimization.

**FIGURE 14.5**

A lift mechanism for a delivery truck and a redundant joint (inset) that could be included in the system.

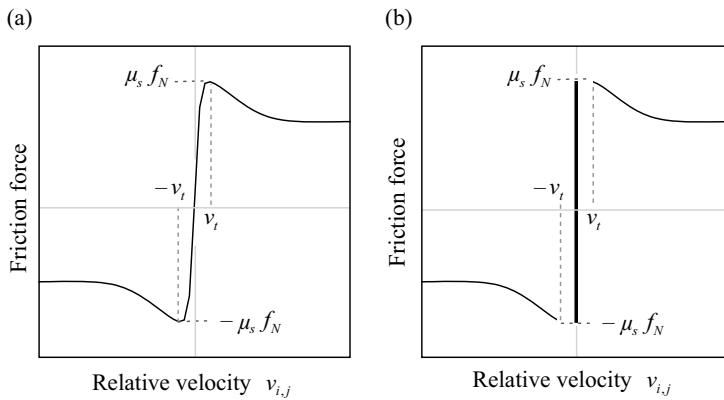
Next we consider a modified design of the lift mechanism as shown in the inset of Figure 14.5. In this design, to eliminate any undesirable vibrations of the container due to any imperfections in the joints, the container is attached to a vertical support by an additional revolute-translational joint. If we model this system, we will have  $n_v = 9$  and  $n_c = 9$ , where the system still has one DoF. Obviously one of the constraints must be redundant.

When redundancy is encountered in a formulation, if we know exactly which constraint is redundant, we can manually remove it from the model. If we are not aware of the redundancy, or if we are not certain which constraint should be removed, we can identify and remove the redundant constraint(s) computationally. For this purpose, we can perform some form of matrix factorization, such as Gaussian elimination or L-U factorization with pivoting (row exchange), on the Jacobian matrix of the constraints. This subject is briefly discussed in Appendix A.

## 14.5 Friction

In Section 4.5, we discussed how to determine the friction force between two contacting bodies. A friction formula is a function of several parameters, the relative velocity, and the normal force between two contacting bodies. Examples of such friction formulas are given in Eqs. (4.43) and (4.44). These or other similar formulas provide a continuous representation of the friction force as a function of the relative velocity. Although these models can provide an acceptable measure of the friction force, their representation of the static friction when the relative velocity is zero, or close to zero, can lead to computational inefficiency in forward dynamics.

In forward dynamic analysis, a variable time-step integrator, such as `ode45` in MATLAB, adjusts the size of the integration time step based on the predicted highest frequency of the system. When the friction model in a multibody system operates in the vicinity of the static friction, due to the large slope of the friction force in the static region, the integrator takes very small time steps, leading to relatively long computation time. A method to improve the computational efficiency for such analyses is discussed in the following.

**FIGURE 14.6**

(a) The B-M friction model and (b) its revised representation.

Let us assume that we have selected the B-M friction model of Eq. (4.44), as shown graphically in Figure 14.6a. In this model, the static friction is active when the relative velocity is very small and close to zero:

$$-v_t < v_{ij} < v_t \quad (14.10)$$

If  $v_{ij}$  is not in this narrow region, we apply Eq. (4.44) to determine the friction force. But if  $v_{ij}$  satisfies the condition of Eq. (14.10), we perform the following steps:

- We add a constraint to the equations of motion to eliminate the relative DoF between the two contacting surfaces. We solve the equations of motion and determine the reaction force associated with the added constraint, that is, the corresponding Lagrange multiplier,  $f\lambda$ . We compare  $f\lambda$  against the maximum possible static friction force:

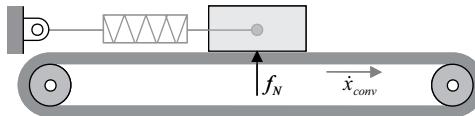
$$-{}^s\mu f_N < {}^f\lambda < {}^s\mu f_N \quad (14.11)$$

- If the condition of Eq. (14.11) is satisfied, then we have determined the friction force, and therefore, there is no need to use the friction formula.
- If the condition of Eq. (14.11) is not satisfied, that is,  $|{}^f\lambda| > {}^s\mu f_N$ , we set the friction force to its maximum value as  $|{}^f\lambda| = {}^s\mu f_N$ . We implement this friction force in the equation of motion in the same direction as that of the computed  $f\lambda$ , remove the added constraint, and recompute the accelerations.

This revised friction model is illustrated in Figure 14.6b.

### Example 14.3

The spring–mass system shown in Figure 14.7 is a typical example that has been used in literature for testing and comparison between friction formulas. The block rests on a conveyor belt that moves with a constant speed. As the block moves with the belt, it stretches the spring until the force of the spring overcomes the friction force and pulls

**FIGURE 14.7**

A spring–mass system moving on a conveyor belt.

the block back, and the process is repeated. For this system, let us consider the friction model of Eq. (4.44) and the following data:

$$\begin{aligned}m &= 1 \text{ kg}, k = 10 \text{ N/m}, {}^0L = 0.5 \text{ m}, g = 9.81 \text{ m/s}^2, \dot{x}_{conv} = 0.1 \text{ m/s} \\ \mu_s &= 0.2, \mu_d = 0.15, v_t = 0.001 \text{ m/s}\end{aligned}$$

In the friction model, we consider no viscous damping. The initial conditions for the block are  ${}^0x = 0.5 \text{ m}$  and  ${}^0\dot{x} = 0.1 \text{ m/s}$ . Simulate the dynamic response of the system for 10 s:

- Consider the friction model of Eq. (4.44).
- Consider the revised method in conjunction with Eq. (4.44).

### *Solution*

(a) For this problem, we consider the equation of motion for the block only in the  $x$ -direction, where the normal force is a constant and equal to the weight of the block:

$$m\ddot{x} = -k(L - {}^0L) + {}^{(f)}f$$

The friction force is computed using the M-file `Friction_B` from Chapter 4. For the friction force model, we compute the relative velocity as  $v_{i,j} = v_{conv} - \dot{x}$ . We construct two M-files as `Example_14_3` and `Ex_14_3`, using `ode45` to perform forward dynamic analysis.

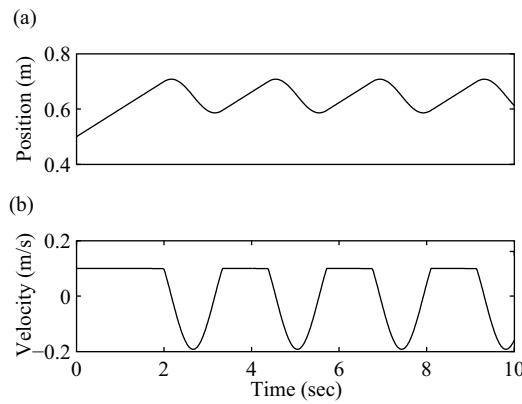
#### MATLAB/Chapter 14/Example\_14\_3\_a, Ex\_14\_3\_a

In the M-file `Example_14_3_a`, we provide the constant data and the initial conditions, and then we invoke `ode45` to integrate the equation of motion. The constant data are passed to the next M-file through a global statement. We guide the integrator to find the equation of motion in the M-file `Ex_14_3_a`.

```
function ud = Ex_14_3_a(t, u)
global m k L0 fN mu_s mu_d v_t v_conv f_eval
c = u(1); c_d = u(2);
f_s = k*(c - L0); % force of the spring
v_ij = v_conv - c_d; % relative velocity
f_f = fN*Friction_B(mu_s, mu_d, v_t, v_ij); % B-M friction
fx = -f_s + f_f; % total force acting on the block
c_dd = fx/m; % acceleration
ud = [c_d; c_dd];
f_eval = f_eval + 1; % number of function evaluations
```

To obtain a measure for computational efficiency and to make comparison between different methods, in this function we keep track of the number of function evaluations ( $f\_eval$ ). Since a fourth-order Runge–Kutta algorithm requires four function evaluations in every time step, the number of function evaluations divided by four is a reasonable representation of the number of time steps taken by the integrator.

Executing the program provides the results in the form of two plots as shown in Figure 14.8. We note that the flat portions of the velocity plot indicate that the block is moving with the conveyor belt due to static friction. The program also reports the number of function evaluation to be approximately 33,000.



**FIGURE 14.8**

Position and velocity of the block versus time.

(b) For the revised friction model, we evaluate the velocity as  $v_{i,j} = v_{conv} - \dot{x}$ . If  $|v_{ij}| > v_t$ , we use the function `Friction_B` to compute the friction force, and then we use the following equation of motion to determine the acceleration:

$$m\ddot{x} = -k(L - {}^0L) + {}^{(f)}f$$

If  $-v_t < v_{ij} < v_t$ , meaning that the friction is in the static region, we solve the following constrained equation of motion:

$$\begin{bmatrix} m & -1 \\ 1 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{x} \\ {}^{(f)}\lambda \end{Bmatrix} = \begin{Bmatrix} -k(L - {}^0L) \\ 0 \end{Bmatrix}$$

We solve this equation to determine  $\ddot{x}$  and  ${}^{(f)}\lambda$ . If the computed value of  ${}^{(f)}\lambda$  does not exceed the limits of static friction, then the solution for  $\ddot{x}$  is acceptable. But if  ${}^{(f)}\lambda$  exceeds the limits, we solve the following equation of motion to obtain  $\ddot{x}$ :

$$m\ddot{x} = -k(L - {}^0L) + \mu_s f_N \operatorname{sign}(v_{i,j})$$

We construct two M-files as `Example_14_3_b` and `Ex_14_3_b`.

**MATLAB/Chapter 14/Example\_14\_3\_b, Ex\_14\_3\_b**

This program is almost identical to Example\_14\_3\_a.

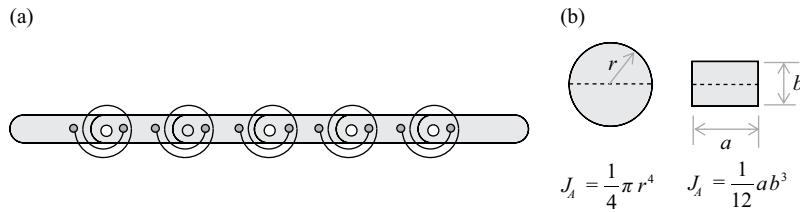
```
function ud = Ex_14_3_b(t, u)
global m k L0 fN mu_s mu_d v_t v_conv f_eval
    c = u(1); c_d = u(2);
    f_s = k*(c - L0); % force of the spring
    v_ij = v_conv - c_d; % relative velocity
if abs(v_ij) > v_t
    f_f = Friction_B(mu_s, mu_d, v_t, v_ij); % B-M friction
    fx = -f_s + fN*f_f; % total force acting on the block
    c_dd = fx/m; % acceleration
else
    fx = -f_s;
    DMD = [m -1
            1 0];
    rhs = [fx; 0];
    solution = DMD\rhs;
    if abs(solution(2)) < fN*mu_s
        c_dd = solution(1); % acceleration
    else
        c_dd = (fx + fN*mu_s*sign(solution(2)))/m; % acceleration
    end
end
ud = [c_d; c_dd];
f_eval = f_eval + 1; % number of function evaluations
```

Executing the program Example\_14\_3\_b provides the results in the form of two plots that are identical (considering some roundoff error) to those shown in Figure 14.8. However, the number of function evaluations is reported to be approximately 2,000. This shows a large reduction in the required computational time with the revised method.

## 14.6 Deformable Body

In some applications of multibody dynamics, the structural deformation of a body may not be negligible, and it could influence the overall response of a system. In these applications, the deformation of a body must be incorporated into its equations of motion. Finite-element method provides a powerful tool to determine structural deformations, which can also be used to model the structural deformation in a multibody system. However, it is outside the scope of this textbook to discuss the finite-element formulation within the subject of multibody dynamics. Instead, we introduce a simple approximation process that could be considered as a *crude* finite-element representation for a deformable body.

The approximation method is demonstrated here for a deformable rod (a beam). Assume that the rod shown in Figure 14.9a, with a length  $L$ , exhibits bending deformation. The rod

**FIGURE 14.9**

(a) A deformable rod can be represented as an assembly of smaller rods connected by pin joints and torsional springs; (b) area moment of inertia about the neutral axis for two common cross sections.

can be split into  $n$  smaller rods, each having a length  $L/n$ , that are connected to each other by  $n - 1$  pin joints. A torsional spring is defined about the axis of each pin joint having the rotational stiffness

$$^{(r)}k_{n-1} = \frac{EJ_A}{\ell} \frac{(n-1)(2n-1)}{2n} \quad (14.12)$$

where  $E$  is Young's modulus and  $J_A$  is the area moment of inertia of the rod's cross section. Values of Young's modulus, or the modulus of elasticity, for different materials can be found in engineering handbooks. The area moment of inertia for most common cross-sectional shapes can be found in most textbooks on statics and dynamics. For circular and rectangular cross sections, the area moment of inertia about the *neutral axis* is shown in Figure 14.9b.

It should be obvious that representing a deformable body as an assembly of several smaller bodies, connected by joints and springs, increases the size of a problem, and hence the computation time. If a multibody system is modeled with the joint-coordinate formulation, the increase in the problem size may not be as significant as with the body-coordinate formulation. Another contributor to the increase in the computation time is that realistic values for Young's modulus yield high stiffness values for the springs. This in turn requires small integration time steps during forward dynamic analyses.

## 14.7 Problems

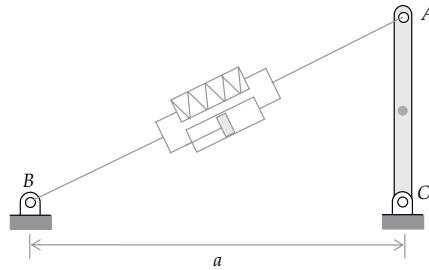
- 14.1 Use the program DAP \_ BC from Chapter 8 to model the structure in Figure 14.2 with the body-coordinate formulation. Execute the program to determine the reaction forces at the joints.
- 14.2 For the structure in Figure 14.2, assume that we only know an approximate position and orientation data for the bodies as

$$\mathbf{c}_1 = \begin{Bmatrix} 1.6 & 1.9 & 45^\circ \end{Bmatrix}, \mathbf{c}_2 = \begin{Bmatrix} 2.9 & 2.1 & 90^\circ \end{Bmatrix}$$

Solve the constraint equations using MATLAB function `fsolve` and then determine the reaction forces.

- 14.3 The rod shown is 0.3 m long and is pinned to the ground at one end. The rod has a mass  $m = 4.0 \text{ kg}$  and a moment of inertia  $J = 0.03 \text{ N m}^2$ . A spring-damper connects the other end of the rod to the ground as shown, where  $a = 0.6 \text{ m}$ . The force element has the following characteristics:  $k = 100 \text{ N/m}$ ,  ${}^0L = 0.8 \text{ m}$ ,  $d_c = 20 \text{ N s/m}$ .

- Construct the static equilibrium equations for this system.
- Determine the equilibrium state using MATLAB function `fsolve`.



- 14.4 For each of the systems shown, consider the gravity as an applied force. Use the following data as applicable:

$$a = 0.01, b = 0.03, c = 0.02, L_2 = 0.06, L_3 = 0.04 \text{ m}$$

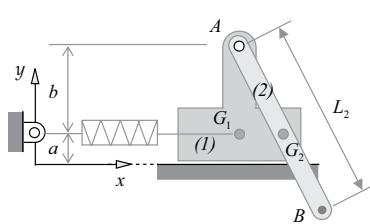
$$k = 100 \text{ N/m}, {}^0L = 0.08 \text{ m}$$

$$m_1 = 0.3, m_2 = 0.2, m_3 = 0.15 \text{ kg}, J_1 = 0.05, J_2 = 0.001, J_3 = 0.0008 \text{ kg m}^2$$

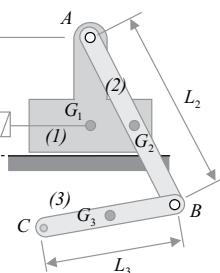
Construct the static equilibrium equations for each system.

Determine the equilibrium state using MATLAB function `fsolve`.

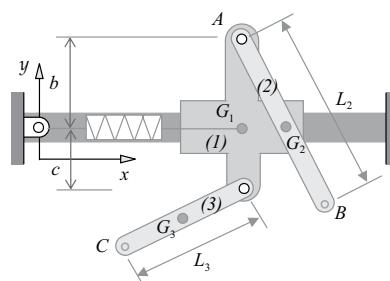
(a)



(b)



(c)



- 14.5 In the shown configurations, construct the free body diagram and the equations of motion. Consider the slider–crank mechanism in Example 6.6 and revise the developed program for this example to solve the corresponding static equilibrium equations. Use MATLAB function `fsolve` to solve the nonlinear equilibrium equations.
- 14.6 Consider the equations of motion for the four-bar mechanism in Example 6.7. Revise the developed program for this example to solve the corresponding static equilibrium equations. Use MATLAB function `fsolve` to solve the nonlinear equilibrium equations.
- 14.7 For the systems shown in Problem 14.4, our estimates for the coordinates of the bodies are (apply as applicable)

$$\mathbf{r}_1 = \begin{Bmatrix} 0.6 \\ 0.1 \end{Bmatrix}, \mathbf{r}_2 = \begin{Bmatrix} 0.7 \\ 0.15 \end{Bmatrix}, \mathbf{r}_3 = \begin{Bmatrix} 0.6 \\ -0.3 \end{Bmatrix}$$

Our estimates for the orientations are as follows: link (2) makes a  $20^\circ$  angle with the vertical axis, and link (3) makes a  $10^\circ$  angle with the horizontal axis. For the velocities, we have the following:

$$\dot{x}_1 = 0.2 \text{ m/s}, \dot{\phi}_2 = -0.3 \text{ rad/s}, \dot{\phi}_3 = 0.4 \text{ rad/s} \text{ (as applicable)}$$

- Apply the iterative process of Eq. (14.7) to determine a correct set of coordinates.
- Determine a correct set of initial velocities while keeping the known velocities unchanged.

# 15

---

## *Application Examples*

---

In this chapter, a variety of examples are provided to demonstrate the application of multibody systems in mechanisms, automotive, biomechanics, robotics, and machinery. The examples are accompanied by a short description and the necessary data for modeling and analysis using the methodologies that have been discussed in this textbook. Some of the examples are very simple systems that have been used repeatedly throughout this textbook to clarify the concepts and formulations. Although these examples reflect the actual multibody systems with applications, they are simple enough not to distract us from understanding a concept. Through these examples, we learn how to construct analytical models, formulate multibody systems using different formulations, and observe major and subtle differences between formulations.

Other examples of more complex multibody systems are also presented in this chapter that can be used as simulation projects. For some of these examples, a scenario for simulation is suggested, but for some others, the scenario for simulation is left to the user's imagination and creativity. A simulation could be performed with the program DAP \_ BC from Chapter 8, or with the program DAP \_ JC from Chapter 9, or the user may construct the equations of motion with other formulations and develop a specific program for simulating the dynamics of that particular system. These examples should also give the reader some ideas for modeling and analyzing the response of other multibody systems.

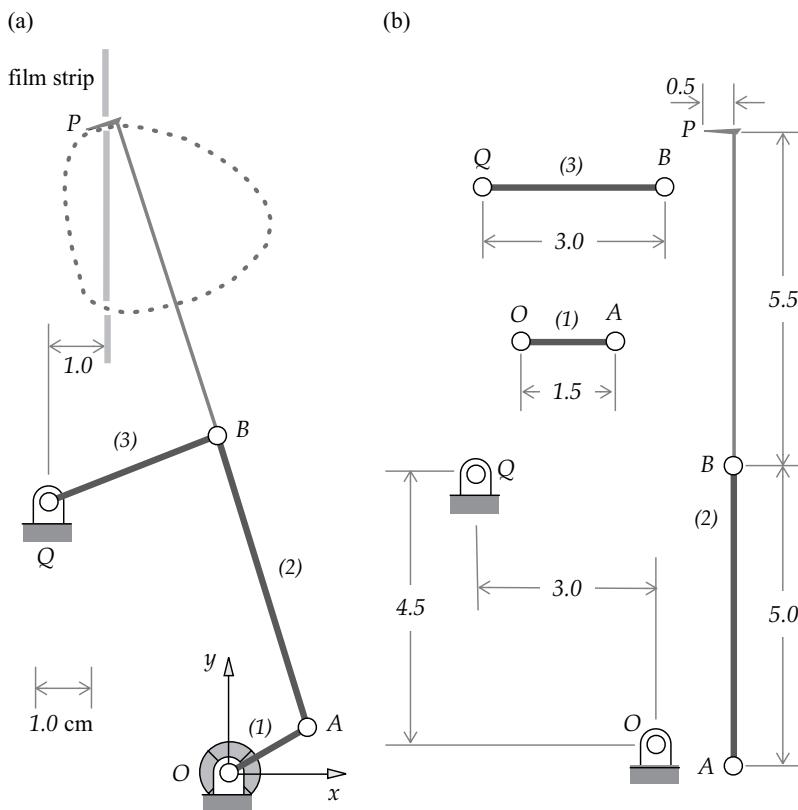
Some of the listed data for the examples are realistic, but some are not. Some data have been scaled or changed completely to make the corresponding figure not to be confusing, or to simplify the numerical values that appear in the formulations, or to avoid encountering specific numerical issues.

---

### **15.1 Film-Strip Advancer**

The simple mechanism shown in Figure 15.1a can be found in most textbooks on mechanisms. This four-bar mechanism provides an intermittent motion to advance a film strip in a primitive type movie projector. The coupler point  $P$  periodically engages and disengages from the sprocket holes in the film strip as the crank rotates clockwise. If the crank is rotated at  $1800\text{ rpm}$  ( $60\pi \text{ rad/s}$ ), the film is advanced 30 frames a second.

The mass center of the combined link (1) and the motor is at  $O$ . The mass center of link (2) could be considered to be at  $B$ , and that of link (3) is at its the geometric center. The dimensions are shown in Figure 15.1b, and the inertial data are listed in Table 15.1.



**FIGURE 15.1**  
(a) A film-strip mechanism; (b) the corresponding dimensions in centimeters.

**TABLE 15.1**

Inertial Properties of the Film-Strip Advancer

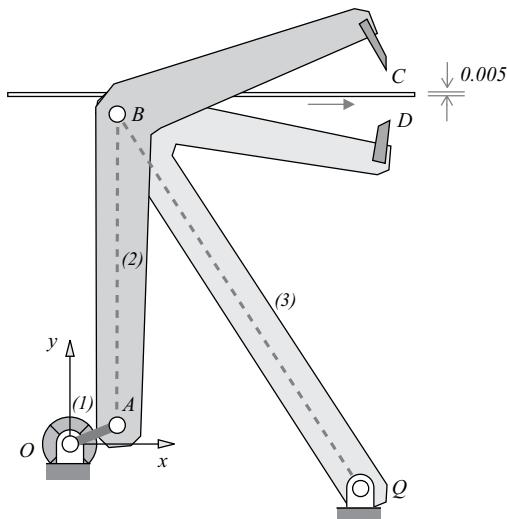
Body (Index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1) + motor	0.4	0.5
(2)	0.2	0.02
(3)	0.05	0.0005

## 15.2 Web-Cutter Mechanism

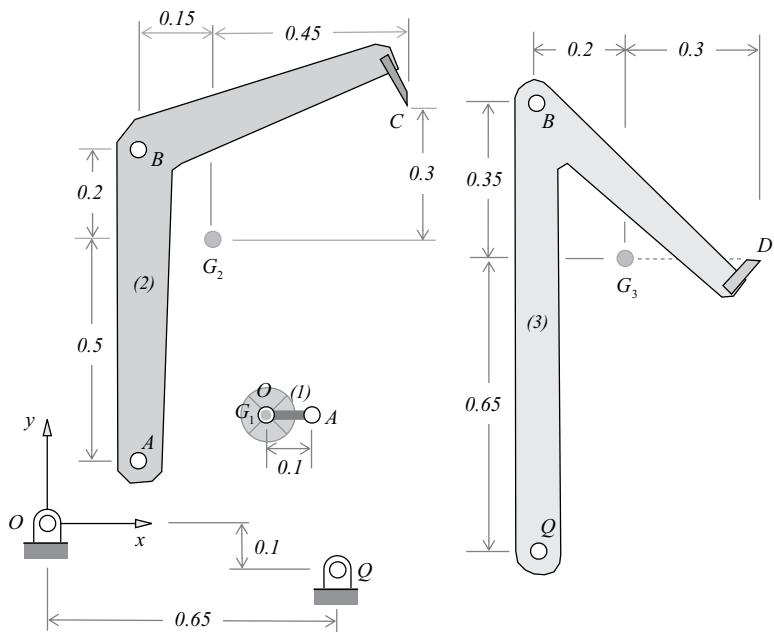
The crank–rocker four-bar mechanism shown in Figure 15.2a has been adopted as a web cutter. As the crank rotates, the cutting blades at C and D cut through a sheet moving from left to right. The velocity of the web at the time of cut must be adjusted to match the average velocity of the two blade points along the horizontal axis. It is assumed that the crank rotates with a constant angular velocity of 60 rpm counterclockwise.

The individual bodies and dimensions are shown in Figure 15.2b. The inertial data are listed in Table 15.2.

(a)



(b)

**FIGURE 15.2**

(a) A web-cutter mechanism; (b) the corresponding dimensions in meters.

**TABLE 15.2**

Inertial Properties of the Web-Cutter Mechanism

Body (Index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1) + motor	1.0	1.0
(2)	10.0	5.0
(3)	10.0	5.0

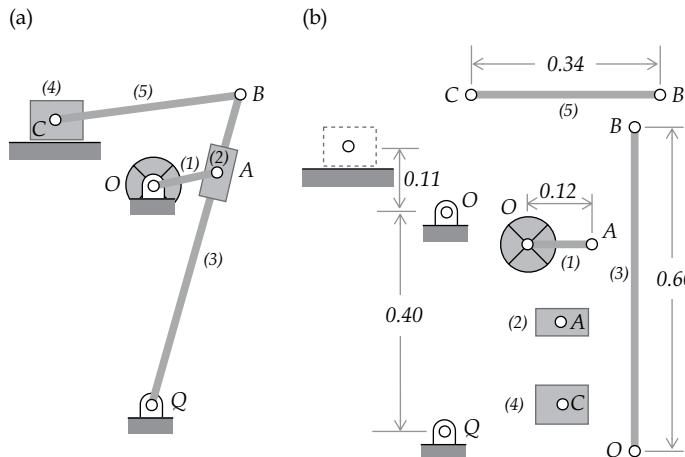
### 15.3 Six-Bar Quick-Return Mechanism

A mechanism that its output link moves slowly in one direction but faster in return is called a *quick-return* or a *quick-forward* mechanism depending on the rotational direction of the motor. A four-bar or a slider-crank could be designed as a quick-return mechanism. The example shown in Figure 15.3a is a six-bar quick-return where, for a constant speed motor rotating link (1), the slider block (4) which is the output link exhibits a quick-return motion.

The individual bodies and dimensions are shown in Figure 15.3b. The inertial data are listed in Table 15.3.

### 15.4 Six-Bar Dwell Mechanism

A mechanism that the motion of its output link comes to a temporary halt while its input link keeps it in motion is called a *dwell* mechanism. A four-bar or a six-bar can be designed to be a dwell mechanism. As an example, the output link of the six-bar



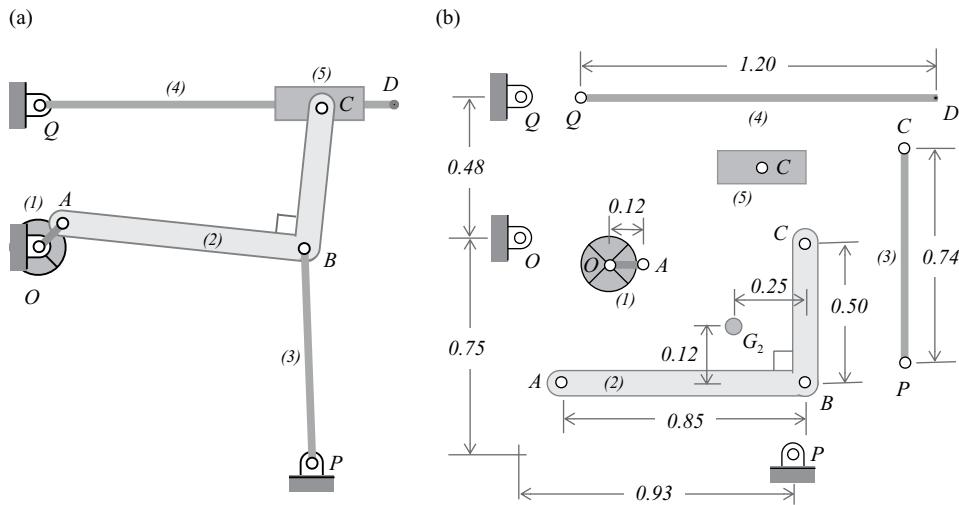
**FIGURE 15.3**

(a) A quick-return six-bar mechanism; (b) the corresponding dimensions in meters.

**TABLE 15.3**

Inertial Properties of the Six-Bar Quick-Return Mechanism

Body (Index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1) + motor	1.0	1.0
(2)	0.3	0.0003
(3)	0.2	0.006
(4)	0.4	0.0004
(5)	0.1	0.003

**FIGURE 15.4**

(a) A dwell six-bar mechanism; (b) the corresponding dimensions in meters.

**TABLE 15.4**

Inertial Properties of the Six-Bar Dwell Mechanism

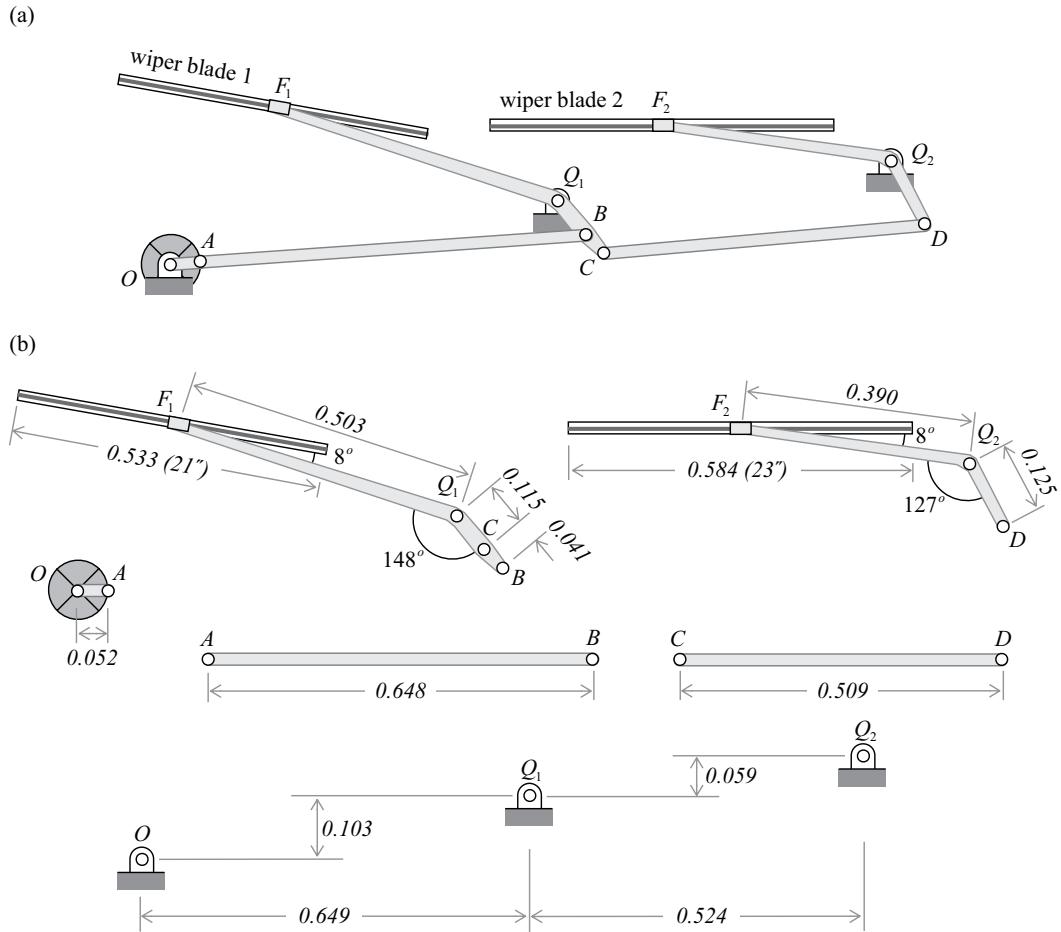
Body (Index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1) + motor	1.0	1.0
(2)	0.3	0.0003
(3)	0.2	0.004
(4)	0.3	0.006
(5)	0.4	0.0004

mechanism shown in Figure 15.4a, link (4), dwells as the motor rotates with a constant angular velocity.

The individual bodies and dimensions are shown in Figure 15.4b. The inertial data are listed in Table 15.4.

## 15.5 Windshield Wiper Mechanism

Applications of multibody dynamics in the automotive industry have been demonstrated throughout this textbook with several examples of suspension systems. Multibody dynamics can be applied to other applications in the automotive industry, such as a steering system, an engine, or a windshield wiper mechanism. In this example, as shown in Figure 15.5a, a typical windshield wiper system is presented as a six-bar mechanism where the wiper blades are attached to the two rocker links. The lengths of the links and the positions of the attachment points to the frame are shown in Figure 15.5b. Although all of the links are slender, their inertial data are negligible. We may assume a mass density of 0.3 kg/m.

**FIGURE 15.5**

(a) A windshield wiper mechanism; (b) the corresponding dimensions in meters.

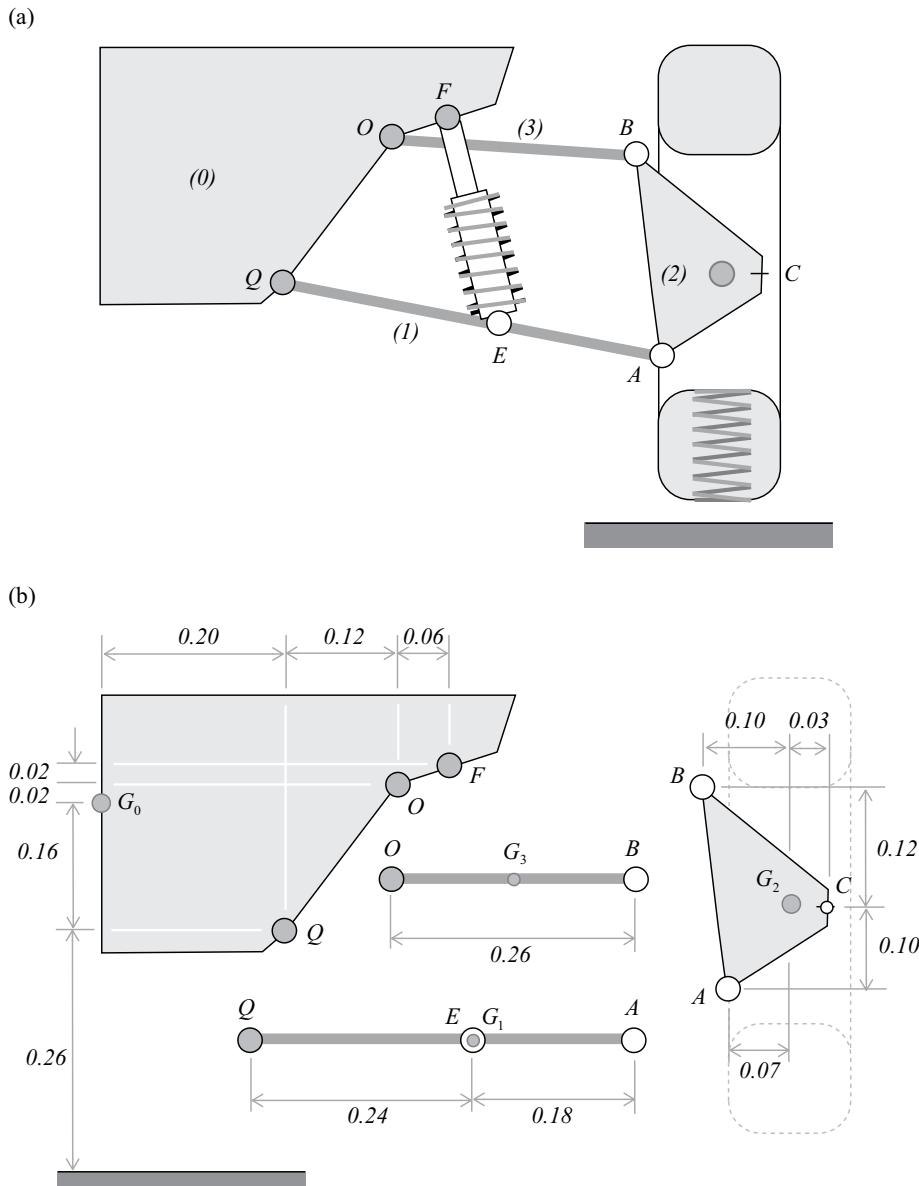
The main resistive force in this system comes from the friction between the wiper blades and the windshield. Each wiper blade is pressed against the windshield by a spring. Each spring causes a force of  $(^w)f = 10\text{ N}$ , perpendicular to the surface of the windshield, at  $F_1$  and  $F_2$ . We can assume worst-case maximum Coulomb coefficients of friction for dry conditions as  $\mu_s = 0.4$  and  $\mu_d = 0.3$ . The two friction forces must be applied to  $F_1$  and  $F_2$  in the opposite direction of their respective velocities. Typical angular velocity for the motor-gearbox combination is  $2\pi \text{ rad/s}$ .

One objective of analyzing this system could be to determine the power requirement for its motor-gearbox unit.\* An inverse dynamic analysis of this model can provide the torque requirement from the motor-gearbox combination.

\* A simple formula for computing the power requirement for windshield wiper motors can be found in *BOSCH Automotive Handbook, 6th ed.*, Robert Bosch GmbH, 2005.

## 15.6 Double A-Arm Suspension

One of the most commonly used independent front and rear suspension systems in automobiles is the double A-arm, also known as double wishbone or short–long arm. As a rear suspension, the system provides an up–down motion of the wheel assembly with respect to the frame. As a front suspension, in addition to the up–down motion, it allows a steering



**FIGURE 15.6**

(a) Planar frontal view of the double A-arm suspension; (b) the corresponding dimensions in meters.

**TABLE 15.5**

Inertial and Tire Properties for the Double A-Arm Suspension

Body (Index)		Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )	
1/4 chassis (0)		340	325	
(1)		2	0.5	
(2)		30	2.5	
(3)		1	0.5	
Suspension	Free Length (m)	Stiffness (N/m)	Damping (N s/m)	
	0.23	90,000	1,100	
Tire	Undeformed Radius (m)	Deformed Radius at Rest (m)	Radial Stiffness (N/m)	Radial Damping (N s/m)
	0.35	0.34	50,000	1,000

degree of freedom (DoF). When this system is viewed from the front or the back, it exhibits planar motion as a four-bar mechanism, as depicted in Figure 15.6a. The steering or tie rod does not have any effect on the planar motion of this system. The multibody system shown in this figure is referred to as a quarter-car representation, which could be either the front view or the rear view of a car. We assume that this is the frontal view of the left-front suspension.

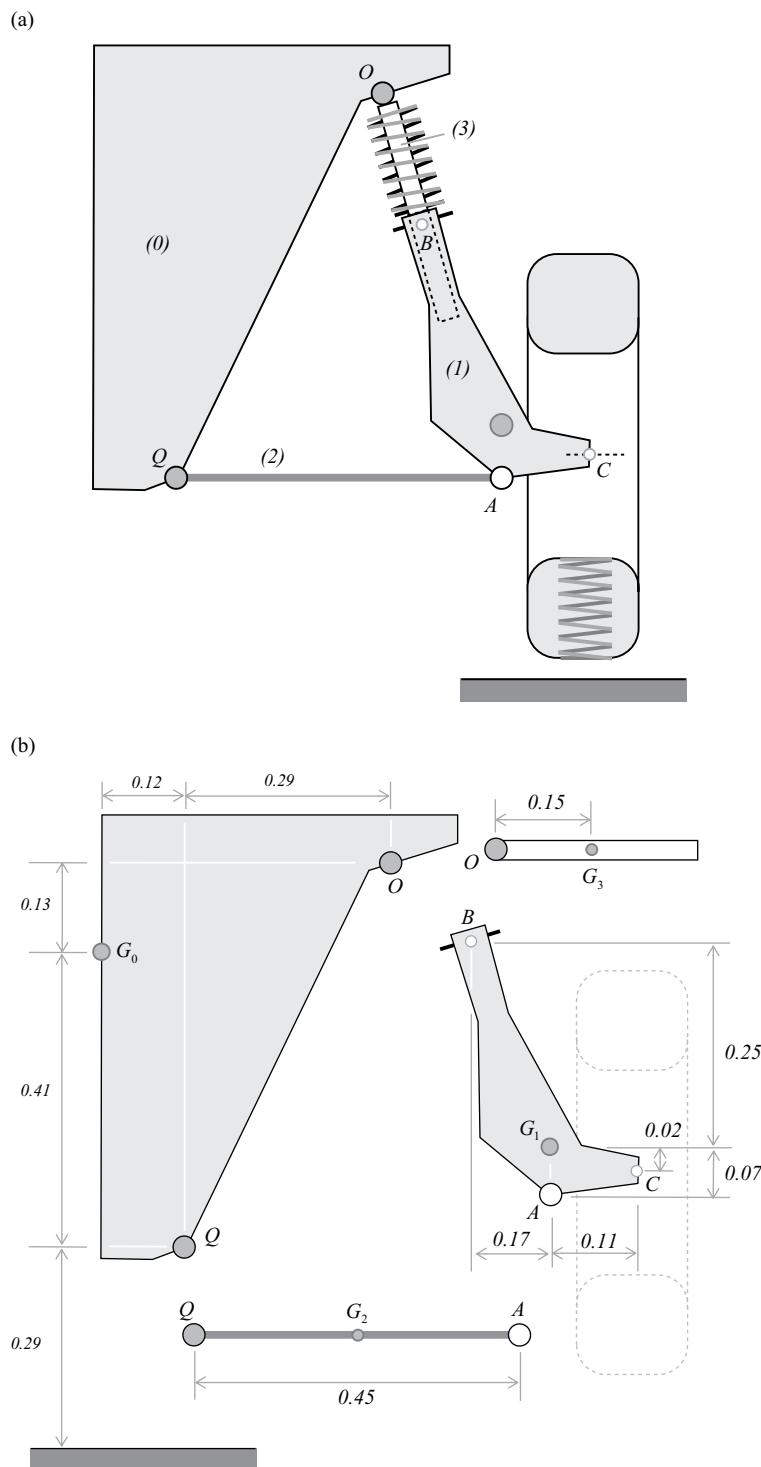
For modeling purposes, the joints and the attachment points are labeled in the shown schematic view. In this presentation, body (0) is the frame. Bodies (1) and (3) are the lower and upper arms respectively, and body (2) is the wheel assembly. The wheel–tire combination is shown in scaled size in the background for reference. The coil-over-shock absorber is the suspension spring–damper.

The individual multibody components of this system and the corresponding dimensions are shown in Figure 15.6b. The mass centers for the three moving bodies are at  $G_1$ ,  $G_2$ , and  $G_3$ . The mass and roll moment of inertias, the characteristics of the spring–damper for the suspension, and tire spring–damper data in the radial direction (vertical) are listed in Table 15.5.

## 15.7 MacPherson Strut Suspension

Another common type of the front suspension system in automobiles is the MacPherson strut and link. The fundamental principles of this system are very similar to those of the double A-arm suspension. Variations of this basic strut and link can be found in different automobiles.

When this system is viewed from the front (or the back), it exhibits planar motion as an inverted slider–crank mechanism. The steering rod does not have any effect on the planar motion of this system. For modeling purposes, the joints and the attachment points are labeled in the schematic presentation in Figure 15.7a. In this presentation, body (0) is the frame, body (1) is the lower arm, and body (2) is the wheel assembly. The coil-over-shock absorber that includes link (3) acts as a sliding joint (a cylindrical joint in 3D).

**FIGURE 15.7**

(a) Planar view of the MacPherson suspension; (b) the corresponding dimensions in meters.

**TABLE 15.6**

Inertial and Tire Properties for the MacPherson Suspension

Body (Index)		Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
1/4 chassis (0)	300	100	
	(1)	20	2.5
	(2)	2	0.5
	(3)	0.5	0.2
Suspension	Free Length (m)	Stiffness (N/m)	Damping (N s/m)
Tire	0.34	20,000	1,100
	Undeformed Radius (m)	Deformed Radius at Rest (m)	Radial Stiffness (N/m)
	0.30	0.29	100,000
			Radial Damping (N s/m)
			1,000

The dimensions of the links and the positions of the attachment points are given in Figure 15.7b. The mass centers for the three moving bodies are at  $G_1$ ,  $G_2$ , and  $G_3$ . For link (2), point  $B$  is defined on the strut axis at a convenient location as a reference point for some of the dimensions. The mass and moment of inertia of the bodies, the characteristics of the suspension spring–damper, and the tire spring–damper data in the radial direction are listed in Table 15.6.

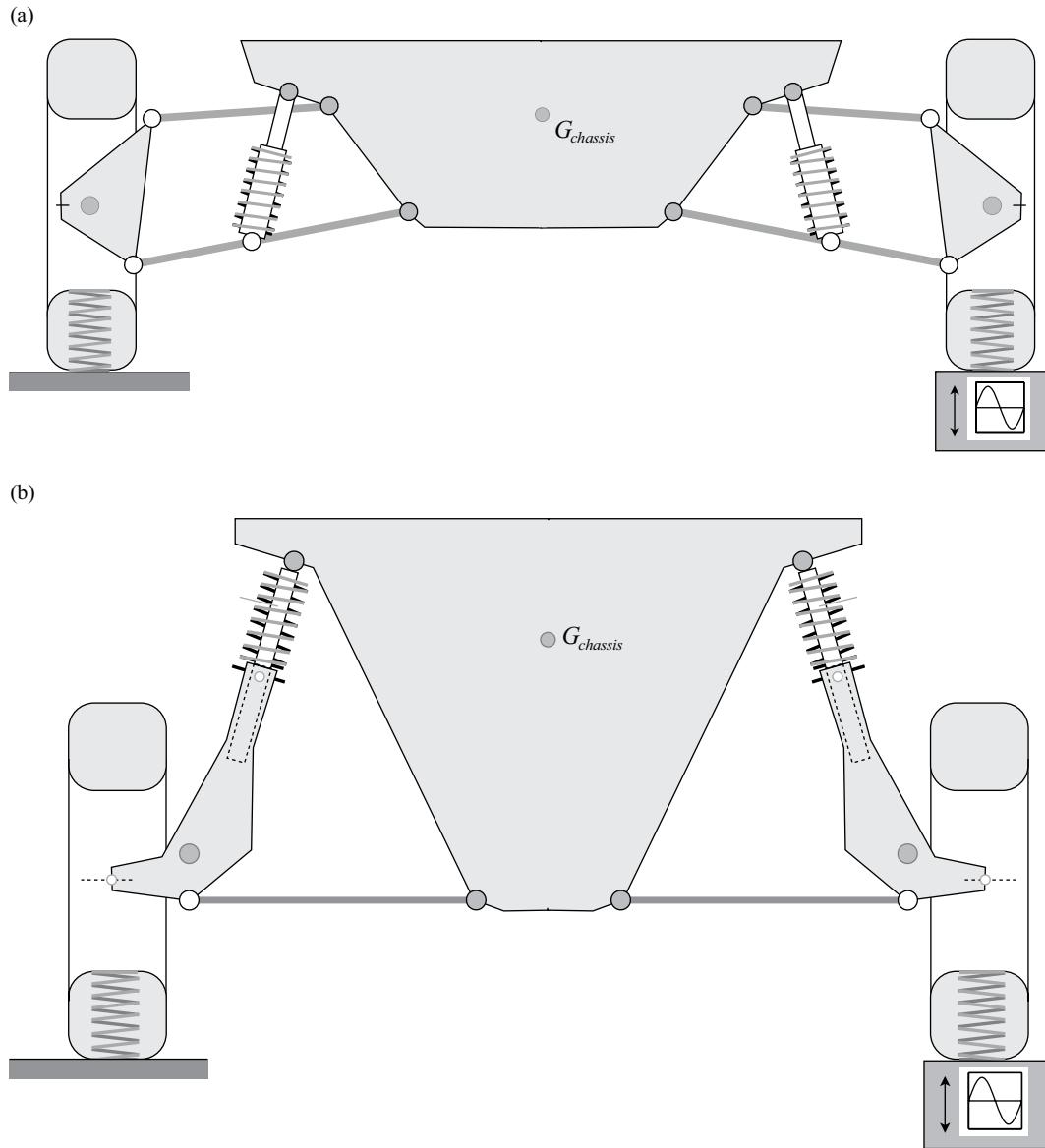
## 15.8 Half-Car

The quarter-car double A-arm and MacPherson suspensions of Sections 15.6 and 15.7 are represented here as two half-car models as shown Figure 15.8. In these models, the main chassis is no longer assumed stationary as it was considered in the quarter-car models. The data for these half-car models can be obtained from their corresponding quarter-car models. The only required adjustment is for the mass and the moment of inertia of the chassis.

With a half-car model, we can perform a vibration test by applying a sinusoidal excitation to one wheel, as shown, or to both wheels. If we apply excitations to both wheels, they could be either in-phase or out-of-phase. The frequency of the excitation could be far or close to any of the natural frequencies of the vehicle.

## 15.9 Mountain Bike

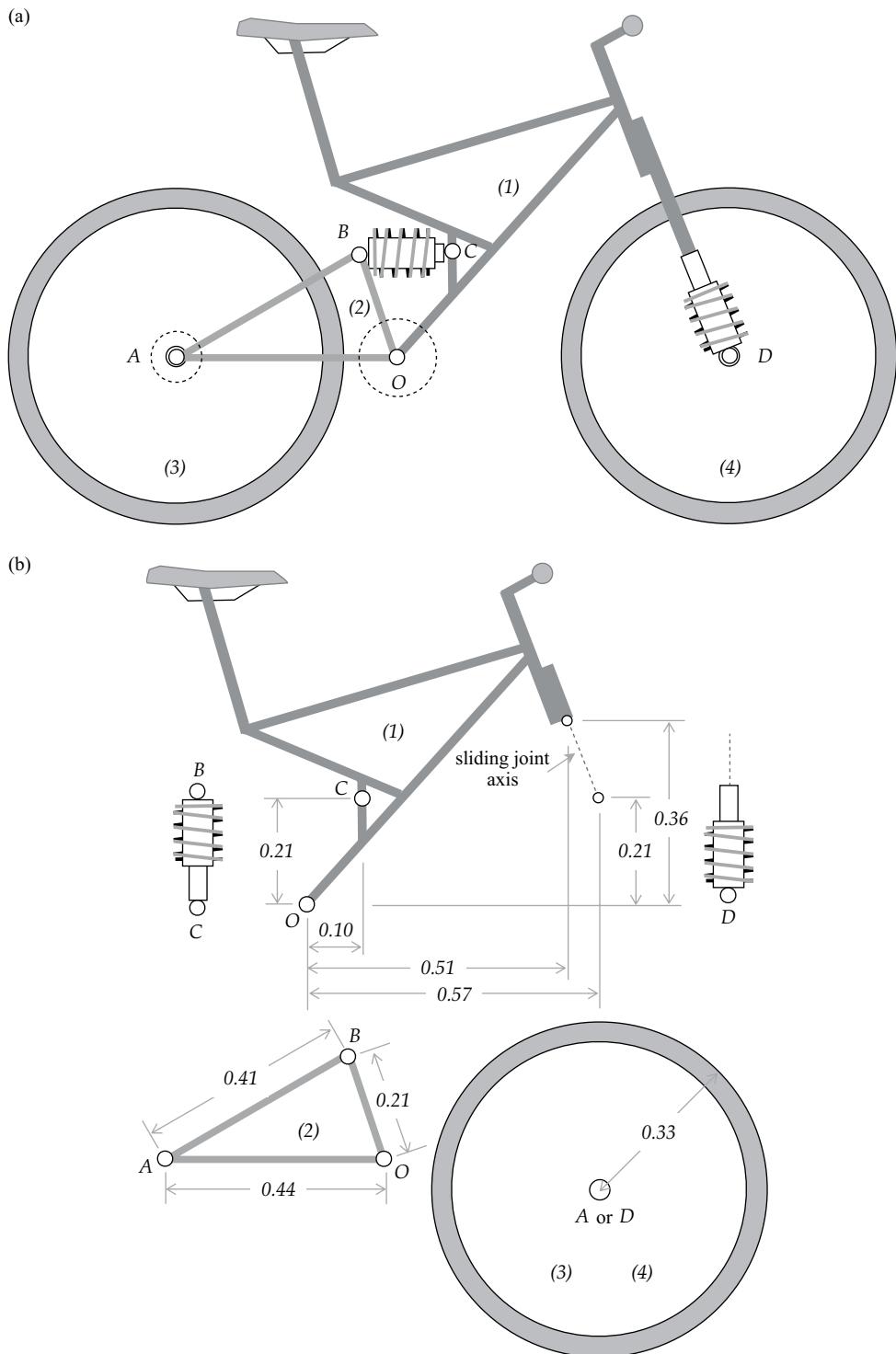
In this section, we provide data for constructing multibody models of two mountain bikes. The two models are very similar except for the rear suspension mechanism. A schematic



**FIGURE 15.8**  
Half-car representations of (a) double A-arm and (b) MacPherson suspensions.

planar view of bike 1 is shown in Figure 15.9a, and the individual components are shown in Figure 15.9b. There are two shock absorbers in this bike where the dampers act as sliding joints. The front suspension spring-damper can be placed between the wheel center and a point on the frame along the axis of the sliding joint. The necessary data are provided in Table 15.7.

The second model, bike 2, is very similar to bike 1 except for the rear suspension mechanism. A schematic planar view of bike 2 is shown in Figure 15.10a, and the individual



**FIGURE 15.9**

(a) Multibody representation of mountain bike 1; (b) the corresponding individual components and dimensions.

**TABLE 15.7**

Inertial, Suspension, and Tire Properties (Bike 1)

Body (index)		Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )	
(1)		3.2		
(2)		1.8		
Each wheel		1.5		
Suspension	Free Length (m)	Stiffness (N/m)	Damping (N s/m)	
Front		10,000	1,350	
Rear		120,000	5,300	
Tire	Undeformed Radius (m)	Deformed Radius at Rest (m)	Radial Stiffness (N/m)	Radial Damping (N s/m)
	0.33		60,000	2,700

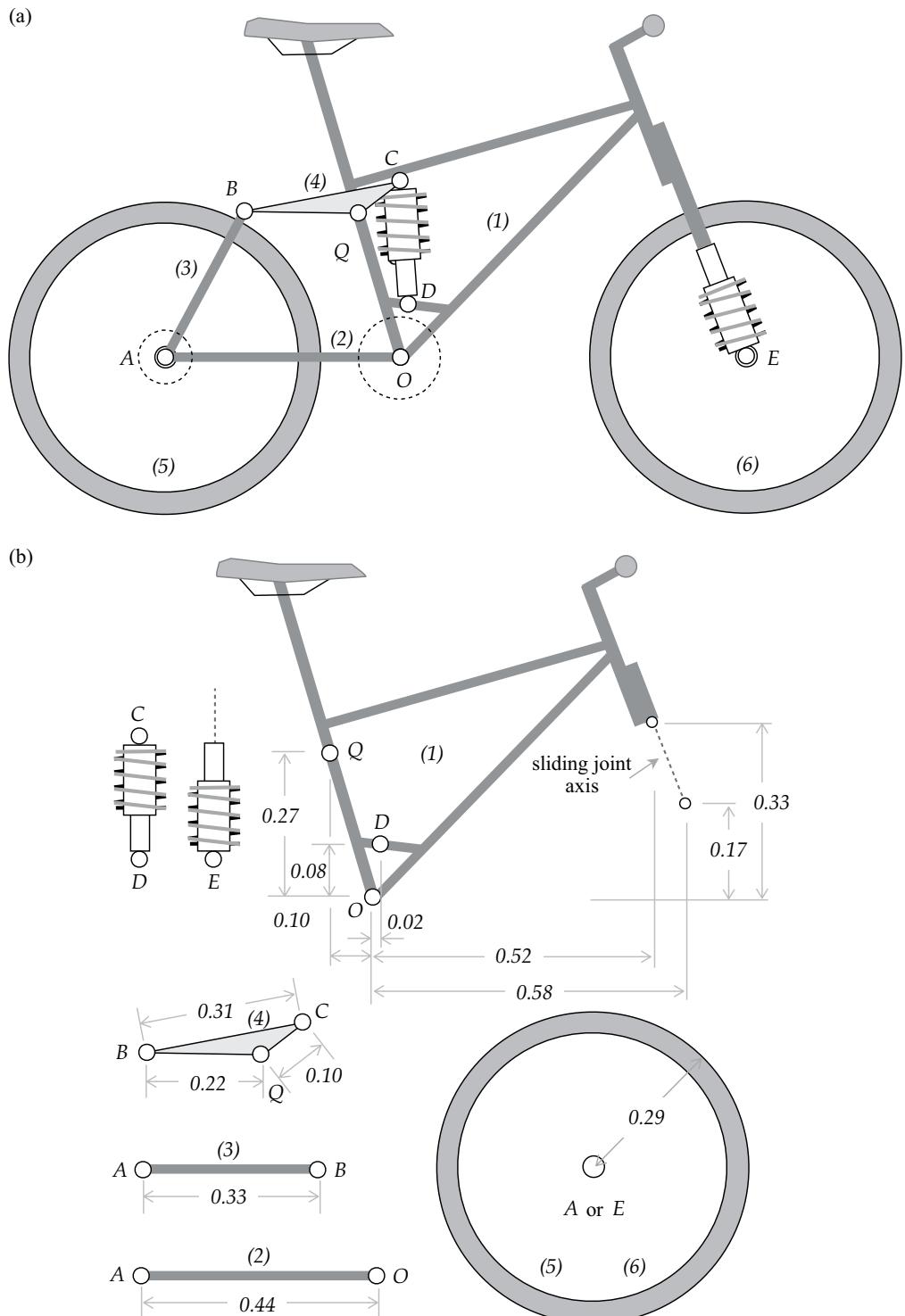
components are shown in Figure 15.10b. The necessary data for this bike are provided in Table 15.8.

## 15.10 Motorcycle

This example provides data to construct a multibody model of a motorcycle. As shown in Figure 15.11a, the front wheel suspension system is a four-bar mechanism in which two trailing links, attached to the frame, enable the wheel fork to move. The front suspension spring-damper is attached between the lower of the two trailing links and the frame. The rear suspension spring-damper is attached between a one-arm rocker and the frame. Individual components of the model with the necessary dimensions are shown in Figure 15.11b. The inertial, suspension, and tire data are provided in Table 15.9. In the multibody model, we need to decide on the location of the mass center of the frame depending on whether we include a rider in the model.

## 15.11 Dump Truck

A hydraulic actuator controls the unloading process of the dump truck shown in Figure 15.12a. The dimensions for the link lengths and attachment points are provided in Figure 15.12b, and the corresponding inertial data are listed in Table 15.10. The actuator can be modeled as a linear driver that its length can vary from a minimum to a maximum value in a given period of time, as shown in the inset (refer to the function “type b” in DAP \_ BC program, Chapter 8).



**FIGURE 15.10**

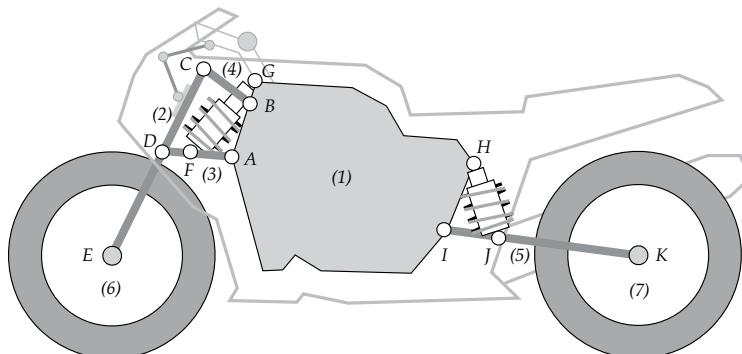
(a) Multibody representation of mountain bike 2; (b) the corresponding individual components and dimensions.

**TABLE 15.8**

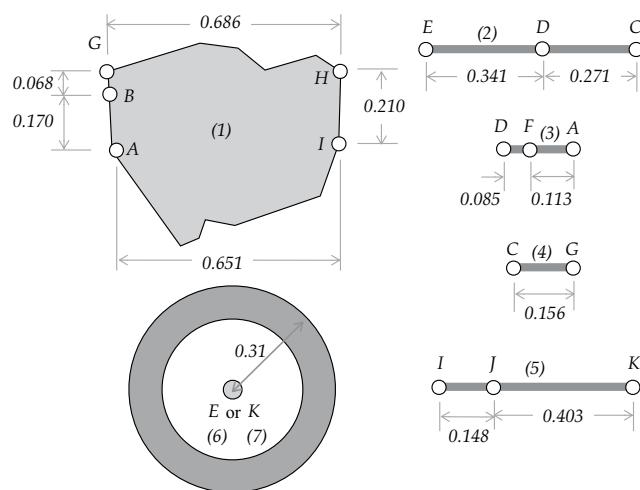
Inertial, Suspension, and Tire Properties (Bike 2)

Body (Index)		Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1)		3.2	
(2)		0.5	
(3)		0.5	
(4)		0.8	
Each wheel		1.2	
Suspension	Free Length (m)	Stiffness (N/m)	Damping (N s/m)
Front		10,000	1,350
Rear		120,000	5,300
Tire	Undeformed Radius (m)	Deformed Radius at Rest (m)	Radial Damping (N s/m)
	0.29		60,000
			2,700

(a)



(b)

**FIGURE 15.11**

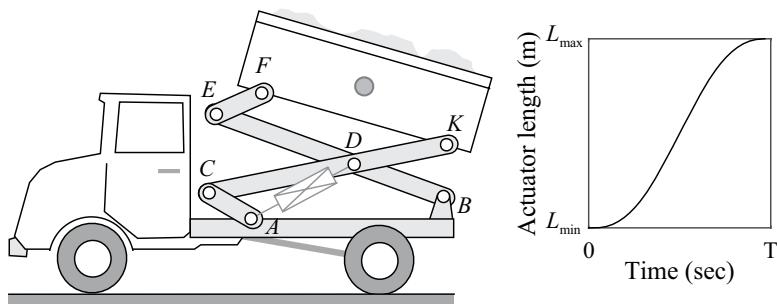
(a) A multibody representation of a motorcycle; (b) the corresponding dimensions in meters.

**TABLE 15.9**

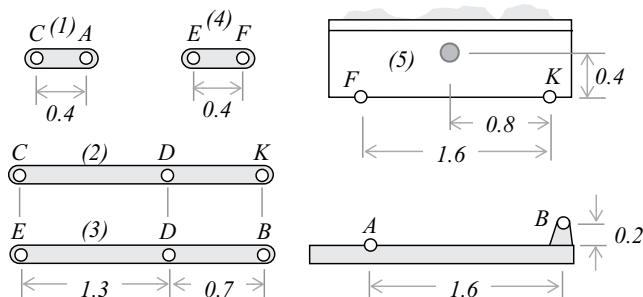
Inertial, Suspension, and Tire Properties for the Motorcycle

Body (Index)		Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1)		230	
(2)		10	
(3)		2	
(4)		2	
(5)		8	
Each wheel		10	
Suspension	Free Length (m)	Stiffness (N/m)	Damping (N s/m)
Front	0.29	75,000	6,700
Rear	0.25	120,000	8,500
Tire	Undeformed Radius (m)	Deformed Radius at Rest	Stiffness (Radial) (SI Units)
	0.31		175,000
			Damping (Radial) (SI units)
			11,000

(a)



(b)

**FIGURE 15.12**

(a) A dump truck; (b) the corresponding dimensions in meters.

**TABLE 15.10**

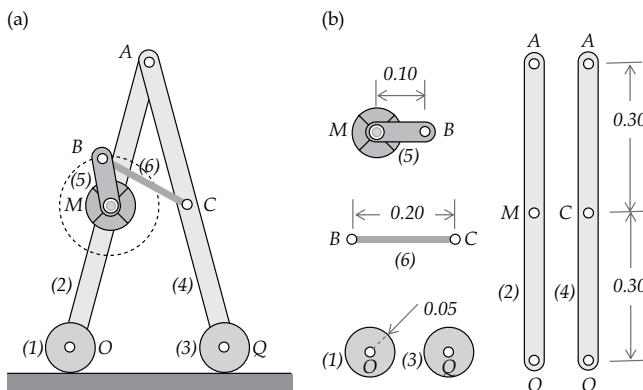
Inertial Data for the Dump Truck

Body (Index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1)	3	0.04
(2)	15	5.0
(3)	15	5.0
(4)	3	0.04
(5)	300	125

## 15.12 Creeping Robot

This example represents a creeping robot as shown in Figure 15.13a.\* A motor acts between links (2) and (5) about the axis of the pin joint  $M$ . The legs of the robot are supported on two discs (wheels), where brakes (not shown) are applied on one disc, whereas the other disc is free to roll without slip; then the sequence is reversed. The switching of the brake from one disc to the other is done for every  $180^\circ$  rotation of the motor. The switching must take place exactly when the legs are at their farthest and closest orientations, which causes the robot to move either forward or backward.

The dimensions for this robot are shown in Figure 15.13b, and the inertial data are listed in Table 15.11. We may assume that the mass centers are at the geometric center of each body except for link (5) where the mass center is at  $M$ .

**FIGURE 15.13**

(a) A creeping robot; (b) the corresponding dimensions in meters.

**TABLE 15.11**

Inertial Properties of the Creeping Robot

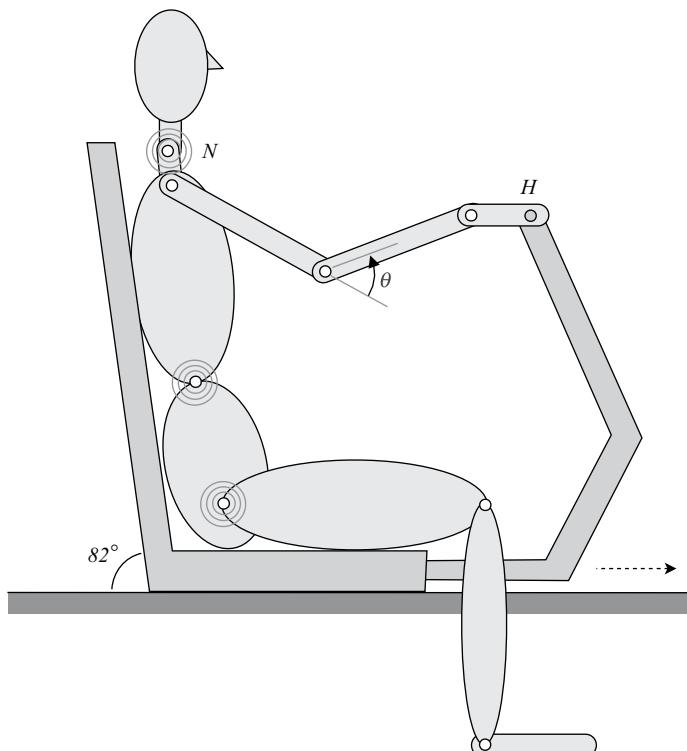
Body (Index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )
(1) (3)	0.3	0.001
(2) (4)	0.2	0.001
(5) + motor	0.6	0.01
(6)	0.1	0.001

\* This model is adopted from a YouTube video ([www.youtube.com/watch?v=Ob59kaOGlbs](http://www.youtube.com/watch?v=Ob59kaOGlbs)) named Robot Gusano.

### 15.13 Sled Test and Belted Dummy

Laboratory and field crash tests are performed in a variety of forms to gain better understanding of the dynamics of a crash to design and manufacture safer vehicles. Valuable data are collected from crash tests using crash dummies. In this example, we attempt to set up a multibody model of a crash dummy and to simulate a sled crash test.\* A multibody representation of the dummy and the seat/sled is shown in Figure 15.14. The simulation scenario would allow the belted dummy and the seat to move forward with a specific speed and then to come to a stop in a short duration of time.

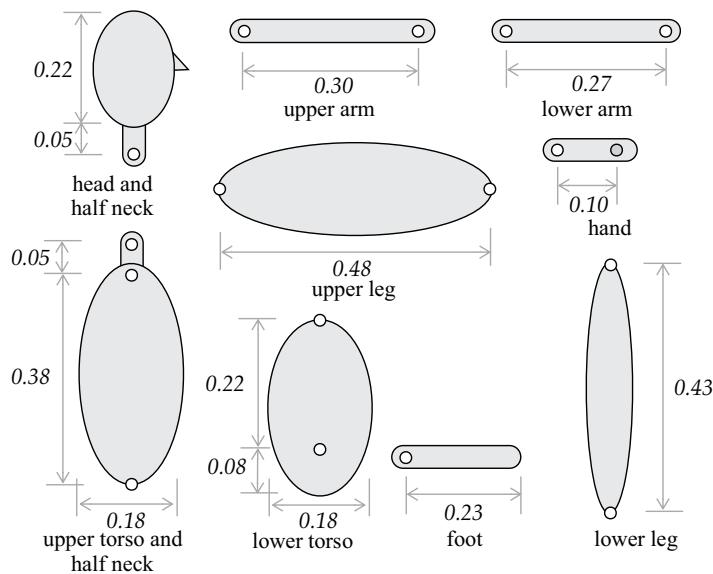
Geometric data for individual bodies of the model are shown in Figure 15.15. It is assumed that pin joints connect the bodies of the dummy. The neck is not considered as a body—it is represented as a pin joint and a torsional spring–damper between the head and the upper torso. In the model, a sliding joint should be provided between the seat and the ground. The steering column is rigidly attached to the seat, and the dummy's hand can be connected to the steering wheel by a pin or a bracket joint. The inertial and spring–damper data are provided in Table 15.12.



**FIGURE 15.14**

A multibody representation of a belted dummy.

\* The concept for this model and some of the data have been adopted from the website: [www.ftss.com/pcat/fem](http://www.ftss.com/pcat/fem) (First Technology Safety Systems, Plymouth, MI, USA)

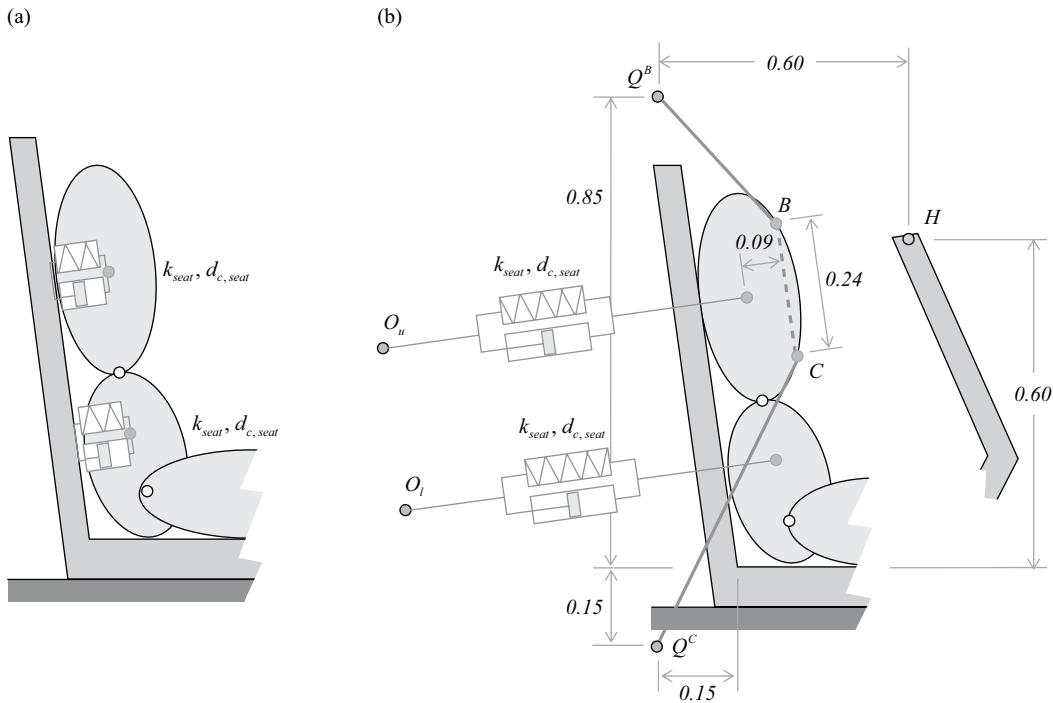


**FIGURE 15.15**  
Individual bodies and attachment pin joints for the dummy.

The upper and lower torsos rest on the seatback. Unilateral spring-dampers, as shown in Figure 15.16a, can provide the necessary compliance. To model the torsos resting on the seatback, the distance of the mass center of each torso from the surface of the seatback should be monitored. When this distance becomes smaller than a known length,  ${}^0L_{seat}$ , the unilateral spring-damper should be activated. Although it is not difficult to determine the distance between each mass center and the seatback, the deformation of each spring-damper can be approximated, with acceptable accuracy, using a simple modeling “trick.”

**TABLE 15.12**  
Inertial and Spring-Damper Data for the Sled and Dummy

	<b>Body</b>	<b>Mass (kg)</b>	<b>Moment of Inertia (kg m<sup>2</sup>)</b>
	Head	5.25	0.04
	Upper torso	18.0	0.35
	Lower torso	23.0	0.13
	Upper arm	2.0	0.02
	Lower arm	1.7	0.01
	Hand	0.6	0.002
	Upper leg	6.0	0.13
	Lower leg	4.3	0.07
	Foot	1.4	0.007
<b>Spring-Damper</b>	<b>Body</b>	<b>Stiffness (SI Units)</b>	<b>Damping (SI units)</b>
	Neck (N)	34.0	1.5
	Seatback	$k_{seat} = 50,000$	$d_{c,seat} = 150$
	Seatbelt	$K_{belt} = 60,000$	$d_{c,belt} = 200$



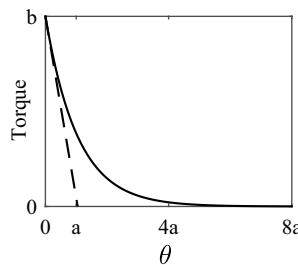
**FIGURE 15.16**  
Modeling of the compliance of the seatback and the seatbelts.

Assume a *long* spring–damper element is connected between a mass center and an imaginary point fixed to the seat at some distance, as shown in Figure 15.16b. The force of this element, regardless of any slight up or down motion of the dummy, remains practically perpendicular to the seatback. The undeformed length of the spring should be adjusted depending on the location of the attachment points.

The seatbelt can be included in the model as another form of unilateral spring–damper element connecting points  $Q^B$ ,  $B$ ,  $C$ , and  $Q^C$  as shown in Figure 15.16b. Points  $Q^B$  and  $Q^C$  are attached to and move with the seat, and points  $B$  and  $C$  are defined on the upper torso. We can assume a spring–damper attached between  $Q^B$  and  $B$ , and another between  $C$  and  $Q^C$ . There is no need to consider a spring–damper between  $B$  and  $C$  since both points are on the same body, and therefore, the distance between them will not change. We monitor the lengths  $L^B$  and  $L^C$ , and if  $L^B + L^C > 0L^B + 0L^C$ , then the spring–damper element should be activated.

For simulating different crash scenarios, we can consider either a simple or a full model. In the simple model, we do not include the lower leg and the foot. We rigidly fix the upper leg to the seat. In the full model, we allow the upper leg to slide relative to the seat, for which we need to include friction.

In a forward dynamic analysis, we must specify the same initial velocity in the horizontal direction to all of the bodies including the seat/sled. The sudden deceleration or impact can be modeled as a continuous contact force analysis as discussed in Section 11.2. The parameters of the contact force model should be adjusted such that the dummy would experience a deceleration of roughly 10g.

**FIGURE 15.17**

Nonlinear characteristics of a torsional spring can limit the relative rotation between two bodies.

We can include other refinements in the model. For example, there should be a limit on the angle between the upper and lower arms. As shown in Figure 15.14, the angle  $\theta$  should not become negative. For this purpose, a torsional spring–damper can be included about the pin joint between the two bodies having nonlinear spring characteristics. A function such as  $(^S)n = be^{-\theta/a}$  can produce this nonlinear behavior as shown in Figure 15.17. Similar refinements can be included between the upper and lower torsos, and the lower torso and the upper leg.

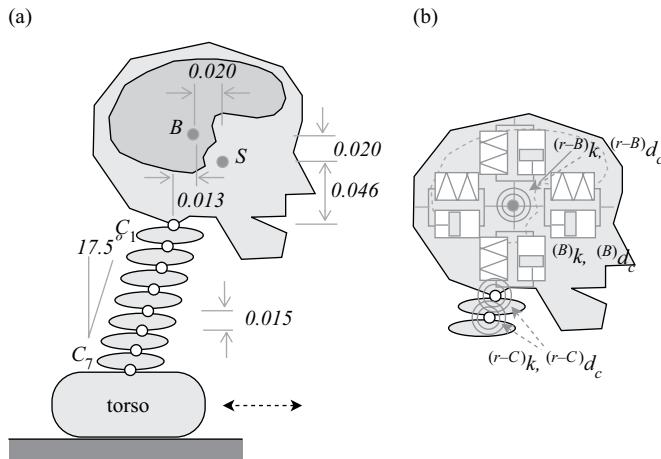
### 15.14 Head and Neck

Numerous analytical and computational models for the head and neck have been developed to study whiplash or other accident related trauma. Most of these models are based on the finite-element method, and a few are based on multibody dynamics. In this example, a planar multibody model for the human head–neck is presented to examine its dynamics under the action of arbitrary forcing conditions.\*

The proposed multibody model, as shown in Figure 15.18, is composed of ten bodies (torso, seven vertebrae–disc combinations, skull, and brain), eight pin joints, one sliding joint, nine torsional spring–dampers, and four point-to-point spring–dampers. The torso is allowed to translate horizontally with respect to the ground (sliding joint). Seven vertebrae–disc combinations attach the torso to the skull by pin joints. Torsional spring–damper elements are placed about each pin joint. The body representing the brain is connected to the skull by spring–damper elements. Proper attachment points on the skull must be chosen to place four point-to-point spring–damper elements. A torsional spring–damper element must also be defined between these two bodies. These five force elements roughly represent the compliance between the brain and the skull. The data for the model are provided in Table 15.13.

In a forward dynamic simulation of this model, the torso could be subjected to either an external applied load or a predefined motion (a driver function) such as a shaking function  $x_{torso} = {}^0x + a \cos(\omega t)$ . We can experiment with different values for the parameters  $a$  and  $\omega$ . Range of values around  $a = 0.1\text{ m}$  and  $\omega = 2\text{ Hz}$  are suggested. If the model is adjusted to

\* This model has been adopted from: Tsai, D-L. and Arabyan, A., "A Multibody Dynamic Model of the Human Head and Neck," Technical Report No. CAEL-91-1, Department of Aerospace and Mechanical Engineering, University of Arizona, Tucson, Arizona, January 1991.



**FIGURE 15.18**  
Multibody representation of the head and neck.

**TABLE 15.13**

Inertial and Spring–Damper Data for the Head and Neck

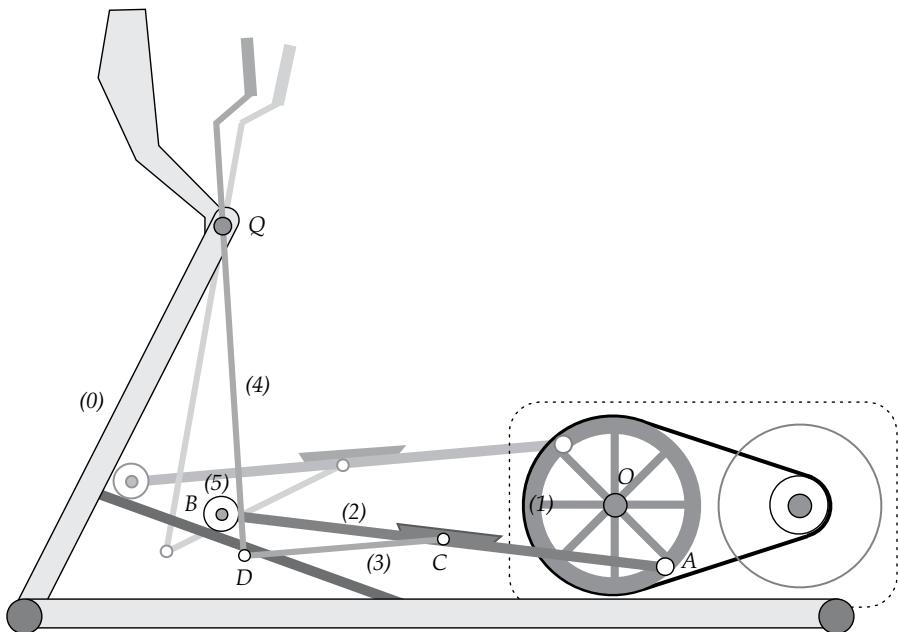
Body	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )	
Skull	2.0	0.03	
Brain	1.4	0.004	
One vertebra	0.25	0.0004	
Torso	50.0	1.2	
Spring-Damper	Bodies	Stiffness (SI units)	Damping (SI units)
	Brain-skull	$(B)k = 8,750$ $(r-B)k = 23$	$(B)d_{(c)} = 90$ $(r-B)d_{(c)} = 0.05$
	Skull-C <sub>1</sub>	$(r-C)k = 271$	$(r-C)d_{(c)} = 0.22$
	Two vertebrae	$(r-C)k = 271$	$(r-C)d_{(c)} = 0.11$
	C <sub>7</sub> -torso	$(r-C)k = 271$	$(r-C)d_{(c)} = 0.11$

represent a young child, a refined version of the model could be used to study a medical trauma known as the shaken baby syndrome.

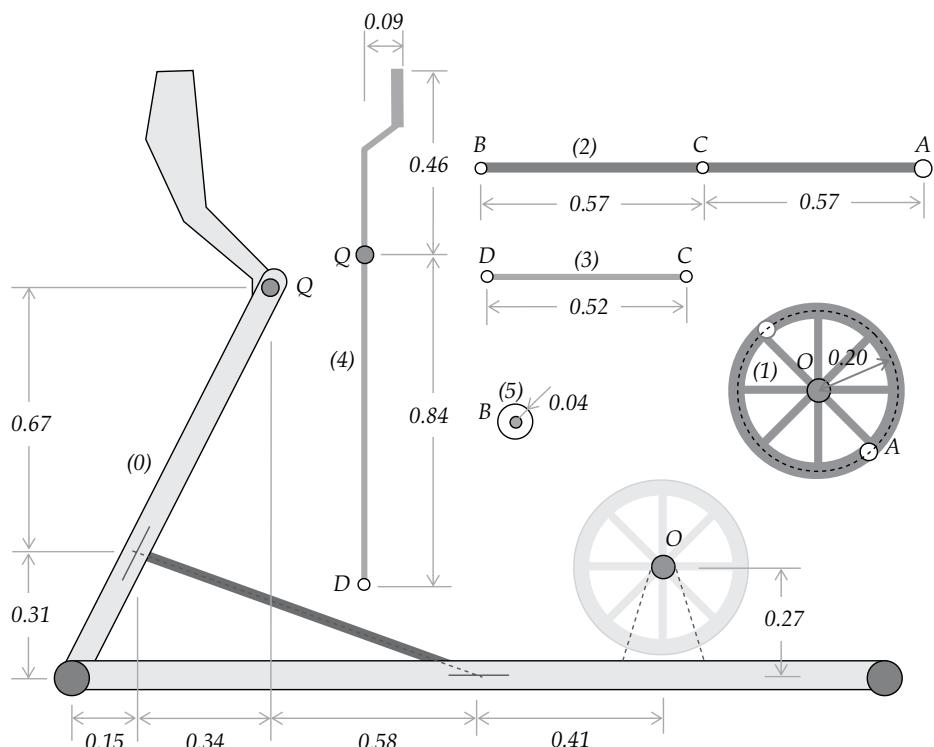
## 15.15 Elliptical Exercise Machine

A popular fitness equipment is the elliptical exercise machine. The machine mainly comprises two slider–crank mechanisms, 180° out-of-phase with one another. A multibody representation of such a machine is shown in Figure 15.19a. Identical left and right footrests cause a flywheel to turn when we shift our weight from one leg to the other. Each footrest sits on a bar connected to the flywheel at one end and slides on an inclined surface through a roller. Two additional links provide support for each hand, which may or may not be included in the model. An adjustable resistive torque is applied to the flywheel via

(a)



(b)

**FIGURE 15.19**

(a) A multibody representation of an exercise machine; (b) the corresponding dimensions in meters.

a system within the dotted box. There is no need to include a multibody representation of this braking system in the model—we can apply the resistive torque directly on the flywheel.

The individual bodies with dimensions for constructing a model are shown in Figure 15.19b. It is left to the reader to assign appropriate inertial values to each component. In the multibody model, we can take advantage of composite joints, such as revolute–revolute and revolute–translational, to reduce the size of the model.

In a forward dynamic analysis, the weight of a person could be shifted from one footrest to the other based on a simple strategy as

$$f_L = -w \cos^2(2\pi t/\tau), f_R = -w \sin^2(2\pi t/\tau)$$

where  $w$  is the weight of the person and  $\tau$  is the time period that can be adjusted. In different simulations, we can experiment with changing: the moment of inertia of the flywheel, the value of the resistive torque, the timing period  $\tau$ , or the inclination angle of the slider.

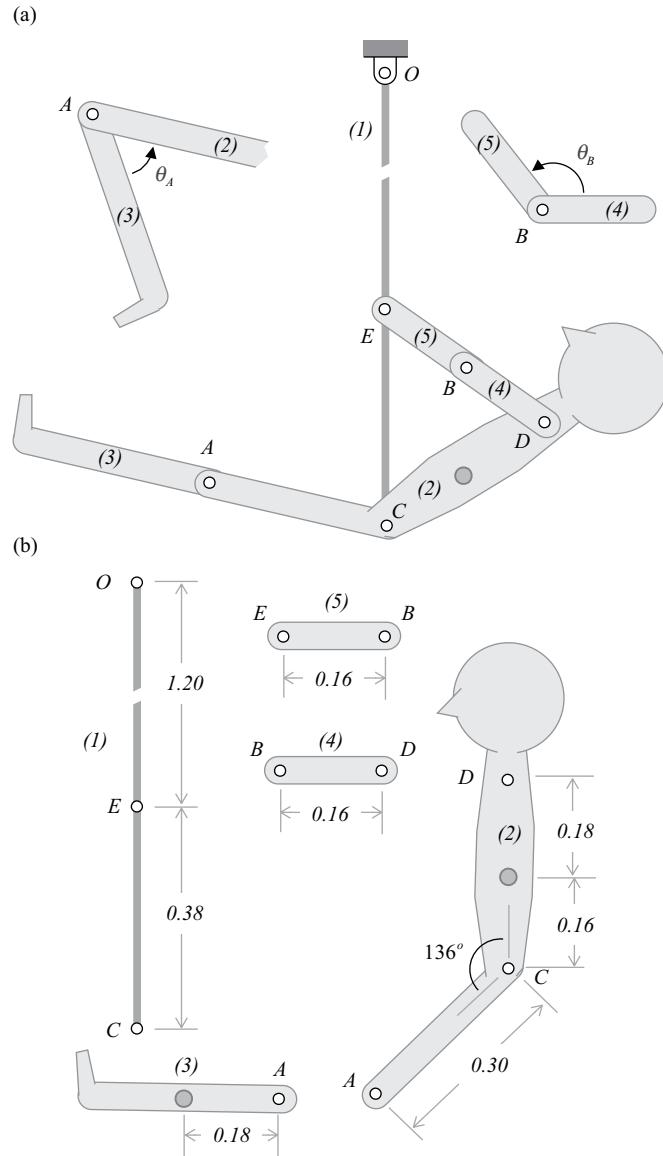
## 15.16 Swing

Children are quick to figure out how to swing higher and higher without being pushed. They intuitively learn to pump their legs and torsos at certain points in the swing cycle. This *body-pump* changes the child's overall mass center with respect to the swing and alters the child's moment of inertia. In this example, we construct a multibody model of a child on a swing and include a body-pump strategy for forward dynamic analyses.

A simple multibody representation of a child on a swing is shown in Figure 15.20a. The model contains five bodies and six pin joints. This represents a system with three DoFs where one is the swinging DoF, and the other two can be used to control the body-pump motion. The dimensions for the model are shown in Figure 15.20b, and a set of inertial data is provided in Table 15.14.

When a child swings forward, the legs and arms are kept straight for maximizing the moment of inertia. We refer to this as the *straight leg-arm* configuration, as shown in Figure 15.20a. When the child swings backward, the legs and arms are tucked in to minimize the moment of inertia. We refer to this as the *bent leg-arm* configuration. In our multi-body model, we need to have two driver functions to switch the orientation of the legs and arms between the two configurations. For this purpose, we define two angles,  $\theta_A$  and  $\theta_B$ . These angles could be controlled and varied from a minimum to a maximum value, and vice versa, using the function “type b” in the DAP \_ BC program, Chapter 8 (refer to the program manual). The change of the angle could take place in a short but specified time period, for example,  $\Delta t = 0.5$  s.

One strategy to determine when to initiate the change of the angles would be to monitor the angular velocity and acceleration of the swing, body (1). When the swing is moving forward ( $\dot{\phi}_1 < 0$ ), the child should be in the straight leg-arm configuration. When the swing is moving backward ( $\dot{\phi}_1 > 0$ ), the child should be in the bent leg-arm configuration. When the swing reaches the highest orientation at either end; i.e., when  $\dot{\phi}_1 = 0$ , it is time to initiate a change. During the integration of the equations of motion, such exact time to initiate a change can be determined using MATLAB® function event (refer to Section 13.4).

**FIGURE 15.20**

(a) A multibody representation of a child on a swing; (b) the corresponding dimensions in meters.

### 15.17 Satellite Panels

Some satellites when in orbit deploy certain instruments such as antennas and solar panels. This example is a planar representation of the opening process of solar panels of a satellite\*. The satellite with its panels completely unfolded is shown in Figure 15.21. There

\* This example is based on the SAE-Antenna developed by Dornier GmbH for the European Research Satellite ESR-1. All dimensions and other data have been revised.

**TABLE 15.14**

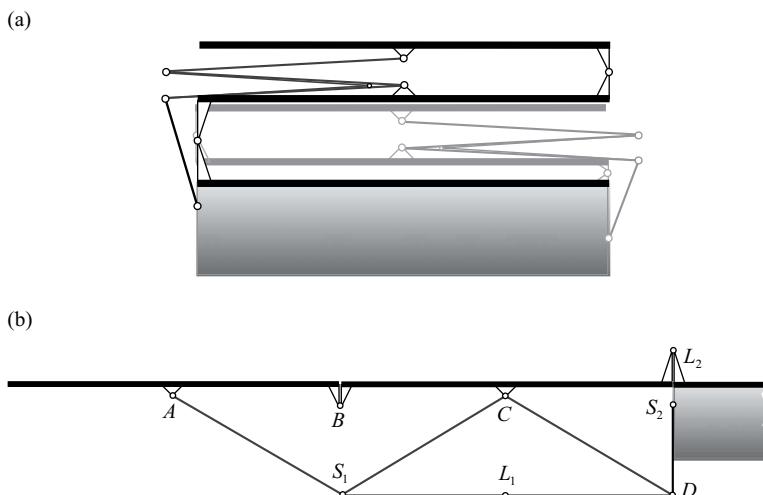
Inertial and Angle Data for the Swing Application

Body (index)	Mass (kg)	Moment of Inertia ( $\text{kg m}^2$ )	
(1)	3.0	0.50	
(2)	30.0	2.1	
(3)	2.5	0.04	
(4)	1.8	0.007	
(5)	1.4	0.006	
Angles	$\theta$	Minimum	Maximum
	$A$	$20^\circ$	$175^\circ$
	$B$	$30^\circ$	$175^\circ$

**FIGURE 15.21**

Solar panels of a satellite in the unfolded state.

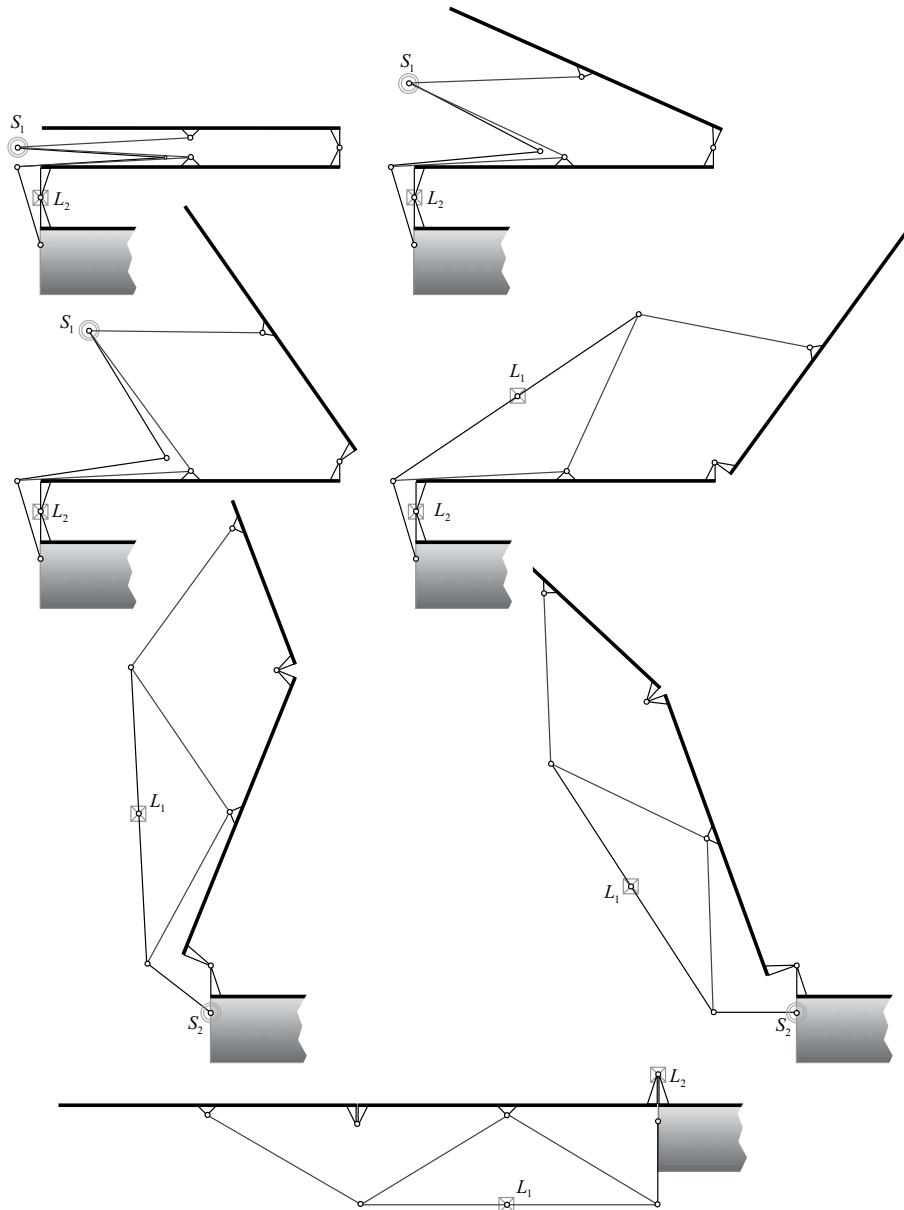
are five panels—one stationary in the middle, which is attached rigidly to the body of the satellite, and two deployable panels on each side. The panels are supported by slender links and joints, which are folded and stowed during the launch, and unfolded when in orbit. The two folded panels of one side fold over the two folded panels of the other side, as shown in Figure 15.22a, minimizing the storage space. When panels are unfolded, as shown in Figure 15.22b for only one side, the links form a truss to support the panels as a structure. In this figure, the pin joints are labeled as  $A, B, C, D, S_1, S_2, L_1$ , and  $L_2$ . The joints  $S_1$  and  $S_2$  contain torsional springs that are compressed when the panels are folded, but become uncompressed when the panels are completely unfolded. These torsional springs

**FIGURE 15.22**

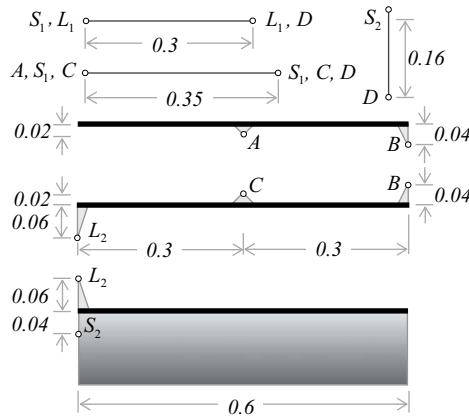
(a) Folded panels and (b) unfolded panels on one side.

provide the necessary torques to unfold the panels, which can also be locked or unlocked. The joints  $L_1$  and  $L_2$  can be locked and unlocked switching between pin and rigid joints.

The unfolding process on each side is done in two phases, as shown in Figure 15.23. In the first phase, although joint  $L_2$  is locked, the loaded spring at  $S_1$  is released and the unfolding of the first panel begins. When the system comes to rest, due to the presence of damping, the joint at  $L_1$  is locked. The second phase begins by unlocking the joint at  $L_2$  and activating the loaded spring  $S_2$ , which starts to unfold the second panel. After the system comes to rest, the joint at  $L_2$  is locked. At this point, the mechanisms become a



**FIGURE 15.23**  
Unfolding process of the panels on one side.

**FIGURE 15.24**

Dimensions for the panels and the link in meters.

truss structure supporting the panels. The same process is repeated for the panels on the other side.

The dimensions for the panels and the links are provided in Figure 15.24. In a model, we may consider the mass of each panel to be 1.0 kg and for the links to be 0.1 km/m. For a dynamic simulation, we can assume the stiffness of 10.0–50.0 N m/rad and the damping coefficient of 5.0–10.0 N m s/rad. Damping is also present in the pin joints, which could be experimented with in a model.

In simulating this system with a program such as `DAP _ BC` of Chapter 8, we may consider two approaches:

- We can split the bodies into two sets: set up a model for phase 1 and a second model for phase 2. The coordinates of the bodies at the end of simulation of phase 1 could be used to determine the initial conditions for the coordinates of the bodies for the second model.
- We can set up one complete model and take advantage of MATLAB function `event`, which would require some programming adjustment in `DAP _ BC`.

---

## Appendix A: L-U Factorization

---

In Section 2.3.2, we briefly discussed how to solve a set of linear algebraic equations in the form

$$\mathbf{A}\mathbf{x} = \mathbf{c} \quad (\text{A.1})$$

In MATLAB®, we can use either matrix inversion  $\mathbf{x} = \text{inv}(\mathbf{A})^* \mathbf{c}$  or the backslash operator  $\mathbf{x} = \mathbf{A}\backslash\mathbf{c}$ . There are other computational methods that can be applied to solve Eq. (A.1). Gaussian elimination, Gauss–Jordan reduction, L-U factorization, and Q-R decomposition are examples of such methods. For a detailed description of these methods, interested reader should refer to any textbook on numerical methods. In this appendix, we briefly discuss the L-U factorization method, which provides several useful features in computational kinematics and dynamics.

The Gaussian elimination or the Gauss–Jordan reduction operates on matrix  $\mathbf{A}$  and array  $\mathbf{c}$  simultaneously while transforming  $\mathbf{A}$  to a triangular or diagonal matrix. Then a back (or forward) substitution on the transformed matrix provides the solution. However, the L-U factorization operates on the  $\mathbf{A}$  matrix without any consideration for the array  $\mathbf{c}$ .

For any nonsingular matrix  $\mathbf{A}$ , it can be proven that there exists an *upper triangular matrix*  $\mathbf{U}$  with nonzero diagonal elements and a *lower triangular matrix*  $\mathbf{L}$  with unit diagonal elements such that

$$\mathbf{A} = \mathbf{LU} \quad (\text{A.2})$$

The process of factorizing  $\mathbf{A}$  into the product  $\mathbf{LU}$  is called L-U factorization.

Once the  $\mathbf{L}$  and  $\mathbf{U}$  matrices are obtained, the operation  $\mathbf{Ax} = \mathbf{LUx} = \mathbf{c}$  is performed in two steps:

$$\mathbf{Ly} = \mathbf{c} \quad (\text{A.3})$$

and

$$\mathbf{Ux} = \mathbf{y} \quad (\text{A.4})$$

Equation (A.3) is first solved for  $\mathbf{y}$ , and Eq. (A.4) is then solved for  $\mathbf{x}$ .

### Example A.1

Solve the following set of linear algebraic equations using L-U factorization:

$$\left[ \begin{array}{ccc} 2 & -1 & 3 \\ -4 & 1 & -8 \\ 6 & -2 & 13 \end{array} \right] \left[ \begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right] = \left[ \begin{array}{c} 7 \\ -18 \\ 27 \end{array} \right] \quad (\text{A.5})$$

***Solution***

An L-U factorization (the method is not discussed here) without interchanging rows or columns (called pivoting) yields the following **L** and **U** matrices:

$$\left[ \begin{array}{ccc} 2 & -1 & 3 \\ -4 & 1 & -8 \\ 6 & -2 & 13 \end{array} \right] \Rightarrow \left[ \begin{array}{ccc} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & -1 & 1 \end{array} \right] \left[ \begin{array}{ccc} 2 & -1 & 3 \\ 0 & -1 & -2 \\ 0 & 0 & 2 \end{array} \right]$$

We can now perform a forward substitution to solve the following set of equations:

$$\left[ \begin{array}{ccc} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & -1 & 1 \end{array} \right] \left\{ \begin{array}{c} y_1 \\ y_2 \\ y_3 \end{array} \right\} = \left\{ \begin{array}{c} 7 \\ -18 \\ 27 \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} y_1 \\ y_2 \\ y_3 \end{array} \right\} = \left\{ \begin{array}{c} 7 \\ -4 \\ 2 \end{array} \right\}$$

Then a back substitution in solving the following set of equations yields the unknowns:

$$\left[ \begin{array}{ccc} 2 & -1 & 3 \\ 0 & -1 & -2 \\ 0 & 0 & 2 \end{array} \right] \left\{ \begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right\} = \left\{ \begin{array}{c} 7 \\ -4 \\ 2 \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right\} = \left\{ \begin{array}{c} 3 \\ 2 \\ 1 \end{array} \right\}$$

**Example A.2**

Perform L-U factorization on the coefficient matrix of Example A.1 in MATLAB.

***Solution***

```
A = [2 -1 3;-4 1 -8; 6 -2 18];
[L, U] = lu(A) .....
```

L =	0.3333	1.0000	0
	-0.6667	1.0000	1.0000
	1.0000	0	0
U =	6.0000	-2.0000	13.0000
	0	-0.3333	-1.3333
	0	0	2.0000

The MATLAB's function `lu` provides the factorized matrices in permuted form due to pivoting. We make the observation that if we move the third row of the **L** matrix to the first row, **L** is definitely a lower triangular matrix. We can verify the results by testing **L**\* **U** to obtain the original matrix.

Function `lu` can also provide the permutation order:

```
A = [2 -1 3;-4 1 -8; 6 -2 18];
[L1, U, p] = lu(A, 'vector')
...
L1 =
1.0000 0 0
0.3333 1.0000 0
-0.6667 1.0000 1.0000
U =
...
p =
3 1 2
```

We note that the L matrix appears as a lower triangular, and the permutation array indicates that the third row of the original matrix has been moved to the top. The product testing  $L1^*U$  results in the permuted original matrix.

In any matrix factorization, at some steps of the process, the elements of a row or column are divided by the value of the so-called pivot element. To lower the amount of truncation error in the process, it is better for the pivot element not to be too small (absolute value). Therefore, the pivot element is selected to be the largest element of a row, or a column, or the entire matrix. This process is called pivoting by reordering the rows, or columns, or both. The following example shows how pivoting can be useful in computational kinematics.

### Example A.3

Consider the following two equations where  $\mathbf{x} = \{3 \ 2 \ 1\}'$  is one of the infinite number of solutions:

$$\left[ \begin{array}{ccc} 2 & -1 & 3 \\ -4 & 1 & -8 \end{array} \right] \left\{ \begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right\} = \left\{ \begin{array}{c} 7 \\ -18 \end{array} \right\} \quad (\text{A.6})$$

Assume that we need to pick one of these three unknowns as the independent variable, having a known value, and solve the equations for the other two unknowns. In the following exercise, we demonstrate how pivoting can be used to identify the best choice of the independent variable (out of three) in order to lower the truncation error.

Let us assume that the known value of the independent variable contains an error  $e = 0.1$ . We solve the equations three times, each time with a different independent variable containing the error. In each case, we compare the resulting answer to the correct answer by computing the norm of the total error.

In the first trial, we choose  $x_1 = 3.0 + e$  as the independent variable.

```
B = [2 -1 3;-4 1 -8];
A1 = [B; 1 0 0];
e = 0.1;
c1 = [7 -18 3.0+e]';
x1 = A1\c1 ..... .
norm1 = norm(x1 - x) .....
```

```
x1 =
3.1000
2.0800
0.9600
norm1 =
0.1342
```

Next we pick  $x_2 = 2.0 + e$  as the known independent variable.

```
A2 = [B; 0 1 0]; c2 = [7 -18 2.0+e]';
x2 = A2\c2 ..... .
norm2 = norm(x2 - x) .....
```

```
x2 =
4.6250
2.1000
-0.0500
norm2 =
1.9373
```

Finally, we pick  $x_3 = 1.0 + e$  as the known independent variable.

<pre>A3 = [B; 0 0 1]; c3 = [7 -18 1.0+e] ' x3 = A3\c3 .....</pre>	<pre>x3 = 1.7500 -0.2000 1.1000 norm3 = 2.5323</pre>
---	--

If we compare the norms from the three answers, we find that choosing  $x_1$  as the independent variable yields the smallest overall error. We could have found the best choice for this example by performing an L-U factorization on the transpose of the coefficient matrix as

<pre>[LB, UB, pB] = lu(B', 'vector') .....</pre>	<pre>pB = 3     2     1</pre>
--	-------------------------------

The last element of the permutation array refers to the first variable being the best choice. Further explanation is provided below.

When a matrix is rectangular with more columns than rows, such as the coefficient matrix in Eq. (A.6), an L-U factorization yields an upper triangular matrix that contains a residual portion. For example,

$$\left[ \begin{array}{ccc} 2 & -1 & 3 \\ -4 & 1 & -8 \end{array} \right] \Rightarrow \left[ \begin{array}{cc|c} 1 & 0 & 2 \\ -2 & 1 & 0 \end{array} \right] \left[ \begin{array}{cc|c} 2 & -1 & 3 \\ 0 & -1 & -2 \end{array} \right]$$

The third column of the  $2 \times 3$   $\mathbf{U}$  matrix is the residual portion. If we perform L-U factorization (or Gaussian elimination) with pivoting that includes column interchange, the index of the columns that ends up in the residual portion of  $\mathbf{U}$  refers to the best choice of variable(s) that should be considered independent. Since function `lu` in MATLAB performs pivoting with row interchange, we can still use the function but perform L-U factorization on  $\mathbf{B}'$ .

A similar process can be performed on a coefficient matrix to identify any redundant rows (or columns). In the case of redundancy, the row(s) of the  $\mathbf{U}$  matrix associated with the redundant rows will be zeros. The permutation array identifies the row(s) that is (are) redundant.

#### Example A.4

Consider the following  $3 \times 4$  matrix:

$$\mathbf{D} = \left[ \begin{array}{cccc} 2 & -1 & 3 & 1 \\ -4 & 1 & -8 & 2 \\ -2 & 0 & -5 & 3 \end{array} \right] \quad (\text{A.7})$$

We want to identify whether the rows are independent.

We perform L-U factorization using function `lu`.

```
D = [2 -1 3 1;-4 1 -8 2; ...  
     -2 0 -5 3];  
[L, U, p] = lu(D,'vector') ....  
  
L =  
    1.000      0      0  
   -0.500    1.000      0  
    0.500    1.000    1.000  
  
U =  
   -4.000    1.000   -8.000    2.000  
     0   -0.500   -1.000    2.000  
     0       0       0       0  
  
p =  
    2       1       3
```

We note the third row of  $U$  is all zeros, and the permutation array points to the third row of the original matrix.

We can perform this process on the Jacobian matrix of the constraints of a multibody system to identify whether all the constraints are independent.

---

## *Appendix B: Dynamic Analysis Program: Body Coordinates (DAP\_BC)*

---

In Chapter 8, we discussed several examples of multibody systems for modeling and simulation of their dynamic responses with the dynamic analysis program based on the body-coordinate formulation (DAP\_BC). The user manual for this program is provided in this appendix that can be downloaded from the textbook's website.

# Appendix B

## DAP\_BC User Manual

(December 2018)

The Dynamic Analysis Program: Body Coordinates (DAP\_BC) is a general-purpose program that is not developed around a specific problem—it can simulate the dynamics of a variety of problems based on the description and the data provided by the user. The program contains most of the features that have been discussed in Chapter 7 of the textbook, such as kinematic joints and force elements. A user can modify the program to include new features and capabilities. The program is capable of performing kinematic, inverse dynamic, and forward dynamic analyses by integrating the equations of motion (refer to Chapters 12 and 13). The program can also solve static and static equilibrium problems, and correct initial conditions on the coordinates and velocities prior to performing a forward dynamic analysis (refer to Chapter 14). An animation program that provides a simple animated stick drawing of the dynamic response of the analyzed model accompanies the analysis program.

### B.1 USING DAP\_BC

The program DAP\_BC and the accompanying examples are all in one folder named DAP\_BC followed by the date of its last revision. As shown in Figure B.1(a), the contents of this folder are three script M-files and two other folders. The folder named Models contains several examples of simple multibody systems, as shown in Figure B.1(b). One of these models, SP (Sliding Pendulum), is discussed in this manual in detail as a guide in learning how to use the program. Some of the other models are discussed in Chapter 8 of the textbook. If we construct any new models, they should be saved inside this folder.

The script M-file `dap` is the main program to execute in order to simulate the response of a system. The program prompts the user with a few questions prior to performing a simulation. After a simulation is completed, the user may execute the script M-file `anim` to visualize a crude animation of the simulated model. Following a simulation, we can also execute the script M-file `post` for post-processing of the simulated response. This script transforms the results into a more recognizable set of arrays for further analyses or plots.

The folder Formulation contains other subfolders, script and function M-files. These are the body-coordinate formulations from Chapter 7 for constructing and evaluating constraints, Jacobian sub-matrices, forces, mass matrix, etc. To simulate the response of a multibody system, using the existing capabilities of the program, there is no need for a user to revise anything in this folder.

It is highly recommended to make the folder DAP\_BC the “Current Folder” in MATLAB, as indicated in Figure B.1. This facilitates setting the “paths” to different folders and M-files. The program will then automatically set all the necessary paths.

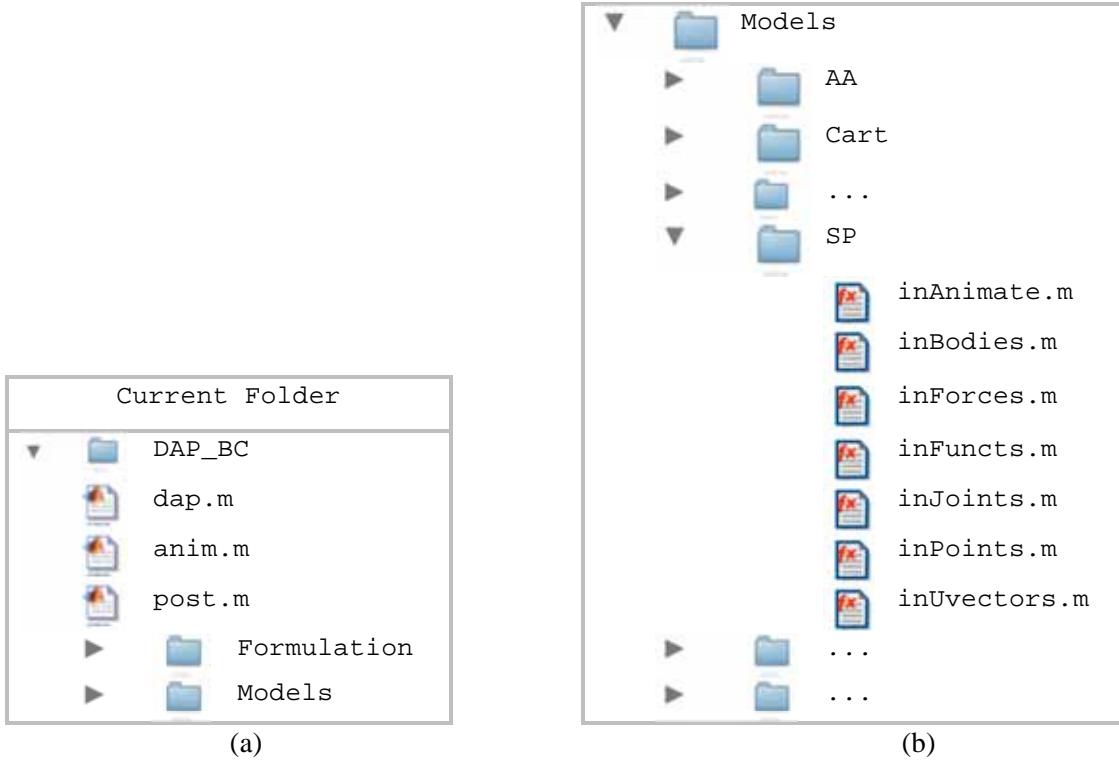


Figure B.1 Folders and script files of DAP\_BC.

### B.1.1 Example

This sliding-pendulum example, shown in Figure B.2(a), was considered for FBD construction in Section 6.1.3. Body (1), the slider, is connected to the ground by a frictionless sliding joint, and to body (2), the pendulum, by a pin joint. A spring is attached between the ground and body (1), and the gravity acts on the system. All the necessary data are listed in Example 6.3. The body-coordinate model of this system is provided in the subfolder SP, which resides inside the folder Models. To simulate the response of this system, we perform the following steps.

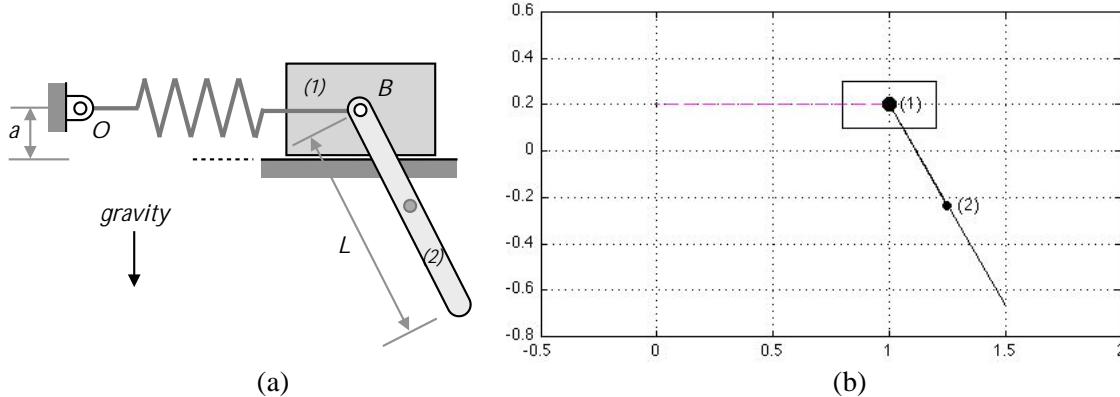


Figure B.2 (a) Sliding-pendulum example and (b) its animated presentation.

In the Command Window we type dap and then respond to each question, as appears in the right column.

>> dap Which folder contains the model? .....	SP
--	----

We entered SP for Sliding Pendulum. Next, the program asks if we want the program to correct the initial conditions on the coordinates and velocities. If we are certain that the provided initial conditions satisfy the position and velocity constraints, we may answer no, otherwise we should respond yes.

Do you want to correct the initial conditions? [(y)es/(n)o] ....	y
---	---

Since we asked the program to correct the initial conditions, the corrected set of initial conditions will be reported:

Corrected coordinates
x                y                phi
1                0.2                0
1.25            -0.23301        0.5236
Corrected velocities
x-dot            y-dot            phi-dot
0                0                0
0                0                0

Next, the program asks for the final time of integration (the starting time is “0”) and the reporting time steps of the results (all in seconds).

Final time = ? .....	4
Reporting time-step = ? .....	0.02

At this point, the program starts integrating the equations of motion from “0” to “4” seconds. To keep the user informed of the progress, the program reports the integration time every 100 function-evaluations. (A function evaluation means that the equations of motion have been constructed and solved for the accelerations.) At the completion of integration, the program reports the number of function evaluations. This information is reported as a measure for the efficiency (or inefficiency) of the method of solution.

0
0.4605
1.1473
1.8110
2.5231
3.1091
3.7231
Number of function evaluations = 655
>>

At the completion of the simulation, dap saves the simulated coordinates and velocities for each moving body from  $t = 0$  to  $t = 4.0$  at every  $\Delta t = 0.02$  seconds. We can run either the script anim.m or post.m to view or process the response.

To view an animation of the results, in the Command Window we type:

```
>> anim
```

A graphical window will display a crude representation of the system in its initial configuration, as shown in Figure B.2(b). The body centers, the pin joints, and some of the defined points are marked on the plot. The spring is displayed in dashed-line. Pressing any keys on the keyboard initiates the animation of the response.

We can also execute the M-file `post` to recover some the results that were not reported by the integration process, such as the accelerations and Lagrange multipliers. The program `post` organizes the results in separate recognizable arrays (these arrays will be discussed later in this manual). We can, for example, plot the angle of body (2) versus time by typing in the Command Window the followings:

```
>> plot(T,p(:,2), 'k')
>> grid on
```

The resulting plot is shown in Figure B.3.

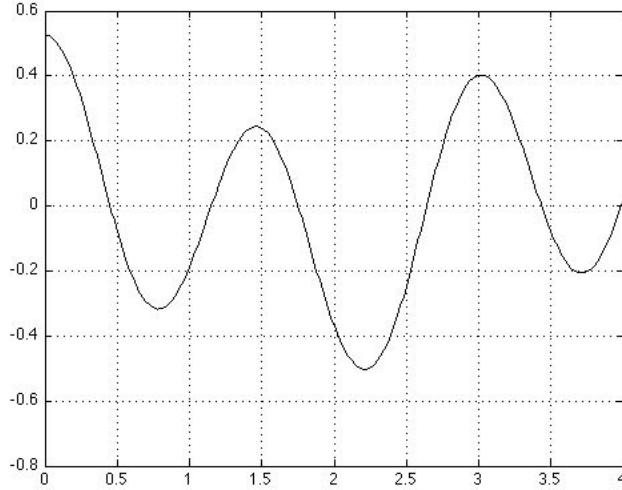


Figure B.3 Angle of the pendulum versus time.

Through this example, we demonstrated that it is simple to simulate the dynamic response of an existing model. We can repeat the same steps to simulate the response of other models that are already available in the folder `Models`. Next we describe how a model is constructed.

## B.2 CONSTRUCTING A MODEL

A multibody model is organized in seven function M-files as depicted in Figure B.1(b). If we look at the existing examples in the folder `Models`, we find that all of them contain exactly the same named function M-files. Some models may have an extra M-file named `user_force`, which will be discussed later in this section.

The function M-files provide data for the bodies, points, unit vectors, joints, forces, and any analytical functions that may be needed (for example, for a driver motor). Each set of data must be organized in a specific form, using MATLAB's `structure` array, or `struct`. The structure for each element (a body, a joint, etc.) contains certain fields, where some of the fields require data to be provided by the user. The required data and the structure of each element are described in the followings, using the sliding pendulum example.

### Example B.1

To begin constructing a model for the sliding pendulum, we first separate the bodies, assign indices, attach to each a body-fixed frame, and define a global  $x$ - $y$  frame. As shown in Figure B.4, the two moving bodies are marked as (1) and (2). We have identified three points as  $O_0$ ,  $B_1$  and  $B_2$  that are needed for the joints and the spring attachment points. We have assigned indices to these points as [1], [2], [3] (the order of numbering is up to us). We also need two unit vectors for the translational joint:  $\vec{u}_0$  on the ground and  $\vec{u}_1$  on the slider. These unit vectors are numbered [[1]] and [[2]]. We are now ready to set up the M-files for this model. The constant dimensions are  $a = 0.2$  m and  $L = 1.0$  m.

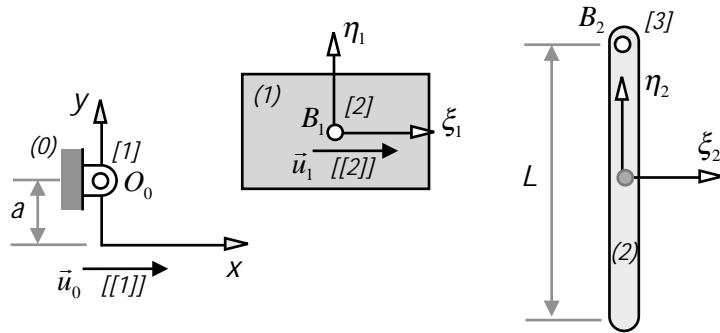


Figure B.4 Individual bodies and the assigned indices for the sliding pendulum example.

**M-file inBodies:** In this file we define the mass, moment of inertia, and initial conditions of the coordinates and velocities for all the bodies, except for the ground. To construct this file, we must follow the structure listed in Table B.1. There are six fields in this structure that need data from the user. The comment for each field should state clearly what the data is: for example, ‘m’ is mass and ‘J’ is moment of inertia. Each field also contains a default value; for example, if we do not state the mass for a body, the program will use the default value of 1.0 unit. The entries for the fields ‘m’ and ‘J’ will not be altered by the program. However, the values for other fields, for example ‘r’, will be updated by the program during a simulation, as the bodies move. There are additional fields in this structure that are not listed in this table—the program will use those fields to save various data during a simulation.

Table B.1

```
function Body = Body_struct
Body = struct ( ...
    'm' , 1 , ... % mass
    'J' , 1 , ... % moment of inertia
    'r' , [0;0] , ... % x and y coordinates
    'p' , 0 , ... % angle phi
    'r_d' , [0;0] , ... % time derivative of x and y
    'p_d' , 0 , ... % time derivative of phi
    ...
);
```

### Example B.1 continued (DAP\_BC/Models/SP)

The following data for the sliding pendulum are provided in the function M-file inBodies.

$m_1 = 5.0$   
 $J_1 = 4.0$   
 $x_1 = 1.0, y_1 = 0.2$   
 $\phi_1 = 0$   
  
 $m_2 = 2.0$   
 $J_2 = 0.2$   
 $x_2 = 1.25, y_2 = -0.233$   
 $\phi_2 = \pi / 6$

```
function inBodies
    include_global

    B1 = Body_struct;
    B1.m = 5.0;
    B1.J = 4.0;
    B1.r = [1.0; 0.2];

    B2 = Body_struct;
    B2.m = 2.0;
    B2.J = 0.2;
    B2.r = [1.25; -0.233];
    B2.p = pi/6;

    Bodies = [B1; B2];
```

In this M-file we have created two bodies, arbitrarily named B1 and B2, which have been assigned the structure named Body\_struct. We have not stated an angle for the first body or velocities for either body since their values are zero (defaults). The two created bodies are saved in the array Bodies.

The statement `include_global` must appear in this and all other input M-files. Through the global statements in the script “`include_global`”, the array `Bodies` will become available to the rest of the program. From this point on, `Bodies(1)` will refer to the first body saved in this array, which is body (1), and `Bodies(2)` will refer to body (2). For example, we can access the coordinates of body (2) from any M-file that contains the `include_global` statement, by stating `Bodies(2).r`. In general, the order in which the defined bodies are placed in the array `Bodies` becomes body indices for a model.

**M-file `inPoints`:** In this file we provide body-fixed coordinates,  $\mathbf{x}_i^P$ , for all the defined points, including the points on the ground. The data must be entered according to the structure listed in Table B.2.

Table B.2

```
function Point = Point_struct
Point = struct ( ...
    'Bindex' , 0      , ... % body index
    'sPlocal' , [0;0] , ... % body-fixed (local) coordinates
    ...
);
```

### Example B.1 (cont.)

The following body-fixed coordinates are extracted from the figure.

$$\begin{aligned}\mathbf{x}_0^O &= \begin{Bmatrix} 0 \\ 0.2 \end{Bmatrix} \\ \mathbf{x}_1^B &= \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \\ \mathbf{x}_2^B &= \begin{Bmatrix} 0 \\ 0.5 \end{Bmatrix}\end{aligned}$$

Based on the data, we construct the function M-file `inPoints`. The defined points are saved in the array `Points`, which is the output of this file. The order in which the defined points are placed in this array becomes point indices for this model.

```
function inPoints
    include_global

P1 = Point_struct;
P1.Bindex = 0;
P1.sPlocal = [0; 0.2];

P2 = Point_struct;
P2.Bindex = 1;
P2.sPlocal = [0; 0];

P3 = Point_struct;
P3.Bindex = 2;
P3.sPlocal = [0; 0.5];

Points = [P1; P2; P3];
```

**M-file `inUvectors`:** In this file we provide the body-fixed components,  $\mathbf{u}_i$ , of the defined unit vectors, including the unit vectors on the ground. The data for the vectors must be entered according to the structure listed in Table B.3.

Table B.3

```
function Unit = Unit_struct
Unit = struct ( ...
    'Bindex' , 0      , ... % body index
    'ulocal' , [1;0] , ... % body-fixed (local) components
```

	);
--	----

**Example B.1 (cont.)**

The following body-fixed components are defined for the two unit vectors in the model:

$$\mathbf{u}_0 = \begin{Bmatrix} 1.0 \\ 0 \end{Bmatrix}, \quad \mathbf{u}_1 = \begin{Bmatrix} 1.0 \\ 0 \end{Bmatrix}$$

Based on the data, we construct the function M-file **inUvectors**.

The output of this file is the array **Uvectors**.

```
function inUvectors
    include_global

U1 = Unit_struct;
U1.Bindex = 0;
U1.ulocal = [1.0; 0];

U2 = Unit_struct;
U2.Bindex = 1;
U2.ulocal = [1.0; 0];

Uvectors = [U1; U2];
```

**M-file inForces:** In this file we define the forces that act on the bodies, such as gravity, spring-dampers, or other force elements. Different force elements require different type of data; therefore the structure for forces, as is listed in Table B.4, contains more fields than those for bodies or points.

Table B.4

<pre><b>function</b> Force = Force_struct Force = struct ( ...     'type' , 'ptp' , ... % element type: ptp, weight, f, T, etc.     'iPindex' , 0 , ... % index of the point on body (i)     'jPindex' , 0 , ... % index of the point on body (j)     'iBindex' , 0 , ... % index of body (i)     'jBindex' , 0 , ... % index of body (j)     'k' , 0 , ... % spring stiffness     'L0' , 0 , ... % undeformed length     'theta0' , 0 , ... % undeformed angle     'dc' , 0 , ... % damping coefficient     'f_a' , 0 , ... % constant actuator force     'T_a' , 0 , ... % constant actuator torque     'gravity' , 9.81 , ... % gravitational constant     'wgt' , [0;-1] , ... % gravitational direction     'flocal' , [0;0] , ... % constant force in local frame     'f' , [0;0] , ... % constant force in x-y frame     'T' , 0 , ... % constant torque     'iFunct' , 0 , ... % analytical function index );</pre>
---

The first field for a force element is its **type**. The available force elements types are:

- Gravitational force (**.type = 'weight'**)
- Point-to-point spring-damper-actuator (**.type = 'ptp'**)
- Rotational spring-damper-actuator (**.type = 'rot-sda'**)
- Constant force described in body-fixed frame (**.type = 'flocal'**)
- Constant force described in x-y frame (**.type = 'f'**)
- Constant torque (**.type = 'T'**)
- User supplied forces (**.type = 'user'**)

Each type requires different set of data as described in the followings:

.type	Required entries	Comments
'ptp'	iPindex jPindex k L0 dc f_a	There is no need to provide body indices. Since the point indices have already been provided, the program can determine the corresponding body indices. The program constructs a vector $\vec{d}$ between the two defined points.
'rot-sda'	iBindex jBindex k theta0 dc T_a	
'weight'	gravity wgt	The gravitational force to be applied on all bodies.
'flocal'	iBindex flocal	This is a constant force defined in the body-fixed frame that will be applied continuously on the body.
'f'	iBindex f	This is a constant force defined in the x-y frame that will be applied continuously on the body.
'T'	iBindex T	This is a constant torque that will be applied continuously on the body.
'user'		The M-file user_force.m is required. Such a file will be discussed in some of the examples. Also see the note following the example.

**Note:** The default settings for the gravitational constant and the direction of the gravitational force are  $9.81 \text{ m/s}^2$  and in the negative y-direction. With this default setting for the gravitational constant, the program assumes that the SI units are used in all the input M-files. If we wish to switch to a different system of units, it should be done here.

### Example B.1 (cont.)

The spring characteristic data is:

$$k = 20, {}^0L = 0.6$$

The spring is connected between points [1] and [2].



Figure B.5

The program constructs a vector  $\vec{d}$  between the two defined points. We also need to apply gravitational forces on the bodies in the default direction.

Based on the data, we construct the function M-file `inForces`. The output of this file is the array `Forces`.

```
function inForces
    include_global

F1 = Force_struct;
F1.type = 'ptp'; % default
F1.iPindex = 2;
F1.jPindex = 1;
F1.k = 20;
F1.L0 = 0.6;

F2 = Force_struct;
F2.type = 'weight';

Forces = [F1; F2];
```

**Note on Contact:** In the folder Forces (inside the folder Formulations) there are three function M-

files for modeling contact between a body and the ground. The files are Contact, Contact\_LN, and Contact\_FM. The M-files Contact\_LN and Contact\_FM describe the continuous contact force models of Eqs. 11.42 and 11.43 respectively. The M-file Contact uses these force models to determine to determine the contact force between a point on a body and the ground in the y-direction. It then applies the force and its corresponding moment to the force-array of that body. The M-file Contact can be used in a user\_force file to model contact. The function Contact requires the following input arguments:

```
Contact(Ci, Pi, Bi, k, e, Mi)
```

where,

Ci: a given index for the contact

Pi: point index

Bi: body index

Mi: model index = 1 Eq. 11.42; = 2 Eq. 11.43

k, e: model parameters

As an example, assume in a model point A on body (3) and point B on body (2) may contact the ground (not necessarily simultaneously). Assume the indices for points A and B are [5] and [7] respectively. Assume for a given set of parameters we want to use Eq. 11.42 for both contact points. For this purpose we need the following statements in the user\_force file:

```
k = 10^11; e = 0.95;
Contact(1, 5, 3, k, e, 1)
Contact(2, 7, 2, k, e, 1)
```

An example is provided in the model **Rod**.

**M-file inJoints:** In this file we define the constraints that are applied on the bodies, either as kinematic joints or as drivers. Since different joints or drivers require different type of data, the structure for these constraints contains different fields, as listed in Table B.5.

Table B.5

```
function Joint = Joint_struct
Joint = struct ( ...
    'type' , 'rev' , ... % joint type: rev, tran, rev-rev, etc.
    'iBindex' , 0 , ... % body index i
    'jBindex' , 0 , ... % body index j
    'iPindex' , 0 , ... % point Pi index
    'jPindex' , 0 , ... % point Pj index
    'iUindex' , 0 , ... % unit vector u_i index
    'jUindex' , 0 , ... % unit vector u_j index
    'iFunct' , 0 , ... % function index
    'L' , 0 , ... % constant length
    'R' , 1 , ... % constant radius
    'x0' , 0 , ... % initial condition for a disc
    'p0' , 0 , ... % initial condition for a disc or rigid
    ...
);
```

The first field for a joint element is its type. The available joint elements are:

Revolute or pin (.type = 'rev')

Translational or sliding (.type = 'tran')

```

Revolute-revolute (.type = 'rev-rev')
Revolute-translational (.type = 'rev-tran')
Rigid or bracket (.type = 'rigid')
Rolling disc (.type = 'disc')
Relative rotational driver (.type = 'rel-rot')
Relative translational driver (.type = 'rel-tran')

```

Each type of joint requires different set of data as described in the followings:

.type	Required entries	Comments
'rev'	iPindex jPindex	There is no need to enter body indices, since the point indices can provide that information.
'tran'	iPindex jPindex iUindex jUindex	
'rev-rev'	iPindex jPindex L	
'rev-tran'	iPindex jPindex iUindex L	
'rigid'	iBindex jBindex	Refer to Eq. 7.35.
'disc'	iBindex R x0 p0	A circular disc rolling on the horizontal ground. Refer to Eq. 7.38.
'rel-rot'	iBindex jBindex iFunct	A rotational motor or actuator (driver) that requires an analytical function.
'rel-tran'	iPindex jPindex iFunct	A translational motor or actuator (driver) that requires an analytical function.

### Example B.1 (cont.)

In this model we have a translational and a revolute joint. For the translational joint we state the type to be 'tran', and we define one point and one unit vector on each of the bodies.

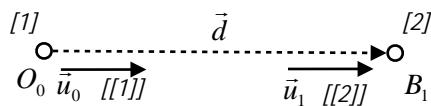


Figure B.6

For the revolute joint we need to define the indices of the two coinciding points as [2] and [3].

The output of this file is the array Joints.

```

function inJoints
    include_global

    J1 = Joint_struct;
    J1.type = 'tran';
    J1.iPindex = 2;
    J1.jPindex = 1;
    J1.iUindex = 2;
    J1.jUindex = 1;

    J2 = Joint_struct;
    J2.type = 'rev';
    J2.iPindex = 2;
    J2.jPindex = 3;

    Joints = [J1; J2];

```

For the translational joint, we must make sure that the point `iPindex` and the vector `iUindex` have previously been defined on the same body, and the same for the point and vector on the other body. The program will construct a vector  $\vec{d}$  between the two points with the arrow pointing to point `iPindex`.

**M-file `inFuncts`:** A user may define a function such as  $f = f(t)$  to describe the kinematics of a driver motor as function of time, for example, or the force of an actuator. The parameter  $t$  could be the time or a coordinate such as  $x$  of a body. The structure for these elements contains different fields, as listed in Table B.6.

Table B.6

```
function Funct = Funct_struct
    include_global
    Funct = struct ( ...
        'type' , 'a' , ... % function type a, b, or c
        't_start' , 0 , ... % required for functions b, c
        'f_start' , 0 , ... % required for functions b, c
        't_end' , 1 , ... % required for functions b, c
        'f_end' , 1 , ... % required for function b
        'dfdt_end' , 1 , ... % required for function c
        'ncoeff' , 3 , ... % number of coefficients
        'coeff' , [ ] , ... % required for function a
    );
)
```

The first field for a function element is its `type`. In this version of the program, three types of functions are available. Based on the type of the function, the corresponding parameters and coefficients are saved in the other fields.

**Type “a”:** This function can be used, for example, to define a constant angular velocity motor, such as  $\phi_i = \pi + 2\pi t$ . The general expression for this function is:

$$f = c_1 + c_2 t + c_3 t^2 \quad (8.1)$$

For the driver function  $\phi_i = \pi + 2\pi t$ , the coefficients are  $c_1 = \pi$ ,  $c_2 = 2\pi$ , and  $c_3 = 0$ . The user must supply values for the three coefficients; e.g.,

```
F1.type = 'a';
F1.coeff = [pi 2*pi 0];
```

This is the only function that the user must provide the coefficients directly. For the other two function types, the program determines the coefficients based on several parameters that are provided by a user.

**Type “b”:** This function can be used to vary the coordinate of a point or the length of an actuator from an initial value to a final value in a specified period of time. The general expression for this function is:

$$f = c_1 t^3 + c_2 t^4 + c_3 t^5 \quad (8.2)$$

This function and its derivatives are depicted in Figure B.7. The function has zero first and second derivatives at the start and end points. For this function we do not provide values for the three coefficients—we must supply four parameters: the `start` and the `end` values for  $t$  and  $f$ . The program computes the three coefficients in the M-file `functData` and saves them in the array `coeff`. The following is an example for the required data:

```
F1.type = 'b';
F1.t_start = 0;
F1.f_start = 0.3;
F1.t_end = 2;
F1.f_end = 1.5;
```

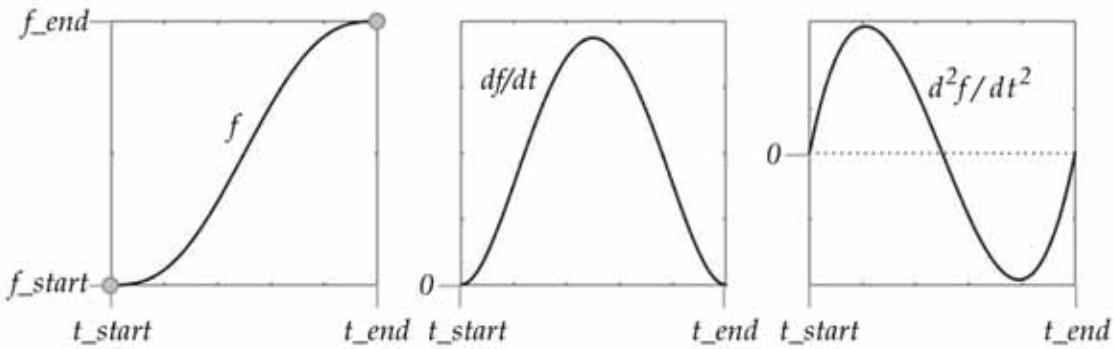


Figure B.7: Type “b” function and its first and second derivatives

**Type “c”:** The first derivative of this function rises from zero to a final value during a specified time period and keeps that value for the remaining duration of simulation. This function can be useful in simulations where a driver should raise the velocity (first derivative) from zero to a specified speed. The general expression for this function is:

$$f = c_1 + c_2 t^4 + c_3 t^5 + c_4 t^6 \quad (8.3)$$

This function and its derivatives are depicted in Figure B.8, where the function has zero first, second, and third derivatives at the start, and zero second and third derivatives at the end. The first derivative has a non-zero value at the end point. The user must supply four parameters: the *start* and *end* values for *t*, the start value for *f*, and the end value for the first derivative. The program adjusts the *ncoeff* from the default value of 3 to 4, and computes the four coefficients of the function in the M-file *functData* and saves them in the array *coeff*. The following is an example for the required data:

```
F1.typ2 = 'c';
F1.t_start = 0;
F1.f_start = 0;
F1.t_end = 2;
F1.dfdt_end = 1.5;
```

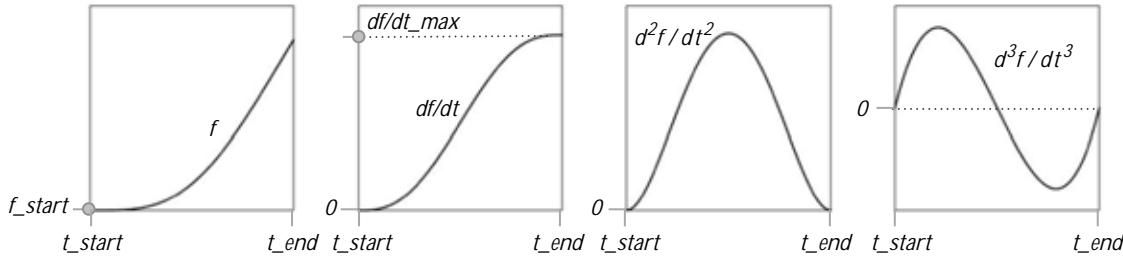


Figure B.8: Type “c” function and its first, second, and third derivatives

### Example B.1 (cont.)

There are no analytical functions in the sliding pendulum model. However, we still need to construct the M-file *inFuncs* and provide a *blank* array *Functs* as the output of this M-file.

```
function inFuncs
    include_global
    Functs = [];
```

**Note:** This is a short note on how the program uses a function during a simulation. There is a function named `functs` that the program uses as

```
[fun, fun_d, fun_dd] = functs(n, t)
```

The first input to this function is the function number (index) `n`, which could be 1, 2, ... based on the order of the functions that have been defined in `inFunct`. The second input entry `t` is the parameter time or a coordinate. `functs` returns the value of the function and its first and second derivatives as `fun`, `fun_d`, and `fun_dd` respectively.

This information becomes useful if the user wants to apply one of the available functions in a user supplied force function, or to add other capabilities to the program, or to develop a driver constraint in a model in the program DAP\_JC (refer to Chapter 9).

**M-file `inAnimate`:** One of the best ways to determine whether a constructed model performs as expected is by visualization. Looking at the orientation of bodies relative to each other at the initial time can help identify some obvious errors in a model. Observing an animation of the system in motion can reveal more about the model. The M-file `anim` takes the output of `dap` and provides an animation of the simulated response. Additional points or outlines for improving the presentation of an animation, but are not required for the analysis, can be provided in the M-file `inAnimate`. In this file we may provide data for shape and color of bodies, additional points for better visualization, and plot parameters. There is no new structure associated with this file—we use the existing structures of bodies and points.

The program `anim` displays the body mass centers and all the defined points as small circles. The program draws lines between a mass center and every point that is defined on that body. The assigned color to a body will be used for the lines as well (the default color is *black*). We can specify a different color, based on MATLAB's color abbreviations.

We may choose a shape, out of the three available shapes shown in Figure B.9, and assign it to a body. For any of these shapes, the program assumes that the geometric center of the shape is the origin of the  $\xi-\eta$  frame (mass center). The data for a shape is saved in additional fields in the body structure as stated in Table B.7, and additional explanation in Table B.8.

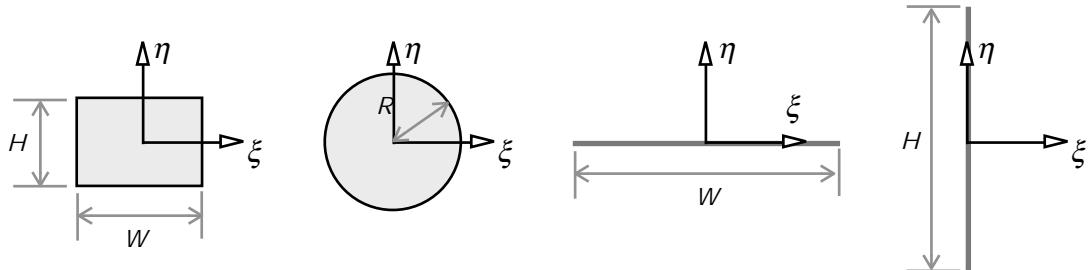


Figure B.9 Available standard shapes.

Table B.7

```
function Body = Body_struct
Body = struct ( ...
    ...
    'shape' , ' ' , ... % 'circle', 'rect', 'line'
    'R' , 1 , ... % radius of the circle
    'W' , 0 , ... % width of the rectangle
    'H' , 0 , ... % height of the rectangle
    'color' , 'k' , ... % default color for the body
    ...
);
```

Table B.8

Shape	Required statements	Comments
Rectangle	Bodies(i).shape = 'rect'; Bodies(i).W = 0.4; Bodies(i).H = 0.2;	
Circle	Bodies(i).shape = 'circle'; Bodies(i).R = 1.2;	
Line	Bodies(i).shape = 'line'; Bodies(i).W = 1.4;	Line along the $\xi$ -axis
Line	Bodies(i).shape = 'line'; Bodies(i).H = 0.8;	Line along the $\eta$ -axis

In addition to the defined attachment points of joints and application points of force elements, other points could be defined on a body for enhancing the presentation of its *nonstandard* shape. As an example, assume the triangular shaped body shown in Figure B.10(a) is connected to other bodies at pin joints *A* and *B*. With only two points, the body will be displayed as shown in Figure B.10(b). We may define a third point, such as *C*, to display the shape shown in Figure B.10(c), which may be a better description for a triangle.

Any point that is defined in the M-file `inAnimate` must be saved in the array `Points_anim`. These points are in addition to the points that have already been defined in M-file `inPoints` and saved in the array `Points`. The array `Points_anim` will be appended to the array `Points` for the animation.

For a rotating circular body, such as a rolling disc, we may want to define one arbitrary point on its circumference. The constructed line, as shown in Figure B.10(d), will provide a reference to realize the rotation of the circular body.

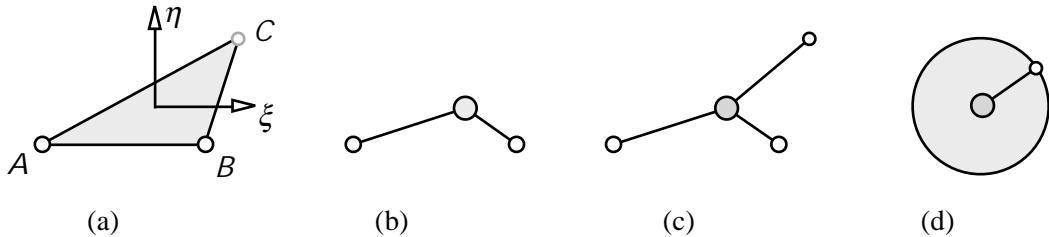


Figure B.10 Defining additional points for better visualization of animated objects.

In the M-file `inAnimate` we must set the scaling for the axes of the animation plot window. The scaling should be based on the dimensions used in a model and the expected range of its motion.

#### Example B.1 (cont.)

In this example there is no need to define any new points for animation. Therefore, we set `Points_anim` to be a *blank* array.

We have set the shape of body (1) to be a rectangle having a width of 0.4 and a height of 0.2. The

```
function inAnimate
    include_global
    Points_anim = [];
    Bodies(1).shape = 'rect';
    Bodies(1).W = 0.4;
```

shape of body (2) is a line along the body's  $\eta$ -axis, with a length (height) of 1.0 unit.

The scaling parameters for the animation plot window are also defined in this file.

```
Bodies(1).H = 0.2;
Bodies(2).shape = 'line';
Bodies(2).H = 1.0;
xmin = -0.5; xmax = 2.0;
ymin = -1.0; ymax = 0.5;
```

### B.3 POSTPROCESSING

The program dap simulates the dynamic response of a system by integrating the equations of motion from the initial time  $t = 0$  to the final time stated by the user. The program uses the MATLAB's integrator `ode45`, which reports the results at every  $\Delta t$  seconds (time period) as specified by the user<sup>1</sup>.

At every reporting time step, the integrator saves the time in an array named `T`, and saves the coordinates and velocities for all the bodies in an array named `uT`. None of the other results, such as accelerations and Lagrange multipliers, are saved. The coordinates and velocities are saved column-wise. For example, if we integrate the equations of motion of the sliding-pendulum system that contains two bodies for 4.0 seconds, and ask for  $\Delta t = 0.02$  reporting intervals, the arrays will be formed as listed in Table B.9. Note that for  $n_b$  bodies, there will be  $6n_b$  columns in the array `uT`.

Table B.9 Arrays `T` and `uT` contain the output from the integrator `ode45`.

$t$	$x_1$	$y_1$	$\phi_1$	$x_2$	$y_2$	$\dots$	$\dot{x}_2$	$\dot{y}_2$	$\dot{\phi}_2$
0.00	1.00	0.20	0.00	1.25	-0.23	$\dots$	0.00	0.00	0.00
0.02	0.99	0.20	0.00	1.24	-0.23	$\dots$	-0.06	-0.03	-0.13
...						$\dots$			
3.98	1.06	0.20	0.00	1.06	-0.30	$\dots$	0.21	0.01	1.32
4.00	1.05	0.20	0.00	1.06	-0.29	$\dots$	0.20	0.01	1.37

The program `post` takes the computed values for the coordinates and velocities from `uT` at each reporting time step. It recovers the missing results by solving the equations of motion again at every time step, and then saves all the results in several arrays as listed in

**Note:** In this version of the program the potential energy is only computed for the gravitational force and the point-to-point springs with linear characteristics. Contributions to the potential energy from other conservative forces could be included in the M-file `post` by the user.

Table B.10. This program also computes the kinetic energy, the potential energy, and the total energy at every time steps and saves the results. The user can refer to these arrays to perform further analyses with the simulated results.

**Note:** In this version of the program the potential energy is only computed for the gravitational force and the point-to-point springs with linear characteristics. Contributions to the potential energy from other conservative forces could be included in the M-file `post` by the user.

Table B.10

Array	Size	Description
<code>r</code>	$nt \times nB \times 2$	Translational coordinates

<sup>1</sup> The reporting time step is not the same as the integration time step. Integration time step used by `ode45` is variable and most likely much smaller than the reporting time step.

rd	nt × nB × 2	Translational velocities
rdd	nt × nB × 2	Translational accelerations
p	nt × nB	Rotational coordinates
pd	nt × nB	Rotational velocities
pdd	nt × nB	Rotational acceleration
rP	nt × nP × 2	Coordinates of points
rPd	nt × nP × 2	Velocity of points
Jac	nt × nConst × nB3	Jacobian matrix
Lam	nt × nConst	Lagrange multipliers
eng	nt × 3	Energy (kinetic, potential, total)

nt	Number of time steps (rows) including $t = 0$
nB	Number of bodies
nP	Number of points
nConst	Number of constraints
nB3	3 times nB

### Examples

To plot the acceleration of body (3) in the  $y$ -direction; that is  $\ddot{y}_3$ , versus time, we use:

```
plot(T,rdd(:,3,2))
```

To plot the  $x$ -component of the velocity of the 5<sup>th</sup> point in the model versus time, we use:

```
plot(T,rPd(:,5,1))
```

To plot the kinetic, potential, and total energies, we can use:

```
plot(T,eng(:,1)'r')
hold on
plot(T,eng(:,2),'g')
plot(T,eng(:,3))
```

## B.4 APPLICATION EXAMPLES

These example models are discussed in Chapter 8 of the textbook. All the models reside in the folder **Models**.

### B.4.1 Double A-Arm Suspension (DAP\_BC/Models/AA)

Refer to Section 8.1.1 of the textbook for figures and further discussion.

The input M-files are developed according to the body and point indices that are discussed in Section 8.1.1. In the M-file **inForces** two sets of initial conditions for the coordinates are provided: one correct set and one incorrect set. In the M-file **inForces** we state that the second force elements is described in a user supplied force function (**user\_force**) where we provide the logic to determine whether the tire is in contact with the ground, and if so, to determine the tire force.

```
function inBodies
    include_global

B1 = Body_struct;
B1.m = 2;
B1.J = 0.5;
```

```
function inPoints
    include_global

Q1 = Point_struct;
Q1.Bindex = 1;
Q1.sPlocal = [-0.24; 0];
```

```

B1.r = [0.4398; 0.2512]; % correct
B1.p = -0.0367; % correct
% B1.r = [0.51; 0.28]; % incorrect
% B1.p = 340*pi/180; % incorrect

B2 = Body_struct;
B2.r = [0.6817; 0.3498]; % correct
B2.p = 0.0783; % correct
% B2.r = [0.75; 0.35]; % incorrect
% B2.p = 0; % incorrect
B2.m = 30;
B2.J = 2.5;

B3 = Body_struct;
B3.r = [0.4463; 0.4308]; % correct
B3.p = 6.5222; % correct
% B3.r = [0.49; 0.41]; % incorrect
% B3.p = 350*pi/180; % incorrect
B3.m = 1;
B3.J = 0.5;

Bodies = [B1; B2; B3];

```

```

A1 = Point_struct;
A1.Bindex = 1;
A1.sPlocal = [0.18;0];

A2 = Point_struct;
A2.Bindex = 2;
A2.sPlocal = [-0.07;-0.10];

A3 = Point_struct;
A3.Bindex = 2;
A3.sPlocal = [-0.10;0.12];

B3 = Point_struct;
B3.Bindex = 3;
B3.sPlocal = [0.13;0];

O3 = Point_struct;
O3.Bindex = 3;
O3.sPlocal = [-0.13;0];

O0 = Point_struct;
O0.Bindex = 0;
O0.sPlocal = [0.32;0.40];

Q0 = Point_struct;
Q0.Bindex = 0;
Q0.sPlocal = [0.20;0.26];

E1 = Point_struct;
E1.Bindex = 1;
E1.sPlocal = [0;0];

F0 = Point_struct;
F0.Bindex = 0;
F0.sPlocal = [0.38;0.43];

Points = [Q1; A1; A2; B2; ...
          B3; O3; O0; Q0; E1; F0];

```

```

function inJoints
    include_global

J1 = Joint_struct;
J1.type = 'rev'; %Q
J1.iPindex = 1;
J1.jPindex = 8;

J2 = Joint_struct;
J2.type = 'rev'; % A
J2.iPindex = 2;
J2.jPindex = 3;

J3 = Joint_struct;
J3.type = 'rev'; % B
J3.iPindex = 4;
J3.jPindex = 5;

```

```

function inForces
    include_global

S1 = Force_struct;
S1.type = 'ptp';
S1.iPindex = 9;
S1.jPindex = 10;
S1.k = 90000;
S1.L0 = 0.23;
S1.dc = 1100;

S2 = Force_struct;
S2.type = 'user';
S2.k = 50000;
S2.L0 = 0.35;
S2.dc = 1000;

S3 = Force_struct;

```

```
J4 = Joint_struct;
J4.type = 'rev'; % O
J4.iPindex = 6;
J4.jPindex = 7;

Joints = [J1; J2; J3; J4];
```

```
S3.type = 'weight';

Forces = [S1; S2; S3];
```

```
function user_force
    include_global

    % Unilateral spring-damper representing the radial tire force
    del = Bodies(2).r(2) - Forces(2).L0;
    if del < 0
        fy = Forces(2).k*del + ...
            Forces(2).dc*Bodies(2).r_d(2);
        fsd = [0; -fy];
        Bodies(2).f = Bodies(2).f + fsd;
    end

    % ...
    % del = Bodies(2).r(2) - 0.35;
    % ...
    % fy = 40000*del + 1000*Bodies(2).r_d(2);
    % ...
```

Following a simulation, to plot the response for one of the coordinates, for example the y-coordinate of body (2) versus time, we can use the following command:

```
plot(T, uT(:,5))
```

Or, we can execute the M-file post and then use this command:

```
plot(T, r(:,2,2))
```

#### B.4.2 MacPherson Suspension (DAP\_BC/Models/MP\_A, MP\_B, MP\_C)

Refer to Section 8.1.2 of the textbook for figures and further discussion.

**DAP\_BC/Models/MP\_A:** This model consists of 3 moving bodies, 3 pin joints, and 1 translational joint. The input M-files are developed according to the body and point indices that are discussed in Section 8.1.2.

```
function inBodies
    include_global

B1 = Body_struct;
B1.r = [0.5840; 0.3586];
B1.p = 6.0819;
B1.m = 20;
B1.J = 2.5;

B2 = Body_struct;
B2.m = 2;
B2.J = 0.5;
B2.r = [0.3450; 0.2900];
B2.p = 0;

B3 = Body_struct;
```

```
function inPoints
    include_global

A1 = Point_struct;
A1.Bindex = 1;
A1.sPlocal = [ 0.00; -0.07];

B1 = Point_struct;
B1.Bindex = 1;
B1.sPlocal = [-0.17; 0.25];

C1 = Point_struct;
C1.Bindex = 1;
C1.sPlocal = [ 0.11; -0.02];

O0 = Point_struct;
```

```
B3.r = [0.4528; 0.6862];
B3.p = 5.0019;
B3.m = 0.5;
B3.J = 0.2;

Bodies = [B1; B2; B3];
```

```
O0.Bindex = 0;
O0.sPlocal = [ 0.41; 0.83];

Q0 = Point_struct;
Q0.Bindex = 0;
Q0.sPlocal = [ 0.12; 0.29];

Q2 = Point_struct;
Q2.Bindex = 2;
Q2.sPlocal = [-0.225; 0.00];

A2 = Point_struct;
A2.Bindex = 2;
A2.sPlocal = [ 0.225; 0.00];

O3 = Point_struct;
O3.Bindex = 3;
O3.sPlocal = [-0.15; 0.00];

Points = [A1; B1; C1; O0; ...
          Q0; Q2; A2; O3];
```

```
function inUvectors
    include_global

V1 = Unit_struct;
V1.Bindex = 1;
V1.ulocal = [0.47; -0.88];

V2 = Unit_struct;
V2.Bindex = 3;
V2.ulocal = [1; 0];

Uvectors = [V1; V2];
```

```
function inJoints
    include_global

J1 = Joint_struct;
J1.type = 'rev'; %A
J1.iPindex = 1;
J1.jPindex = 7;

J2 = Joint_struct;
J2.type = 'rev'; % Q
J2.iPindex = 5;
J2.jPindex = 6;

J3 = Joint_struct;
J3.type = 'rev'; % O
J3.iPindex = 4;
J3.jPindex = 8;

J4 = Joint_struct;
J4.type = 'tran'; % O-B
J4.iUindex = 1;
J4.jUindex = 2;
J4.iPindex = 2;
J4.jPindex = 8;

Joints = [J1; J2; J3; J4];
```

```
function inForces
    include_global

S1 = Force_struct;
S1.type = 'ptp';
S1.iPindex = 2; % B1
S1.jPindex = 4; % O0
S1.k = 20000;
S1.L0 = 0.34;
```

```
function user_force
    include_global

del = Bodies(1).r(2) - Forces(2).L0;

if del < 0
    fy = Forces(2).k*del +
Forces(2).dc*Bodies(1).r_d(2);
    fsd = [0; -fy];
```

```
S1.dc = 1100;
S2 = Force_struct;
S2.type = 'user'; % tire
S2.k = 100000;
S2.L0 = 0.30;
S2.dc = 1000;

S3 = Force_struct;
S3.type = 'weight';

Forces = [S1; S2; S3];
```

```
Bodies(1).f = Bodies(1).f + fsd;
Bodies(1).n = Bodies(1).n + ...
    s_rot(Points(3).sp)/*fsd;
end
```

Following a simulation, we can plot the coordinate and velocity of the mass center of body (2) versus time using the following commands:

```
plot(T, uT(:,5))
```

Or,

```
plot(T, uT(:,14))
```

We note that since the model contains 3 bodies,  $y_2$  is in the 5<sup>th</sup> column of the  $uT$  matrix and  $\dot{y}_2$  is in the 14<sup>th</sup> column.

We can also execute `post` and then plot the y-coordinate of point C versus time using the command:

```
plot(T,rP(:,3,2))
```

Or use the following command for the velocity of C in the y-direction.

```
plot(T,rPd(:,3,2))
```

We note that C is point number [3].

**DAP\_BC/Models/MP\_B:** This model consists of 2 moving bodies, 2 pin joints, and 1 revolute-translational joint. The input M-files are developed according to the body and point indices that are discussed in Section 8.1.2.

```
function inBodies
    include_global

B1 = Body_struct;
B1.r = [0.5840; 0.3586];
B1.p = 6.0819;
B1.m = 20;
B1.J = 2.5;

B2 = Body_struct;
B2.m = 2;
B2.J = 0.5;
B2.r = [0.3450; 0.2900];
B2.p = 0;

Bodies = [B1; B2];
```

```
function inPoints
    include_global

A1 = Point_struct;
A1.Bindex = 1;
A1.sPlocal = [ 0.00; -0.07];

B1 = Point_struct;
B1.Bindex = 1;
B1.sPlocal = [-0.17; 0.25];

C1 = Point_struct;
C1.Bindex = 1;
C1.sPlocal = [ 0.11; -0.02];

O0 = Point_struct;
O0.Bindex = 0;
O0.sPlocal = [ 0.41; 0.83];

Q0 = Point_struct;
Q0.Bindex = 0;
Q0.sPlocal = [ 0.12; 0.29];

Q2 = Point_struct;
Q2.Bindex = 2;
```

```

Q2.sPlocal = [-0.225; 0.00];

A2 = Point_struct;
A2.Bindex = 2;
A2.sPlocal = [ 0.225; 0.00];

Points = [A1; B1; C1; O0; ...
          Q0; Q2; A2];

```

```

function inUvectors
    include_global

V1 = Unit_struct;
V1.Bindex = 1;
V1.ulocal  = [0.47; -0.88];

Uvectors = [V1];

```

```

function inJoints
    include_global

J1 = Joint_struct;
J1.type = 'rev'; %A
J1.iPindex = 1;
J1.jPindex = 7;

J2 = Joint_struct;
J2.type = 'rev-tran'; % O-B
J2.iPindex = 2;
J2.jPindex = 4;
J2.iUindex = 1;

J3 = Joint_struct;
J3.type = 'rev'; % Q
J3.iPindex = 5;
J3.jPindex = 6;

Joints = [J1; J2; J3];

```

```

function inForces
    include_global

S1 = Force_struct;
S1.type = 'ptp';
S1.iPindex = 2; % B1
S1.jPindex = 4; % O0
S1.k = 20000;
S1.L0 = 0.34;
S1.dc = 1100;
S1.f_a = 0;

S2 = Force_struct;
S2.type = 'user'; % tire
S2.k = 100000;
S2.L0 = 0.30;
S2.dc = 1000;

S3 = Force_struct;
S3.type = 'weight';

Forces = [S1; S2; S3];

```

```

function user_force
    include_global

del = Bodies(1).r(2) - Forces(2).L0;

if del < 0
    fy = Forces(2).k*del +
Forces(2).dc*Bodies(1).r_d(2);
    fsd = [0; -fy];
    Bodies(1).f = Bodies(1).f + fsd;
    Bodies(1).n = Bodies(1).n + ...
                  s_rot(Points(3).sp)'*fsd;
end

```

**DAP\_BC/Models/MP\_C:** This model consists of 1 moving body, 1 revolute-revolute and 1 revolute-translational joint. The input M-files are developed according to the body and point indices that are discussed in Section 8.1.2.

```
function inBodies
    include_global

B1 = Body_struct;
B1.r = [0.5840; 0.3586];
B1.p = 6.0819;
B1.m = 20;
B1.J = 2.5;

Bodies = [B1];
```

```
function inPoints
    include_global

A1 = Point_struct;
A1.Bindex = 1;
A1.sPlocal = [ 0.00; -0.07];

B1 = Point_struct;
B1.Bindex = 1;
B1.sPlocal = [-0.17; 0.25];

C1 = Point_struct;
C1.Bindex = 1;
C1.sPlocal = [ 0.11; -0.02];

O0 = Point_struct;
O0.Bindex = 0;
O0.sPlocal = [ 0.41; 0.83];

Q0 = Point_struct;
Q0.Bindex = 0;
Q0.sPlocal = [ 0.12; 0.29];

Points = [A1; B1; C1; O0; Q0];
```

```
function inUvectors
    include_global

V1 = Unit_struct;
V1.Bindex = 1;
V1.ulocal = [0.47; -0.88];

Uvectors = [V1];
```

```
function inJoints
    include_global

J1 = Joint_struct;
J1.type = 'rev-rev'; % Q-A
J1.iPindex = 1;
J1.jPindex = 5;
J1.L = 0.45;

J2 = Joint_struct;
J2.type = 'rev-tran'; % O-B
J2.iPindex = 2;
J2.jPindex = 4;
J2.iUindex = 1;

Joints = [J1; J2];
```

```
function inForces
    include_global

S1 = Force_struct;
S1.type = 'ptp'; % default
S1.iPindex = 2; % B1
S1.jPindex = 4; % O0
S1.k = 20000;
S1.L0 = 0.34;
S1.dc = 1100;
S1.f_a = 0;
```

```
function user_force
    include_global

del = Bodies(1).r(2) - Forces(2).L0;

if del < 0
    fy = Forces(2).k*del +
Forces(2).dc*Bodies(1).r_d(2);
    fsd = [0; -fy];
    Bodies(1).f = Bodies(1).f + fsd;
    Bodies(1).n = Bodies(1).n + ...
```

```
S2 = Force_struct;
S2.type = 'user'; % tire
S2.k = 100000;
S2.L0 = 0.30;
S2.dc = 1000;

S3 = Force_struct;
S3.type = 'weight';
Forces = [S1; S2; S3];
```

```
s_rot(Points(3).sP)'*fsd;
end
```

### B.4.3 Cart (DAP\_BC/Models/Cart\_A, Cart\_B, Cart\_C)

Refer to Section 8.1.3 of the textbook for figures and further discussion.

**DAP\_BC/Models/Cart\_A:** This model consists of 3 moving bodies, 2 pin joints, 2 disc joints with no-slip condition, and a driver constraint for the motor. The input M-files are developed according to the body and point indices that are discussed in Section 8.1.3. It is assumed that the rear wheel, body (2), rolls with a constant angular velocity of  $2\pi$  rad/sec in the clockwise direction and zero angular acceleration.

A *relative-rotation* constraint is defined as the driver. This constraint acts on body (2) and it refers to “function 1” for its analytical description and parameters. In the M-file *inFunct* we state the function type as “a” and, therefore, its parameters are: zero initial angle,  $-2\pi$  (clockwise) angular velocity, and zero angular acceleration.

```
function inBodies
    include_global

B1 = Body_struct;
B1.r = [0.5840; 0.3586];
B1.p = 6.0819;
B1.m = 20;
B1.J = 2.5;

B2 = Body_struct;
B2.m = 2;
B2.J = 0.5;
B2.r = [0.3450; 0.2900];
B2.p = 0;

B3 = Body_struct;
B3.r = [0.4528; 0.6862];
B3.p = 5.0019;
B3.m = 0.5;
B3.J = 0.2;

Bodies = [B1; B2; B3];
```

```
function inPoints
    include_global

P1 = Point_struct;
P1.Bindex = 1;
P1.sPlocal = [-0.3; -0.1];

P2 = Point_struct;
P2.Bindex = 1;
P2.sPlocal = [ 0.3; -0.1];

P3 = Point_struct;
P3.Bindex = 2;
P3.sPlocal = [ 0; 0];

P4 = Point_struct;
P4.Bindex = 3;
P4.sPlocal = [ 0; 0];

Points = [P1; P2; P3; P4];
```

```
function inJoints
    include_global

J1 = Joint_struct;
J1.type = 'rev';
J1.iPindex = 1;
J1.jPindex = 3;
```

```
function inForces
    include_global

F1 = Force_struct;
F1.type = 'weight'; % include the weight

Forces = [F1];
```

```

J2 = Joint_struct;
J2.type = 'rev';
J2.iPindex = 2;
J2.jPindex = 4;

J3 = Joint_struct;
J3.type = 'disc';
J3.iBindex = 2;
J3.R = 0.1;
J3.x0 = 0.2;

J4 = Joint_struct;
J4.type = 'disc';
J4.iBindex = 3;
J4.R = 0.1;
J4.x0 = 0.8;
J4.p0 = 0; % default

J5 = Joint_struct;
J5.type = 'rel-rot'; % motor driver
J5.iBindex = 2;
J5.jBindex = 1;
J5.iFunct = 1;

Joints = [J1; J2; J3; J4; J5];

```

```

function inFuncts
    include_global

F1 = Funct_struct;
F1.type = 'a';
F1.coeff = [0 -2*pi 0];

Functs = [F1];

```

**DAP\_BC/Models/Cart\_B:** This model is the same as the model of Cart\_A except for the driver constraint that refers to function “c” instead of “a”.

```

function inFuncts
    include_global

F1 = Funct_struct;
F1.type = 'c';
F1.t_end = 2.0;
F1.dfdt_end = -2*pi;

Functs = [F1];

```

After executing post we can plot the Lagrange multiplier associated with the driver constraint, the no-slip constraint on the rear wheel, and the no-slip constraint on the front wheel using the following commands respectively:

```

>> plot(T,Lam(:,9))
>> figure
>> plot(T,Lam(:,6))
>> hold on
>> plot(T,Lam(:,8), 'r')

```

**DAP\_BC/Models/Cart\_C:** In this model we consider an applied torque on the rear wheel, where it is assumed that the torque is generated by an electric motor with a known torque-speed characteristic. We remove the driver constraint (`rel-rot`) from the `inJoints` M-file of the original model. In the M-file `inForces` we add a second force element as `user type`. We then provide the following `user_force` M-file. In this file, based on the angular velocity of the rear wheel and the torque-speed characteristics of the motor we determine the torque of the motor and apply it on the wheel. Note that we apply this torque with a negative sign since we want to rotate the wheel CW and move the cart from left to right.

```
function user_force
    include_global
% Motor
    omega_max = 4*pi; T_max = 20;
    omega = abs(Bodies(2).p_d);
    T_motor = T_max*(1 - omega/omega_max);
    if T_motor > T_max
        T_motor = T_max;
    end
    Bodies(2).n = Bodies(2).n - T_motor;
    Bodies(1).n = Bodies(1).n + T_motor;
```

**Models/Cart\_D:** In this model we add a resistive force to the model `Cart_C`. This resistive force is added to the `user_force` M-file and include it in the array of force in the *x*-direction for body (1).

```
function user_force
    ...
% Aerodynamic resistive force
    damp_aero = 10;
    x_d = Bodies(1).r_d(1);
    f_aero = damp_aero*x_d^2;
    Bodies(1).f(1) = Bodies(1).f(1) - f_aero;
```

#### B.4.4 Conveyor Belt and Friction (DAP\_BC/Models/CB)

Refer to Section 8.1.4 of the textbook for figures and further discussion.

```
function inBodies
    include_global

B1 = Body_struct;
B1.m = 1.0;
B1.J = 1.0;
B1.r = [1.0; 0.2];
B1.r_d = [ 0.0; 0.0];

Bodies = [B1];
```

```
function inPoints
    include_global

P1 = Point_struct;
P1.Bindex = 0;
P1.sPlocal = [ 0; 0.2];

P2 = Point_struct;
P2.Bindex = 1;
P2.sPlocal = [ 0; 0];

Points = [P1; P2];
```

```
function inUvectors
    include_global

U1 = Unit_struct;
```

```
function inJoints
    include_global

J1 = Joint_struct;
```

```

U1.Bindex = 0;
U1.ulocal  = [ 1.0; 0];

U2 = Unit_struct;
U2.Bindex = 1;
U2.ulocal  = [ 1.0; 0];

Uvectors = [U1; U2];

```

```

J1.type = 'tran';
J1.iPindex = 2;
J1.jPindex = 1;
J1.iUindex = 2;
J1.jUindex = 1;

Joints = [J1];

```

```

function inForces
    include_global

F1 = Force_struct;
F1.type = 'ptp';
F1.iPindex = 2;
F1.jPindex = 1;
F1.k = 10;
F1.L0 = 0.8;
F1.dc = 0;

F2 = Force_struct;
F2.type = 'weight';

F3 = Force_struct;
F3.type = 'user';

Forces = [F1; F2; F3];

```

```

function user_force
    include_global

% Anderson et al. friction model
mu_d = 0.15; mu_s = 0.2; k_v = 0.0;
v_s = 0.001; p = 2; k_t = 10000;
fy = 9.81; % normal force
v_conv = 0.1;
v = v_conv - Bodies(1).r_d(1);
ff = Friction_A(mu_s, mu_d, v_s, p, k_t, v);
fx = fy*(ff + k_v*v);
fs = [fx; 0];
Bodies(1).f = Bodies(1).f + fs;

```

#### B.4.5 Rod Impacting the Ground (DAP\_BC/Models/Rod)

Refer to Section 8.1.5 of the textbook for figures and further discussion.

```

function inBodies
    include_global

B1 = Body_struct;
B1.m = 1.0;
B1.J = 0.01;
B1.r = [0; 1];
B1.p = pi/4;
B1.r_d = [0; -6];

Bodies = [B1];

```

```

function inPoints
    include_global

P1 = Point_struct;
P1.Bindex = 1;
P1.sPlocal = [ 0; -1];

P2 = Point_struct;
P2.Bindex = 1;
P2.sPlocal = [ 0; 1];

Points = [P1; P2];

```

```

function inForces
    include_global

S1 = Force_struct;
S1.type = 'weight';

S2 = Force_struct;
S2.type = 'user';

Forces = [S1; S2];

```

```
function user_force
    include_global
    global pen1_d0 pen2_d0

% Parameters for the contact model
    k = 10^11; e = 0.95;

% Contact(Contact index, Point index, Body index, k, e, Model index)
%     Model-index = 1: Eq. 11.42
%     Model-index = 2: Eq. 11.43

% Point [1] on Body (1)
    Contact(1, 1, 1, k, e, 1)
% Point [2] on Body (1)
    Contact(2, 2, 1, k, e, 1)
```

**Note:** To observe the peaks of the acceleration response, we must select the reporting time steps to be very small, such as 0.0001 seconds.

---

## *Appendix C: Dynamic Analysis Program: Joint Coordinates (DAP\_JC)*

---

In Chapter 9, we discussed a method for constructing the equations of motion for multibody systems called the joint-coordinate formulation. A MATLAB® program based on this formulation has been developed, named DAP\_JC (dynamic analysis program with joint coordinates). The program and its user manual are provided in this appendix that can be downloaded from the textbook's website.

# Appendix C

## DAP\_JC User Manual

(September 2018)

The Dynamic Analysis Program: Joint Coordinates (DAP\_JC) is a semi-general-purpose program that is developed to assist a user in learning how the joint-coordinate equations of motion are formulated. The program has many similarities and also many differences with the Dynamic Analysis Program: Body Coordinates (DAP\_BC). The similarities are in the structure of the two programs, in the input data, and in the post-processing and the animation of the results. The main difference is that DAP\_JC does not automatically construct the equations of motion—these equations for a multibody model must be provided by the user.

In this appendix we first discuss the structure of the program. Then we show how to use the program to simulate any of the models that are provided with the program. Finally we discuss in detail how to construct a model.

Most of the M-files that form the program DAP\_JC have names similar to the M-files in DAP\_BC however, they are not identical. The M-files from the two programs that share the same name perform the same task but with different formulations.

### C.1 STRUCTURE OF DAP\_JC

The program follows the formulation of the equations of motion as presented in Chapter 9 of the textbook. A model could be a system without any constraints (for example an open-chain system without a driver constraint), or with constraints (for example closed-chain system with or without a driver constraint).

For an open chain system with no added constraint such as a driver, the program considers the equations of motion from Eqs. (9.22)-(9.24):

$$\mathbf{M}\ddot{\boldsymbol{\theta}} = {}^{(a)}\mathbf{h} \quad (9.22)$$

where

$$\mathbf{M} = \mathbf{B}'\mathbf{M}\mathbf{B} \quad (9.23)$$

$${}^{(a)}\mathbf{h} = \mathbf{B}'({}^{(a)}\mathbf{h} - \mathbf{M}\dot{\boldsymbol{\theta}}) \quad (9.24)$$

Based on the model data provided by the user, the program constructs and evaluates the mass matrix  $\mathbf{M}$  and the array of applied forces  ${}^{(a)}\mathbf{h}$  containing the forces of gravity, springs, dampers, etc. The user must provide complete description of forward kinematics, the matrix  $\mathbf{B}$ , and the array  $\dot{\boldsymbol{\theta}}$ . The program constructs matrix  $\mathbf{M}$ , array  ${}^{(a)}\mathbf{h}$ , and performs integration of the equations of motion.

For a closed chain system, or any system containing constraints, the program constructs the equations of motion according to Eq. (9.34):

$$\begin{bmatrix} \mathbf{M} & -\mathbf{C}' \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\boldsymbol{\theta}} \\ {}^*\boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} {}^{(a)}\mathbf{h} \\ -\dot{\mathbf{C}}\dot{\boldsymbol{\theta}} \end{Bmatrix} \quad (9.34)$$

where  $\mathbf{C}$  is the Jacobian of the constraints (refer to Eq. (9.28)). In addition to the complete description of forward kinematics, the matrix  $\mathbf{B}$  and the array  $\dot{\mathbf{B}}\theta$ , the user must also supply a program to construct and evaluate the constraints, matrix  $\mathbf{C}$ , and the array  $\dot{\mathbf{C}}\theta$ . Then the program constructs the equations of motion as in Eq. (9.34) and integrates the equations as discussed in Chapter 13.

In Chapter 9 we discussed that the rotational joint coordinate associated with a revolute (pin) joint could be defined either as a relative coordinate or an absolute coordinate. In the program DAP\_JC we use **relative** joint coordinates. Therefore, if we decide to model some of the examples from Chapter 9 that are formulated based on the absolute joint coordinates, we must first reformulate them with relative joint coordinates before constructing the necessary M-files for the program DAP\_JC.

## C.2 USING DAP\_JC

The program DAP\_JC and the corresponding example models can be found in the folder named DAP\_JC followed by the date of its last revision. As shown in Figure C.1(a), similar to DAP\_BC, this folder contains three script M-files and two sub-folders. The folder named Models contains several examples of simple multibody systems, as shown in Figure C.1(b). Some of these models are discussed in detail in this manual as a guide in learning how to use the program, and some models are discussed briefly. If we construct a new model, we should save it in this folder.

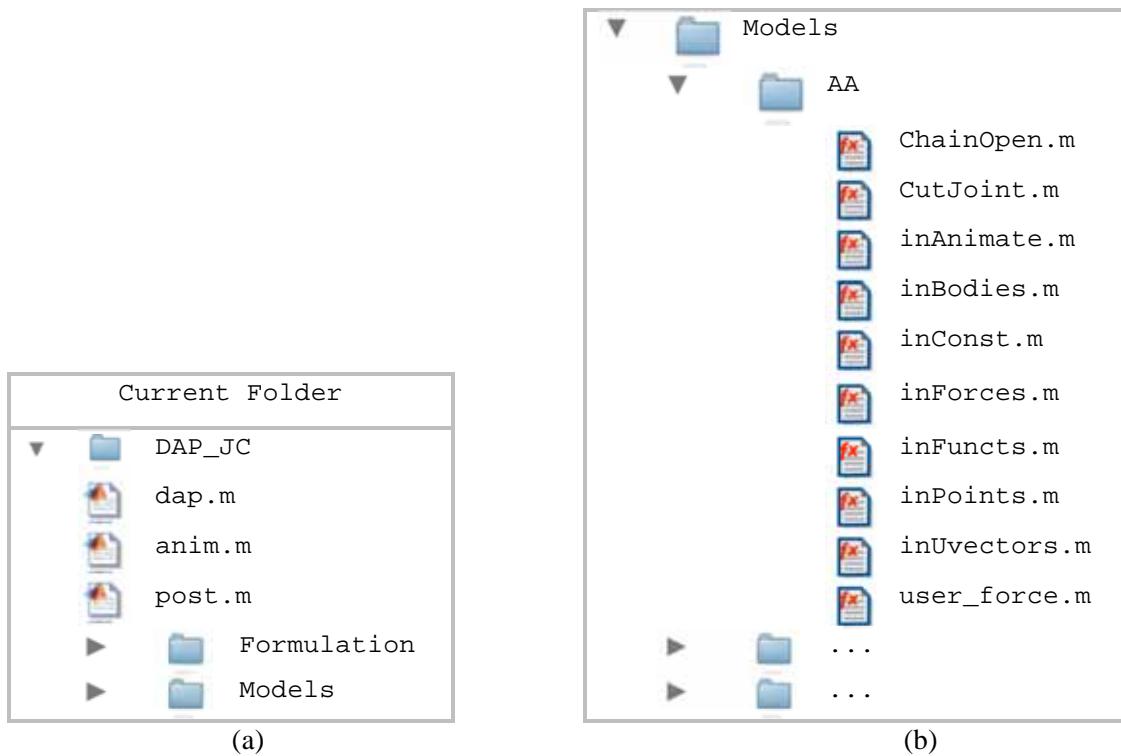


Figure C.1 Folders and M-files files of DAP\_JC.

The script M-file `dap` is the main program to execute. The program prompts the user with a few questions prior to performing a simulation. After a simulation is completed, the user may execute the script M-file `anim` to visualize a crude animation of the simulated response. Or, following a simulation,

we may execute the script M-file `post` for post-processing of the results. This script transforms the results into a more recognizable set of arrays for further analyses or plots.

The folder `Formulation` contains other subfolders, script and function M-files. To simulate the response of a multibody system, using the existing capabilities of the program, there is no need for a user to revise any of the M-files in this folder.

It is highly recommended to make the folder `DAP_JC` the “Current Folder” in MATLAB, as indicated in Figure C.1. This facilitates setting the “paths” to different folders and M-files.

Executing the programs in `DAP_JC` is very similar to that in the programs in `DAP_BC`. In the Command Window we type `dap` and then respond to each question:

```
>> dap
Which folder contains the model?
```

If we enter the name of a folder that contains an open-chain model, the program will continue with the next question for the final time and time step. But if the model is a closed-chain system, the program will ask if we want to correct the initial conditions on the “joint” coordinates and velocities:

```
Do you want to correct the initial conditions? [(y)es/(n)o]
```

If we are certain that the provided initial conditions satisfy the position and velocity constraints of the cut joint(s) or any other existing constraints, we may answer no, otherwise we should respond yes. If we respond yes, the program will report the corrected values for the joint coordinates and velocities. Next, the program will ask for the final time of integration and the reporting time steps (all in seconds):

```
Final time = ?
Reporting time-step = ?
```

During the integration process for the requested time period, the program keeps the user informed of the progress by reporting the integration time every 100 function-evaluations. (A function evaluation means that the equations of motion have been constructed and solved for the accelerations.) At the completion of integration, the program reports the number of function evaluations. This information is reported as a measure for the efficiency (or inefficiency) of the method of solution.

After the completion of a simulation, we can observe an animation of the results by typing in the Command Window `anim`, or executing `post` for further post-processing of the results:

```
>> anim
>> post
```

Some of the existing models in the `Models` folder are:

AA:	Double A-arm suspension (closed-chain)
Cart_C:	Cart (closed-chain)
MPA:	MacPherson suspension with three moving bodies (closed-chain)
MPB:	MacPherson suspension with two moving bodies (closed-chain)
Rod:	A single rod falling and impacting the ground (open-chain)
Rod2:	Two connected rods falling and impacting the ground (open-chain)
SC:	Slider-crank (closed-chain)
SCd:	Slider-crank with driver motor (closed-chain)
TP:	Triple pendulum (open-chain)

### C.3 CONSTRUCTING A MODEL

Most of the function M-files for a model are exactly the same as those for `DAP_BC`. As an example, the input files for the double A-arm system `AA` are shown in Figure C.1(b) where most of the file names should be familiar. The model input files for both programs are listed next to each other in the following

table for comparison. The files that are marked with an asterisk are exactly the same in both programs. The M-file `inBodies` for `DAP_JC` contains slightly different data as in `DAP_BC`, which will be discussed later in this manual. The M-file `inJoints` that is required in `DAP_BC` is not needed for `DAP_JC`, but instead we have a file named `inConst` in `DAP_JC`.

There are two additional files that a model in `DAP_JC` may require that contain formulations—not only data like the other input files. The M-file `ChainOpen`, that is required for every model, must describe the forward kinematics of an open-chain or a cut open-chain system. The M-file `CutJoint` is only required for a closed-chain system describing the cut-joint and/or if we have any driver constraints in a system. This file is not needed if the system is open-chain and there are no driver constraints.

<b>DAP_JC</b>	<b>DAP_BC</b>	<b>Description</b>
<code>inAnimate *</code>	<code>inanimate *</code>	Animation data
<code>inBodies</code>	<code>inBodies</code>	Body data
<code>inConst</code>		Stating the number of constraints if any
<code>inForces *</code>	<code>inForces *</code>	Applied force data
<code>inFuncs *</code>	<code>inFuncs *</code>	Time function type and data
	<code>inJoints</code>	Kinematic joint data
<code>inPoints *</code>	<code>inPoints *</code>	Point data
<code>inUvectors *</code>	<code>inUvectors *</code>	Unit vector data
<code>user_force *</code>	<code>user_force *</code>	Non-standard user force model
<code>ChainOpen</code>		Open chain kinematics
<code>CutJoint</code>		Cut joint(s) constraint entities

\* These files are the same in both programs. For a description of these files refer to the user manual for `DAP_BC`.

To construct a model, we should follow practically the same steps as in `DAP_BC` for indexing bodies, points, and unit vectors. The main difference here is that for an open-chain system, whether it is cut or originally open, we **must** index the bodies in an **ascending** order starting from the ground (as body “0”) and continuing towards the leaves.

**M-file `inBodies`:** In this file we define the mass and moment of inertia for the bodies, and initial conditions for all the joint coordinates and velocities. If a body is the owner of a floating joint, we must define three joint coordinates—the translational and rotational coordinates of the body—and the corresponding joint velocities. If the body is the owner of a revolute or a translational joint, then only one joint coordinate and one joint velocity needs to be defined.

```

function inBodies
    include_global

    B1 = Body_struct;
    B1.m = 5.0; B1.J = 4.0;
    B1.theta = [1.0 0.5 pi/6]; % floating joint
    B1.theta_d = [0 0 0];    % default values

    B2 = Body_struct;
    B2.m = 3.0; B2.J = 2.0;
    B2.theta = 0.5;   % revolute or translational joint
    B2.theta_d = 0;  % default value

    ...

```

```
Bodies = [B1; B2; ...];
```

**M-file inPoints:** In this file we provide body-fixed coordinates,  $s_i^P$ , for all the defined points, including the points on the ground. Construction of this file is the same as that of DAP\_BC.

**M-file inUvectors:** In this file we provide the body-fixed components,  $u_i$ , of the defined unit vectors, including the unit vectors on the ground. Construction of this file is the same as that of DAP\_BC.

**M-file inForces:** In this file we define the forces that act on the bodies, such as gravity, spring-dampers, or other force elements. Construction of this file is the same as that of DAP\_BC.

**M-file inFuncts:** A user may define a function such as  $f = f(t)$ , and apply it to describe the kinematics of a driver motor, for example, as function of time. The structure for these elements is the same as that of DAP\_BC.

**M-file inAnimate:** This file provides data for the animation of the simulated response. Construction of this file is the same as that of DAP\_BC.

**M-file user\_force:** In this file we define nonstandard forces/torques that act on the bodies. Construction of this file is the same as that of DAP\_BC.

There is no need to construct an M-file inJoints as needed in DAP\_BC. The necessary cut-joint constraints are formulated in the M-file CutJoint as will be described in the upcoming sections.

**M-file inConst:** In this file we state the number of constraints that exists in a model either due to the cut joints or drivers. The number of constraints associated with different cut joints and driver is listed in Table C.1.

Table C.1 Cut joints and the corresponding number of constraints

	Constraint type	Number of constraints
Cut joint	Revolute (pin)	2
	Translational (sliding)	2
	Revolute-revolute	1
	Revolute-translational	1
	Driver	1

```
function inConst
include_global

nConst = 2;
```

If there are no constraints in a model, we still need to provide this M-file and state that `nConst = 0`.

**M-file ChainOpen:** The user must provide this M-file for any model, whether an open-chain or a cut open-chain system. The M-file contains three parts: (a) describing the forward kinematics of the system; (b) constructing the  $\mathbf{B}$  matrix; and (c) constructing the  $\dot{\mathbf{B}}\dot{\theta}$  array.

```

function ChainOpen(check, theta, theta_d, t)
% Recursive coordinate transformations
    include_global

if check <= 2
    (a) Provide statements for forward kinematics to update coordinates
        of points and components of vectors
end
if check == 2
    (b) Provide statements to construct and evaluate matrix B
elseif check == 3
    (c) Provide statements to construct and evaluate  $\dot{\theta}$  array
end
end

```

In this function M-file the value of the parameter `check` directs the function to evaluate the required entities. This parameter is set by the parent function to 1, 2 or 3. The following is a more detailed description of the three required statements.

(a) *Forward kinematics* (`check = 1 or 2`)

In this part the user must provide statements to compute the coordinates of every point in the system starting from the base and moving toward every leaf. This computation requires updating:

$\mathbf{A}_i$  for all the bodies

$\mathbf{s}_i^P$  and  $\mathbf{r}_i^P$  for all the points

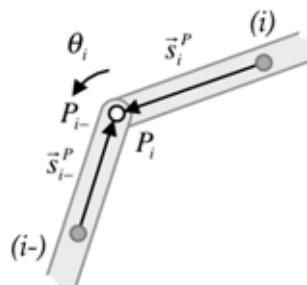
$\mathbf{u}_i$  for all the unit vectors

For these computations the user should follow the recursive kinematic process that is described in Section 9.1.2. To simplify this process for the user, the program provides the following three functions based on the type of kinematic joint that exists in the open-chain path.

### Revolute Joint

```
function Coord_Rev(Pim, Pi, theta)
```

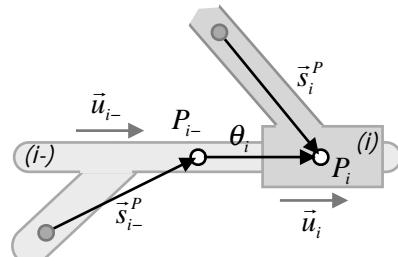
- The user must provide to this function indices of points  $P_{i-}$  and  $P_i$ , and the value of the joint coordinate  $\theta_i$ .
- This function assumes that the rotational transformation matrix  $\mathbf{A}_{i-}$  has already been evaluated. The function evaluates  $\mathbf{s}_{i-}^P$ ,  $\bar{\mathbf{s}}_{i-}^P$ ,  $\mathbf{r}_{i-}^P$ ,  $\phi_i$ ,  $\mathbf{A}_i$ ,  $\mathbf{s}_i^P$ ,  $\bar{\mathbf{s}}_i^P$ ,  $\mathbf{r}_i^P$ , and  $\mathbf{r}_i$ .



### Translational Joint

```
function Coord_Tran(Pim, Uim, Pi, Ui, theta)
```

- The user must provide to this function indices of points and unit vectors  $P_{i-}$ ,  $\vec{u}_{i-}$ ,  $P_i$  and  $\vec{u}_i$ , and the value of the joint coordinate  $\theta_i$ .
- This function assumes that the rotational transformation

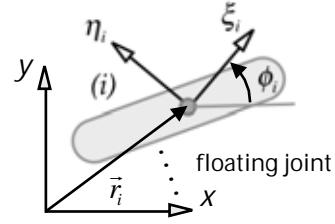


matrix  $\mathbf{A}_{i-}$  has already been evaluated. The function evaluates  $\mathbf{s}_i^P$ ,  $\check{\mathbf{s}}_i^P$ ,  $\mathbf{r}_i^P$ ,  $\mathbf{u}_{i-}$ ,  $\check{\mathbf{u}}_{i-}$ ,  $\phi_i$ ,  $\mathbf{A}_i$ ,  $\mathbf{s}_i^P$ ,  $\check{\mathbf{s}}_i^P$ ,  $\mathbf{r}_i^P$ ,  $\mathbf{u}_i$ ,  $\check{\mathbf{u}}_i$  and  $\mathbf{r}_i$ .

### Floating Joint

```
function Coord_Float(Bi, theta)
```

- The user must provide to this function the index of the body, and the value of three joint coordinates.
- This function assigns the joint coordinates to  $\phi_i$  and  $\mathbf{r}_i$ , then it evaluates matrix  $\mathbf{A}_i$ .



In addition to the points and vectors in the open chain path that are updated, any other points and vectors that are not in the path (for example those that belong to the force elements), their coordinates and components must also be updated. To simplify this process for the user, the program provides the following two functions.

### Update Point

```
function Update_P(Pi)
```

- The user must provide to this function the index of the point that its coordinates need to be updated.
- This function updates  $\mathbf{s}_i^P$ ,  $\check{\mathbf{s}}_i^P$ , and  $\mathbf{r}_i^P$ .

### Update Vector

```
function Update_U(Ui)
```

- The user must provide to this function the index of the vector that its components need to be updated.
- This function updates  $\mathbf{u}_i$ , and  $\check{\mathbf{u}}_i$ .

#### (b) $\mathbf{B}$ matrix (check = 2)

In this part the user must provide statements to compute the  $\mathbf{B}$  matrix. At this point all the coordinates and vectors  $\mathbf{s}_i^P$ ,  $\check{\mathbf{s}}_i^P$ ,  $\mathbf{r}_i^P$ ,  $\mathbf{u}_i$  and  $\check{\mathbf{u}}_i$  have been updated. We need to define and compute the components of any  $\dot{\mathbf{d}}_{i,j}$  vector as needed.

#### (c) $\dot{\mathbf{B}}\dot{\theta}$ array (check = 3)

In this part the user must provide statements to compute the  $\dot{\mathbf{B}}\dot{\theta}$  array. At this point all the coordinates and velocities have been updated. We need to compute components of any  $\dot{\mathbf{d}}_{i,j}$  vector as needed.

As a simple example for an open chain system for modeling with DAP\_JC, we consider the sliding pendulum from Chapter 8 that we used to demonstrate how to set up a model for DAP\_BC. Originally this example was considered for FBD construction in Section 6.1.3.

### Example C.1 (open-chain)

To begin constructing a model for the sliding pendulum, we follow the process that is described in detail in Chapter 9. We first separate the bodies, assign indices to the bodies, attach to each a body-fixed frame, and define a global  $x$ - $y$  frame. As shown in Figure C.2, the two moving bodies are marked as (1) and (2) in an ascending order that is essential for the program DAP\_JC. We have identified three points as  $O_0$ ,  $B_1$  and  $B_2$  that are needed for the joints and the spring attachment points. We have assigned indices to these points as [1], [2], and [3] (the order of numbering is up to us). We also need two unit vectors for the translational joint:  $\vec{u}_0$  on the ground and  $\vec{u}_1$  on the slider. These unit vectors are numbered [[1]] and [[2]]. We are now ready to set up the M-files for this model. The constant dimensions are  $a = 0.2$  m and  $L = 1.0$  m.

The constructed M-files for this example can be found in `Models/SP`.

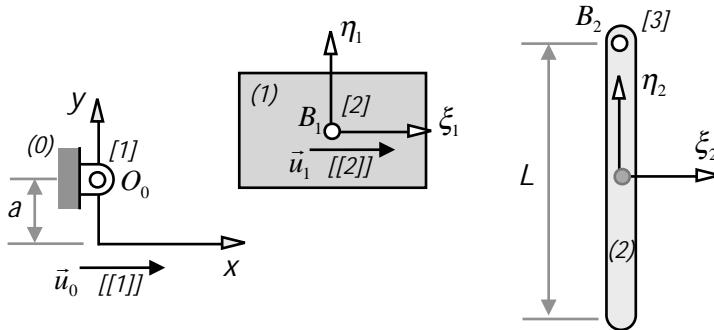


Figure C.2 Individual bodies and the assigned indices for the sliding pendulum example.

The data, for the mass, moment of inertia, and initial conditions on the joint coordinates and velocity for each body is provided in the function M-file `inBodies`.

$$\theta_1 = 1.0$$

$$\dot{\theta}_1 = 0$$

$$\theta_2 = \pi/6$$

$$\dot{\theta}_2 = 0.3\pi \text{ rad/sec}$$

```
function inBodies
    include_global

    B1 = Body_struct;
    B1.m = 5.0; B1.J = 4.0;
    B1.theta = 1.0;

    B2 = Body_struct;
    B2.m = 2.0; B2.J = 0.2;
    B2.theta = pi/6;
    B2.theta_d = 0.3*pi;

    Bodies = [B1; B2];
```

The following function M-files are the same as in the DAP\_BC model:

`inAnimate`, `inForces`, `inFuncts`, `inPoints`, `inUvectors`

Since there are no constraints in this model, we need to provide `nConst = 0` in this function M-file.

```
function inConst
    include_global

    nConst = 0;
```

The last function M-file to construct is the following:

(a) *Forward kinematics*

The forward kinematics starts from the ground

```
function ChainOpen(check, theta,
                    theta_d, t)
```

and goes through the translational joint using the translation joint function `Coord_Tran`. The indices refer point  $O_0$ , unit vector  $\bar{u}_0$ , point  $B_1$ , and unit vector  $\bar{u}_1$  respectively, and the last argument contains the value of  $\theta_1$ .

Next the path goes through the revolute joint that uses the revolute joint function `Coord_Rev`. The indices refer to points  $B_1$  and  $B_2$ , and the value of  $\theta_2$ .

(b) **B matrix**

The **B** matrix for this example is determined as

$$\mathbf{B} = \begin{bmatrix} \mathbf{u}_0 & \mathbf{0} \\ 0 & 0 \\ \mathbf{u}_0 & \check{\mathbf{d}}_{2,2} \\ 0 & 1 \end{bmatrix}$$

(c)  **$\dot{\mathbf{B}}\dot{\theta}$  array (check = 3)**

For the sliding pendulum example, the  $\dot{\mathbf{B}}\dot{\theta}$  array is determined as

$$\dot{\mathbf{B}}\dot{\theta} = \begin{Bmatrix} \mathbf{0} \\ 0 \\ \check{\mathbf{d}}_{2,2}\dot{\theta}_2 \\ 0 \end{Bmatrix}$$

```
include_global

if check <= 2

    Coord_Tran(1, 1, 2, 2, theta(1));

    Coord_Rev(2, 3, theta(2));
end

if check == 2
    u1 = Uvectors(2).u;
    d22 = -Points(3).sP;
    z2 = [0; 0];
    Bmat = [u1 z2
             0 0
             u1 s_rot(d22)
             0 1];

elseif check == 3
    d22d = -Points(3).sP_d;

    Bd = [0
           0
           0
           s_rot(d22d)*theta_d(2)
           0];
end
end
```

Now we have all the files to simulate the response of this system.

For systems containing closed chains or drivers, we construct all the necessary M-files for the cut system, as for any other open chain system. Then we provide the following function M-file to describe the necessary entities for the cut joint and/or driver constraints.

**M-file CutJoint:** The user must provide this M-file for any model that contains cut joints and/or driver constraints. The M-file contains four parts defining: (a) the Jacobian matrix **C**; (b) the coordinate constraints; (c) the right-hand-side array of the velocity constraints; and (d) the right-hand-side array of the acceleration constraints; i.e.,  $-\dot{\mathbf{C}}\dot{\theta}$ .

```
function CutJoint(check, theta, theta_d, t)
% Cut joint and driver constraints
    include_global

        (a) Provide the Jacobian matrix of the constraints
if check == 1
        (b) Provide the coordinate constraints
end
if check == 2
        (c) Provide the right-hand-side array of the velocity constraints
```

```

elseif check == 3
    (d) Provide the right-hand-side array of the acceleration
        constraints
end
end

```

In this function M-file the value of the parameter `check` directs the function to evaluate the required entities. This value, which could be 1, 2 or 3, is set by the parent function. The following is a more detailed description of the four required set of statements.

(a) *Jacobian matrix C*

In this part the user must provide statements to compute the Jacobian matrix **C** for all the constraints. This matrix will be evaluated whether the parameter `check` is equal to 1, 2 or 3.

(b) *Cut-joint coordinate constraints*

The user must provide statements to evaluate all the constraints, whether cut joint or driver. If the coordinate values do not satisfy these constraints, the program will correct them; e.g., in the initial condition correction step. These constraints will be evaluated when the parameter `check` is set to 1 by the program.

(c) *Right-hand side array of velocity constraints*

The right-hand side array of velocity constraints for any cut joint should be zero, but for a driver constraint it could be a constant non-zero value or a time dependent function. This array will be evaluated when the parameter `check` is set to 2 by the program.

(d) *Right-hand side array of acceleration constraints  $-\dot{\mathbf{C}}\dot{\theta}$*

The right-hand side of acceleration constraints for a cut joint most likely contains quadratic velocity terms, but for a driver constraint it could be zero, constant non-zero value, or a time dependent function. This array will be evaluated when the parameter `check` is set to 3 by the program.

Next example shows how to incorporate constraints for a cut joint.

### Example C.2 (closed-chain)

In this example we consider the slider-crank mechanism from Chapter 9, Example 9.4. In this model, however, we use relative angle for the rotational joint coordinate at pin joint *A* and not an absolute angle as used in Example 9.4.

To begin constructing a model for the slider-crank mechanism, we cut the system at revolute joint *B*, we separate the bodies, assign indices to the bodies, attach to each a body-fixed frame, and define a global *x-y* frame. We have identified six points as  $O_0$ ,  $O_1$ ,  $A_1$ ,  $A_2$ ,  $B_2$  and  $B_3$ , and two unit vectors  $\vec{u}_0$  and  $\vec{u}_3$ . We then assign indices to the points and the unit vectors as shown on the figure.

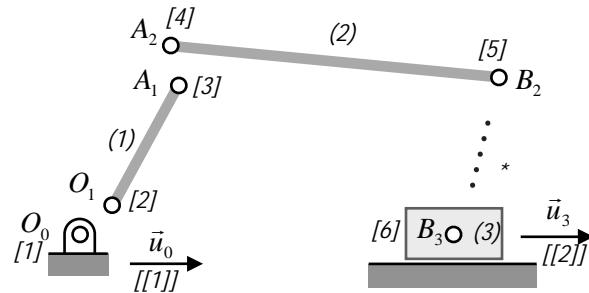


Figure C.3 The slider-crank mechanism from Example 9.4.

The constructed M-files for this example can be found in `Models/SC`. Most of the input M-files are the same as the model for the DAP\_BC program. In the function M-file `inBodies`, for the defined three bodies, the initial conditions are stated based on the joint coordinates as  $\theta_1 = \phi_1$ ,  $\theta_2 = \phi_2 - \phi_1$ , and  $\theta_3 = x_3$ . However, when we execute the program, we should ask the program to correct the initial conditions in case the stated values do not satisfy the cut-joint constraints.

Since we have cut a revolute joint there are two constraints in this model.

```
function inConst
    include_global
    nConst = 2;
```

The function M-file `ChainOpen` is constructed as in the following.

(a) *Forward kinematics*

Since there are two trees in this system, the forward kinematics for the first tree starts from the ground through the revolute joint  $O$  ( $O_0$  and  $O_1$ ) to body (1), then through the revolute joint  $A$  ( $A_1$  and  $A_2$ ) to body (2). For the second tree, from the ground we go through the translational joint (point  $O_0$ , unit vector  $\vec{u}_0$ , point  $B_3$ , and unit vector  $\vec{u}_3$ ) to body (3).

Point  $B_2$  on body (2) is not in the path of the forward kinematics. Therefore we update its coordinates using the `Update_P` function.

(b) **B** matrix

The **B** matrix for this example is determined as

$$\mathbf{B} = \begin{bmatrix} \check{\mathbf{d}}_{1,1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 \\ \check{\mathbf{d}}_{2,1} & \check{\mathbf{d}}_{2,2} & \mathbf{0} \\ 1 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{u}_3 \\ 0 & 0 & 0 \end{bmatrix}$$

```
function ChainOpen(check, theta, theta_d, t)
    include_global

    if check <= 2
        Coord_Rev(1, 2, theta(1));
        Coord_Rev(3, 4, theta(2));
        Coord_Tran(1, 1, 6, 2, theta(3));

        Update_P(5);

    end

    if check == 2
        d11 = -Points(2).sp;
        d21 = Bodies(2).r;
        d22 = -Points(4).sp;
        u3 = Uvectors(2).u;
        z2 = [0; 0];
        Bmat = [s_rot(d11) z2 z2
                 1 0 0
                 s_rot(d21) s_rot(d22) z2
                 1 1 0
                 z2 z2 u3
                 0 0 0];
    end
```

(c)  $\dot{\mathbf{B}}\dot{\theta}$  array ( $\text{check} = 3$ )

The  $\dot{\mathbf{B}}\dot{\theta}$  array is constructed as

$$\dot{\mathbf{B}}\dot{\theta} = \begin{pmatrix} \ddot{\mathbf{d}}_{1,1}\dot{\theta}_1 \\ 0 \\ \ddot{\mathbf{d}}_{2,1}\dot{\theta}_1 + \ddot{\mathbf{d}}_{2,2}\dot{\theta}_2 \\ 0 \\ \mathbf{0} \\ 0 \end{pmatrix}$$

```

elseif check == 3
d11d = -Points(2).sP_d;
d21d = Bodies(2).r_d;
d22d = -Points(4).sP_d;
Bd = [s_rot(d11d)*theta_d(1)
      0
      s_rot(d21d*theta_d(1) +
             d22d*theta_d(2))
      0
      0
      0];
end
end

```

Since this model contains a cut joint, we need to provide the function M-file `CutJoint` as shown in the following. In this file we provide statements for (a) the Jacobian matrix, (b) the constraints on the coordinates, (c) the right-hand side array of the velocity constraints, and (d) the right-hand side array of the acceleration constraints.

(a) *Jacobian matrix C*

The  $\mathbf{C}$  matrix for this example is constructed as

$$\mathbf{C} = \begin{bmatrix} \ddot{\mathbf{d}}_{*,1} & \ddot{\mathbf{d}}_{*,2} & -\mathbf{u}_3 \end{bmatrix}$$

Note that this matrix is different from the one in Example 9.4 because we are using relative joint coordinate for  $\theta_2$ .

(b) *Coordinate constraint*

The only constraints are from the cut-joint at the pin joint *B*.

(c) *Right-hand side array of velocity constraints*

For a cut joint this is a zero array.

(d) *Right-hand side array of acceleration constraints*

For the cut pin joint this array is constructed as

$$-\dot{\mathbf{C}}\dot{\theta} = -(\ddot{\mathbf{d}}_{*,1}\dot{\theta}_1 + \ddot{\mathbf{d}}_{*,2}\dot{\theta}_2)$$

```

function CutJoint(check, theta, theta_d, t)
    include_global

    dcu1 = Points(5).rP;
    dcu2 = Points(5).rP - Points(4).rP;
    u3 = Uvectors(2).u;
    C = [s_rot(dcu1)   s_rot(dcu2)   -u3];

    if check == 1
        Phi = Points(5).rP - Points(6).rP;

    elseif check == 2
        rhsV = [0
                  0];

    elseif check == 3
        dcu1d = Points(5).rP_d;
        dcu2d = Points(5).rP_d -
                 Points(4).rP_d;
        rhsA = -s_rot(dcu1d*theta_d(1) +
                      dcu2d*theta_d(2));
    end
end

```

Now we have all the files to simulate the response of this system.

Next example shows how to incorporate a driver constraint in a model.

### Example C.3 (closed-chain and a driver)

In this example we add a driver constraint to the preceding slider-crank mechanism. The driver represents a constant angular speed motor that acts about the pin joint *O*. The constructed M-files for this example can be found in `Models/SCd`. Most of the input M-files are the same as those in the SC model. In the following we show only the M-files that have been revised.

Since we have cut a revolute joint and a driver constraint, the model contains three constraints.

The driver function for this example is  $\theta_1 = \pi/3 + 2\pi t$ . We add this constraint and its corresponding entities to the cut-joint constraints in the following function M-file.

The added entities for the driver constraints are:

(a) Jacobian matrix  $\mathbf{C}$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

(b) Coordinate constraints

$$\theta_1 - \pi/3 - 2\pi t = 0$$

(c) Right-hand side array of velocity constraints

$$\dot{\theta}_1 = 2\pi$$

(d) Right-hand side array of acceleration constraints

$$-\dot{\mathbf{C}}\dot{\theta} = 0$$

```
function inConst
    include_global
    nConst = 3;

function CutJoint(check, theta, theta_d, t)
    ...
    C = [ s_rot(dcut1) s_rot(dcut2) -u3
          1                 0                 0];
    ...

    if check == 1
        ...
        Phi = [(Points(5).rP - Points(6).rP)
                (theta(1) - pi/3 - 2*pi*t)];

    elseif check == 2
        rhsV = [0
                 0
                 2*pi];

    elseif check == 3
        ...
        rhsA = -[s_rot(dcut1d*theta_d(1) +
                        dcut2d*theta_d(2))
                  0];
    end
```

We now have all the files to simulate the response of this system.

## C.4 POSTPROCESSING

The program `dap` simulates the dynamic response of a system by integrating the equations of motion from the initial time  $t = 0$  to the final time. The program uses the MATLAB's integrator `ode45`, which reports the results at every  $\Delta t$  seconds (time period) as specified by the user<sup>1</sup>. At every reporting time step, the integrator saves the time in an array named `T`, and saves the joint coordinates and velocities for all the open-chain joints in an array named `uT`. None of the other results, such as joint accelerations and Lagrange multipliers associated with the cut joints or drivers, are saved. The joint coordinates and velocities are saved column-wise. For example, if we integrate the equations of motion of the sliding-pendulum system that contains two joints for 4.0 seconds, and ask for  $\Delta t = 0.02$  reporting intervals, the arrays will be formed as listed in Table C.2.

The program `post` takes the computed values for the joint coordinates and velocities from `uT` at each reporting time step. It recovers the missing results by solving the equations of motion again at every time step, and saving all the results in several arrays as listed in Table C.3. The user should refer to these arrays for performing further analyses with the simulated results.

---

<sup>1</sup> The reporting time step is not the same as the integration time step. Integration time step used by `ode45` is variable and most likely much smaller than the reporting time step.

Table C.2 Arrays T and uT contain the output from the integrator `ode45`

$t$	$\theta_1$	$\theta_2$	$\dot{\theta}_1$	$\dot{\theta}_2$
0.00	1.00	0.52	0.00	0.00
0.02	0.99	0.52	-0.00	-0.13
...	...	...	...	...
3.98	1.06	-0.01	-0.45	1.32
4.00	1.05	0.02	-0.48	1.37

Table C.3 List of arrays constructed by the program post

Array	Size	Description
theta	nt $\times$ nJC	Joint coordinates
theta_d	nt $\times$ nJC	Joint velocities
theta_dd	nt $\times$ nJC	Joint accelerations
r	nt $\times$ nB $\times$ 2	Translational body coordinates
rd	nt $\times$ nB $\times$ 2	Translational body velocities
p	nt $\times$ nB	Rotational body coordinates
pd	nt $\times$ nB	Rotational body velocities
rP	nt $\times$ nP $\times$ 2	Coordinates of points
rPd	nt $\times$ nP $\times$ 2	Velocity of points
Jac	nt $\times$ nConst $\times$ nJC	Jacobian matrix
Lam	nt $\times$ nConst	Lagrange multipliers

nt	Number of time steps (rows) including $t = 0$
nB	Number of bodies
nP	Number of points
nConst	Number of constraints
nJC	Number of joint coordinates

## C.5 APPLICATION EXAMPLES

In addition to the three examples that we have already discussed for open and closed chain systems, several other examples are provided in this package. All of the models reside in the folder `Models`. Some of these additional models are discussed in Chapter 8 of the textbook and in the user manual for the program `DAP_BC`. In the following, we provide a very short comment for each of these models. It is recommended that the reader should simulate each model for a few seconds and then observe the animation of the simulated response. This should provide a simple visual description of each model. Detailed construction of each model is similar to the ones in Examples C.1, C.2, and C.3.

### Double A-Arm Suspension (`DAP_JC/Models/AA`)

This model was discussed in detail in Section 8.1.1 of the textbook for modeling with `DAP_BC`. To model this system for `DAP_JC` simulation, the pin joint at *B* is cut resulting in two constraints in this model.

### MacPherson Suspension (DAP\_JC/Models/MP\_A, MP\_B)

These models were discussed in detail Section 8.1.2 of the textbook for modeling with DAP\_BC.

**MP\_A:** In this model that contains three moving bodies, the translational joint is cut resulting in two constraints.

**MP\_B:** In this model that consists of two moving bodies, the revolute-translational joint is cut. This cut joint requires only one constraint equation.

### Cart (DAP\_JC/Models/Cart\_C)

In Section 8.1.3 we discussed a simple example of a cart to demonstrate how to model a disc (wheel) rolling on the ground without slip. Four versions of the model were presented with different types of drivers. To demonstrate how to model a rolling disc without slip in DAP\_JC program, we have selected one of the models, Cart\_C.

The model contains three bodies, two revolute joints, and two wheels that roll on the flat ground without slip. Since the system is a closed chain, we cut the rolling disc on the front wheel; i.e., between body (3) and the ground, to form a cut open-chain system. Therefore we have one rolling disc in the forward kinematics of the open tree between body (1) and the ground, and one rolling disc that is cut and must be represented by constraints. A review of the M-files ChainOpen and CutJoint should clarify these concepts.

### Rod Impacting the Ground (DAP\_JC/Models/Rod)

A rod impacting the ground repeatedly was discussed for modeling with DAP\_BC in Section 8.1.5. The same example is considered here for modeling with DAP\_JC. Since there are no kinematic joints in this system, the joint coordinate model is practically identical to the body coordinate model.

### Two Connected Rods Impacting the Ground (DAP\_JC/Models/Rod2)

This example represents two rods that are connected by a pin joint drop under the force of gravity and contact the ground. This model allows three points to be candidates for impacting the ground.

### Triple Pendulums (DAP\_JC/Models/TP)

The triple pendulum shown in Figure 5.7 of the textbook (with relative joint coordinates) is modeled in this example with DAP\_JC. This is an open chain system that does not require cutting any joints.

# **Index**

---

## **A**

Absolute angle  
open-chain systems, 195–197  
unit vectors, 90–91

Absolute coordinates, 47

Acceleration

angular, 3, 4  
of body, 44–46, 199  
of mass centers, 95  
of planar rigid body, 44–45  
slider-crank mechanism, 127, 128  
vectors, 38, 44, 45, 92, 93, 99

AC method *see* Appended constraint (AC) method

Actuator, 69–70

Adding constraints, 337–340

Algebraic conditions, 139

Algebraic constraints, 97, 148, 233, 236, 305, 311

Algebraic vector, 13, 20, 37, 40

Angular momentum, 262

Appended constraint (AC) method, 288, 290, 294

Applied forces, multibody dynamics, 69–74

Arrays, 136, 137–138, 152, 222

Cartesian reference axes, 20–21  
force, 254  
partial derivatives, 28–31  
time derivatives, 27–28  
used by idap for saves, 303  
used by program kap saves, 296

## **B**

Belted dummy, multibody model, 381–385

Block matrices, 193, 198

B-M friction model, 357

Body-coordinate formulation, 135, 185, 229, 333, 338

equations of motion  
constrained, 152–160  
unconstrained, 136, 149–152

kinematic joints (*see* Kinematic joints)

planar multibody system, 135–136

problems, 161–168

total energy, 160–161

Body-fixed frame, 38, 42–43, 145, 170, 187, 190  
Bracket joint *see* Rigid joint

## **C**

Cart

models/cart\_A, 177–178  
models/cart\_B, 178–179  
models/cart\_C, 179–180  
models/cart\_D, 180

Cartesian reference axes, 13, 14

CCW direction *see* Counterclockwise (CCW) direction

Centroidal equations of motion, 63–67

Classical method, point-coordinate formulation, 229–231

Clockwise (CW) direction, 14

Closed-chain systems, 46

cut-joint constraints, 203–206  
driver constraint, 220–221  
equations of motion, 206–213  
four-bar mechanism, 101–105  
initial conditions, 217–218  
Jacobian matrix, 213–217  
kinematic analysis, 108  
reaction forces, 218–220  
six-bar mechanism, 105–108  
slider-crank mechanism, 97–101  
vector kinematics in, 89

Coefficient matrix, 26, 107, 130, 190, 199, 234, 249, 307

Collinear vectors, 17

Column matrix, 23

Compact forms, equation, 31–32

Complementary analyses

deformable body, 360–361  
friction, 356–360  
initial condition correction, 353–355  
problems, 361–363  
redundant constraints, 355–356  
static analysis, 349–350  
static equilibrium, 351–353

Constrained equations, body-coordinate formulation, 49–51, 152–160

Constraint violation stabilization method, 323–326

Constructing vectors, 97

Contact force model, 274, 278, 279  
 Continuous analysis method  
   body contacts surface, 274–276  
   two-body contact, 277–281  
 Conveyor belt, 180–182, 358  
 Coordinate partitioning (CP) method, 288, 290, 326–328  
 Coulomb friction, 76–77  
 Counterclockwise (CCW) direction, 14, 17  
 CP method *see* Coordinate partitioning (CP) method  
 Crank-rocker four-bar mechanism, 366–367  
 Crash tests, 381–385  
 Creeping robot, 381–382  
 Cut-joint constraints, closed-chain systems, 203–206, 215

**D**

DAP\_BC *see* Dynamic analysis program with body coordinates (DAP\_BC)  
 DAP\_JC *see* Dynamic analysis program with joint coordinates (DAP\_JC)  
 Deformable body, multibody dynamics, 360–361  
 Deformable rod (a beam), 360–361  
 Degrees of freedom (DoFs), 7, 48–49, 89, 142, 185, 305, 321, 349  
 Deleting constraints, 337–340  
 Dependent coordinates, 48, 49, 326, 327  
 Diagonal matrix, 23, 115, 149  
 DoFs *see* Degrees of freedom (DoFs)  
 Dot product vectors, 17  
 Double A-arm suspension, 170–173, 371–372  
 Double-loop system, 46  
 Double pendulum, 5–9, 46  
 Double slider mechanism, schematic presentation of, 215–216  
 Driver constraint  
   closed-chain systems, 220–221  
   kinematic joints, 147–148  
 Dump truck, multibody model, 377–381  
 Dwell mechanism, 107–108, 368–369  
 Dynamic analysis program with body coordinates (DAP\_BC), 169–170, 222  
   cart model, 177–180  
   conveyor belt and friction, 180–182  
   double A-arm suspension, 170–173  
   MacPherson suspension system, 173–177  
   problems, 183–184  
   slender rod, 182–183  
 Dynamic analysis program with joint coordinates (DAP\_JC), 221–222

**E**

Elliptical exercise machine, 386–388  
 EQMs *see* Equations of motion (EQMs)  
 Equations  
   constraints, 233–238  
   expanded forms, 31–32  
 Equations of motion (EQMs), 113, 114, 124, 126, 128  
   body-coordinate formulation  
     constrained, 152–160  
     unconstrained, 149–152  
   centroidal, 63–67  
   closed-chain systems, 206–213  
   free-body diagram, 120  
   inverse dynamic, 297–298  
   open-chain system, 197–203  
   point-coordinate formulation, 241–242  
   unconstrained formulation, 311–312  
 Equilibrium, 58  
 Expressions, 31–32

**F**

FBDs *see* Free-body diagrams (FBDs)  
 Fictitious damping method, 352, 353  
 Film-strip advancer, 365–366  
 Finite-element method, 360  
 Force analysis, 126–127 *see also* Free-body diagrams (FBDs)  
   four-bar mechanism, 128–129  
   process of force analysis, 130  
   slider–crank mechanism, 127–128  
 Force distribution, 242–245, 254–255  
 Forward dynamic analysis  
   adding/deleting constraints, 337–340  
   combined analyses, 340–341  
   constrained formulation  
     integration arrays, 319–320  
     integration procedure, 321–323  
     variable-length pendulum, 320–321  
   constraint violation  
     coordinate partitioning method, 326–328  
     joint-coordinate method, 331  
     momentum method, 331–333  
     penalty method, 328–331  
     stabilization method, 323–326  
   contact and impact concepts, 333–337  
   problems, 341–347  
   unconstrained formulation  
     equations of motion, 311–312  
     initial value problems, 312–313  
     integration time-step size, 315–318

- procedure, 318–319
- Runge–Kutta algorithm, 313–315
- F**
  - Four-bar
    - inverse dynamics, 302–305
    - kinematics, MATLAB program for, 294–297
    - linkage, 122–123
    - mechanism, 46–49, 101–105, 107, 238
      - free-body diagram for, 122–125, 128–129
      - schematic presentation of, 213–214
  - Free-body diagrams (FBDs), 5–9, 135, 150, 152, 154, 156, 158, 298, 301
    - equations of motion, 126
    - force analysis, 126–127
      - four-bar mechanism, 128–129
      - process of force analysis, 130
      - slider–crank mechanism, 127–128
    - four-bar mechanism, 122–125
    - problems, 130–133
    - representation, 300
    - slider–crank mechanism, 120–122
    - sliding pendulum, 118–120
    - two-body system, 113–118
  - Friction force, 180–182, 356–360
    - coulomb or viscous, 76–77
    - dynamic friction, 78–79
    - model, 273
    - motor/driver, 82–83
    - wheel/tire, 80–82
- G**
  - Geometric vector, 13, 16
  - Grashof, 296
  - Gravitational force, 69, 302
  - Gravity, 5, 69, 351
- H**
  - Half-car, multibody model, 374
  - Head and neck, multibody model, 385–386
  - Hertz contact force model, 277
  - Hydraulic actuator, 355–356, 377
  - Hysteresis damping factor, 278
- I**
  - ID1 algorithm, 298–299, 301
  - ID2 algorithm, 299–302
  - Independent coordinates, 48, 326, 328
  - Initial condition correction, 353–355
  - Inverse dynamic analysis
    - application of, 305–307
  - equations of motion, 297–298
  - ID1 algorithm, 298–299, 301
  - ID2 algorithm, 299–302
  - program for four-bar, 302–305
- Inverted slider–crank mechanism, 207, 215, 216, 237, 372
- J**
  - Jacobian matrix, 49, 138, 148, 149, 213–217, 292, 294
  - Joint-coordinate formulation, 186–187
    - closed-chain systems (*see* Closed-chain systems)
    - MATLAB program, 221–222
    - open-chain systems (*see* Open-chain systems)
    - problems, 222–227
    - recursive kinematics, 188–190
    - reference point, 187–188
  - Joint-coordinate method, 185, 331
- K**
  - KAP\_IDAP *see* Kinematic analysis program–inverse dynamic analysis program
  - Kinematic analysis, 2, 3, 49, 108
    - chain, 46
    - constraints, 135
    - four-bar mechanisms, MATLAB program for, 294–297
    - general formulation for, 285
    - joints, 51–53, 57, 69, 75
    - K algorithm, 286–287
    - modeling process, rule of thumb in, 135
    - nonlinear algebraic equations, 290–291
      - NR1 algorithm, 291–292
      - NRn algorithm, 293–294
    - of planar mechanical systems, 4, 5
    - problems, 307–309
    - purpose of, 108
    - solution procedures, 288–290
  - Kinematic analysis program–inverse dynamic analysis program (KAP\_IDAP), 294–297
  - Kinematic joints, 136–139
    - driver constraint, 147–148
    - Jacobian matrix, 148–149
    - revolute (pin) joint, 139–140
    - revolute–revolute joint, 142–143
    - revolute–translational joint, 144
    - rigid
      - circular disc, 146–147

- Kinematic joints (*cont.*)
  - joint, 145–146
  - simple constraint, 146
  - translational (sliding) joint, 141–142
- L**
- Lagrange multipliers, 153–160, 212, 218–219, 298, 299, 328, 349–350
  - Linear actuator, 147–148
  - Linear algebraic equations, 26, 99, 149, 217, 285, 288, 290, 353
  - Linear momentum, 261–263
  - Logical point-to-point spring–damper, 274
  - L-U factorization, 290, 356
- M**
- MacPherson suspension system, 372–374
    - models/MP\_A, 173–175
    - models/MP\_B, 175–176
    - models/MP\_C, 176–177
  - Magnitude of torque, 63
  - Mass
    - centers, velocity and acceleration of, 95
    - condensation
      - three primary points, 253–254
      - two primary points, 248–253
    - distribution, 246–248, 254–255
    - matrix, 202–203, 250, 252–254, 268
    - moment of inertia, 63, 65
    - spring contact model, 274–275
  - Massless link, 116–117, 155, 176
  - MATLAB®
    - four-bar kinematics, program for, 294–297
    - functions, 10–11, 71
    - numerical calculations, 32–35
  - Matrices, 21
    - forms, 31–32
    - linear algebraic equations, 26
    - in MATLAB®, 22
    - operations, 23–26
    - partial derivatives, 28–31
    - problems, 32–35
    - square, 23
    - time derivatives, 27–28
  - Mechanical systems, multibody, 1–2
    - analyses, types of, 2
    - computer programs, 9–11
    - formulation, methods of, 2–9
  - Middle pendulum, “T-section” of, 95
  - Momentum method, 331–333
- Motorcycle multibody model, 377
- Motor/driver, multibody dynamics, 82–83
- Mountain bike multibody model, 374–377
- Moving reference frame, 38
- Multi-loop closed-chain system, 46
- N**
- Negative vector, 16
  - Neutral axis, 361
  - Newton–Raphson method, 291–292, 326
  - Newton’s laws of motion
    - first law, 57, 58
    - second law, 57–59, 62, 63, 262
    - third law, 57, 75
  - Noncentroidal equations of motion, planar dynamics, 67–69
  - Non-Grashof, 296
  - Non-Linear algebraic equations, 49–50, 288, 290–294
  - Nonmoving reference frame, 38
  - NR1 algorithm, 291–292
  - NRn algorithm, 293–294
  - Numerical integration algorithm, 333, 334
- O**
- One-dimensional mass–spring system, 315, 316
  - Open-chain systems, 46, 94–97, 190–195
    - absolute angle, 195–197
    - equations of motion, 197–203
    - in problems, 222–227
    - structure of, 185
    - topologies of, 186
    - vector kinematics in, 89, 90
  - Orthonormal matrix, 26, 40
- P**
- Parallel vectors, 17, 19
  - Particle dynamics, system of particles, 58–61
  - Penalty method, 328–331
  - Pendulum, 5, 46, 48, 95, 328
    - double, 5–9
    - single, 316, 327, 351
    - sliding, 94, 118–120
    - variable-length, 158, 199, 312, 332, 352
  - Piecewise analysis
    - constrained bodies, 271–272
    - friction force, 273
    - impact of two particles, 262–266
    - linear momentum, 261–262

- problems, 281–283
  - two bodies, 267–271
  - Pin-in-slot joint** *see* Revolute-translational joint
  - Planar dynamics**
    - friction force, 76–80
    - motor and driver, 82–83
    - wheel and tire, 80–82
  - multibody**
    - variety of force elements, 69–74
    - variety of kinematic joints, 75–76
  - Newton's laws of motion**, 57–58
  - problems, 84–87
  - rigid body**, 61–62
    - centroidal equations of motion, 63–67
    - force and torque, 62–63
    - noncentroidal equations of motion, 67–69
  - system of particles, 58–61
  - work and energy, 83–84
  - Planar kinematics**
    - array of coordinates, 46–48
    - definitions, 46–53
      - array of coordinates, 46–48
      - constraint equations, 49–51
      - degrees of freedom, 48–49
      - kinematic joints, 51–53
    - particle, 37–38
  - problems, 53–56
  - rigid body**, 38–39
    - acceleration of, 44–45
    - coordinates of, 39–43
    - velocity of, 43–44
  - Planar rigid body**
    - acceleration of, 44–45
    - coordinates of, 39–43
    - velocity of, 43–44
  - Point-coordinate formulation**
    - classical method, 229–231
    - constraints, 233–234
      - angle, 235–236
      - length, 235
      - simple, 236–238
    - equations of motion, 241–242
    - force
      - and mass addition, 254–255
      - and torque distribution, 242–245
    - mass
      - condensation, 248–254
      - distribution, 246–248
    - primary and stationary points, 231–233
    - problems, 255–259
    - secondary points, 238–241
  - Point–follower joint**, 51–52
  - Point-to-point spring-damper-actuators**, 167, 172
  - Position vector**, 89–90, 92, 93
  - Primary points, point-coordinate formulation**, 231–233
  - Programming effort**, AC method, 290
- R**
- Reaction forces, body-coordinate formulation**, 153–160
  - Rectangular matrix**, 199
  - Recursive kinematics**
    - revolute joint, 188–189, 195
    - translational joint, 189–190
  - Redundant constraints**, 355–356
  - Reference frames** *see* Cartesian reference axes
  - Reference point**, 187–188
  - Relative coordinates**, 47
  - Relative rotation (motor)**, 177
  - Relative translation (actuator)**, 75, 76, 145
  - Revolute (pin) joint**, 139–140
    - recursive kinematics, 188–189
    - schematic representations of, 51
  - Revolute–revolute joint**, 142–143, 145, 155
  - Revolute–translational joint**, 144, 155
  - Rigid body**
    - dynamics, 61–62, 262
      - centroidal equations of motion, 63–67
      - moment of force and torque, 62–63
      - noncentroidal equations of motion, 67–69
    - kinematics
      - acceleration of, 44–45
      - coordinates of, 39–43
      - definition, 38–39
      - velocity of, 43–44
  - Rigid circular disc**, 146–147
  - Rigid joint**, 145–146
  - Robotics application, inverse dynamics in**, 305–307
  - Rotational actuator**, 73–74
  - Rotational transformation matrix**, 40
  - Row rank (column rank)**, 25
  - Runge–Kutta algorithm**, 313–315
- S**
- Satellite panels**, 389–392
  - Scalars**, 13, 14, 17, 236
    - multiplication of matrix, 24

- Scalars (*cont.*)
  - vector, 16
  - product, 17, 19–21
- Schematic representations
  - revolute joint, 51
  - translational joint, 51–52
- Secondary points, point-coordinate formulation, 238–241
- Simple constraint, kinematic joints, 146
- Single loop system, 46
- Six-bar
  - dwell mechanism, 105–108, 203–204, 368–369
  - quick-return mechanism, 367–368
- Sled test, 381–385
- Slender rod, 182–183, 248–249, 269
- Slider–Crank mechanism, 3, 4, 10, 229–230, 286, 300
  - free-body diagram for, 120–122, 127–128
  - inverted, 207, 215
  - vector loop, 205–206
- Slider *versus.* time, acceleration of, 287
- Sliding pendulum, 94, 118–120
- Spatial, 1, 46
- Special-purpose program, 9–10
- Spring-damper, 80, 165, 377, 383, 384
  - characteristics of, 372
  - force element, 261
  - logical point-to-point, 274, 385
  - representation, 173
- Spring-mass system, 180–181
- Square matrix, 23, 25, 321, 349
- Stationary points, point-coordinate formulation, 231–233
- Stribeck effect, 77
- Swing, multibody model, 388–389
- 
- T**
- Torque, 74, 82
  - distribution, 242–245
  - magnitude of, 63
  - moment of force and, 62–63, 65
- Total energy, body-coordinate formulation, 160–161
- Transition force, 79
- Translational joint
  - recursive kinematics, 189–190
  - schematic representations of, 52
  - sliding, 141–142, 154
- 
- U**
- Unconstrained equations, body-coordinate formulation, 149–152
- Unit vectors, 16, 17, 70, 90–91, 96, 143, 170, 173, 239
- 
- V**
- Variable-length pendulum, 158
- Vector kinematics
  - closed-chain systems
    - four-bar mechanism, 101–105
    - kinematic analysis, 108
    - six-bar mechanism, 105–108
    - slider–crank mechanism, 97–101
  - open-chain systems, 94–97
  - position vector, uses of, 89–90, 92, 93
  - problems, 108–111
  - velocity vector, uses of, 93
- Vector-loop method, 4, 5
- Vectors, 13, 188
  - acceleration, 44
  - algebraic representation of, 15, 17, 20
  - angle of, 14–15
  - components of, 15–16, 39–40
  - definition for, 44
  - direction of, 14
  - loop equations, 89, 90, 97, 106, 107
  - loop representation
    - four-bar mechanism and, 101
    - offset slider–crank mechanism and, 100
    - slider–crank mechanism and, 97
  - magnitude of, 41
  - multiplication, 16
  - partial derivatives, 28–31
  - projection of, 19, 20
  - rotation, 17–18
  - time derivatives, 27–28
  - translational coordinates of body, 39
  - unit, 16–17, 96, 170, 173, 239
  - zero, 17, 61, 192
- Velocity
  - constraints, 49, 98–99, 106, 108, 140, 203, 236, 271, 299, 330, 331, 339, 354
  - of particle, 37–38
  - of planar rigid body, 43–44
  - transformation matrix, 192, 193, 198, 200, 209
  - vector, 3, 14, 38, 93, 267
  - of wheel/tire, 175
- Viscous friction, 76–78

**W**

- Web-cutter mechanism, 366–367  
Welded joint *see* Rigid joint  
Wheel/tire, multibody dynamics, 80–82  
Windshield wiper mechanism, 369–370  
Work and energy, planar dynamics, 83–84

**Y**

- Young's modulus, 278, 361  
**Z**  
Zero vector, 17, 61, 192

## Errata (December 2018)

### PLANAR MULTIBODY DYNAMICS Formulation, Programming with MATLAB, and Applications; Second Edition

P. E. Nikravesh  
CRC Press, 2018

During the preparation of this textbook, a great deal of attention was paid to minimize the number of errors that normally occur either in the preparation phase or in the publication phase of a book. Despite that effort, some errors in the manuscript were not detected, as well as other errors that appeared during the publication phase. As such errors become known, they will be reported in this document. This document will be updated on a regular basis as new errors are found.

**A note to the reader:** If you come across any errors, please inform the author by sending an email to [pen@email.arizona.edu](mailto:pen@email.arizona.edu). Thank you!

#### Chapter 1

- Page 4, line 5 following figure: “... 1.4 rad/s” should be corrected to “... 0.4 rad/s<sup>2</sup>”
- Page 5, line 1: change Eq. (1.1) to Eq. (1.2)

#### Chapter 2

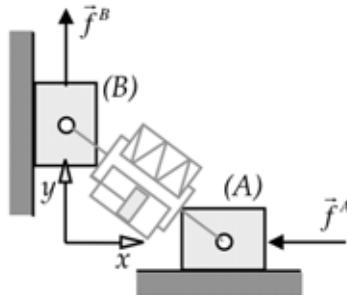
- Page 22, line 6: change “ $a_{ij}$  is the  $ij$ th element of its transpose  $\mathbf{A}'$ ” to “ $a_{ij}$  is the  $j$ ith element of its transpose  $\mathbf{A}'$ ”
- Page 23, MATLAB results following Eq. (2.28), right panel: the top “M” should be lower-case “m”
- Page 25, 1<sup>st</sup> line of text: change  $m \times P$  to  $m \times p$
- Page 25, 2<sup>nd</sup> line of text: change  $\boldsymbol{\alpha}' = \begin{Bmatrix} \alpha_1 & \alpha_2 & \alpha_m \end{Bmatrix} \neq \mathbf{0}$  to  
$$\boldsymbol{\alpha}' = \begin{Bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \end{Bmatrix} \neq \mathbf{0}$$
- Page 28; line before Eq. (2.57): change “The second time derivative ...” to “The time derivative ...”

#### Chapter 3

- Page 43, 2<sup>nd</sup> line of Sec. 3.2.2: change  $\dot{\phi}_i$  to  $\dot{\phi}$

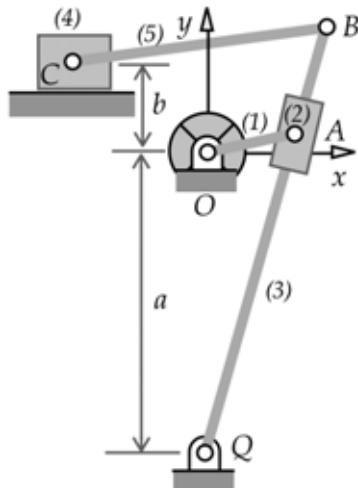
#### Chapter 4

- Page 67, last line: correct Eq. (4.19) from  $n^O = \bar{s}'^{C,O} \mathbf{f}^P$  to  $n^O = \bar{s}'^{P,O} \mathbf{f}^P$
- Page 84, last line: remove “L” from  $x^B = 0.33 \text{ mL}$
- Page 85, last line in Problem 4.4: remove “L” from  $y^B = 3 \text{ mL}$
- Page 85, figure in Problem 4.4: move the reference frame  $x-y$  to the position shown

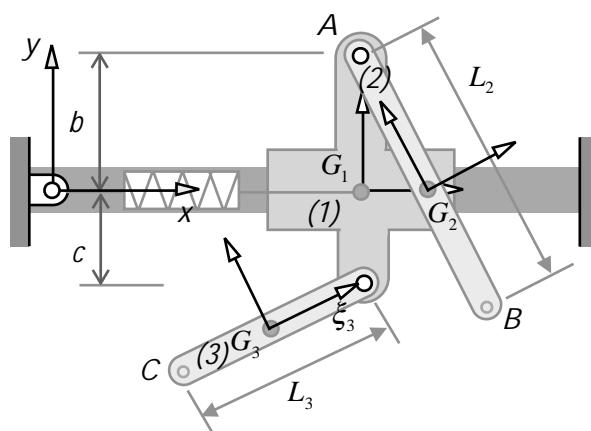


## Chapter 5

- Page 92, Figure 5.4: in several of the figures,  $\theta$  does not show properly
- Page 92, Figure 5.4: The character Sigma should be removed from the description
- Page 104, Figure 5.13:  $\theta$ 's do not show properly
- Page 105, Figure 5.14(a): the labels  $a$  and  $b$  should be interchanged as shown



- Page 105, Figure 5.14(c):  $\theta$ 's do not show properly
- Page 107, Figure 5.15(b):  $\theta_5$  does not show properly
- Page 109, figure (c): the  $x$ - $y$  axes should be relocated as shown

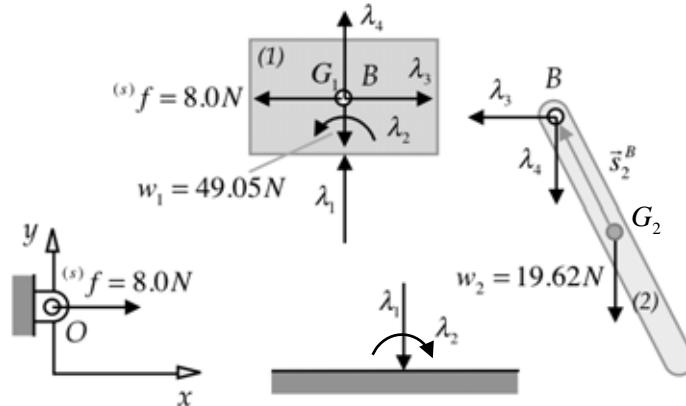


- Page 110, Problem 5.3, second line of data: remove "rad" at the end of the line
- Page 110, Problem 5.4, second line of data, change " $\theta_2 = 285^\circ$ " to  $\theta_2 = 315^\circ$

- Page 110, Problem 5.4, second line of data: remove “rad” at the end of the line
- Page 110, Problem 5.4, third line of data: change “ $\theta_3 = 50^\circ$  rad/s” to  $\dot{\theta}_3 = -0.3$  rad/s

### **Chapter 6**

- Page 113, last line of data: change “20 N sx/m” to “20 N s/m”
- Page 119, Figure 6.6: position of  $x$ - $y$  frame should be corrected,  $x$ -axis should be at the ground level



- Page 122, third line from the bottom: adjust the size of  ${}^aT$  ( $T$  is not the subscript of  $(a)$ ;  $(a)$  is the left superscript of  $T$ )
- Page 124, middle of the page:  $\boldsymbol{\lambda}_{1,2} = \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix}$  is repeated twice; remove one of them
- Page 131, figure (c) at the bottom of the page: the  $x$ - $y$  axes should be relocated (same as on Page 109)
- Page 132, line 4: Insert the following before “In the configuration ...”: “Assume initial position of body (1) to be  ${}^0x_1 = 0.07\text{ m}$ .”
- Page 132, middle of the page: change “ $k = \text{L/m}$ ” to “ $k = \text{N/m}$ ”

### **Chapter 7**

- Page 161, Problem 7.2, first line, remove “s” from “Vectors  $\vec{a}$  ...”
- Page 162, Problem 7.3, include  $\phi_2 = 135^\circ$  to the data
- Page 168, Problem 7.27, first line following data: change “Assume the deformed length of the spring to be  ${}^0L = 0.07\text{ m}$ ” to “Assume initial position of body (1) to be  ${}^0x_1 = 0.07\text{ m}$ ”
- Page 168, figure (c) at the bottom of the page: the  $x$ - $y$  axes should be relocated, same as on Page 109

### **Chapter 8**

- Page 177, line 4 from the bottom: change “... and (0) respectively” to “... and (1) respectively”
- Page 182, in the middle of the page: correct  $\text{N/m}^{-0.5}$  to  $\text{N/m}^{1.5}$

### **Chapter 9**

- Page 188, following Figure 9.4: replace the four stated equations with  $\mathbf{r}_{i,j} = \mathbf{r}_i - \mathbf{r}_j$
- Page 196, 2<sup>nd</sup> line of the *Solution*: replace  $\theta = \phi_3$  with  $\theta_3 = \phi_3$

- Page 206, following the line “For the cut-joint, we can write ...”, correct the 5<sup>th</sup> entry of the matrix to  $-\mathbf{I}$ ; i.e.,  ${}^*\mathbf{D} = \left[ \begin{array}{cc|cc|cc} \mathbf{0} & \mathbf{0} & \mathbf{I} & \frac{L_2}{2} \check{\mathbf{u}}_2 & -\mathbf{I} & \mathbf{0} \end{array} \right]$
- Page 223, Problem 9.1, first line of data, change “ $\overset{\text{def}}{L} = 0.07 \text{ m}$ ” to “ ${}^0x_1 = 0.07 \text{ m}$ ”
- Page 223, Problem 9.2, first line of data, change “ $\overset{\text{def}}{L} = 0.07 \text{ m}$ ” to “ ${}^0x_1 = 0.07 \text{ m}$ ”
- Page 223, Problem 9.3, first line of data, change “ $\overset{\text{def}}{L} = 0.07 \text{ m}$ ” to “ ${}^0x_1 = 0.07 \text{ m}$ ”
- Page 223, Problem 9.3, figure: the  $x$ - $y$  axes should be relocated as on Page 109

## Chapter 10

- Page 241, third equation: replace  $-3^2$  with  $-3.0^2$

## Chapter 11

- Page 264, Eq. (11.12): correct the equation to  $\Delta\dot{\mathbf{r}}^i = {}^{(+)}\dot{\mathbf{r}}^i - {}^{(-)}\dot{\mathbf{r}}^i$ ,  $\Delta\dot{\mathbf{r}}^j = {}^{(+)}\dot{\mathbf{r}}^j - {}^{(-)}\dot{\mathbf{r}}^j$
- Page 265, 2<sup>nd</sup> line following Eq. (11.14): in both integrals, zero should be in bold-face
- Page 267, line 4 of the text: “... a discontinuous change” should be corrected to “... a discontinuous change”.
- Page 267: correct Eq. (11.20) to  $\mathbf{u}'(\Delta\dot{\mathbf{r}}_i^P - \Delta\dot{\mathbf{r}}_j^P) = -(e+1)\mathbf{u}'{}^{(-)}\mathbf{v}_{i,j}^P$
- Page 278, in Eqs. (11.36) and (11.37): correct  $h_i = \frac{1-v_k^2}{\pi E_k}$  to  $h_k = \frac{1-v_k^2}{\pi E_k}$
- Page 279, 2<sup>nd</sup> line in the shaded box: correct  $0 < e < 1.0$  to  $0 \leq e \leq 1.0$
- Page 280, line 6 of Example 11.6: correct  $\text{N}/\text{m}^{-0.5}$  to  $\text{N}/\text{m}^{1.5}$
- Page 280, results should be corrected:  
... for  $e = 0.15$ :  
Eq. (11.42):  $f_N = \dots = 4.05 \times 10^6 \text{ N}$   
Eq. (11.43):  $f_N = \dots = 1.42 \times 10^7 \text{ N}$

For  $e = 0.95$  we have:

$$\text{Eq. (11.42): } f_N = 3.25 \times 10^6 \text{ N}$$

$$\text{Eq. (11.43): } f_N = 3.26 \times 10^6 \text{ N}$$

## Chapter 12

## Chapter 13

- Page 315, second line od *Solution*: change Ex\_9\_3 to Example\_9\_3
- Page 328, last line: change “stiff” to “stiffness”
- Page 334, Example 13.9, line 4: “... coefficient or restitution ...” should be “... coefficient of restitution ...”
- Page 339; two lines before Eq. (13.56): correct the equation to  
$$\mathbf{M}({}^{(+)}\dot{\mathbf{c}} - {}^{(-)}\dot{\mathbf{c}}) - \mathbf{D}'({}^{(+)}\boldsymbol{\sigma} - {}^{(-)}\boldsymbol{\sigma}) - \mathbf{D}'_{new}({}^{(+)}\boldsymbol{\sigma}_{new} - {}^{(-)}\boldsymbol{\sigma}_{new}) = \mathbf{0}$$

## Chapter 14

- Page 352, line 8 from the bottom, change “Ex\_14\_2\_eqm” to “Ex\_14\_2”
- Page 357, Eq. (14.10) and in the following 2 lines:  $v_{ij}$  should be corrected to  $v_{i,j}$
- Page 357, Eq. (14.11) and in the following lines 3 and 4:  ${}^s\mu$  should be corrected to  $\mu_s$

- Page 358, line 6 of the *Solution*: “Example\_14\_3 and Ex\_14\_4” should be corrected to “Example\_14\_3\_a and Ex\_14\_3\_a”
- Page 359, 1<sup>st</sup> line following Figure 14.8: correct  $|v_{ij}| \geq v_t$  to  $|v_{i,j}| \geq v_t$
- Page 360, line 6 of the program: correct “if abs(v\_ij) > v\_t” to “if abs(v\_ij) >= v\_t”
- Page 362, figure (c) at the bottom of the page: the x-y axes should be relocated (same as on Page 109)

## **Chapter 15**

### **Appendix A**

### **Appendix B**

### **Appendix C**