# Parallel Robots

## Mechanics and Control

SECOND EDITION

Hamid D. Taghirad

SUPPORT MATERIAL

CRC Press
Taylor & Francis Group

# Parallel Robots

## Mechanics and Control

SECOND EDITION

Hamid D. Taghirad

SUPPORT MATERIAL

CRC Press
Taylor & Francis Group

# Parallel Robots

In today’s rapidly evolving industrial landscape, robotics has become essential for meeting the demands of large-scale production. Parallel robots, with their closed-loop kinematic structures, offer unmatched precision, rigidity, and load-bearing capabilities, making them indispensable for tasks requiring high accuracy and efficiency. This book explores the unique advantages of parallel robots, providing a comprehensive resource for engineers, researchers, and students interested in mastering their design, analysis, and control.

Building on the success of its first edition, this second edition has been extensively restructured and updated to reflect over a decade of progress in robotics. It features expanded chapters on dynamics, new sections on simulation and calibration, and a detailed exploration of control techniques, ranging from introductory linear methods to advanced force control. With nearly 45% updated references, the text ensures readers are equipped with cutting-edge knowledge.

This book is both a comprehensive guide and a gateway to innovation, providing detailed insights into the design, simulation, calibration, and control of parallel robots. Whether you are a newcomer to robotics or an experienced

professional, this text equips you with the knowledge to harness the full potential of parallel robots, helping you stay ahead in the dynamic field of industrial automation.

# Parallel Robots
# Mechanics and Control
# Second Edition

Hamid D. Taghirad

CRC Press
Taylor & Francis Group
Boca Raton London New York

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

Designed cover image: Shutterstock

Second edition published 2026
by CRC Press
2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

*CRC Press is an imprint of Taylor & Francis Group, LLC*

© 2026 Hamid D. Taghirad

[First edition published by CRC Press 2013]

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, access www.copyright.com or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. For works that are not available on CCC please contact mpkbookspermissions@tandf.co.uk

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

MATLAB® and Simulink® are trademarks of The MathWorks, Inc. and are 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® or Simulink® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® and Simulink® software.

ISBN: 978-1-032-96209-2 (hbk)
ISBN: 978-1-032-96666-3 (pbk)
ISBN: 978-1-003-59040-8 (ebk)

DOI: 10.1201/9781003590408

Typeset in CMR10 Roman
by KnowledgeWorks Global Ltd.

Publisher's note: This book has been prepared from camera-ready copy provided by the author.

Access the Support Material: www.routledge.com/9781032962092

*To four remarkable women who have filled my life with love and joy: In memory of my dear mother, Fatemeh, who taught me resilience; to my beloved wife, Azam, my greatest strength; to my wonderful daughter, Matineh, who gifted me the scent of heaven; and for the bright future of my cherished granddaughter, Zeinab.*

# *Contents*

**Preface**

**Author Biography**

**I Preliminary Studies**

**1 Introduction**

1.1 What is a Robot

1.1.1 Robot Definitions

1.1.2 The History of Robotics

1.2 Robot Components

1.2.1 Links

1.2.2 Joints

1.2.3 Kinematic Chains

1.3 Robot Degrees of Freedom

1.4 Robot Classification

1.4.1 Serial Robots

1.4.2 Parallel Robots

1.4.3 Hybrid Structures

1.4.4 Cable–Driven Parallel Robots

1.5 Aims and Scope of the Book

Exercises
Problems

**2 Motion Representation**
2.1 Introduction
2.2 Spatial Motion Representation
2.2.1 Position of a point
2.2.2 Orientation of a Rigid Body
2.2.3 Rotation Matrix
2.2.4 Euler Angles
2.3 Rigid Body's Pose
2.4 Homogeneous Transformation
2.4.1 Homogeneous Coordinates
2.4.2 Homogeneous Transformation Matrix
2.4.3 Homogeneous Transformation Arithmetic
Problems

**3 Advanced Representations**
3.1 Introduction to Screw Theory
3.2 Orientation Representation
3.2.1 Screw Representation
3.2.2 Quaternion
3.3 Screw Representation for Motion
3.4 Linear and Angular Velocities
3.4.1 Angular Velocity of a Rigid Body
3.4.2 Angular Velocity and the Rate of Rotation Matrix
3.4.3 Angular Velocity and Rate of Euler Angles
3.4.4 Angular Velocity and Rate of Quaternion

3.4.5 Linear Velocity of a Point
3.5 Screw Coordinates
3.6 Linear and Angular Accelerations
3.6.1 Angular Acceleration of a Rigid Body
3.6.2 Linear Acceleration of a Point
Problems

**II Kinematics**

**4 Kinematic Analysis**
4.1 Introduction
4.2 Parallel Robot Types
4.3 Loop Closure Method
4.4 Case Study I: Planar Cable Robot
4.4.1 Mechanism Description
4.4.2 Geometry of the Manipulator
4.4.3 Inverse Kinematics
4.4.4 Forward Kinematics
4.4.5 Simulations
4.5 Case Study II: Shoulder Manipulator
4.5.1 Mechanism Description
4.5.2 Geometry of the Manipulator
4.5.3 Inverse Kinematics
4.5.4 Forward Kinematics
4.5.5 Simulations
4.6 Case Study III: Stewart–Gough Platform
4.6.1 Mechanism Description
4.6.2 Geometry of the Manipulator

4.6.3 Inverse Kinematics
4.6.4 Forward Kinematics
4.6.5 Simulations
Problems

**5 Jacobian Matrix and Singularity**
5.1 Introduction
5.2 Jacobian Matrices of a Parallel Manipulator
5.3 Jacobian Matrix Derivation
5.4 Singularity Analysis
5.4.1 Inverse Kinematic Singularity (Type I)
5.4.2 Forward Kinematic Singularity (Type II)
5.4.3 Combined Kinematic Singularity (Type III)
5.4.4 Singularity Avoidance
5.5 Case Study I: Planar Cable Robot
5.5.1 Velocity Loop Closure
5.5.2 Singularity Analysis
5.5.3 Sensitivity Analysis
5.6 Case Study II: Shoulder Manipulator
5.6.1 Velocity Loop Closure
5.6.2 Singularity Analysis
5.7 Case Study III: Stewart–Gough Platform
5.7.1 Velocity Loop Closure
5.7.2 Singularity Analysis
Problems

**6 Jacobian Matrix: Mappings & Characteristics**
6.1 Introduction
6.2 Static Forces in Parallel Manipulators

6.2.1 Free-Body Diagram Approach
6.2.2 Virtual Work Approach
6.2.3 Case Study I: Planar Cable Robot
6.2.4 Case Study II: Shoulder Manipulator
6.2.5 Case Study III: Stewart–Gough Platform
6.3 Jacobian Matrix: Intrinsic Characteristics
6.3.1 Dexterity Ellipsoid
6.3.2 From Singularity to Isotropy
6.3.3 Dexterity Measure
6.3.4 Manipulability Measure
6.3.5 Global Measures
6.4 Stiffness Analysis
6.4.1 Stiffness Matrix
6.4.2 Stiffness Ellipsoid
6.4.3 Case Study I: Planar Cable Robot
6.4.4 Case Study II: Shoulder Manipulator
6.4.5 Case Study III: Stewart–Gough Platform
6.5 Inverse Problems
6.5.1 Inverse Velocity Problem
6.5.2 Inverse Force Problem
6.6 Redundancy Resolution
6.6.1 Problem Formulation
6.6.2 Lagrange and Karush–Kuhn–Tucker Multipliers
6.6.3 Iterative Solutions
6.6.4 Numerical Methods
6.6.5 An Iterative-Analytical Method
Problems

# III Dynamics


## 7 Dynamics Analysis: Newton-Euler Approach

7.1 Introduction

7.2 Dynamics of Rigid Bodies: A Review

7.2.1 Center of Mass

7.2.2 Moments of Inertia

7.2.3 Linear and Angular Momentum

7.2.4 Kinetic Energy

7.2.5 Newton–Euler Laws

7.3 Newton–Euler Formulation

7.4 Case Study I: Planar Cable Robot

7.4.1 Acceleration Analysis

7.4.2 Dynamic Formulation of the Limbs

7.4.3 Dynamic Formulation of the Moving Platform

7.5 Case Study II: Stewart–Gough Platform

7.5.1 Acceleration Analysis

7.5.2 Dynamic Formulation of the Limbs

7.5.3 Dynamic Formulation of the Moving Platform

7.5.4 Closed-Form Dynamics

Problems

## 8 Dynamics Analysis: Euler-Lagrange Approach

8.1 Introduction

8.2 Euler-Lagrange Formulation

8.2.1 Generalized Coordinates and Forces

8.2.2 Euler-Lagrange Equations of the Second Kind

8.2.3 Euler-Lagrange Equations of the First Kind

8.3 Properties of Dynamic Formulations
8.3.1 Properties of Mass Matrix
8.3.2 Christoffel Matrix
8.3.3 Linearity in Parameters
8.4 Case Study I: Planar Cable Robot
8.4.1 Dynamic Formulation of the Limbs
8.4.2 Dynamic Formulation of the Moving Platform
8.4.3 Dynamic Formulation of the Whole Manipulator
8.5 Case Study II: Stewart–Gough Platform
8.5.1 Dynamic Formulation of the Limbs
8.5.2 Dynamic Formulation of the Moving Platform
8.5.3 Dynamic Formulation of the Whole Manipulator
Problems

**9 Dynamics Analysis: Virtual Work Approach**
9.1 Introduction
9.2 D’Alembert's Principle
9.3 Principle of Virtual Work
9.4 Closed-Form Formulation
9.5 Case Study I: Planar Cable Robot
9.5.1 Implicit Dynamics
9.5.2 Formulation Verification
9.6 Case Study II: Stewart–Gough Platform
9.7 Case Study III: The 3UPU Manipulator
9.7.1 Manipulator Structure and Kinematics
9.7.2 Jacobian Analysis
9.7.3 Acceleration Analysis

9.7.4 Implicit Dynamics
9.7.5 Closed-Form Formulation
Problems

**10 Dynamic Simulation and Calibration**
10.1 Introduction
10.2 Motion Planning
10.2.1 Cubic Polynomials
10.2.2 Cubic Polynomials with Via Points
10.2.3 Quintic Polynomials
10.2.4 Linear Trajectory with Parabolic Blends
10.2.5 Trajectory Planning in Joint Space
10.3 Motion Simulation
10.3.1 Forward Dynamics
10.3.2 Inverse Dynamics
10.4 Case Study I: Planar Cable Robot
10.4.1 Forward Dynamics Simulations
10.4.2 Closed-Loop Dynamics Simulations
10.4.3 Inverse Dynamics Simulation
10.5 Case Study II: Stewart–Gough Platform
10.5.1 Forward Dynamics Simulation
10.5.2 Closed-Loop Dynamics Simulation
10.5.3 Inverse Dynamics Simulation
10.6 Dynamic Calibration
10.6.1 Linear Regression Models
10.6.2 Least Squares Method
10.6.3 Evaluation of Calibrated Model
Problems

# IV Control


## 11 Linear Motion Control

11.1 Introduction
11.2 Controller Topologies
11.2.1 Motion Control in Task Space
11.2.2 Motion Control in Joint Space
11.2.3 Topology Selection
11.3 Linear Control: Concepts and Methods
11.3.1 Magic of Feedback
11.3.2 PD Control
11.3.3 PID Control
11.3.4 Performance Limitations
11.3.5 Feed-Forward Control
11.4 PD Control in Parallel Robots
11.5 Multivariable PID Control
11.6 Case Study I: Planar Cable Robot
11.6.1 Decentralized PD Control
11.6.2 Multivariable PID Control
11.7 Case Study II: Stewart Gough Platform
11.7.1 Decentralized PD Control
11.7.2 Multivariable PID Control
Problems

## 12 Multivariable and Nonlinear Control

12.1 Introduction
12.2 Multivariable Control with Feed-Forward
12.3 Inverse Dynamics Control
12.4 Partial Feedback Linearization

12.5 Robust and Adaptive Control
12.5.1 Model Uncertainty
12.5.2 Robust Inverse Dynamics Control
12.5.3 Chattering Mitigation
12.5.4 Adaptive Inverse Dynamics Control
12.6 Summary of Motion Control Techniques
12.7 Case Study I: Planar Cable Robot
12.7.1 Basic PID Control
12.7.2 PID Control with Feed-Forward
12.7.3 Inverse Dynamics Control
12.7.4 Partial Feedback Linearization
12.7.5 Robust Inverse Dynamics Control
12.7.6 Adaptive Inverse Dynamics Control
12.8 Case Study II: Stewart–Gough Platform
12.8.1 Basic PID Control
12.8.2 Feed-Forward Control
12.8.3 Inverse Dynamics Control
12.8.4 Partial Feedback Linearization
12.8.5 Robust Inverse Dynamics Control
Problems

**13 Force Control**
13.1 Introduction
13.2 Controller Topologies
13.2.1 Cascade Control
13.2.2 Force Feedback in the Outer Loop
13.2.3 Force Feedback in the Inner Loop
13.3 Stiffness Control
13.3.1 Single-Degree-of-Freedom Stiffness Control

13.3.2 General Stiffness Control
13.3.3 Case Study I: Planar Cable Robot
13.3.4 Case Study II: Stewart–Gough Platform
13.4 Direct Force Control
13.4.1 Case Study I: Planar Cable Robot
13.4.2 Case Study II: Stewart–Gough Platform
Problems

**14 Impedance Control**
14.1 Introduction
14.2 Definition of Impedance
14.3 Impedance Control Concept
14.3.1 Automated Fish Fillet Production
14.3.2 Flight Simulator Yoke
14.4 Impedance Control Structure
14.5 Special Case in Impedance Control
14.6 Case Study I: Planar Cable Robot
14.7 Case Study II: Stewart–Gough Platform
Problems

**V Appendices**

**A Appendix A: Linear Algebra**
A.1 Vectors and Matrices
A.2 Vectors and Matrices Arithmetic
A.3 Eigenvalues and Singular Values
A.4 Pseudo-Inverse
A.4.1 Pseudo-Inverse Properties
A.4.2 Linear Inverse Problem

A.5 Kronecker Product

**B Appendix B: Nonlinear Control Review**

B.1 Dynamical Systems

B.2 Stability Definitions

B.3 Lyapunov Stability

B.4 Krasovskii–Lasalle Theorem

**Bibliography**

**Index**

# *Preface*

Robots have profoundly reshaped human life in the 21st century, particularly within industrial automation. They are essential for maintaining both production quantity and quality, while also bringing much-needed flexibility to manufacturing processes. Beyond boosting production capabilities, robots minimize human labor in dangerous environments, enhance efficiency, and ensure consistency in tasks requiring high precision. The advancement of robotics technology continually pushes boundaries by integrating sophisticated sensors, artificial intelligence, and machine learning to enhance robot functionalities further.

Typically, industrial robots possess an articulated structure made up of interconnected links that provide a large workspace. The movement of each link is controlled by individual actuators, enabling precise motion control. Despite these benefits, including a large workspace and flexibility, the serial structure can significantly impact the accuracy of the robot's end effector. As innovation progresses, efforts are increasingly focused on overcoming

the limitations of serial structures and exploring alternative configurations that provide both precision and adaptability.

Parallel structures are ideal for applications demanding high precision, low compliance, or high load capacity relative to the robot's weight. Unlike serial robots, which consist of a single continuous chain of links and joints, parallel robots feature multiple kinematic chains connected to a shared moving platform, providing a unique structural advantage. Many industrial applications have adopted parallel robots, but there are few resources that extensively cover their kinematics, dynamics, and control. This book aims to fill that gap, offering comprehensive tools and systematic methods for engineers and researchers focused on the design and application of parallel robots in industry.

This text explores both the theoretical foundations and key technological advancements essential for understanding and developing parallel robots. While the first generation of industrial robots—instrumental in the rise of mass production—predominantly utilized serial architectures, a new generation has embraced parallel structures. This book is dedicated to the analysis and design of these second-generation, parallel-structured robots. For readers seeking foundational knowledge on robotics, particularly on serial robot design, a separate book by the same author offers a detailed introduction to that field [1]. Here, the focus is specifically on parallel robots, with a detailed exploration of their kinematics, dynamics, and control, equipping readers with the skills

needed to analyze complex, multi-degree-of-freedom structures.

Since the release of the first edition in 2012, there has been notable progress in the study and application of parallel robots, which this book reflects. The second edition has been completely restructured, now organized into four main parts and fourteen chapters. The lengthy chapters from the previous edition have been divided to distinguish introductory content from advanced topics. The dynamics chapter has been expanded into three distinct chapters, and new chapters on simulation and calibration have been added. The Control chapter is now split into four chapters, emphasizing linear control design techniques in the introductory part and categorizing force control methods into two main approaches. Additionally, the reference section has been updated with about 45% new sources that have appeared since the first edition.

An overview of the redesigned chapters is provided as follows. Accurately defining the motion characteristics of parallel robots requires a structured approach to representing the 3D motion of the robot's moving platform in space. This entails a systematic method for describing the position, orientation, and location of rigid bodies. Chapter 2 introduce the essential concepts and basic representation methods, while Chapter 3 expands on more advanced techniques. Special emphasis is placed on screw representation to simplify the description of general motion. These foundational concepts are critical for

representing linear and angular velocities and accelerations, as well as the forces and torques exerted on a robot's moving platform using a six-dimensional array.

Kinematic analysis involves studying the geometric aspects of motion in a robot, focusing solely on the relationship between the manipulator's geometric structure and parameters with the resulting motion of the moving platform. Chapter 4 offers a comprehensive treatment of this analysis, accompanied by detailed case studies of three parallel robots, one of which is a planar cable-driven parallel robot. By this means, this book formally explores the analysis of cable-driven parallel robots, highlighting them as a promising new generation of parallel structures known for their expansive workspace capabilities.

The introduction of the Jacobian matrix in robotics and the examination of differential kinematics are other essential topics thoroughly explored in this book. Numerous key aspects of parallel robot design are closely tied to velocity analysis and the Jacobian matrix. Differential kinematic analysis is also critical in designing robotic manipulators that avoid singularities. Jacobian analysis reveals the connection between joint variable velocities and the linear and angular velocities of the moving platform.

This book presents a systematic method for conducting Jacobian analysis for parallel manipulators, outlined in Chapter 5. Jacobian analysis further aids in the transformation needed to calculate actuator forces based

on task-space forces and torques acting on the moving platform. Advanced topics, including stiffness analysis and redundancy resolution schemes, are elaborated upon in Chapter 6. In these chapters, the proposed methods and analyses are demonstrated through the same case studies introduced in earlier chapters.

The dynamic analysis of parallel manipulators is inherently complex due to their closed-loop structure and kinematic constraints. Nonetheless, dynamic modeling is essential for their simulation and control, as parallel manipulators are favored in applications that demand precise positioning and strong dynamic performance under heavy loads. Although substantial research has focused on the kinematics of parallel manipulators, relatively few studies emphasize their dynamics and control, and detailed coverage in these areas is scarce. This book addresses these critical issues comprehensively.

Chapters 7 to 9 explore the dynamic analysis of parallel robots using three main approaches: the Newton-Euler method, the Euler-Lagrange formulation, and the principle of virtual work. These chapters also introduce an innovative approach for deriving the dynamic equations of parallel robots in closed form, improving the manageability of dynamic matrices and facilitating the verification of their dynamics. Additionally, a new chapter in this edition (Chapter 10) elaborates on dynamic simulation and calibration methods, with thorough demonstrations of

simulation techniques applied to the same case studies covered in previous chapters.

The last four chapters of the book elaborate on the control of parallel robots, covering both motion and force control schemes. The control design techniques start with the basic and conceptual methods of linear control in Chapter 11. More advanced nonlinear methods are then introduced in Chapter 12, alongside detailed explanations of robust and adaptive control methods. These chapters apply these control techniques to the same case studies explored earlier in this book. The final two chapters provide in-depth coverage of force control for parallel robots. Chapter 13 elaborates on stiffness and direct force control, while Chapter 14 provides comprehensive studies on impedance control schemes.

This book stands out with two distinctive features: the comprehensive examination of multiple case studies and the practical use of engineering software such as MATLAB® and the Python programming language to work out these case studies. The first feature involves the thorough implementation of theoretical concepts discussed in the book on various industrial robots across its chapters. The case studies focus on the analysis and design of parallel robots, including a planar cable-driven robot, a 3DOF shoulder manipulator, and the Stewart-Gaugh platform. At the conclusion of each chapter, readers are encouraged to apply the theoretical concepts to twelve different parallel robots. This practical approach enables

readers to gain proficiency in the design and analysis of parallel robots by actively working on these problems.

The second distinguishing feature revolves around the utilization of MATLAB and Python programming for the analysis of the presented case studies. The book goes beyond theory by offering complete solutions and programming codes for the featured case studies in these software platforms. This approach empowers readers to acquire practical expertise in the analysis and design methods for parallel robots. The inclusion of ready-to-use codes facilitates the seamless implementation of the discussed methods on the robots introduced in each chapter's problem section, enhancing accessibility for readers and promoting hands-on learning. The codes and their updates could be accessed in the Author's Github: github.com/aras-labs/Robotics

This book has been typeset using LaTeX, with numerous figures and diagrams created using Inkscape software. Despite careful efforts to produce an error-free version, some imperfections may remain. The author invites readers to report any identified issues via email to hamid@cim.mcgill.ca. Your feedback is appreciated and will help to enhance the overall quality of the book. Moreover, all additional materials accompanying this book, such as MATLAB and Python codes, teaching slides, and any post-print corrections, will be made easily accessible to readers on the publisher's website.

## For Instructors

Given its well-structured content, this book is anticipated to be a valuable resource for two primary groups of readers. Firstly, it serves as a suitable reference for teaching advanced robotics courses in the first year of graduate studies. Additionally, the content is suitable for an advanced course in dynamics and control of robots, specifically designed for senior graduate students. The interdisciplinary approach of this book enhances its accessibility for students in electrical engineering, mechanical engineering, and computer science. This approach allows students from diverse backgrounds to easily understand and acquire substantial knowledge in robotics engineering. The second category of readers includes industry professionals who use robotic technology in their careers and are interested in gaining the necessary skills for the operation, design, and control of industrial robots for their specific applications.

The author has continuously taught the content of this book in parallel robot and robot dynamics and control courses at various universities for several years. The chapters' content and size have been tailored to meet the specific needs of these courses. Consequently, by selecting specific chapters from this book, it can be effectively used in the following courses.

**1- Parallel Robots: Kinematics and Dynamics** 500 Level; Junior graduate students, for a one-semester

course (16 weeks)

Chapter 1: Introduction

Chapter 2: Motion Representation

Chapter 3: Advanced Representations

Chapter 4: Kinematics Analysis

Chapter 5: Jacobian Matrix and Singularity

Chapter 6: Jacobian Matrix: Mappings & Characteristics

Chapter 8: Dynamics Analysis: Euler Lagrange Approach

Chapter 10: Dynamics Simulation and Calibration

2- **Parallel Robots: Dynamics and Control:** 600 Level; Senior graduate students for a one semester course (16 weeks)

Chapters 1-6: Review of Selected Sections

Chapter 7: Dynamics Analysis: Newton-Euler Approach

Chapter 8: Dynamics Analysis: Euler Lagrange Approach

Chapter 9: Dynamics Analysis: Virtual Work Approach

Chapter 10: Dynamics Simulation and Calibration

Chapter 11: Linear Motion Control

Chapter 12: Multivariable and Nonlinear Control

Chapter 13: Force Control

Chapter 14: Impedance Control

## Acknowledgments

I am deeply grateful to the many individuals who supported me technically and spiritually throughout the writing of this book and revising its second edition. Given its interdisciplinary nature, numerous people have contributed to this work, and while it is not feasible to list all their names here, their contributions are sincerely appreciated. I dedicate this book to the late Professor G. Zames and Professor P. R. Bélanger, not only for what they taught me in control theory but also for the profound impact they had on my personal philosophy of *making a difference*. I am also indebted to Professor J. Angeles and Professor C. Gosselin, whose encouragement was instrumental in guiding this endeavor. Many of the results presented in this book are derived from the contributions of J. Angeles, C. Gosselin, J.-P. Merlet, L-W. Tsai, and other distinguished researchers in this field, whose work I have had the privilege to organize and expand upon. Any errors in the presentation of their work are my own responsibility.

I am grateful for the enjoyable collaborations I had with Professors M. Nahon, I. Bonev, and Mahdi Tavakoli, and I wish to express my sincere gratitude to them for providing me with visiting opportunities during critical periods, allowing me to temporarily escape my regular tight schedule and focus on this book. The content of this book has been reviewed by numerous students who took postgraduate courses at various universities. Their valuable

comments and corrections have significantly enhanced the quality of the material. Special thanks are extended to all those who contributed to the compilation and revision of this book, as well as to the students whose feedback has been valuable throughout the years of teaching this content. I extend special thanks to Dr. H. Sadjadian for dedicating considerable time to reviewing the manuscript of first edition. Additionally, the author expresses appreciation to Mr. Mohammad Javad Ahmadi and Mr. Mohammad Mehdi Nazari Ardakani for their significant efforts in completing the codes of the second edition and uploading them to the book's GitHub repository.

I cannot conclude without acknowledging the unwavering support and encouragement I received from my wife Azam and my daughter Matineh. Their boundless support and patience have been invaluable to me throughout this endeavor.

HAMID D. TAGHIRAD

# *Author Biography*

Hamid D. Taghirad received his M.Sc. in mechanical engineering (mechatronics) in 1993 and his Ph.D. in electrical engineering (control–robotics) in 1997, both from McGill University, Montreal, Canada. He is currently a Professor and the Director of Applied Robotics and AI Solutions (ARAS), at K. N. Toosi University of Technology. Additionally, he has served as a Visiting Professor at McGill University, ETS, Concordia University, and most recently at the University of Alberta in Canada. As a senior member of IEEE, Hamid D. Taghirad contributes to the field of Robotics as an associate editor of IEEE Transactions on Medical Robotics and Bionics (IEEE – TMRB) and Frontiers in Robotics and AI – Biomedical Robotics. He is also part of the editorial board of the International Journal of Robotics: Theory and Applications. His research interests primarily revolve around robust and nonlinear control applied to robotics, with a focus on medical robotics, as well as the application of VR and AI technologies in medical applications. He boasts a substantial publication record, including seven books and over 300 papers in peer-

reviewed international journals and conference proceedings.

# Part I

# Preliminary Studies

# 1

# *Introduction*

DOI: 10.1201/9781003590408-1

In today's rapidly advancing industrial landscape, the integration of robotics technology has become indispensable to meet the increasing demand for large-scale production. Robotics, a field both intriguing and complex, involves the creation of intelligent machines designed to perform a diverse range of tasks by mimicking natural designs. Initially, the first generation of industrial robots featured serial structures, excelling in tasks that required a sequence of articulated movements. However, the transition to parallel robots has brought enhanced precision, flexibility, and load-bearing capabilities to the industry.

The ever-increasing demand for new product styles, improved quality, and reduced manufacturing costs has led to a greater adoption of robotic equipment across various industries. Initially, automobile manufacturers were the primary users of robots in their production lines. However, in recent years, industries such as home appliances, food,

and pharmaceuticals have also integrated robotic systems into their operations. A key factor driving this growth is the significant decline in the cost of industrial robots. While robot prices have decreased, human labor costs continue to rise. Moreover, robots have become more efficient, faster, smarter, more accurate, and more flexible, further enhancing their appeal and utility in diverse production environments.

Industrial robots typically feature an articulated structure, with a series of interconnected links that offer a large workspace. Each link's motion is controlled by individual actuators, allowing for precise manipulation. While these serial structures achieve design objectives such as extensive workspace and flexibility, they often compromise the accuracy of the robot's end-effector.

For applications requiring high precision, stiffness, or a high load capacity relative to the robot's weight, parallel structures present an ideal alternative. Parallel robots, with their closed-loop kinematic structure, excel in these demanding applications. Despite their advantages, the analysis, design, and control of parallel robots have not been extensively covered in the literature. While many books focus on the theoretical and technological advancements of serial robots [1, 2, 3, 4, 5], few elaborate on parallel robots [6, 7, 8]. This book aims to fill that gap by providing analysis and design tools for the growing number of engineers and researchers interested in the industrial application of parallel robots. It addresses the

unique challenges and opportunities presented by parallel structures, equipping readers with the knowledge needed to innovate and excel in this specialized field.

## 1.1 What is a Robot

The introduction of robotic technology in industry began with the advent of robotic arms. During the industrial revolution and the expansion of mass production for various components and machines, there was an increasing demand for skilled labor to perform repetitive manufacturing tasks. In 1954, the first programmable robot was invented and put into industrial use [9]. This milestone was followed by the establishment of Unimation, the first company dedicated to manufacturing robots. Robotic arms quickly became integral to mass production across various industries, particularly in the automotive sector. For many years, robots were primarily associated with robotic manipulators, drawing inspiration from the human arm [10].

However, it is important to note that robotics technology has evolved far beyond just robotic manipulators. Innovations in the field have led to the development of various advanced technologies. These include the design and construction of hands or end-effectors for grasping and manipulating objects, mobile and legged robots capable of navigating both smooth and rugged terrains, machine

vision systems for visual perception of the environment, sound processing systems that mimic auditory abilities, flying robots inspired by the flight of birds, and humanoid robots. Additionally, the integration of artificial intelligence has enhanced robots' capabilities in perceiving their surroundings and making decisions. This broad scope of advancements signifies the extensive evolution of robotics technology, moving well beyond its initial association with robotic manipulators [11].

While advancements in these domains owe their progress to multidisciplinary research, the contemporary landscape no longer categorizes these sciences and technologies as exclusively part of robotics. However, the remarkable progress in these technologies, particularly in recent years, has brought humanity closer to the aspiration of creating a being resembling itself. In this context, robotics represents an interdisciplinary field within engineering and computer science, focused on the creation, application, and utilization of robots with the ultimate goal of developing machines that can serve and assist humans.

### 1.1.1 Robot Definitions

Despite the integration of robots into daily life and the recognition that each individual has their own interpretation of a robot, there is no universally accepted comprehensive definition of a robot. Generally, a robot is considered a mechanical or virtual entity, often equipped with electromechanical structures, that is programmed by a

computer to appear as if it has independent intent or autonomy. While this description is comprehensive, it does not imply that a robot must possess all of the following features, though it should exhibit some of them:

- It is not *natural* and has been artificially created.
- It has the ability to perceive the environment.
- It can interact with its surrounding environment.
- It is programmable and to some extent intelligent.
- It moves in three-dimensional space.
- It appears to have intent or agency.

The last feature is perhaps the most distinguishing factor between a robot and an automated machine. Generally, the more a robot's motion resembles human-like movement, the more it reinforces this characteristic in the viewer's mind, making it seem closer to a real robot.

As mentioned, a unified definition of a robot has not yet been established. The International Organization for Standardization (ISO) defines a robot as follows:

> “A typical industrial robot is automatically controlled, programmable for various purposes, capable of movement in three or more degrees of freedom, and can be either stationary or mobile on the ground. It is primarily utilized in industrial automation applications.”

This way, it becomes clear that this definition underscores the key features of a robot for the purpose of

offering a specific definition. Joseph Engelberger, a trailblazer in the field of robotics, once expressed that he might not be able to precisely define a robot, but when he sees one, he can identify it [12].

The Cambridge and Oxford dictionaries define a robot as follows:

> "A device that is automatically controlled by a computer to perform tasks."
>
> "A device that is similar to a human and is programmed to perform specific movements automatically."

A more comprehensive definition of a robot is provided by the American Robotics Society, in which a robot is defined as follows [13]:

> "A robot is a multi-functional arm with programmable capabilities that can be used for the transportation of objects, parts, tools, and specific equipment through various programmed movements and employed to perform a wide range of tasks."

As previously discussed, distinguishing between a robot and an automated machine in real-world scenarios can be challenging. The key difference between the two lies in the increased complexity and programmability of robots. A robot, with its adaptable programming, can autonomously perform predefined tasks. In advanced intelligent robots, artificial intelligence plays a role in perception and

decision-making. In contrast, automated machines are designed for repetitive tasks and operate based on on/off sensors, actuators, or human commands.

### 1.1.2 The History of Robotics

The term “robot” was first introduced to the world by the Czech writer Karel Čapek in 1921 in a play titled *Rossum's Universal Robots* [14]. This play depicts a factory where artificial organic materials are used to create emotionless workers who tirelessly labor for their masters. Čapek coined the term “Robota” to describe these diligent laborers. While the robots we know today differ significantly in nature and application from those described in this play, the word “robot” has become integrated into the vocabulary of various countries.

It is important to note that historically, the concept of robotics emerged later within the engineering community. Following World War II, with the rise of mass production, automated machines equipped with digital control systems began to be utilized. During this period, automated machines started replacing human operators in tasks such as material handling, especially in nuclear power plants. The development of digital control technology, alongside the adoption of automated material handling machines, paved the way for the first generation of industrial robots, initially controlled pneumatically. Over time, advancements in technology led to the transition from digital control to computer control systems, and from pneumatic actuators to

electromechanical ones. A summary of significant events and developments in the evolution of robotics technology worldwide is provided in [1].

## 1.2 Robot Components

Typically, a robot's structure consists of the following components:

**Base:** Generally, the robot's chassis, which functions as the primary structure for its mechanical components, is referred to as the robot's base. When the robot's base is fixed on the ground, its movement is confined to a limited area near its installation point. However, if the robot is equipped with wheels or legs for mobility, it is classified as having a mobile base. Depending on its method of movement, it can be categorized as a “mobile” or “legged” robot.

**Limbs:** The limbs of a robot consist of interconnected links and joints that form a structured framework. Much like the joints and bones in human limbs, these robotic components enable the robot to interact with its environment or to move, akin to taking steps. Despite the distinct purposes of human and robotic limbs, their joint structures exhibit striking similarities. Consequently, many contemporary robots draw design inspiration from the structure of the human arm and leg.

**Hand or End-Effector** Industrial robots commonly feature a straightforward component designed for gripping objects, known as an “end-effector.” Positioned at the wrist of the robotic arm, the end-effector marks the termination point of the arm's components. In many cases, industrial end-effectors are equipped with a single degree of freedom, enabling them to grasp and release objects using pneumatic grippers or on/off magnetic actuators. However, tasks involving objects of varying geometries often require more degrees of freedom and designs inspired by the human hand. Unlike robotic hands, which typically have fewer degrees of freedom, the human hand is intricately designed with 27 degrees of freedom, allowing for precise and complex tasks such as painting or surgery.

**Actuators:** Actuators within a robot serve as the counterparts to muscles in the human body. They translate motion commands from the robot's controller into physical movement. Contemporary robots predominantly utilize servomotors as actuators, a departure from earlier times when pneumatic and hydraulic actuators were prevalent. Ongoing technological advancements are focused on developing actuators that mimic human muscles, often integrating materials such as Shape Memory Alloys (SMAs) to achieve this goal.

**Sensors:** Robots utilize various sensors to gather information about their environment, as well as the movements of their joints and arms, and their interactions with surroundings. These sensors play a role analogous to the human senses, enabling the robot's controller to make informed decisions based on acquired data. Human senses, diverse in nature, encompass a broad spectrum of functions from detecting joint positions to perceiving tactile feedback and environmental cues like vision and sound. Initially, industrial robots relied primarily on motion sensors to monitor joint movements. However, advancements in robotics have expanded sensor technologies to include machine vision, laser rangefinders, speech recognition and processing, as well as tactile and haptic sensors, tailored to specific applications and operational needs.

**Controller:** The cerebellum in the human brain, despite its smaller size compared to the entire brain, plays a crucial role in controlling body movements. Similarly, the robot's controller acts akin to the cerebellum, receiving sensor inputs from the robot's main processor (analogous to the human brain) to guide and manage the robot's motions. Central to the performance of robotic movements is the utilization of feedback loops and data from motion or tactile sensors, which control the robot's position and its interactions with the environment. The feedback loop continuously compares the robot's current position with the desired position in real-time,

generating precise control commands that are then transmitted to the robot's actuators. This feedback mechanism ensures that the robot can accurately follow its intended trajectory despite external disturbances, measurement inaccuracies, or imperfect knowledge of the robot's dynamics.

**Processor:** In a robot, the processor functions similarly to the brain. It takes raw sensor data and processes it into meaningful information regarding the positions of the robot's joints and links. This processed information is then used to compare the current state of the robot with the desired state. The processor calculates the necessary movement speeds for each joint to achieve the desired state and sends this information to the robot controller. Processors used in robots can vary depending on the computational requirements. For industrial applications, processors range from standard computing systems to specialized hardware like commercial microprocessors, industrial computers, and programmable logic controllers (PLCs). The choice of processor depends on the specific needs and considerations of the application at hand.

**Software:** In a robot, three types of software are employed. The primary software runs on the robot's processor and includes the operating system, which operates in real-time to control the robot online. The second type of software is responsible for executing control commands

to the actuators. Additionally, the robot's software incorporates task-oriented routines such as returning home, assembly, machine loading, and parts retrieval and delivery, designed for user applications. Operation planning in robots typically occurs in two modes: manual and automatic. In manual mode, control of the robot's automatic system is handed over to the manual robot controller, often referred to as the “teaching pendant.” Using the teaching pendant, each joint of the robot can be manually moved, and its position information can be stored at desired states. This stored information is then used to facilitate smooth transitions between states when employing the predefined routines. In many modern robots, traditional pre-programmed planning has been replaced by flexible programming that integrates artificial intelligence. This approach avoids predetermined motion points stored through manual teaching. Instead, during operation, motion points are dynamically calculated using inputs from machine vision systems, data related to the robot's interaction with its environment, and various artificial intelligence techniques. This allows for more adaptive and responsive robotic operations.

### 1.2.1 Links

Robotic arms are typically structured with multiple links and joints that, through their configuration and geometry, form the fundamental framework of the robot. The range of

motion of a robot's end-effector is determined by the number, type, and arrangement of these links and joints. Various innovative designs have been developed over time to achieve specific motions required for different applications. In industrial settings, rigid links are predominantly used to ensure precise and efficient maneuverability during operations.

However, in specialized applications such as space robots or cable-driven parallel robots, where weight constraints or operational requirements are critical, compliant links are sometimes employed. These types of robots fall under the category of compliant robots, which involve complex mechanical analysis and controller design beyond the scope of this book. For the purposes of this book, rigid links are assumed, and therefore, the analysis of their deformation and flexibility is simplified compared to compliant systems. This simplification facilitates the mechanical analysis and controller design discussed in the subsequent chapters.

From a kinematic perspective, a link in a robot refers to a set of solid components connected between two robot joints in a manner that ideally eliminates relative motion between those joints. For example, a basic robot link can be visualized as a metal rod with rotational joints at each end, allowing for specific movements dictated by its design.

### 1.2.2 Joints

Robot links are interconnected in pairs through joints, which facilitate relative motion between two links in

specific directions while constraining movement in others. The types and characteristics of joints used in robots vary based on the required motion between two links. These joints are described as follows:

## Revolute Joint (R)

The most commonly used joint in robotic arms is the “revolute” joint. As illustrated in the schematic of this joint in Figure 1.1(a), it enables relative rotational motion between two links along the axis of the joint. This joint restricts rotation in directions perpendicular to the joint axis and translational motion in three directions. Therefore, the revolute joint provides one degree of freedom for rotational movement between two links and is classified as one of the primary types of joints. Many industrial robots utilize exclusively this type of joint due to its simplicity in implementing rotation measurement using sensors like encoders and applying rotational torque commands using electric motors.

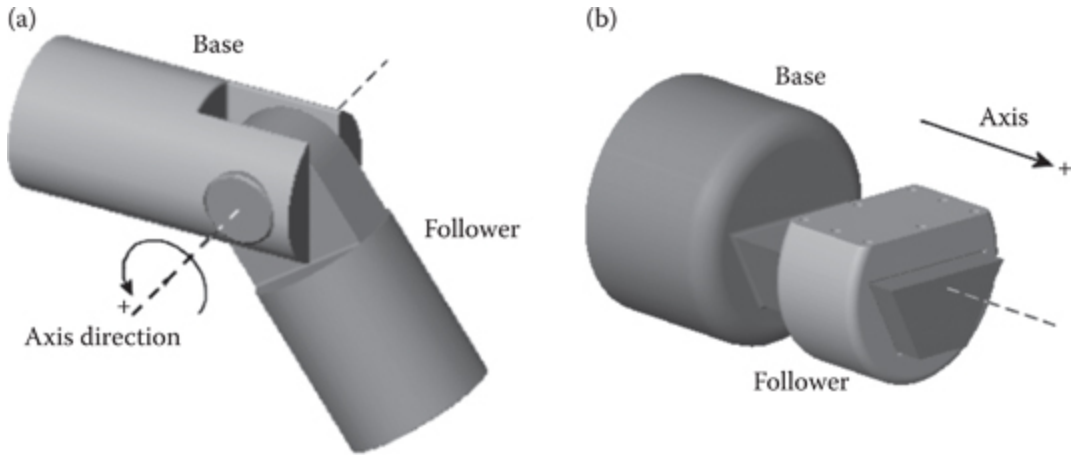


**FIGURE 1.1**
Schematics of a revolute joint (a) and a prismatic joint (b). (From Mathworks Inc. 2024. Mathworks. With permission.) ⏎

## Prismatic Joint (P)

Another essential joint commonly used in robots is the “prismatic” joint. The schematic of this joint is depicted in Figure 1.1(b). This type of joint allows for translational motion along its axis between two links. It restricts translational motion in directions perpendicular to the prismatic axis and rotational movements in three directions. Therefore, the prismatic joint provides one degree of freedom for translational movement between two links and is categorized among the primary joints, as shown in Figure 1.1. In industrial robots that utilize hydraulic actuators, this type of joint is commonly employed due to the natural cylinder-piston structure of such actuators. Additionally, measuring the amount of translational movement can be easily implemented using sensors like the linear variable differential transformer (LVDT) in this joint.

## Cylindrical Joint (C)

A cylindrical joint allows for both rotational motion around an axis and simultaneous translational movement along the same axis between two links, as depicted in Figure 1.2(a). This type of joint provides two degrees of freedom for movement between two links and is therefore not classified

as one of the primary joints in robots. While constructing this joint is not challenging, creating actuators that can achieve two simultaneous degrees of freedom is difficult. Consequently, the use of this joint is limited to specific cases in robots where providing such dual-motion capability is applicable.

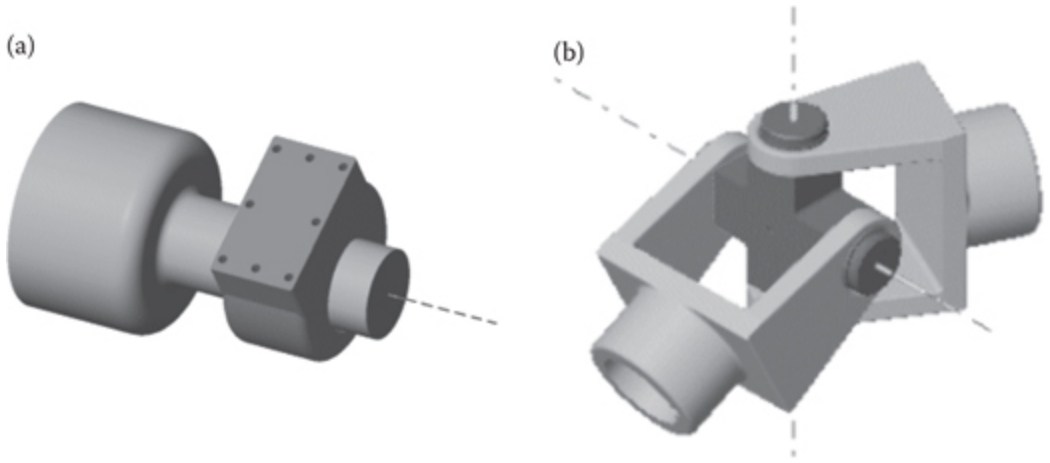



**FIGURE 1.2**
Schematics of a cylindrical joint (a) and a universal joint (b). (From Mathworks Inc. 2024. Mathworks. With permission.) ⏎

## Universal Joint (U)

Similar to the cylindrical joint, the universal joint facilitates movement in two degrees of freedom between two interconnected components. Nevertheless, it differs from the cylindrical joint in that it creates two rotational degrees of freedom between the linked components, often oriented at right angles to each other. As depicted in Figure 1.2(b), this type of joint is also not commonly used in industrial

robots, primarily due to the previously mentioned reasons regarding two-degree-of-freedom joints. However, in some parallel robots, it serves as a passive intermediate joint. In automotive applications, this joint is used in power transmission systems in four-wheel-drive vehicles.

## Spherical Joint (S)

A spherical joint allows the rotational movement of two links about all three principal axes. As shown in Figure 1.3(a), this type of joint restricts translational movements of the two links but enables rotational movement in all three directions, providing three degrees of freedom. A spherical joint is typically constructed by assembling a spherical ball into a precisely dimensioned spherical socket. This joint is commonly used in the structure of parallel robots. However, due to the complexity of constructing an active joint that allows rotational movement in three axes, spherical joints are often used in a passive form, usually at the beginning and end of linear cylinder-piston actuators. In serial robots where each joint corresponds to a connected actuator, instead of using an active spherical joint, three revolute joints are often employed, with their rotational axes intersecting at a single point. This setup allows each joint to be actuated independently by separate motors.

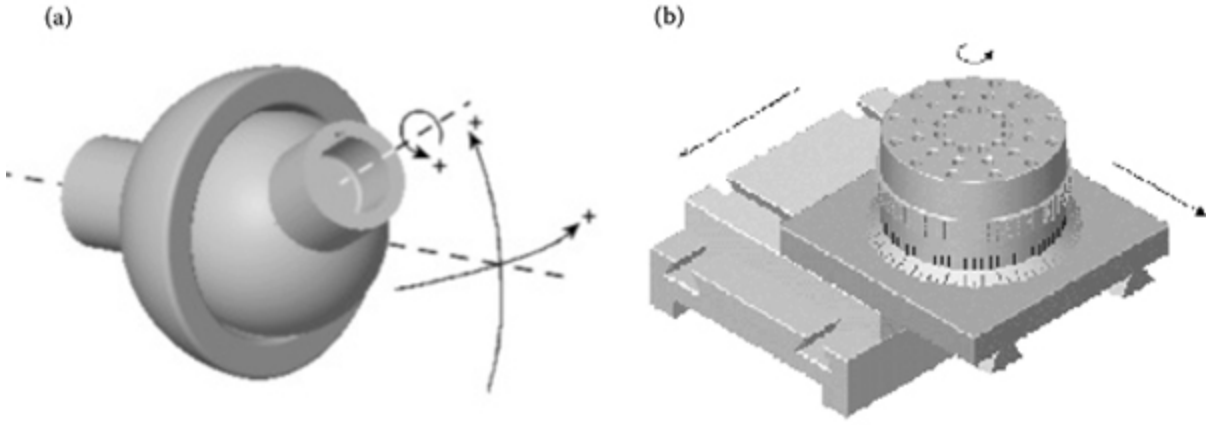



**FIGURE 1.3**
Schematics of a spherical joint (left) and a planar joint (right) (From Mathworks Inc. 2024. Mathworks. With permission.) ⏎

## Planar Joint (E)

A planar joint allows for two translational degrees of freedom along a plane of contact and one rotational degree of freedom about an axis normal to the plane of contact, as shown in Figure 1.3(b). Hence, it imposes three constraints and provides three degrees of freedom. This type of joint provides three degrees of freedom for movement between two links, and therefore, it is not considered one of the primary joints in robots. While constructing this joint is not challenging, creating actuators that can achieve three simultaneous degrees of freedom is difficult. Consequently, the use of this joint is limited to specific cases in robots where such multi-directional motion capabilities are necessary.

### 1.2.3 Kinematic Chains

In robotics, a kinematic chain refers to a sequence of interconnected links and joints that form a structure capable of relative motion. These chains are fundamental to the movement and functionality of robotic systems and mechanical assemblies. Serial kinematic chain is the most common type, where links are connected end-to-end in a linear fashion, with each joint providing one degree of freedom. In a parallel kinematic chain, links are connected to a common base and end-effector simultaneously, typically using multiple independent branches or limbs. Parallel kinematic chains are valued for their ability to distribute loads and for enhanced stiffness.

Furthermore, a kinematic chain is classified based on its connectivity and structure. When each link in a kinematic chain connects to other links through at least two distinct paths, it is termed as a closed-loop kinematic chain. This configuration allows for multiple routes of connection between its components, enhancing redundancy and potential fault tolerance. Conversely, in an open-loop kinematic chain, each link connects to its adjacent link through only one path. This type of chain lacks redundancy in connections and typically follows a straightforward sequential path from one link to the next.

In cases where a mechanism incorporates both closed-loop and open-loop kinematic chains, it is referred to as a hybrid kinematic chain. Hybrid configurations offer a balance of structural integrity and simplicity, leveraging

the advantages of closed-loop redundancy alongside the straightforwardness of open-loop pathways. These distinctions in kinematic chain types play a crucial role in the design and operation of mechanisms and machines, influencing their reliability, flexibility, and overall performance in various applications.

A mechanism refers to a system of rigid bodies (links) connected by joints (also known as kinematic pairs) to manage force and motion. As it is shown in Figure 1.4, a kinematic chain is called a *mechanism*, when one of its links is fixed to the ground, which is called the base. A *machine* is an assembly of one or more mechanisms, along with electrical and/or hydraulic components, used to transform external energy into useful work. Although in many texts the terms mechanism and machine are used synonymously, there is a definite difference between them, according to the given definitions. In other words, mechanisms are used for transmission of the motion, and they can be converted to machines if equipment such as digital controllers, instrumentation systems, actuators and other accessories are incorporated into their structure to produce useful work.

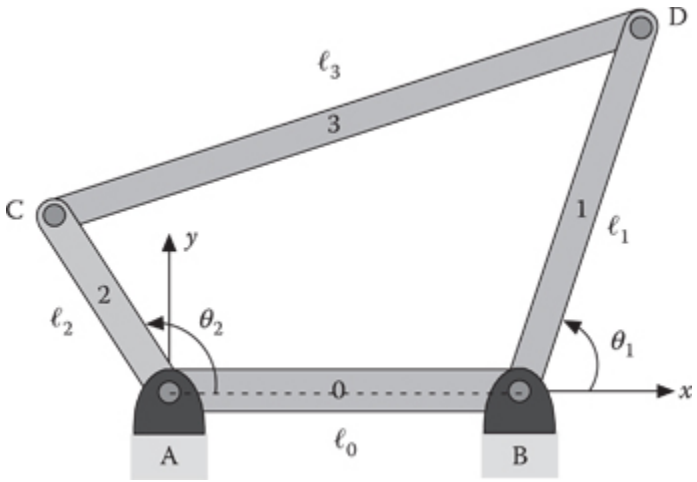



**FIGURE 1.4**
Schematics of a four bar mechanism. ⏎

## 1.3 Robot Degrees of Freedom

The degrees of freedom (DOF) of a mechanism refers to the number of independent inputs required to fully specify its configuration. Determining the DOF of a mechanism typically involves a general relation based on the number of links and the types of joints it incorporates. In this context, a joint with $n$ DOF can be understood as equivalent to $n$ binary joints, each with one independent DOF. For example, a cylindrical joint $C$ can be decomposed into two independent binary joints—one of revolute type and the other of prismatic type. Similarly, a spherical joint $S$ can be decomposed into three independent binary joints, all of the revolute type, with intersecting axes.

Let us define $\lambda$ as the degree of freedom of motion occurring in a space, where for a planar mechanism $\lambda = 3$, and for a general spatial mechanism $\lambda = 6$. Intuitively, the number of degrees of freedom (DOF) of a mechanism equals the sum of DOF associated with all the links (excluding the base), minus the constraints imposed by the joints. Therefore, the degrees of freedom of a mechanism can be determined by the following relation:

$$F = \lambda(n - 1) - \sum_{i=1}^{j} c_i \tag{1.1}$$

in which,

$F$ = degrees of freedom of the mechanism,
$\lambda$ = degree of freedom of the space,
$n$ = number of links in the mechanism including the base,
$j$ = number of binary joints of the mechanism, and
$c_i$ = number of constraints imposed by joint $i$,

However, the number of constraints imposed by a joint is equal to $\lambda$:

$$\lambda = c_i + f_i. \tag{1.2}$$

Therefore, the overall number of constraints imposed by the joint is

$$\sum_{i=1}^{j} c_i = \sum_{i=1}^{j} (\lambda - f_i) = j\lambda - \sum_{i=1}^{j} f_i. \tag{1.3}$$

By substituting Equation 1.3 into Equation 1.1, we arrive at the well-known Chebychev–Grübler–Kutzbach (1917) criterion [15].

$$F = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i \tag{1.4}$$

The Chebychev–Grübler–Kutzbach criterion applies under the condition that there are no passive degrees of freedom in the mechanism. Passive degrees of freedom refers to situations where a link, equipped with joints allowing rotation or translation, can move freely without transmitting torque or force along that particular axis. For instance, a link with spherical joints at both ends allows rotation around the axis defined by these joints without transmitting torque.

While these passive degrees of freedom contribute to the overall motion capability of the mechanism, they do not facilitate active force or torque transmission. Hence, they

are typically excluded from the count of degrees of freedom in mechanisms. If $f_p$ denotes the total number of passive degrees of freedom, then the active degrees of freedom can be expressed as:

$$F = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i - f_p. \tag{1.5}$$

in which,

$F$ = degrees of freedom of the mechanism,
$\lambda$ = degree of freedom of the space,
$n$ = number of links in the mechanism including the base,
$j$ = number of binary joints of the mechanism,
$f_i$ = degrees of relative motion permitted by joint $i$, and
$f_p$ = the total number of passive degrees of freedom.

**Example 1.1 (Piston-crank mechanism)** *Figure 1.5 depicts a planar piston-crank mechanism comprising four links that form a single-loop closed kinematic chain. Determine the active degrees of freedom available in this mechanism.*

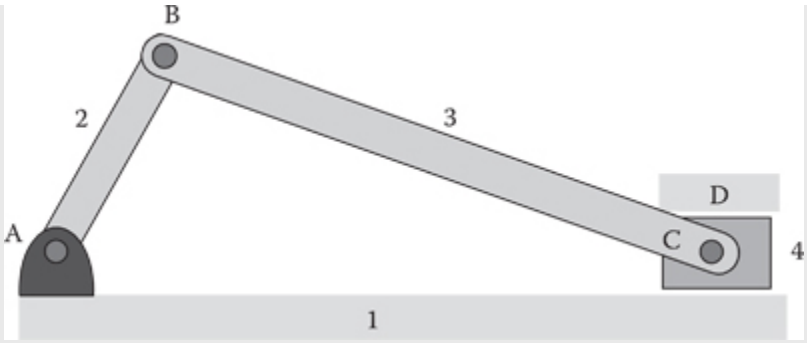



**FIGURE 1.5**
Schematics of a piston–crank mechanism. ⏎

**Solution:** For this mechanism, $\lambda = 3$ since it is planar. The total number of links, including the base, is four, $n = 4$, and there are four binary joints, $j = j_1 = 4$, with three being revolute joints and one being a prismatic joint. Additionally, there are no passive degrees of freedom, $f_p = 0$. Applying the Chebychev–Grübler–Kutzbach criterion (Equation 1.5) yields:

$$F = 3 \times (4 - 4 - 1) + (4 \times 1) - 0 = 1$$

The piston–crank mechanism possesses a single degree of freedom and is primarily employed to convert the rotational motion of the crank into the cyclic translational motion of the piston. In other applications, such as in combustion engines, the cyclic force produced by the piston is transmitted as torque to the crankshaft.

**Example 1.2 (Planar 4RPR parallel mechanism)** *Figure 1.6 depicts a planar 4RPR parallel mechanism, where the movement of the platform is driven by four identical piston–cylinder actuators. In this mechanism, the limbs form multiple closed-loop kinematic chains. Determine the active degrees of freedom available in this mechanism.*

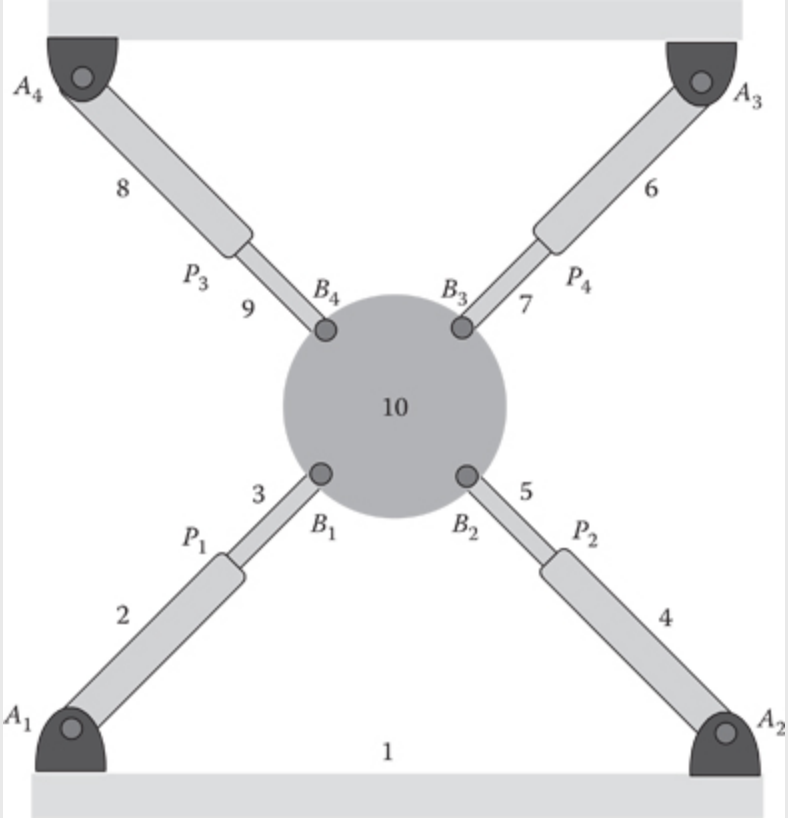



**FIGURE 1.6**
Schematics of a planar 4RPR parallel mechanism. ⏎

**Solution:** For this mechanism, $\lambda = 3$ since it is planar. The number of links, including the base, is ten ($n = 10$) as shown in Figure 1.6. The mechanism also contains

eight binary revolute joints ($A_i$'s and $B_i$'s) and four binary prismatic joints ($P_i$'s), giving us $j = j_1 = 12$. Additionally, there are no passive degrees of freedom ( $f_p = 0$). Applying the Chebychev–Grübler–Kutzbach criterion (1.5) yields:

$$F = 3 \times (10 - 12 - 1) + (12 \times 1) - 0 = 3$$

This planar mechanism has three degrees of freedom. It is important to note that the DOF of the mechanism equals $\lambda$, indicating that the mechanism has no kinematic redundancy. However, since four actuators are employed in its structure, the mechanism possesses one degree of actuator redundancy.

**Example 1.3 (Stewart–Gough Platform)** ⏎ *Figure 1.7 illustrates a spatial parallel manipulator known as the Stewart–Gough platform. In this manipulator, the spatial motion of the moving platform is achieved through six identical piston–cylinder actuators. Determine the active degrees of freedom available in this mechanism.*

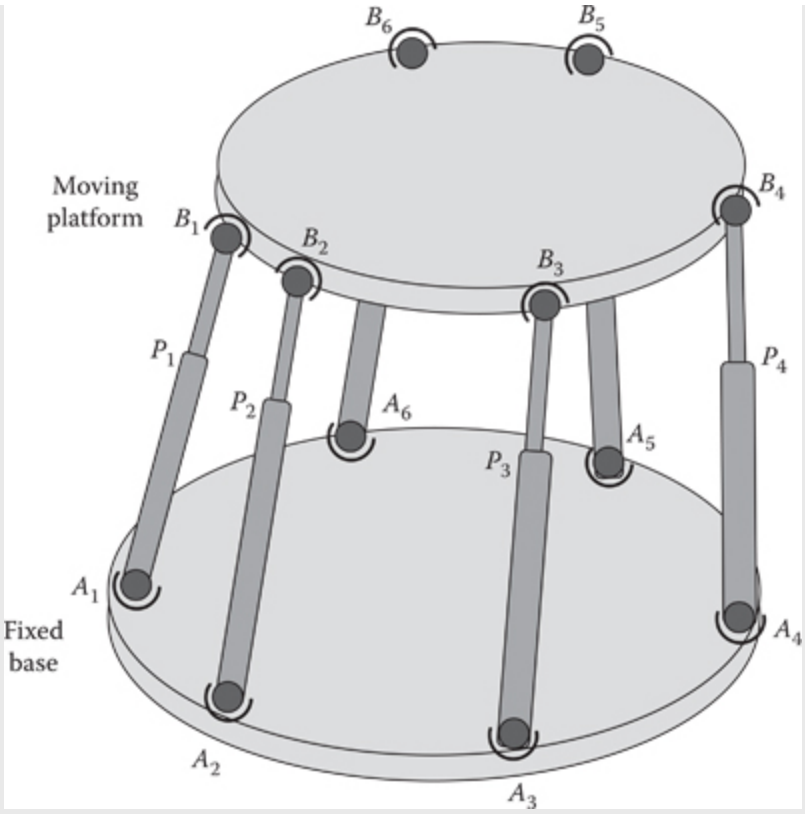



**FIGURE 1.7**
Schematics of a Stewart–Gough Platform. ⏎

**Solution:** For this mechanism $\lambda = 6$, since the mechanism is spatial. The number of links, including the base, is fourteen $(n = 6 \times 2 + 2 = 14)$. There are also $6 \times 2 = 12$ twelve spherical joints $A_i$'s and $B_i$'s and six binary prismatic joints, $P_i$'s on the mechanism; hence, $j_1 = 6$, $j_3 = 12$ and $j = 18$. Furthermore, for each SPS kinematic structure, there exists one passive degree of freedom, and therefore, $f_p = 6$. Applying the Chebychev–Grübler–Kutzbach criterion (Equation 1.5) yields to

$$F = 6 \times (14 - 18 - 1) + (12 \times 3 + 6 \times 1) - 6 = 6$$

This spatial manipulator has six degrees of freedom and can function as a general parallel manipulator, capable of producing complete spatial movement with three degrees of freedom for position and three degrees of freedom for orientation. The number of actuators equals the degrees of freedom of the manipulator, indicating there is no actuator redundancy in this system.

As seen in the examples of parallel mechanisms, the number and placement of actuators must be chosen carefully to achieve complete control over the manipulator's motion. Most robot manipulators have the same number of actuators as the required degrees of freedom to perform their tasks. These are called *fully parallel* manipulators. Although this structure is cost-efficient, it limits the manipulator's ability to perform the required tasks in an agile way. Many biological mechanisms, such as the human shoulder, arm, and hand, benefit from redundancy in actuation.

In general, redundant manipulators use more actuators than the required degrees of freedom to perform tasks with additional desirable user-defined characteristics. To control the required motion in these types of manipulators, an appropriate distribution of actuator effort is necessary. The freedom to optimally utilize the degree(s) of redundancy is

usually formulated as an optimization problem to minimize a user-defined performance criterion, such as optimal force distribution, singularity avoidance, or other performance indices. Such optimization problems and their corresponding solutions are discussed in more detail in this book in Section 6.6.

## 1.4 Robot Classification

Robots can be classified based on various criteria, with one of the primary classifications being based on degrees of freedom. A *general-purpose* robot typically has six degrees of freedom, allowing it to manipulate objects freely in space. For specific applications, such as assembly tasks, only four degrees of freedom might be sufficient. From this perspective, a *redundant robot* is one that has more actuators than the required degrees of freedom, while an *underactuated robot* has fewer actuators than necessary. A redundant robot offers more flexibility in moving objects, helping to avoid obstacles or singularities that may occur within the robot's workspace.

Another widely used approach to categorize robots is by their kinematic structure. A *serial robot* is composed of an open-loop kinematic chain, whereas a *parallel robot* features multiple closed-loop chains. Furthermore, a *hybrid robot* combines both open- and closed-loop chains within its structure. Unlike the human arm and wrist, which use a

complex arrangement of muscles to enable various movements, mechanical robots often employ an open-chain or serial configuration, with one actuator assigned per degree of freedom. In contrast, parallel robots use a closed-chain structure with a coordinated set of actuators to achieve multi-degree-of-freedom motion.

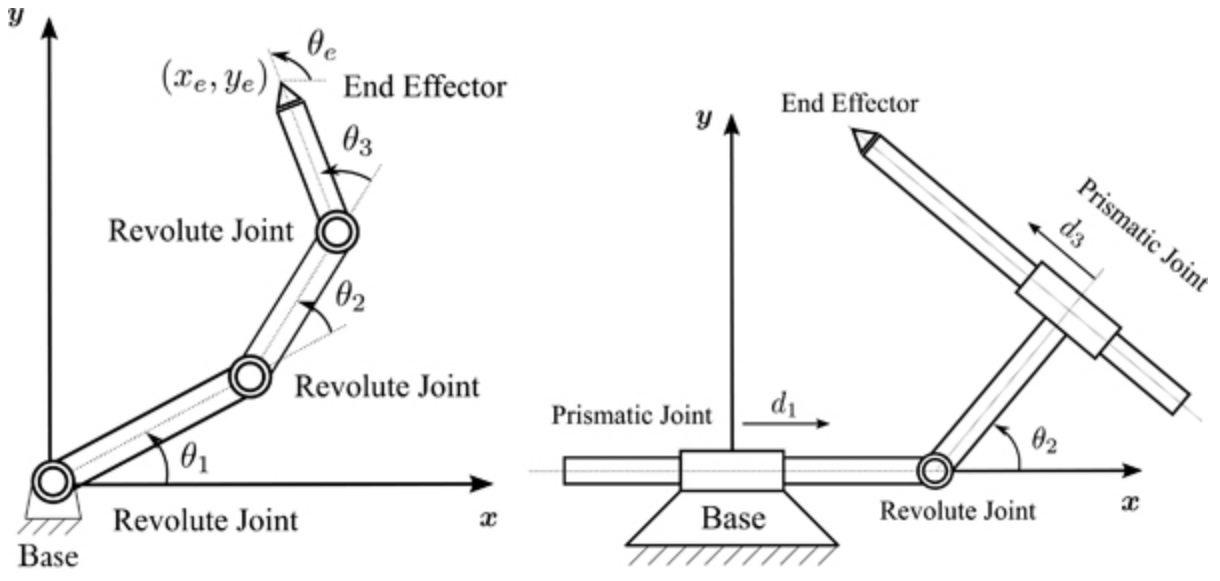
To illustrate, consider the human shoulder and wrist, where multiple muscles work together to produce three degrees of freedom in rotational motion. Meanwhile, the upper arm-elbow-lower arm arrangement uses direct actuation for flexion-extension movements. This demonstrates the use of a hybrid structure in a biological arm, where a serial structure is employed for a larger workspace, and a parallel structure is utilized for precise and stiff movements. This book provides a thorough analysis of parallel structures.

In the robotics literature, it is customary to represent a robot's abbreviated name by depicting the sequence of joint types. For instance, consider a simple planar robot consisting of two primary revolute joints denoted as R. The kinematic structure of this robot can be represented as RR or 2R. To clarify the type of actuation for the robot, the active joint, actuated by an actuator, is underlined. For example, if only the first joint of the two-jointed robot mentioned above is active, it is represented as $\underline{R}$R. If both joints are active, it can be represented as $\underline{RR}$ or simply as 2$\underline{R}$.

### 1.4.1 Serial Robots

Within the robot's kinematic structure, some robots consist of a continuous sequence of interconnected joints in a serial configuration. In this case, by mounting the first joint on a base, the robot's end-effector will be suspended in space. This kinematic structure includes an open kinematic chain, which is known in robotics literature as a “serial robot structure.”

One of the simplest serial robot can be seen in Figure 1.8(a). This robot is planar and has three revolute joints, represented as RRR or 3R. The robot's base is positioned on the ground, and as illustrated in the figure, it can be designed to either face upward or be suspended. By controlling the three robot joints motions with their actuators, the robot's end-effector's position and orientation represented as $(x_e, y_e, \theta_e)$ can be controlled.




(a) Schematics of an RRR robot.

(b) Schematics of an PRP robot.

▶ Long Description for Figure 1.8

**FIGURE 1.8**
Serial planar robots with revolute and prismatic joints. ⏎

The primary prismatic joint can also be employed in the kinematic structure of serial robots. This joint is typically actuated using hydraulic or pneumatic cylinder-piston actuators or linear motors and can be positioned either at the base or in a sequential arrangement of the robot's joints. An example of such robots used for rapid pick and place application on a plane is depicted in Figure 1.8(b), This robot features two prismatic joints and is designed with a PRP structure. The presence of a right angle in the design of the second arm showcases flexibility in the arrangement of the robot's joints. This flexibility aids the designer in creating an appropriate kinematic structure based on the intended application.

To facilitate the movement of objects in three-dimensional space, a common choice is to use a robotic arm with joints that allow movements beyond a single plane. The robot depicted in Figure 1.9 is the PUMA[1], designed by the Unimation company, that has found extensive use in various industries. As observed in this figure, the first joint of the PUMA robot depicted in this figure is oriented vertically on the base, and by rotating this joint, out-of-plane movements are generated in this robot.

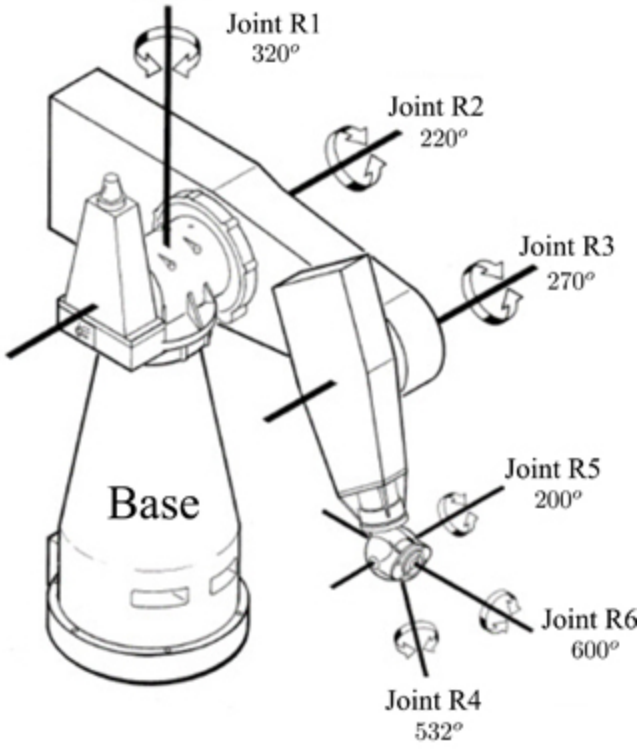



**FIGURE 1.9**
Schematics of a PUMA robot with a general 6R kinematic structure. ⏎

This robot is constructed with six revolute joints and follows a kinematic structure of 6R. The first three joints, along with their corresponding links, form the robot's arm and provide access to the workspace. Meanwhile, the three end-effector joints, designed with intersecting joint axes at a single point, create an appropriate design for the robot's wrist with a spherical actuated joint.

Due to its six degrees of freedom in three-dimensional space, the PUMA robot is highly versatile and has been widely adopted in various industries for different

applications. Unimation produced numerous PUMA models in the classes of 200, 500, and 700. The PUMA 560C model, the best-selling one, had a reach of 878 mm and a payload capacity ranging from 2.5 to 4 kg, making it suitable for a wide range of industrial applications. Unimation, which manufactured various types of these robots until the 1980s, was acquired by Westinghouse in 1980. In 1988, it continued its operations under the name Stäubli. Today, Stäubli is considered one of the leading manufacturers of industrial robots, offering new designs with diverse workspaces.

Another prominent serial robot widely used in the industries is renowned as the SCARA[2]. It is recognized as one of the most successful serial robot structures, particularly acclaimed for its effectiveness in rapid object handling. The kinematic structure of this robot is RRRP, where the first three joints are revolute joints with parallel rotation axes, and the fourth joint is prismatic. As illustrated in Figure 1.10, due to the configuration of three parallel revolute joints, this robot exhibits a degree of compliance in the $x-y$ plane while maintaining complete rigidity along the vertical $z$-axis.

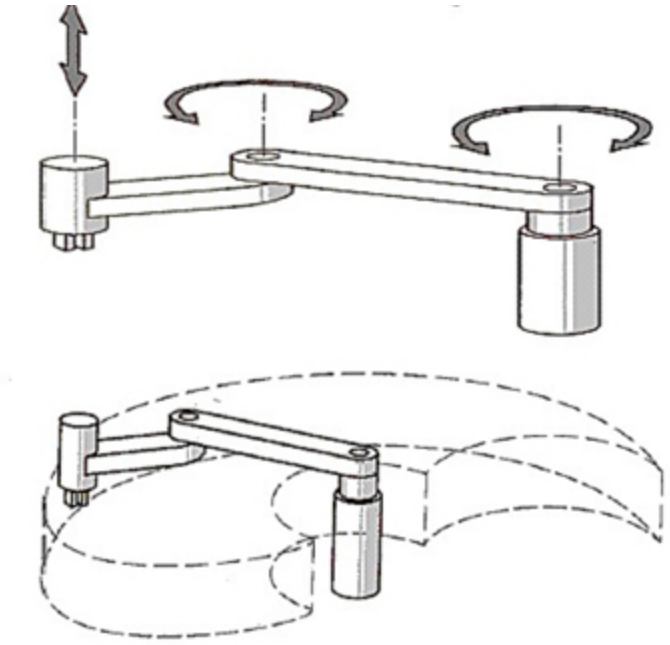


**FIGURE 1.10**
The schematics and workspace of a SCARA robot. ⏎

---

[1]Programmable Universal Manipulation Arm ⏎
[2]Selective Compliance Articulated Robot Arm ⏎

This characteristic allows the robot to efficiently perform tasks involving object manipulation and part assembly. Moreover, the utilization of suitably designed belt-driven power transmission systems for the initial three joints allows for a substantial reduction in the weight of the moving components, and therefore, enhances the speed of the robot. With this structure, it is possible to achieve linear speeds of up to 7 m/s and perform 120 picks per minute. To observe the SCARA robot in action, performing swift pick-and-place maneuvers, just click on the designated link[1] to access the video.

Serial robots have three primary drawbacks: poor load-to-mass ratio, low positioning accuracy, and high structural compliance. For a general six degrees of freedom serial manipulator, the load-to-mass ratio is less than 0.15, whereas for a SCARA-type robot, this ratio can increase to a maximum of 0.25 [6]. This means that a typical serial robot with a payload of 100 kg would weigh more than 670 kg, or at least 400 kg for a SCARA-type robot.

When it comes to positioning accuracy, there are two key definitions. Absolute accuracy refers to the absolute positioning error of the end-effector when moving to a desired pose. Repeatability, on the other hand, is defined as the maximum distance between two positions of the end-effector when reaching the same desired pose from different starting positions. Although users are primarily interested in absolute accuracy, manufacturers generally specify repeatability, which tends to be much better than absolute accuracy.

In serial robots, positioning accuracy is usually poor because errors occurring at each link accumulate toward the end-effector. This accumulation of errors also contributes to poor structural compliance in serial manipulators. However, these drawbacks can be significantly reduced by fundamentally altering the robot's structure. Parallel robots, which are designed based on a different structural concept, can effectively mitigate these issues.

---

[1] https://youtu.be/-m1oKuFkSTE ⏎

### 1.4.2 Parallel Robots

A parallel robot, also referred to as a parallel manipulator, is characterized by its structural arrangement, where multiple kinematic chains are linked to a shared mobile platform. These chains, commonly known as *legs*, with one end affixed to the base and the other end connected to the platform, establish a parallel configuration. In the context of parallel robots, the term “moving platform” is often used instead of end-effector, indirectly representing the multiple connections to the robot's base. The mobile platform is usually employed for tasks like positioning, manipulating objects, or executing specific operations. The kinematic loops of the legs typically exhibit similar kinematic structures and consist of various joints and links, akin to a serial robot. The simplest type of these arms consists of a prismatic actuated joint connected to the base and the moving platform through two passive spherical or universal joints, denoted as S or U. This basic structure allows the use of hydraulic piston-cylinders to create robots with high payload capacities, such as those used in aircraft simulators.

The distinguishing feature of parallel robots is that all of their actuators are located on the base, while the moving platform moves in a coordinated manner due to the interconnected legs. This design provides several advantages, including high precision, stiffness, and the ability to handle heavy loads. Parallel robots are known for their use in applications that require accuracy, speed, and

precision, such as flight simulators, medical surgery devices, and certain types of manufacturing processes.

In Figure 1.11, a schematic of a general-purpose parallel robot is depicted, where six closed kinematic chains connect the base to the moving platform. In this versatile setup, any kinematic structure can be used to link the base to the moving platform. Additionally, although this kinematic structure incorporates several revolute joints denoted as R and one prismatic joint denoted as P, typically only one of these joints is active, while the others remain passive. Consequently, with appropriate design, all the robot's actuators can be installed on the base, reducing the weight of the moving components. Furthermore, as evident in this figure, the weight of the mobile platform is distributed across multiple legs, enabling the use of lighter actuators in the robot. Given the inherent features of parallel robots, the payload-to-weight ratio and the robot's rigidity significantly increase, leading to improved performance accuracy.

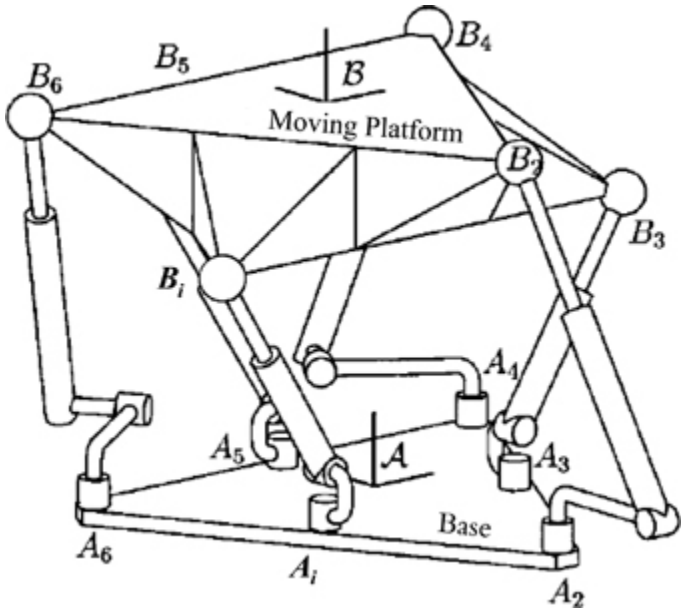


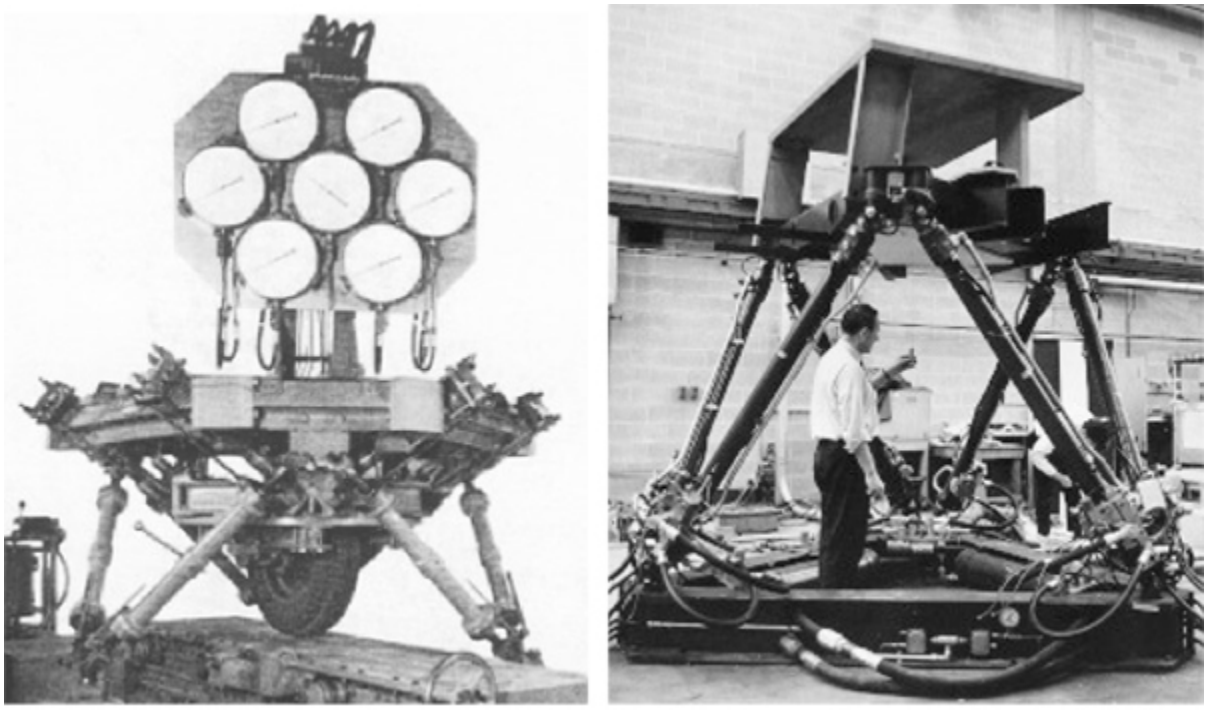

**FIGURE 1.11**
Schematics of a general parallel robot. ⏎

Nevertheless, parallel robots have a notable limitation concerning their confined workspace. The close proximity of neighboring arms and joints introduces the possibility of collisions between various components, including arm components to the base, or the moving platform, consequently restricting the robot's workspace. Furthermore, in parallel robots, at singular points and under specific conditions, the robot may lose its stability and become uncontrollable.

## Stewart–Gough Platform

As reviewed by Bonev [16], the most celebrated parallel robot, commonly known as the Stewart platform, was paradoxically first invented by V. E. Gough in 1947.

Gough's invention, called a universal rig, was inspired by the design of an octahedral hexapod used to determine the properties of tires under combined loads. As illustrated in Figure 1.12(a), the Gough platform features six independently actuated limbs, with the lengths of the legs being controlled to move the platform to a desired position and orientation. According to Gough's 1956 paper [17], the octahedral hexapod was not an entirely new invention at the time; systems with six struts (hexapods) were already in use, typically featuring three vertical and three horizontal struts. These hexapods were quite popular and continue to be used in various applications [18].



(a) Gough's rig to measure the tire wear in 1947. (b) Klaus Cappel's flight simulator in mid sixties.

**FIGURE 1.12**

Two early applications of Stewart–Gough platform structure. ⏎

Gough's innovation lay in the arrangement of the six struts. Seeking relatively large ranges of motion, he selected a symmetrical arrangement forming an octahedron. The machine was built in the early 1950s and became fully operational in 1954. While Gough was the first to invent and construct the popular octahedral hexapod, Klaus Cappel independently designed and patented the same hexapod [19], subsequently licensing it to the first flight simulator companies. A picture of Cappel's first flight simulator is shown in Figure 1.12(b).

The name Stewart became associated with this architecture because Gough's earlier work and a photograph of his platform were mentioned in the reviewer's remarks to a paper published by Stewart in 1965 [20]. In that paper, Stewart introduced a different hybrid design, shown in Figure 1.13, featuring three legs with two actuators each. Stewart proposed this six degrees of freedom motion platform for use as a flight simulator. However, this proposed parallel mechanism differs from the octahedral hexapod invented by Gough, which is often paradoxically referred to as the *Stewart platform*.

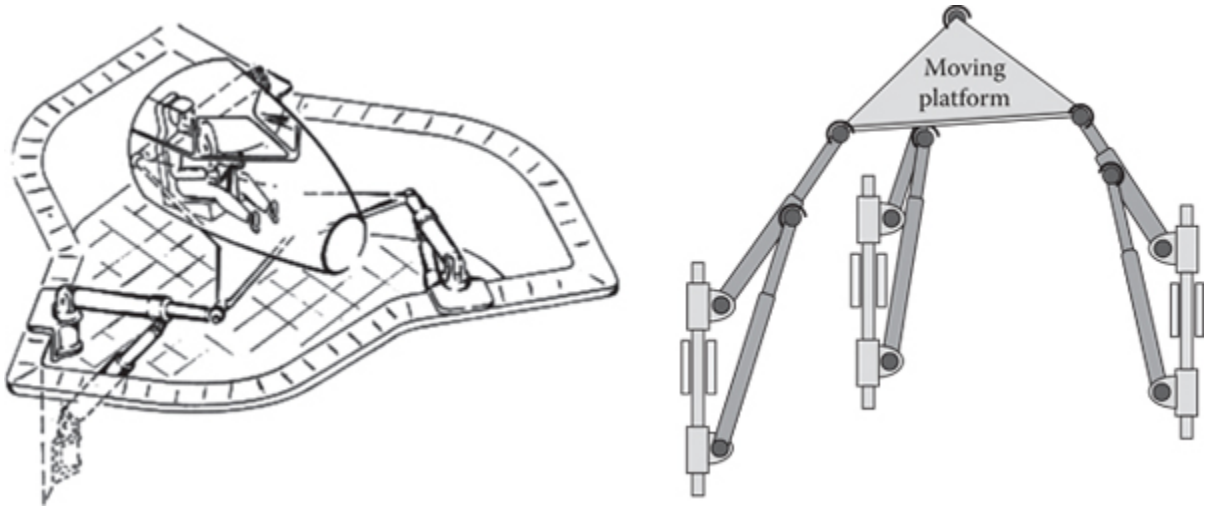



**FIGURE 1.13**
A schematics of the Stewart platform and its use as flight simulator. (From D. Stewart. A platform with six degrees of freedom. *Proceedings of the UK Institute of Mechanical Engineering*, 180(1): 371–386, 1965.) ⏎

There is no doubt that Stewart's remarkable paper had a significant impact on the subsequent development in the field of parallel kinematics. Various suggestions for the use of hexapods were made, many of which accurately predicted future applications. Despite this, in the literature, the famous octahedral hexapod mechanism is mostly referred to as the Stewart platform. In this book, we refer to variations of such a design as the Stewart–Gough platform (SGP) to acknowledge the original inventor, Gough. Given that this structure has six degrees of freedom and is a fully parallel mechanism, it has been extensively analyzed and used as the basis for similar parallel mechanisms in practice. For this reason, a thorough

analysis of this mechanism is provided throughout all chapters of this book.

The Stewart-Gough Platform (SGP) often referred to as the hexapod, holds significant prominence as an industrial example. It finds extensive applications in various domains, including aircraft, helicopters, and automotive simulators [21]. As shown in Figure 1.7, this robot employs a structure consisting of six similar arms with a kinematic structure of SPS or UPS, providing six degrees of freedom in three-dimensional space. Hence, due to the presence of six similar arms in this robot's structure, it is commonly denoted as 6SPS or 6UPS. The utilization of the universal joint (U) on the robot's base is preferred for design and ease of kinematic analysis, although the construction of the robot with a spherical joint (S) is simpler. Hydraulic piston-cylinders or linear electric motors are used as actuators in the prismatic joints of the robot. This enables the robot to provide the necessary six degrees of freedom in three-dimensional space, making it a suitable choice in applications that require a high payload capacity and a smaller workspace compared to serial robots with the structure 6R. You can watch a video clip featuring the 6D motion of a SGP by following this link[1].

[1] https://youtu.be/xiECumcaEx0 ⏎

## Delta Robot

In applications where six degrees of freedom are not required, simpler structures can be employed in parallel robots. Another highly successful example of these robots, commercially produced by various manufacturers, is the Delta robot. The Delta robot is an excellent replacement for the SCARA in pick and place applications and is widely used in the packaging industry. This robot is composed of three similar arms, attached to the base with revolute joints (R). The key idea behind this robot is the parallelogram structure in the second link, which keeps the orientation of the moving platform fixed. With this structure, the links of each of the three arms can be designed with minimal weight, enabling very fast movements. Due to these characteristics, commercial versions of these robots have been produced with a rapid displacement speed of over 200 picks per minute [22]. To witness the delta robot in action, executing rapid pick-and-place movements, simply click on the provided link[1] to access the video.

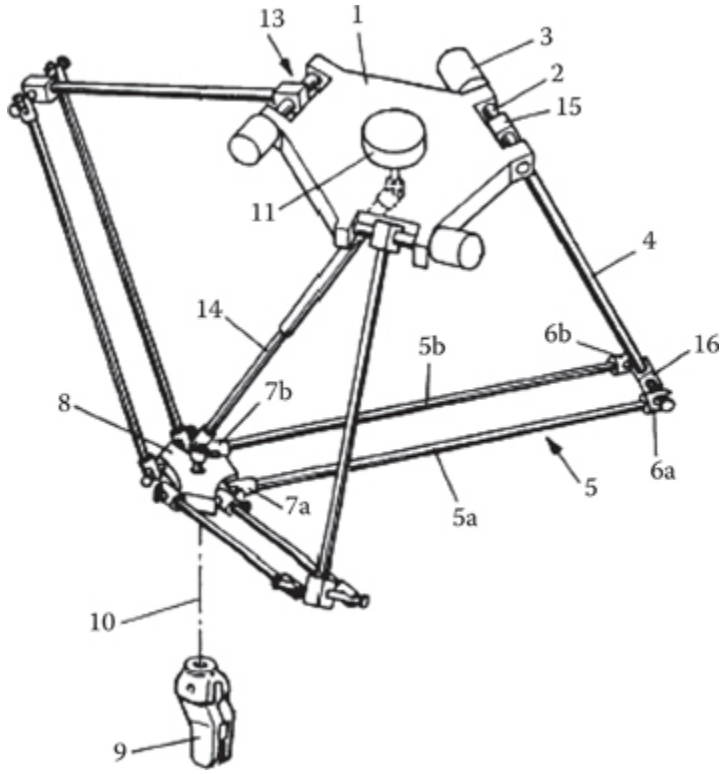



**FIGURE 1.14**
The schematics of a Delta robot. (From R. Clavel. Device for the movement and positioning of an element in space. US Patent No. 4,976,582, December 1, 1990. With permission.)

The Delta robot was invented in the early 1980s by Reymond Clavel at EPFL in Switzerland [23]. This new type of robot was designed to manipulate light and small objects at very high speeds. In 1987, Demaurex company acquired a license for the Delta robot and began producing them for the packaging industry. Reymond Clavel presented his doctoral thesis in 1991 and was awarded the Golden Robot Award in 1999 for his work and development of the Delta robot. Also in 1999, ABB Flexible Automation started

selling its version of the Delta robot, known as the FlexPicker.

The Delta robot is a parallel robot, often considered a spatial generalization of a planar four-bar mechanism. It has four degrees of freedom: three translational and one rotational. The key concept behind the Delta robot is the use of parallelograms, which ensure that an output link maintains a fixed orientation relative to an input link. The implementation of three such parallelograms completely constrains the orientation of the mobile platform, leaving it with only three purely translational degrees of freedom.

---

[1] https://youtu.be/6JIjCpW35CM ⏎

The robot's base is mounted above the workspace, with all actuators located on this base. From the base, three middle-jointed arms extend, typically made of lightweight composite material. The ends of these three arms are connected to a small triangular platform. Actuating the input links moves the triangular platform in the $x$, $y$ or $z$ directions. The actuation can be achieved using either linear or rotational actuators. Additionally, a fourth leg is employed to transmit rotary motion from the base to an end-effector mounted on the mobile platform.

Since the actuators are all located on the base and the arms are made of composite material, the moving parts of the Delta robot have minimal inertia. This design enables very high accelerations, up to 30 g, and speeds reaching 10 $m/s$. These characteristics make the Delta robot an ideal

choice for pick-and-place operations involving light objects ranging from 10 grams to 1 kilogram. Typically, Delta robots available on the market operate within a cylindrical workspace that is 1 meter in diameter and 0.2 meters in height [24].

Industries such as packaging, medical, and pharmaceuticals benefit greatly from the high speed of Delta robots. For instance, Adept has developed its Delta-type parallel robot called Quattro for the packaging industry. This manipulator features one degree of redundancy in actuation and comprises four parallelogram limbs. Additionally, the Delta robot's structure has been adapted to create commercial haptic devices, such as the Novint Falcon.

### 1.4.3 Hybrid Structures

The structure of the human arm and hand, as well as the locomotion mechanisms of certain animals and even insects, surpasses the simplicity of typical serial or parallel structures seen in industrial robots. Take, for instance, the human arm: the shoulder joint, with its ball-and-socket configuration, offers three rotational degrees of freedom. This joint operates akin to a ball joint, facilitated by four muscle groups that enable these rotations.

In contrast, the elbow and wrist joints in the human hand are interconnected by bones, resembling the parallelogram structure found in a delta robot. However, unlike simple hinge joints, each elbow and wrist joint independently

provides two rotational degrees of freedom. The intricate network of muscles and tendons in the forearm coordinates these angular movements, resulting in a combined total of four independent degrees of freedom. Thus, the human arm utilizes seven actuated degrees of freedom to control the six degrees of freedom at the wrist.

This intricate design, blending elements of both serial and parallel structures, demonstrates redundancy in actuation—a unique characteristic contributing to the exceptional motor skills of the human arm. The mathematical analysis of this topic will be explored in forthcoming chapters. Such remarkable designs are prevalent in numerous natural creations.

Another example is the human wrist, consisting of eight bone segments and six muscle groups, leveraging a parallel structure to enhance dexterity for its three degrees of freedom wrist motion. In contrast, each finger of the hand operates like a serial 3R arm. However, their attachment to the palm and the arrangement of muscles and tendons allow for out-of-plane movements for each finger.

Despite technological constraints, human-created robotic systems have not yet matched the performance of these remarkable biological designs. However, to meet specific performance demands and harness the advantages of both serial and parallel structures, engineers have developed specialized robotic designs, which integrate serial components for expansive workspace and flexibility,

alongside parallel mechanisms for enhanced stability and load-bearing capacity.

Hybrid robots embody systems that blend elements from both serial and parallel robot designs into a unified structure. They leverage the broad workspace and flexibility of serial robots and the stability and strength of parallel robots. By combining these distinct design approaches, hybrid robots achieve unique capabilities and adaptability tailored to specific tasks. This makes them ideal for applications requiring a blend of characteristics from both serial and parallel robots to optimize performance. Such robots often introduce innovative solutions by leveraging the diverse features offered by each design type.

To illustrate the concept of hybrid design, consider the quadruped robot “Spot,” which features a robotic arm specialized for manipulating objects in its vicinity. While each of Spot's legs could be designed with a serial structure, their connection to a mobile body functioning as a platform exhibits characteristics typically associated with parallel robots. In contrast, the robotic arm attached to Spot's body demonstrates traits more commonly found in serial robots as it integrates with the mobile platform. This example effectively demonstrates the integration of diverse robotic structures, particularly suited for applications such as exploration and rescue [25].

Another example is the cable-driven“SpiderCam” robot, equipped with a three-degree-of-freedom stabilizing gimbal

designed to eliminate vibrations and ensure camera stability for filming purposes. The cable-driven structure of the robot falls under suspended parallel robots, while the stabilizing gimbal employs a three-degrees-of-freedom serial structure [26]. In the creation of hybrid robots, it is common to incorporate both serial and parallel designs, leveraging the proven operational effectiveness of each design type. This fusion allows for the development of robots equipped with unique features tailored to specific tasks. Therefore, a solid understanding of the design principles underlying both serial and parallel robots is essential for the development of such innovative products. However, it is noteworthy that human-created examples of these robots still have significant ground to cover to replicate the complexities found in natural structures.

### 1.4.4 Cable–Driven Parallel Robots

Meeting the increasing demands for performance requires designing new types of robots with larger workspaces, capable of operating at higher accelerations. As discussed in the previous section, parallel manipulators generally outperform serial manipulators in terms of stiffness and acceleration requirements. However, their limited workspace and the presence of singular regions within that workspace restrict the applicability of parallel manipulators for large workspace requirements [27, 28].

In a cable-driven parallel manipulator (CDPM), the linear actuators of parallel manipulators are replaced with

electrically powered cable winches, resulting in an immediate increase in workspace size. For instance, consider the application of recording a football game using a flying camera over a stadium. In such scenarios, a cable-driven parallel structure can serve as a viable alternative. These systems utilize four pairs of cables to suspend a camera package with pan and tilt control. Commercially available systems with similar designs are known as Cablecam, Skycam, or Spidercam.

An international consortium of radio astronomers and engineers is actively exploring technologies to construct the Square Kilometer Array (SKA)—a groundbreaking cm-to-m wave radio telescope. The SKA aims to revolutionize our understanding of cosmic phenomena [29, 30]. However, a significant challenge lies ahead: the sensitivity barrier currently limits existing telescopes from making deeper observations at these wavelengths, especially when studying the early universe.

The Canadian proposal for the SKA design involves an array of 30-50 individual antennas. These antennas work in concert, combining their signals to achieve the resolution typically associated with a much larger single antenna. To achieve this, they employ the Large Adaptive Reflector (LAR) concept [31, 32] The LAR design, initially proposed by a group led by the National Research Council of Canada, has garnered support from university and industry collaborators. Notably, the LAR concept is applicable to telescopes with diameters of several hundred meters. While

the SKA represents an ambitious endeavor, the National Research Council of Canada pursued the design and construction of a 200m LAR prototype. This prototype holds promise for advancing radio astronomy and pushing the boundaries of our cosmic exploration.

The LAR (Large Adaptive Reflector) prototype comprises two essential components. The first component is a 200-meters-diameter parabolic reflector with a focal length of 500 meters. This reflector consists of actuated panels supported at the ground. The second component is the receiver package, which relies on a tension structure composed of multiple long tethers and a helium-filled aerostat. The schematic representation of this setup is shown in Figure 1.15. Notably, a one-third scale prototype of the multi-tethered aerostat subsystem has been successfully designed and implemented in Penticton, Canada, with funding from the Canada Foundation for Innovation. Despite its reduced scale, this system remains quite large, covering an impressive footprint of approximately one squared kilometer.

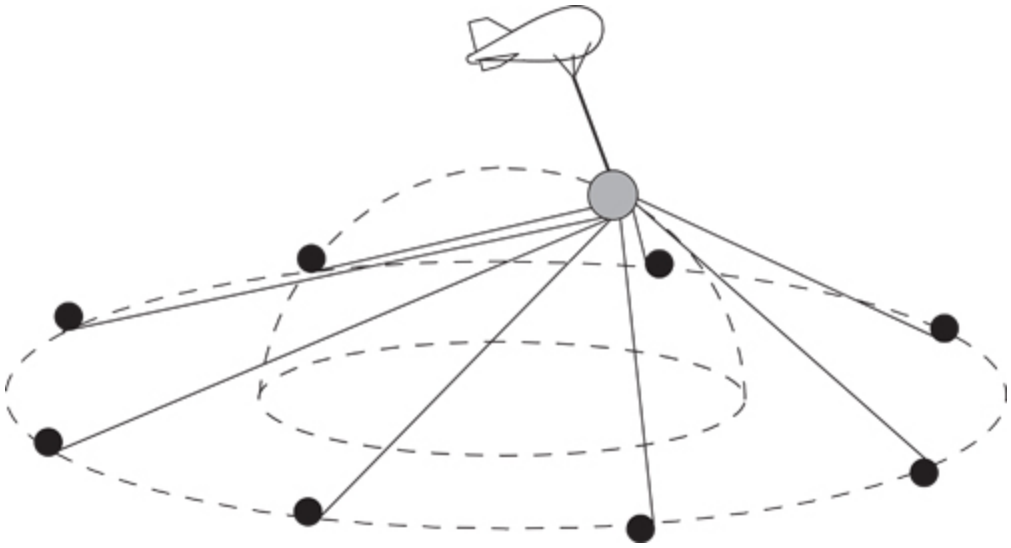


**FIGURE 1.15**
The schematics of cable–driven manipulator with eight cables, and aerostat used in LAR application. ⏎

In this system, achieving precise positioning of the feed (receiver) presents a significant challenge, especially in the presence of disturbances like wind turbulence. As it is illustrated in Figure 1.15, the receiver is strategically moved to different locations on a circular hemisphere, and its positioning is meticulously controlled by adjusting the lengths of eight cables using ground winches. Notably, the cable-driven parallel manipulator employed in this design is a six-degrees-of-freedom (DOF) manipulator with two degrees of actuator redundancy.

Another application of cable–driven robot in astronomy is the Five-hundred-meter Aperture Spherical Telescope (FAST). This telescope, also known as Tianyan (meaning “Sky's/Heaven's Eye”), is a remarkable radio telescope situated in the Dawodang depression, a natural basin in

Guizhou, southwest China. FAST boasts a 500-meters diameter dish, ingeniously constructed within the natural landscape depression. What sets FAST apart is its innovative design: an active surface composed of 4,500 metal panels that dynamically form a moving parabolic shape in real-time. The cabin housing the feed antenna is suspended on cables above the dish, allowing automatic movement to receive signals from various directions. FAST operates across wavelengths ranging from 10 cm to 4.3 m [33].

Construction of FAST began in 2011, and it achieved first light in September 2016. After three years of testing and commissioning, it was officially declared fully operational in January 2020. Notably, FAST made its initial discovery—identifying two new pulsars—in August 2017. Since then, it has continued to contribute significantly to astronomical research [34].

The concept of using cable-driven parallel robots extends beyond applications that solely require a very large workspace, as mentioned previously. This idea is effectively applied in scenarios where precise and rigid robots need to operate at high accelerations within a relatively larger workspace than what conventional parallel robots can achieve. An example of such a robot is the WARP (Wire-Actuated Robotic Platform)—a six-degree-of-freedom cable-driven robot with an over-constrained, lightweight moving platform. Thanks to its low mass structure, the WARP

manipulator achieves ultra-high speeds with acceleration exceeding 40 g [35, 36].

The NIST RoboCrane serves as an intriguing example. It shares similarities with a Stewart-Gough platform, but instead of struts, it employs an octahedral assembly of cables. This remarkable manipulator boasts six degrees of freedom and excels at lifting and precisely manipulating heavy loads across large volumes with fine control in all six dimensions [37]. In laboratory settings, the RoboCrane has demonstrated its ability to manipulate tools such as saws, grinders, and welding torches. Additionally, it can lift and precisely position heavy objects like steel beams and cast iron pipes.

Notably, a commercial version of the RoboCrane has been developed for the Air Force, enabling rapid tasks such as paint stripping, inspection, and repainting of very large military aircraft like the C-5 Galaxy. Looking ahead, the RoboCrane holds promise for diverse applications, including, shipbuilding, construction of high-rise buildings, highways, bridges, and tunnels, port facilities, cargo handling, ship-to-ship cargo transfer at sea, radioactive and toxic waste cleanup, underwater tasks such as salvage, drilling, cable maintenance, and undersea waste site management [38].

Cable-driven robots possess distinctive properties that make them appealing for a wide range of potential applications. However, practical challenges arise during the design and development of these robots. The crucial

distinction between cable-driven robots and conventional parallel robots lies in the fundamental property of cables: they operate exclusively under tension and can only pull objects, not push them. To ensure that cables remain under tension throughout their entire workspace, special care must be taken in the structure of cable-driven robots. One approach is to introduce an external force acting on the robot's moving platform. For instance, the gravity force is utilized in applications like the FAST telescope and RoboCrane, and Helium aerostat lifting force is employed in the LAR application.

However, it's essential to note that if gravity force alone is used to maintain cable tension, the robot's vertical acceleration is limited to no more than one g (gravitational acceleration). While this limitation is acceptable for crane applications where lifting heavy objects is the primary task, it is not recommended for general-purpose six-degrees-of-freedom robots. In such cases, a fully constrained structure with redundancy in actuation may serve as a viable alternative.

To further illustrate the concept of actuator redundancy, consider the motion of a rigid body controlled by cables along the $x$-axis in both positive and negative directions, as depicted in Figure 1.16. If only one cable is used for actuation, the moving object can be displaced in only one direction. However, when a redundant or extra cable is introduced for actuation, the object becomes capable of movement in both directions.

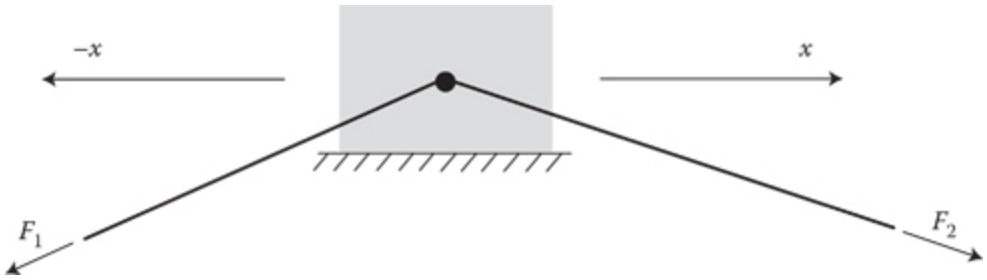



**FIGURE 1.16**
The concept of using two cables to move an object arbitrarily in one direction. ⏎

In a redundant parallel robot, the number of actuators exceeds the number of degrees of freedom. For instance, a redundant planar parallel robot must have at least four actuators, while a general six-degree-of-freedom redundant robot requires at least seven actuators. In conventional parallel robots, actuator redundancy is employed to avoid singularities within the workspace or achieve a more uniform distribution of actuation efforts. However, in cable-driven robots, redundancy is not optional—it is essential to ensure feasible tension forces for all cables throughout the workspace.

When introducing an additional degree of actuation in a robot, the overall analysis of system kinematics, dynamics, and control becomes more intricate. In the workspace analysis of cable-driven robots, specific assessments—such as force-feasible workspace or wrench-closure workspace—are introduced. These analyses examine the feasibility of maintaining tension in all cables within a given workspace volume. Moreover, addressing optimal methods for utilizing

these extra degrees of actuation—while ensuring force feasibility and minimizing other cost functions—becomes crucial. These optimal techniques open up a new area in the analysis of such robots, commonly referred to as redundancy resolution. For further details, you can explore Section 6.6 in the book.

## 1.5 Aims and Scope of the Book

As elaborated within this chapter, robotics, in its entirety, represents a multifaceted research domain that cannot be succinctly encapsulated within the confines of a single book. To provide clarity regarding the subjects addressed within this book, it might be advantageous to initially outline the scientific fields that fall outside the scope of this book. The design of robotic systems encompasses a broad spectrum of topics, many of which extend beyond the boundaries of this book. For instance, fields such as machine vision [39], speech processing [40], artificial intelligence [41], and programming and decision-making [42], are each well-structured scientific domains explored in various dedicated publications.

Moreover, the subjects of mobile and flying robots also draw from related scientific disciplines. In the field of mobile robots, topics like path planning and obstacle avoidance [43], kinematics, dynamics, and control of these types of robots [44], self-driving mobile robots [45],

probabilistic robotics [46], and the sensors and actuators employed in these robots [47] constitute vital areas of study that fall outside the scope of this book. Additionally, the intricate design of complex structures such as multi-fingered hands [48] and the tactile sensors integrated within them [49] necessitates comprehensive exploration, which will not be addressed within this book.

In our previous discussion about classifying robotic structures into serial and parallel robots, we observed that analyzing and designing parallel robots is inherently more complex than their serial counterparts. Moreover, expertise in serial robot analysis and design serves as a foundational prerequisite for understanding parallel robots. Given that this book primarily focuses on the analysis and design of parallel robots, I recommend that readers explore the fundamental concepts of robotic analysis and design in another book authored by the same writer in 2025, as referenced in [1].

In the context of designing parallel robots, this book will focus on the essential knowledge and skills required for their analysis and design in two crucial domains: mechanics and control. While we won’t address the mechanical design, actuator selection, sensor integration, or other aspects, our primary objective is to equip readers with the fundamentals needed to understand and work with parallel robots effectively. Through a comprehensive study of the subjects presented in this book, readers will gain the essential expertise to analyze both simple and intricate

structures within closed kinematic loop structures, commonly referred to as parallel robots. These robots possess multiple degrees of freedom and play a crucial role in various applications, including flight simulators, precision assembly, packaging, additive manufacturing, and surgical robotics.

Given the book's primary emphasis on parallel robots, it will explore the mathematical representation of motion in three-dimensional space. This includes precise definitions for both position and orientation. We'll explore various methods for representing these concepts. Additionally, the book will thoroughly examine the kinematic analysis of parallel robots using two distinct approaches: the conventional approach and the utilization of screw theory. Screw theory offers remarkable advantages when analyzing robot kinematics, seamlessly aligning with the fundamental principles of physical motion.

Kinematic analysis involves studying the geometric aspects of robot motion without considering the forces and torques responsible for that motion. In this analysis, we explore the relationship between the manipulator's geometric parameters and the resulting motion of the moving platform. In a comprehensive treatment, the book elaborates on case studies of three parallel robots, including a planar cable-driven robot. Cable-driven robots represent a promising new generation of parallel robots due to their ability to provide large workspaces. Furthermore, this analysis can be extended to consider

linear and angular velocities, which is known as Jacobian analysis.

The kinematic analysis of robot manipulators goes beyond static positioning and differential kinematics is a focused topic of analysis in this book. The Jacobian analysis not only reveals the map between the active joint variable velocities in a parallel manipulator to the linear and angular velocities of the moving platform but also constructs the necessary transformation to determine actuator forces based on the forces and moments acting on the platform. The book outlines a systematic approach for performing Jacobian analysis of parallel manipulators, and this method is examined using the same case studies introduced earlier in the book.

The Jacobian matrix plays a pivotal role in the field of robotics, particularly in the context of differential kinematic analysis, and this book provides an in-depth exploration of this fundamental concept. Many crucial aspects involved in the design of parallel robots necessitate the analysis of velocities and the Jacobian matrix of the robot. The true beauty of this knowledge emerges when one employs analytical methods to establish connections between the mathematical principles governing the kinematic analysis of robots and their physical characteristics. This process, inspired by the optimal design seen in nature, fosters a deeper appreciation for the contemplative aspects of this knowledge.

This book thoroughly investigates the dynamic analysis of robots, emphasizing the delicate balance between computational complexity and the depth of physical understanding needed to derive the governing dynamic equations. Dynamic analysis of parallel manipulators is inherently complex due to their closed-loop structure and kinematic constraints. Nevertheless, understanding their dynamics is critical, particularly because parallel manipulators excel in applications demanding precise positioning and robust dynamic performance under heavy loads.

Despite substantial research on the kinematics of parallel manipulators, relatively few studies elaborate on their dynamics and control. This book addresses this gap by introducing three distinct methods: Newton-Euler, Euler-Lagrange, and virtual work. The primary emphasis lies on the Euler-Lagrange method. Furthermore, this book elaborates on deriving the robot's dynamic equations in closed form, accompanied by an exploration of the associated properties.

It is interesting to note that the governing dynamic formulation of all parallel robots, regardless of their structure and degrees of freedom, can be expressed in a general form. This form includes a mass matrix that generalizes the mass of a rigid body into high-dimensional space, a gravity vector that encompasses the reacting forces on the robot links due to gravity, and a centrifugal and Coriolis matrix that includes all the interacting forces

in the joints due to the multi-degree of freedom motion of the links. Although these vectors and matrices differ in size and shape for different robots, they share certain properties in all cases. These properties enable the computer simulation and model-based controller design of robots to be done systematically.

Visualizing motion characteristics and assessing controller performance are crucial aspects of robot design. Computer simulations play a pivotal role in achieving these objectives. The general formulation of robot dynamics enables systematic simulations for various robots using standard software. In this book, both MATLAB® and the Python programming language are effectively utilized for simulation and analysis. Additionally, the book elaborates on generating desired trajectories and crafting optimal time paths for robots, considering the inherent speed and acceleration limitations of the actuators.

The standard model for a robot's dynamics is inherently nonlinear and involves multiple variables. Despite this complexity, industrial robots often employ linear controllers to accommodate the characteristics of their actuators. However, designing linear control systems that can achieve multiple objectives remains a significant challenge for control system designers. Feedback loops play a crucial role in achieving control system objectives, regardless of the type of controller used. Continuously observing the system status allows us to achieve the desired tracking even in the presence of disturbances and

uncertainties in the system model. This concept, often referred to as “The magic of feedback,” is extensively discussed in the book. Using feedback loops, the design procedure of PD and PID controller for robots are elaborated in details.

In parallel robots, the governing dynamic equations are inherently nonlinear and multivariable. As a result, a variety of nonlinear controllers have been developed to address these complexities. One fundamental approach to controller design is through full or partial feedback linearization, where the nonlinear dynamics are compensated using an inverse dynamics control scheme. While inverse dynamics control is widely used across applications, it has two primary limitations. First, it is highly knowledge-dependent: the effectiveness of this control approach relies on complete knowledge of the system's dynamic model, requiring precise calibration of its parameters. Second, model uncertainty must be taken into account. In practice, despite efforts to establish an accurate model, uncertainty is unavoidable and can affect control performance. This book explores these concepts in detail, offering a comprehensive analysis of the intricacies of inverse dynamics control for parallel robots.

In the field of robotics, researchers have proposed two general approaches to address the uncertain effects of model dynamics during transient and steady-state behavior. In robust control methods, the uncertainty of the model in the robot workspace is quantified and its upper bound is

determined. A corrective term is then added to the inverse dynamic control to compensate for the uncertain effects in the worst possible case. On the other hand, in adaptive control methods, the nominal model of the system is initially used in the inverse dynamic loop. By observing the tracking error dynamics, the model parameters are updated based on the linear regression form of the model. In this way, the parameters of the model are gradually adjusted to obtain the desired tracking. Both approaches play a crucial role in enhancing control performance while accounting for model uncertainties.

In various robot applications, particularly those involving human-robot interactions, precise force control is essential for safety and alignment with Asimov's three laws of robotics. This book provides detailed insights into control frameworks specifically designed to manage interaction forces between robots and their environment. It explores specific control structures like stiffness, direct force, hybrid, and impedance control. By elaborating these control strategies, the book equips readers with a comprehensive understanding of their application in various robotic scenarios. Whether you’re a researcher, engineer, or enthusiast, this exploration enhances both theoretical knowledge and practical skills for implementing effective force control in your own robotic projects.

This book offers readers a rich collection of illustrative examples and case studies, carefully curated and strategically placed within relevant sections. This

deliberate organization creates a cohesive narrative throughout the chapters, allowing readers to follow the theoretical foundations seamlessly. The aim is to equip readers not only with a solid understanding of the concepts but also with the practical skills needed to apply the methodologies to their own robotic projects. At the end of each chapter, readers are encouraged to extend the theoretical concepts to a range of additional robotic systems.

This practical approach empowers readers to develop proficiency in designing and analyzing parallel robots by actively solving relevant problems. To enhance practical implementation, the book takes an additional step by generously sharing the accompanying codes. These codes, developed in both MATLAB and Python, are readily accessible to the audience. They enrich the learning journey and encourage hands-on exploration. The open accessibility, available on the CRC Book site, serves as a valuable resource for readers seeking to focus deeper into the practical aspects of robotics through real-world examples and applicable code snippets.

## Exercises

1.1 Create a timeline outlining significant developments in robotics research over the last five years, highlighting

the pivotal role of parallel robotics in shaping the future of robotics research.

1.2 What are Isaac Asimov's three laws of robotics, and how have they been incorporated into robotics literature? To what extent have these laws influenced the development of new robots worldwide?

1.3 Share your perspective on the distinctions between serial and parallel robots, outlining the merits and drawbacks of each. Additionally, discuss the challenges associated with designing hybrid robots.

1.4 Explore recent applications of parallel robots beyond what has been discussed, and provide your findings. Include details about the nature of the application and the types of robotic systems employed in these instances.

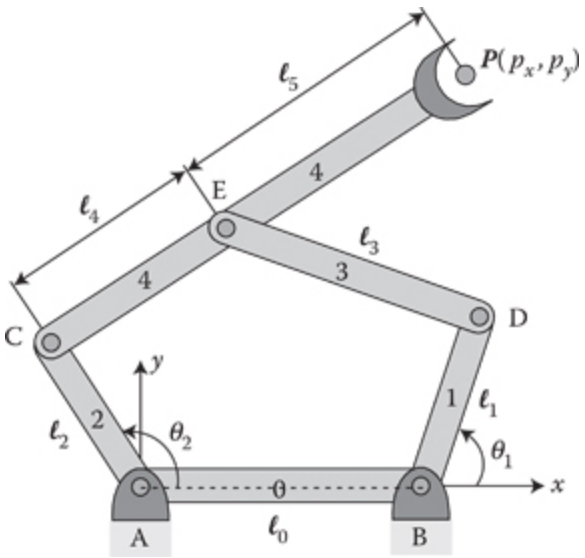
1.5 Investigate the following concepts in robotics and provide brief statements conveying your understanding of each item:

- Degrees of freedom.
- Kinematics: Forward verses inverse kinematics.
- Workspace: Reachable and dexterous workspaces.
- Path and trajectory.
- Differential kinematics: Jacobian matrix and its properties.

- Dynamics: Newton-Euler versus Euler-Lagrange Method.
- Dynamics: Closed-form formulation.
- Control: Motion versus force control.
- Control: Robust versus adaptive control.

## Problems

1.1 For the five-bar mechanism depicted in Figure 1.17, which is constructed from five revolute joints, use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom.




▶ Long Description for Figure 1.17

**FIGURE 1.17**

Five bar mechanism of Problem 1.1. ⏎

1.2 For the five-bar mechanism depicted in Figure 1.18, which is constructed from four revolute joints and one prismatic joint, use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom.

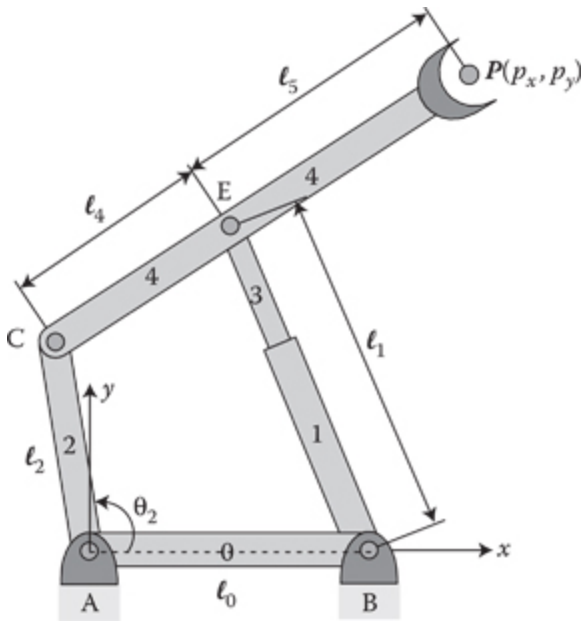



**FIGURE 1.18**

Five bar, 4P1R mechanism of Problem 1.2. ⏎

1.3 Figure 1.19 illustrates the schematics of a planar pantograph mechanism in which the ground–connected joints are prismatic, and the remaining joints are revolute. The structure of this mechanism is based on a parallelogram, in which links 3 and 5 and links 4, and 6 are parallel, respectively. For this mechanism, use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom.

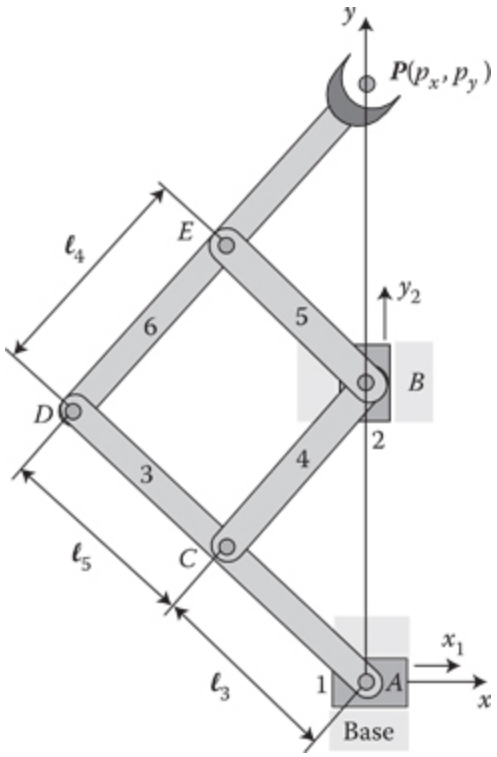



**FIGURE 1.19**

Pantograph mechanism of Problem 1.3. ⏎

1.4 A planar 3RRR parallel manipulator is depicted in Figure 1.20. By utilizing the Chebychev–Grübler–Kutzbach formulation, calculate the number of degrees of freedom of this parallel manipulator.

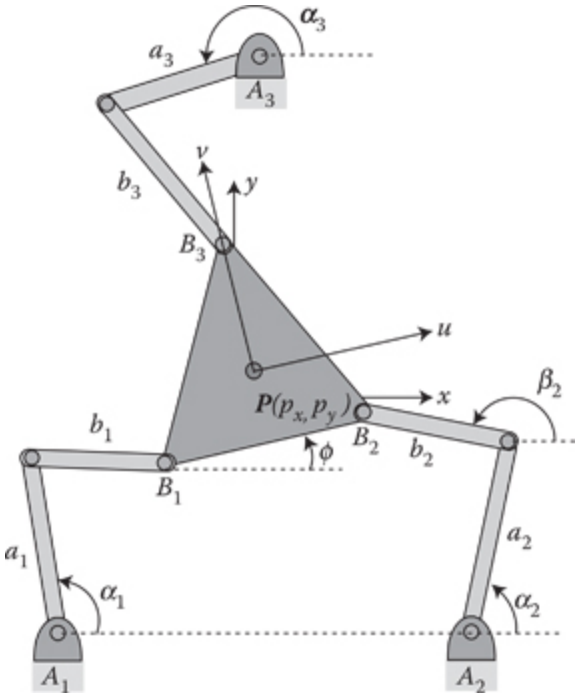



**FIGURE 1.20**

The planar 3RRR parallel manipulator of Problem 1.4. ⏎

1.5 A planar 4RRR parallel manipulator is shown in Figure 1.21. Use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom of this parallel manipulator.

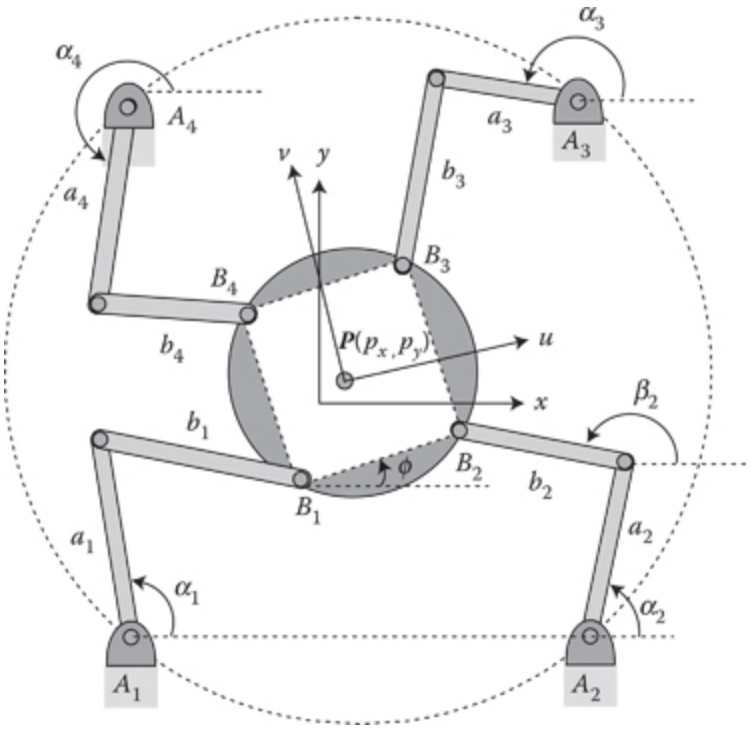



**FIGURE 1.21**

The planar 4RRR parallel manipulator of Problem 1.5. ⏎

1.6 Figure 1.22 illustrates the schematics of a spatial orientation manipulator consists of two tetrahedrons that are pivoted at point $O$ with a spherical joint. Use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom of this parallel manipulator.

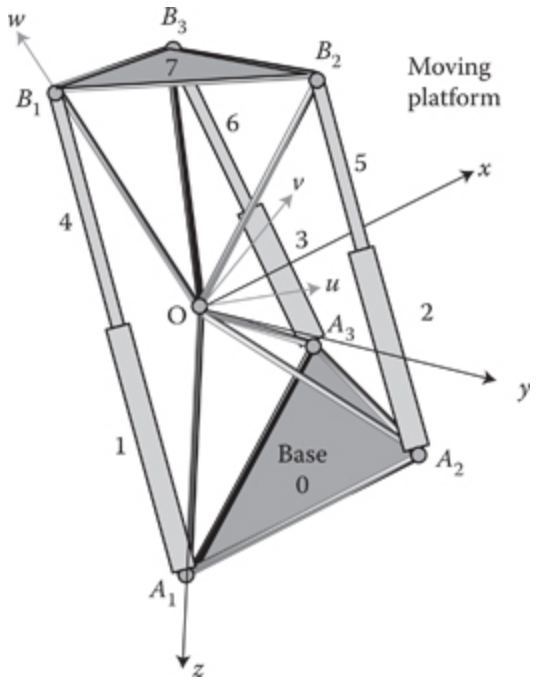



**FIGURE 1.22**

Two tetrahedral orientation manipulator of Problem 1.6. ⏎

1.7 The architecture of a 3UPU parallel manipulator is shown in Figure 1.23. This manipulator consists of a moving platform, a fixed base, and three supporting limbs of identical kinematic structure, [52]. Each limb connects the fixed base to the moving platform by a universal joint followed by an actuated prismatic joint and another universal joint. Use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom of this parallel manipulator.

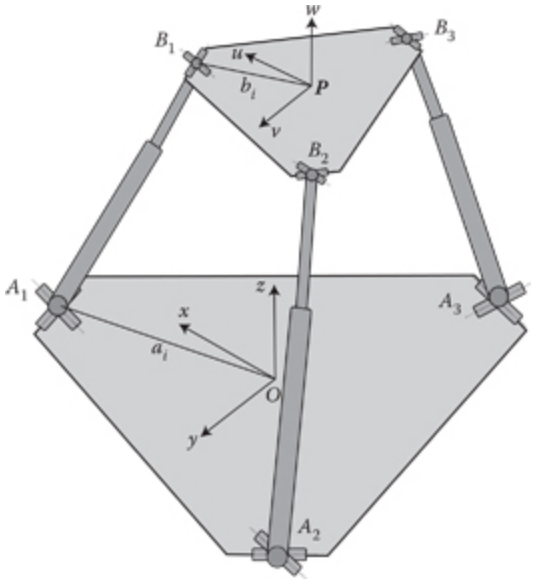



**FIGURE 1.23**

The 3UPU parallel manipulator of Problem 1.7, [50]. ⏎

1.8 Consider the 3UPS parallel manipulator depicted in Figure 1.24, composed of a moving platform (MP), a base platform (BP), and four limbs. Three limbs are connected to MP by spherical joints and coupled to BP by universal joints. The fourth or the central limb is connected to the centroid of MP by a universal joint and is fixed to BP. Each limb contains two links coupled by a prismatic joint. Use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom of this parallel manipulator.

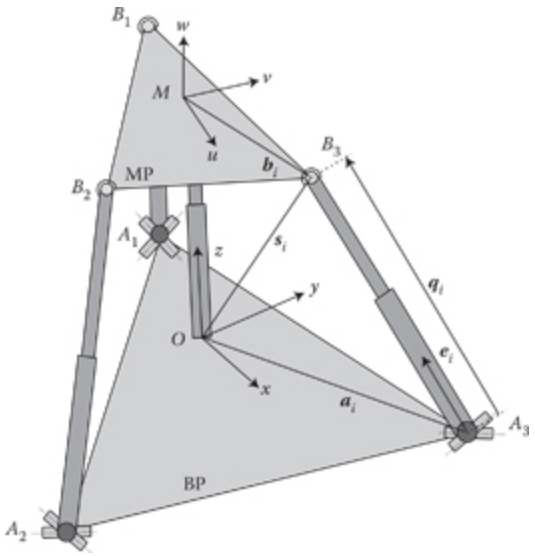



**FIGURE 1.24**

The four–legged 3UPS parallel manipulator of Problem 1.8, [51]. ⏎

1.9 Consider the 3RPS parallel manipulator depicted in Figure 1.25. The manipulator consists of the moving and base platforms that are equilateral triangles with $\ell$ and $k\ell$ as the lengths of the sides, respectively. Three extensible prismatic limbs connect the moving platform by spherical joints and the base by means of rotational ones; at each joint of the base, the revolute axis is parallel to the opposite edge. The 3RPS joint structure corresponds to the revolute joints between the limbs and the base, the actuated prismatic joints in the limbs and the spherical joints between the limbs and the moving platform. Use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom of this parallel manipulator.

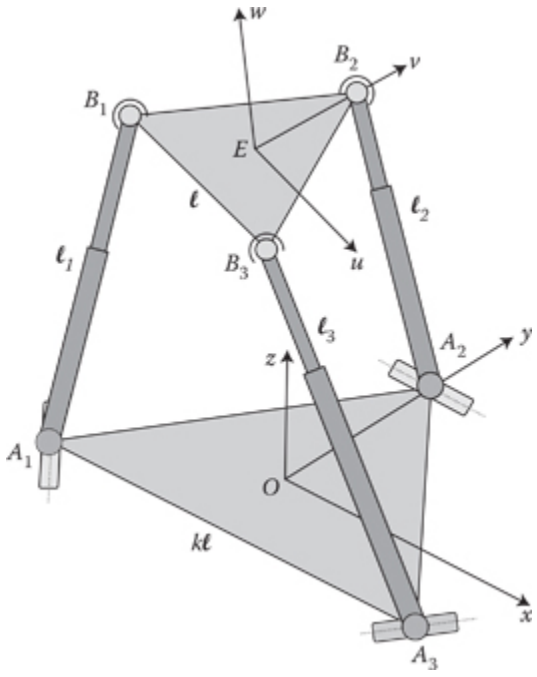
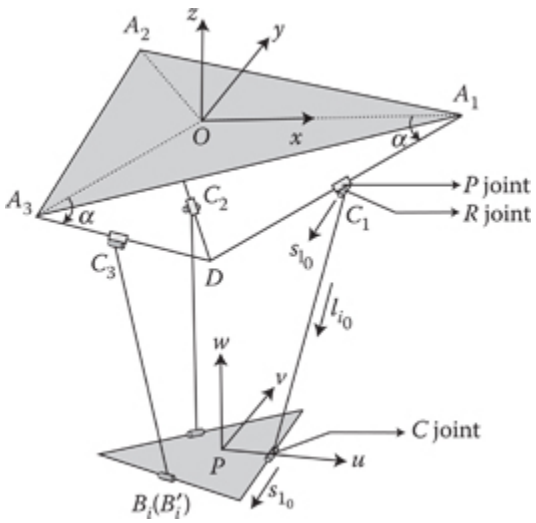



**FIGURE 1.25**

The 3RPS parallel manipulator of Problem 1.8, [53]. ⏎

1.10 The schematic diagram of a 3PRC translational parallel manipulator is shown in Figure 1.26. This manipulator consists of a moving platform, a fixed base, and three limbs with identical kinematic structure. Each limb connects the fixed base to the moving platform by an actuated *P* joint, an *R* joint, and a *C* joint in sequence. Thus, the moving platform is attached to the base by three identical PRC linkages. The mobility analysis of this manipulator shows that in order to avoid any change of orientation in the moving platform, it is sufficient for the three axes of joints within the same limb to satisfy certain geometric conditions [54]. Show that using Chebychev–Grübler–Kutzbach criteria, the degrees of freedom of this

manipulator in general form is zero, and due to the specific geometric design of this manipulator stated in [54], its moving platform possesses three degrees of freedom.




▶ Long Description for Figure 1.26

**FIGURE 1.26**

The schematics of the 3PRC parallel manipulator of Problem 1.10, [54]. ⏎

1.11 The Delta robot was invented in the early 1980's by Reymond Clavel [24], at EPFL, Switzerland. The purpose of this new type of robot was to manipulate light and small objects at a very high speed. As shown in Figure 1.27, this parallel robot consists of a fixed base, a moving platform and three arms. Each arm consists of two parts, the actuated lower arm and the upper arm. The lower arm is connected to the base by revolute joints. The key design feature in the upper arm is the use of parallelograms, which maintain the

orientation of the end effector. These parallelograms are connected to the lower arm and the moving platform by spherical joints. Use the Chebychev–Grübler–Kutzbach formulation to calculate the number of degrees of freedom of this parallel manipulator for this special parallelogram structure.

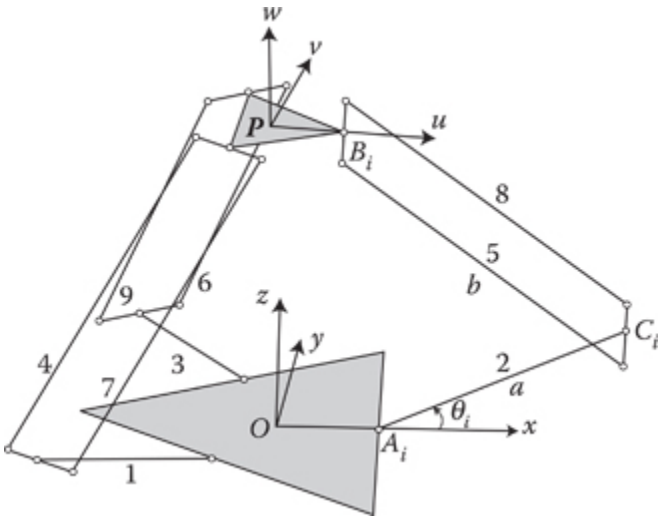
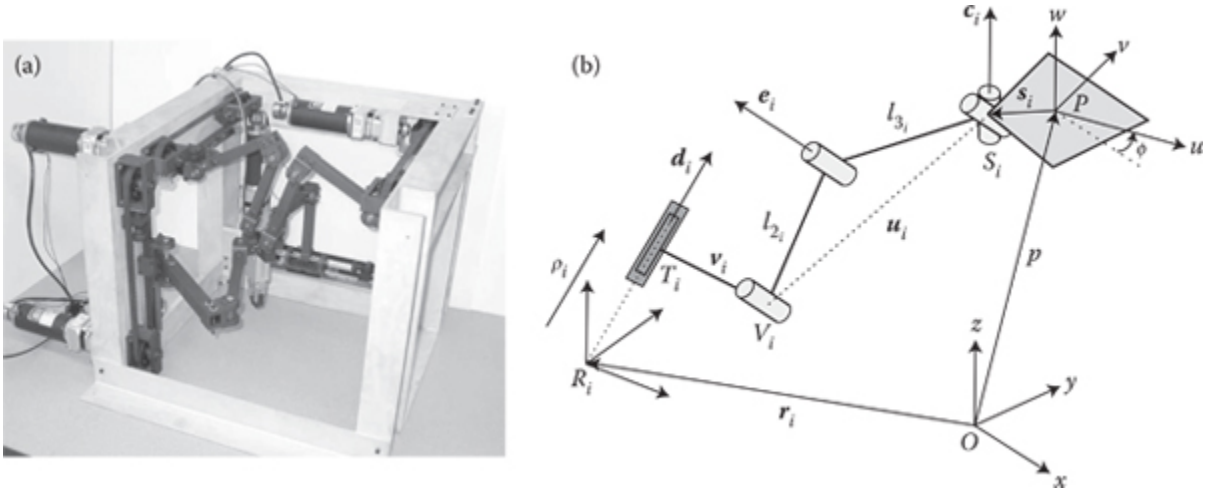



**FIGURE 1.27**

The schematics of the Delta robot of Problem 1.11. ⏎

1.12 A picture of a Quadrupteron robot, and the schematics of one of its limb kinematics is shown in Figure 1.28. This parallel robot consists of a moving platform connected to a fixed base by four limbs each having five joints namely, from base to moving platform, a $P$ joint and four $R$ joints which are labeled from 1 to 5 in sequence. For practical reasons, the axes of the last two $R$ joints within a limb are arranged to have at least one common point. The $P$ joints, mounted directly on the base, are the only actuated

ones. In order for the architecture to produce three translational and one rotational motion and to be drivable by the four actuated *P* joints, a number of geometric constraints must be satisfied [55]. Apply the Chebychev–Grübler–Kutzbach formulation to demonstrate that, given the specific geometric constraints, this parallel manipulator has four degrees of freedom (3T1R).




▶ Long Description for Figure 1.28

**FIGURE 1.28**

A picture of the Quadrupteron and the schematic of its limb's kinematic structure. (Adapted from P. L. Richard and C. M. Gosselin. *ASME Journal of Mechanical Design*, 129(6):611–616, 2007.) ⏎

# 2

# *Motion Representation*

DOI: 10.1201/9781003590408-2

This chapter thoroughly examines the complexities involved in describing motion, with particular emphasis on representing rigid body motion in three-dimensional space. The discussion commences with an introductory overview and subsequently explores the representation of rigid body motion in three dimensions. It elucidates the methodology for determining the position of a point on a rigid body, providing foundational insights into the description of motion. Additionally, the chapter elaborates on the orientation of rigid bodies, introducing concepts and methodologies such as rotation matrices and Euler angles. The discussion on rotation matrices underscores their role in expressing orientation and delineates their associated properties. Conversely, Euler angles present an alternative method, employing fewer parameters for orientation mapping.

Furthermore, the chapter places significant emphasis on defining the pose of a rigid body in three-dimensional

space, integrating both position and orientation. A comprehensive examination of homogeneous transformations is provided, presenting a mathematical framework that unifies translation and rotation. This includes the introduction of homogeneous coordinates for various types of vectors and the definition of the homogeneous transformation matrix, which is crucial for expressing transformations. Detailed coverage of the arithmetic operations involved in homogeneous transformations, such as consecutive transformations and the inverse of such transformations, is also included. Lastly, the chapter offers a variety of problems, enabling readers to practice different representations of rigid body motion and gain proficiency in transitioning between them.

## 2.1 Introduction

In the domain of robotic motion studies, the precise characterization of the position and orientation of various components within the robot's structure is of paramount importance. This necessity arises due to the frequent execution of motion maneuvers by robots within three-dimensional space, requiring a detailed description of their movements within this spatial dimension. Such representations are of substantial relevance across a broad spectrum of robotic applications, including mobile and aerial robotics, humanoid robots, and various engineering

disciplines such as aerospace and underwater applications, wherein entities navigate freely in three-dimensional space. Consequently, a multitude of methods have been developed and refined over time, originating from early considerations of human flight [56]. This section scrutinizes some of these methodologies, as applied in the analysis of robotic motion.

To accurately define the motion of a specific point on an object in space, it is imperative to establish a reference frame for measurement. Traditionally, a coordinate system affixed to the ground serves as the foundational basis for this reference frame. Nevertheless, when analyzing speed and acceleration, it is crucial for this reference frame to remain stationary. Although every point in space experiences some degree of motion, classical mechanics typically disregard the Earth's motion when examining the movements of objects situated upon it. Thus, the ground-based coordinate system is selected as the reference frame for motion measurement. Despite the existence of various coordinate systems for this purpose, the Cartesian coordinate system remains the most widely utilized, in preference to alternatives such as cylindrical and spherical coordinate systems.

## 2.2 Spatial Motion Representation

If the position of every point on a rigid body can be determined with respect to a reference coordinate system,

the complete spatial location, or “pose,” of the body is established. Here, the term ”pose” serves as a succinct representation of both the position and orientation of the rigid body. When the motion of the body is restricted to a plane, a set of three variables suffices to describe the body's pose: two variables to specify the location of any arbitrary point on the rigid body relative to the reference coordinate system, and an additional variable to describe the orientation of the body with respect to the reference frame.

To quantify these variables and achieve an accurate mathematical representation of the body's pose, it is sufficient to affix a movable Cartesian coordinate system to the body. Given the rigidity of the body, all points on it appear stationary to an observer within this coordinate system. Consequently, the position of the origin of this movable coordinate system relative to the reference coordinate system can be expressed using two translational variables, and the orientation of this coordinate system relative to the reference frame can be described using one rotational variable. Thus, three variables are required to depict the pose of a rigid body confined to a plane.

To represent the pose of a rigid body in three-dimensional space, one can similarly affix a mobile coordinate system to the moving body. The position of the origin of this mobile coordinate system relative to the reference coordinate system can be described using three translational variables. Additionally, the orientation of this

coordinate system with respect to the reference frame can be represented by three rotational variables. Therefore, a total of six variables are required to accurately represent the dynamic motion of a rigid body in three-dimensional space.

With this introduction, consider the spatial motion of a rigid body as depicted in Figure 2.1. Envision the Cartesian coordinate system $\{\mathscr{A}\}$ at the origin $O_A$, illustrated with three orthogonal axes $(x, y, z)$, as the reference coordinate system for representing the motion of the moving body. Furthermore, to describe the rotational motion of the rigid body in three-dimensional space, consider another Cartesian coordinate system $\{\mathscr{B}\}$ affixed to the moving body at point $O_B$. This coordinate system is represented with orthogonal axes $(u, v, w)$, indicating its mobility.

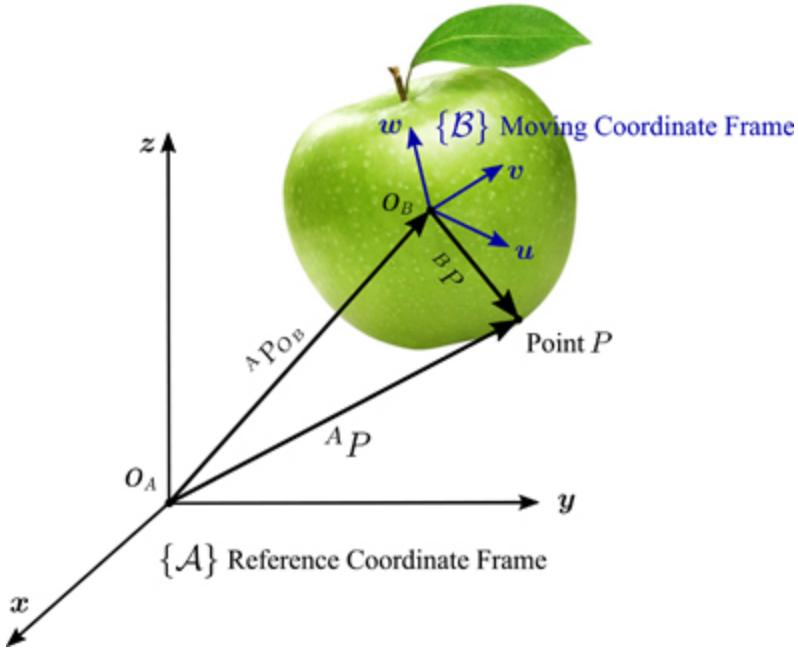



**FIGURE 2.1**

Motion representation of a rigid body in three dimensional Space. ⏎

In this manner, the pose of the rigid body can be mathematically expressed by describing the position and orientation of the mobile coordinate system $\{\mathscr{B}\}$ relative to the reference coordinate system $\{\mathscr{A}\}$. This description requires six variables: three for the translational motion of the mobile coordinate system relative to the reference system and three for the rotational motion between these two coordinate systems. The position can be conveniently represented using vector notation, while the orientation is more complex and can be described using various methods. By accurately describing the translational and rotational motion of the mobile coordinate system relative to the reference system, the absolute position of any arbitrary point $P$ on the rigid body relative to the reference coordinate system can be easily determined.

### 2.2.1 Position of a point

The position of a point $P$ relative to the coordinate system $\{\mathscr{A}\}$ can be represented by a vector containing three components along the three orthogonal axes $(x, y, z)$. If this vector is expressed relative to a fixed coordinate system, it denotes the absolute position of point $P$. Conversely, if it is specified with respect to a moving coordinate system, it indicates the relative position of this point. Given the significance of the chosen coordinate system in

representing the location of a point, the notation ${}^{A}\boldsymbol{P}$ is used to symbolize the position vector of point $P$ relative to the coordinate system $\{\mathscr{A}\}$. This is interpreted as the vector $P$ with respect to $A$. Consequently, the position of point $P$ relative to the coordinate system $A$ is represented as a vector with three components, as follows.

$$ {}^{A}\boldsymbol{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = [P_x,\, P_y,\, P_z]^T \tag{2.1} $$

In this representation, the elements $x$,$y$,$z$ of the vector $P$ determine the components of the vector in the Cartesian coordinates of $\{\mathscr{A}\}$, as shown in Figure 2.2. Similarly, the vector representing the position of $P$ with respect to another coordinate system, such as $\{\mathscr{B}\}$, can be seen in Figure 2.1, which is denoted by ${}^{B}\boldsymbol{P}$.

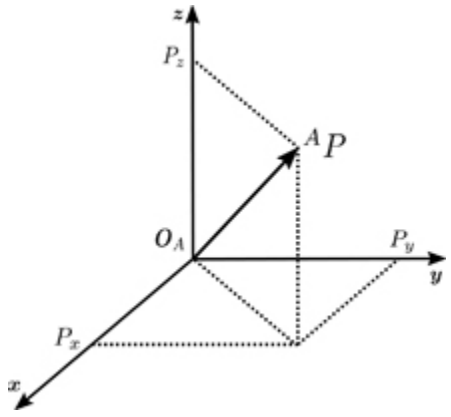



**FIGURE 2.2**

Displaying the vector ${}^{A}\boldsymbol{P}$ along with its individual components. ⏎

### 2.2.2 Orientation of a Rigid Body

The definition of orientation fundamentally differs from the definition of position. Each point on an object in space has its unique position vector, whereas the orientation of a rigid body applies uniformly to all its points. Thus, orientation serves as a representation for the entire body. In the preceding section, we discussed how a movable Cartesian coordinate system attached to the body can be used to express the positions of points on a rigid body. Similarly, in defining the orientation of the body, we rely on the amount of rotation of this coordinate system relative to the reference coordinate system. Unlike position, which can be straightforwardly defined using a vector with three components in space, there exist various methods for describing the orientation of a body, developed from ancient times to the present.

The most prevalent method for representing the orientation of a rigid body involves the use of a *rotation matrix*. In this approach, a $3 \times 3$ matrix is employed, where its columns are derived from the representation of unit vectors of the coordinate axes attached to the body relative to the reference coordinate frame. Although the rotation matrix is widely adopted for representing orientation in space, its description typically requires six parameters, adding complexity. This has prompted the exploration of

alternative representations, despite the rotation matrix's advantageous properties. These alternatives include representations through screw axes, various Euler angles, and quaternions, all of which will be explored further in this chapter and the following one.

### 2.2.3 Rotation Matrix

One of the most common methods for representing the orientation of a body is using a rotation matrix. To better grasp this representation, let's consider a scenario where a rigid body undergoes only rotational motion without any translational movement. Under this assumption, to depict the orientation of the body, envision a Cartesian coordinate system affixed to the body, depicted in Figure 2.3, rotating freely in space. In this figure, the body transitions from its initial state, illustrated as a dotted body with the coordinate system $\{\mathscr{A}\}$, to its final state depicted as a solid body with the moving coordinate system $\{\mathscr{B}\}$. Assuming the origins of both the reference and moving coordinate systems coincide, the unit vector directions $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ will not align with the unit vector directions $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}})$. Here, the symbol $\hat{(\cdot)}$ denotes unit vectors. Based on this assumption, we define the $3 \times 3$ rotation matrix as the operator that represents this rotational motion, defined as follows.

$$ {}^{A}\boldsymbol{R}_{B} := \left[ {}^{A}\hat{\boldsymbol{x}}_{B} \;\middle|\; {}^{A}\hat{\boldsymbol{y}}_{B} \;\middle|\; {}^{A}\hat{\boldsymbol{z}}_{B} \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.2} $$

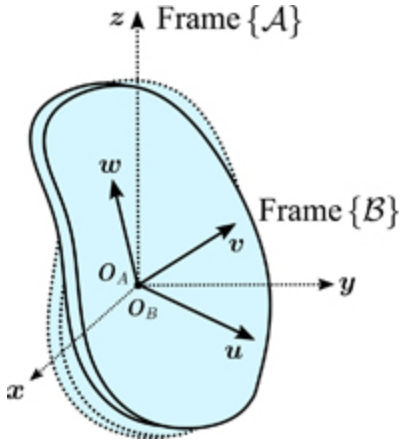



**FIGURE 2.3**
Representation of the pure rotation of a body along with the rotation-oriented coordinate axes within it. ⏎

In this definition, ${}^{A}\hat{\boldsymbol{x}}_{B}$, ${}^{A}\hat{\boldsymbol{y}}_{B}$, ${}^{A}\hat{\boldsymbol{z}}_{B}$ are unit vectors of the coordinate system $\{\mathscr{B}\}$ represented in the reference coordinate system $\{\mathscr{A}\}$. Furthermore, since these vectors belong to the Cartesian coordinate system $\{\mathscr{B}\}$, they are perpendicular to each other. This implies that the elements forming the rotation matrix, denoted by $r_{ij}$ in the right side of this equation, all have magnitudes less than or equal to one. Now, if we represent the unit vectors of the coordinate system $\{\mathscr{B}\}$ by their components, we have:

$$
\begin{aligned}
{}^{A}\hat{\boldsymbol{x}}_{B} = &\quad {}^{A}\hat{\boldsymbol{u}} = u_x\hat{i} + u_y\hat{j} + u_z\hat{k} \\
{}^{A}\hat{\boldsymbol{y}}_{B} = &\quad {}^{A}\hat{\boldsymbol{v}} = v_x\hat{i} + v_y\hat{j} + v_z\hat{k} \\
{}^{A}\hat{\boldsymbol{z}}_{B} = &\quad {}^{A}\hat{\boldsymbol{w}} = w_x\hat{i} + w_y\hat{j} + w_z\hat{k}
\end{aligned}
\tag{2.3}
$$

Thus, the rotation matrix can be expressed as follows:

$$
{}^{A}\boldsymbol{R}_{B} = \left[ {}^{A}\hat{\boldsymbol{u}} \;\middle|\; {}^{A}\hat{\boldsymbol{v}} \;\middle|\; {}^{A}\hat{\boldsymbol{w}} \right] = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}
\tag{2.4}
$$

The elements of the unit vectors comprising the rotation matrix can be derived by computing the dot product between the unit vectors of the coordinate system $\{\mathscr{B}\}$ and those of the coordinate system $\{\mathscr{A}\}$, as illustrated below:

$$
{}^{A}\boldsymbol{R}_{B} = \left[ {}^{A}\hat{\boldsymbol{x}}_{B} \;\middle|\; {}^{A}\hat{\boldsymbol{y}}_{B} \;\middle|\; {}^{A}\hat{\boldsymbol{z}}_{B} \right] = \begin{bmatrix} \hat{x}_B \cdot \hat{x}_A & \hat{y}_B \cdot \hat{x}_A & \hat{z}_B \cdot \hat{x}_A \\ \hat{x}_B \cdot \hat{y}_A & \hat{y}_B \cdot \hat{y}_A & \hat{z}_B \cdot \hat{y}_A \\ \hat{x}_B \cdot \hat{z}_A & \hat{y}_B \cdot \hat{z}_A & \hat{z}_B \cdot \hat{z}_A \end{bmatrix}
\tag{2.5}
$$

Please note that the superscripts of ${}^{A}(\,\cdot\,)$ have intentionally been omitted in representing this vector product. This omission stems from the fact that the result of the dot

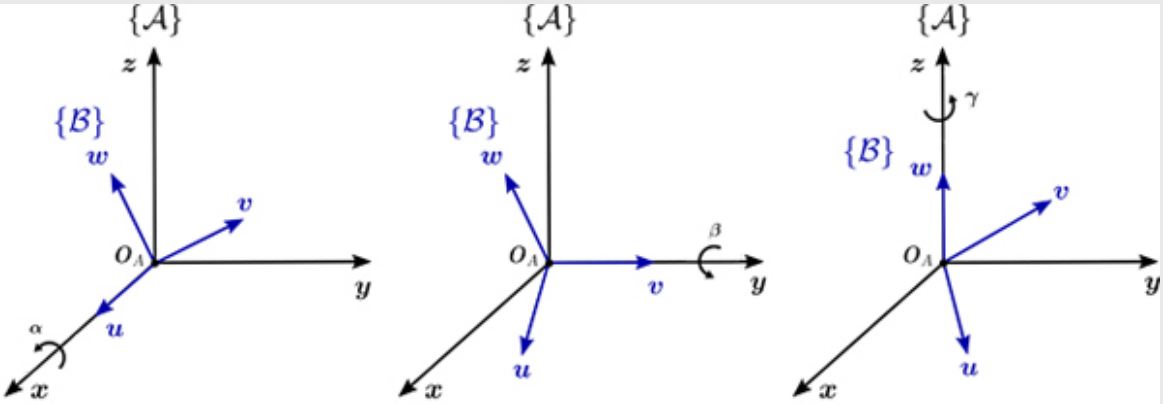
product between two numerical vectors yields a scalar value, which remains consistent regardless of the coordinate system in which these vectors are defined. Geometrically, the dot product of two unit vectors corresponds to the cosine of the angle between them, which underscores why the rotation matrix is also known as the *Cosine direction representation* [57].

**Example 2.1 (Pure rotation about principal axes)** ⏎ *Consider a moving coordinate system* $\{\mathscr{B}\}$ *attached to a rigid body as shown in Figure 2.4, rotating only around one of the principal axes of the reference coordinate system* $\{\mathscr{A}\}$ *with the following angles. In each of the following cases, determine the rotation matrix.*

***A)*** *Rotation around the x-axis by an angle* $\alpha$.

***B)*** *Rotation around the y-axis by an angle* $\beta$.

***C)*** *Rotation around the z-axis by an angle* $\gamma$.




(a) $\alpha$ rotation around the $x$-axis (b) $\beta$ rotation around the $y$-axis (c) $\gamma$ rotation around the $z$-axis

**FIGURE 2.4**

Pure rotation about principal axes. ⏎

**Solution:** The rotation matrix in these three cases can be obtained by specifying the components of the unit vectors of the coordinate axes $\{\mathscr{B}\}$ with respect to the reference coordinate system $\{\mathscr{A}\}$, as:

$$\textbf{A)} \quad {}^{A}\boldsymbol{R}_{B} = \boldsymbol{R}_{\boldsymbol{x}}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$\textbf{B)} \quad {}^{A}\boldsymbol{R}_{B} = \boldsymbol{R}_{\boldsymbol{y}}(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$\textbf{C)} \quad {}^{A}\boldsymbol{R}_{B} = \boldsymbol{R}_{\boldsymbol{z}}(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

As seen in these expressions, the columns of the rotation matrices are formed from the representation of unit vectors along the coordinate axes of $\{\mathscr{B}\}$ in the reference coordinate system $\{\mathscr{A}\}$. These basis rotation matrices will be used in subsequent sections to calculate rotation matrices in more complex rotations.

## Rotation Matrix Properties

**Property I)** Orthonormality

The rotation matrix is an orthonormal matrix; in other words, its columns are formed by vectors with unit magnitudes:

$$\boldsymbol{u} \cdot \boldsymbol{u} = \boldsymbol{v} \cdot \boldsymbol{v} = \boldsymbol{w} \cdot \boldsymbol{w} = 1 \tag{2.6}$$

which are mutually perpendicular:

$$\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{w} = \boldsymbol{w} \cdot \boldsymbol{u} = 0 \tag{2.7}$$

Furthermore, considering the orthogonality property of these vectors, the following relationships hold:

$$\boldsymbol{u} \times \boldsymbol{v} = \boldsymbol{w}, \;\; \boldsymbol{v} \times \boldsymbol{w} = \boldsymbol{u}, \;\; \boldsymbol{w} \times \boldsymbol{u} = \boldsymbol{v} \tag{2.8}$$

**Property II)** Transpose

Considering the definition of the rotation matrix and revisiting Equation 2.5, where the components are derived from the inner product of orthonormal axes within each coordinate system, one can infer that the rows of the rotation matrix can similarly be derived from the components of orthonormal vectors of the

coordinate axes $\{\mathscr{A}\}$ with respect to the reference coordinate system $\{\mathscr{B}\}$.

$$ {}^{A}\boldsymbol{R}_{B} = \left[ {}^{A}\hat{\boldsymbol{x}}_{B} \mid {}^{A}\hat{\boldsymbol{y}}_{B} \mid {}^{A}\hat{\boldsymbol{z}}_{B} \right] = \begin{bmatrix} {}^{B}\hat{\boldsymbol{x}}_{A}^{T} \\ {}^{B}\hat{\boldsymbol{y}}_{A}^{T} \\ {}^{B}\hat{\boldsymbol{z}}_{A}^{T} \end{bmatrix} \tag{2.9} $$

This means that the rotation matrix of the reference coordinate system $\{\mathscr{A}\}$ relative to the coordinate system $\{\mathscr{B}\}$, denoted by ${}^{B}\boldsymbol{R}_{A}$, is the transpose of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$.

$$ {}^{B}\boldsymbol{R}_{A} = {}^{A}\boldsymbol{R}_{B}{}^{T} \tag{2.10} $$

**Property III)** Inverse

From property II, it can be inferred that the inverse of a rotation matrix is equal to its transpose. This property is proven using the rules of linear algebra for orthonormal matrices (for more details on linear algebra, refer to Appendix A). However, considering the physical concept of the rotation of one coordinate system relative to another embedded in the rotation matrix, this property can also be logically deduced from a physical perspective. To mathematically prove this claim, multiply the rotation matrix by its transpose:

$$ {}^{A}\boldsymbol{R}_{B}{}^{T}\ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} {}^{A}\hat{\boldsymbol{x}}_{B}^{T} \\ {}^{A}\hat{\boldsymbol{y}}_{B}^{T} \\ {}^{A}\hat{\boldsymbol{z}}_{B}^{T} \end{bmatrix} \left[ {}^{A}\hat{\boldsymbol{x}}_{B} \,\middle|\, {}^{A}\hat{\boldsymbol{y}}_{B} \,\middle|\, {}^{A}\hat{\boldsymbol{z}}_{B} \right] = \boldsymbol{I}_{3\times 3} $$

(2.11)

As you can see, considering property I, this product results in the identity matrix. Therefore,

$$ {}^{B}\boldsymbol{R}_{A} = {}^{A}\boldsymbol{R}_{B}{}^{-1} = {}^{A}\boldsymbol{R}_{B}{}^{T} $$

(2.12)

**Property IV)** Pure Rotation Mapping

As previously discussed, the rotation matrix inherently encapsulates the concept of rotation about a specific coordinate system. This principle can be utilized as a matrix mapping to transform a vector from one coordinate system to another that is related by pure rotation. To express this transformation mathematically, consider the position vector of a point on the body with respect to the coordinate system attached to it, denoted as ${}^{B}\boldsymbol{P}$ in $\{\mathscr{B}\}$. The objective is to determine the position vector with respect to the reference coordinate system $\{\mathscr{A}\}$, represented by ${}^{A}\boldsymbol{P}$. To ascertain the components of this vector, one can employ its dot product with the unit vectors of the reference coordinate system, as follows.

$$
\begin{aligned}
{}^{A}P_{x} &= {}^{A}\hat{\boldsymbol{x}}_{A} \cdot {}^{A}\boldsymbol{P} \\
{}^{A}P_{y} &= {}^{A}\hat{\boldsymbol{y}}_{A} \cdot {}^{A}\boldsymbol{P} \\
{}^{A}P_{z} &= {}^{A}\hat{\boldsymbol{z}}_{A} \cdot {}^{A}\boldsymbol{P}
\end{aligned}
$$

(2.13)

Given that these components are scalar values, the dot product can be obtained in any other coordinate frame. Thus, by representing these vectors in the $\{\mathscr{B}\}$ frame and calculating their dot product, one can write:

$$
\begin{aligned}
{}^{A}P_{x} &= {}^{B}\hat{\boldsymbol{x}}_{A} \cdot {}^{B}\boldsymbol{P} = {}^{B}\hat{\boldsymbol{x}}_{A}^{T} \; {}^{B}\boldsymbol{P} \\
{}^{A}P_{y} &= {}^{B}\hat{\boldsymbol{y}}_{A} \cdot {}^{B}\boldsymbol{P} = {}^{B}\hat{\boldsymbol{y}}_{A}^{T} \; {}^{B}\boldsymbol{P} \\
{}^{A}P_{z} &= {}^{B}\hat{\boldsymbol{z}}_{A} \cdot {}^{B}\boldsymbol{P} = {}^{B}\hat{\boldsymbol{z}}_{A}^{T} \; {}^{B}\boldsymbol{P}
\end{aligned}
$$

(2.14)

Rewrite Equation 2.14 in matrix form.

$$
{}^{A}\boldsymbol{P} = \begin{bmatrix} {}^{B}\hat{\boldsymbol{x}}_{A}^{T} \\ {}^{B}\hat{\boldsymbol{y}}_{A}^{T} \\ {}^{B}\hat{\boldsymbol{z}}_{A}^{T} \end{bmatrix} {}^{B}\boldsymbol{P}
$$

(2.15)

In view of Equation 2.9, this relationship can be succinctly expressed in matrix form using the rotation matrix:

$$ {}^{A}\boldsymbol{P} = {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} $$

(2.16)

Thus, mathematically, the rotation matrix can be construed as a mapping that facilitates the transformation of a vector from one coordinate system to another. By employing the appropriate notation for vector representations relative to different coordinate frames and for the rotation matrix, this mapping can be interpreted straightforwardly. Specifically, the left superscript $B$ of the position vector ${}^{B}\boldsymbol{P}$ is canceled out with the right subscript of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$, yielding the position vector of point $P$ in the target coordinate system $\{\mathscr{A}\}$, denoted as ${}^{A}\boldsymbol{P}$.

**Example 2.2 (Pure rotational mapping)** ⏎ *Assume that the rotation of the coordinate system attached to a rigid body $\{\mathscr{B}\}$ with respect to the reference coordinate system $\{\mathscr{A}\}$ is given by the following rotation matrix.*

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.933 & -0.067 & -0.354 \\ -0.067 & 0.933 & -0.354 \\ 0.354 & 0.354 & 0.866 \end{bmatrix} $$

*Furthermore, assume that the position vector of point P on this body with respect to the frame $\{\mathscr{B}\}$ is given as*

${}^{B}\boldsymbol{P} = [2 \quad -1 \quad 0]^{T}$. *In this case, determine the representation of this vector with respect to the reference coordinate system* $\{\mathscr{A}\}$, *denoted by* ${}^{A}\boldsymbol{P}$.

**Solution:** By using the mapping relation 2.16, the vector ${}^{A}\boldsymbol{P}$ in the $\{\mathscr{A}\}$ coordinate system can be obtained as follows:

$$
\begin{aligned}
{}^{A}\boldsymbol{P} &= {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} \\
&= \begin{bmatrix} 0.933 & -0.067 & -0.354 \\ -0.067 & 0.933 & -0.354 \\ 0.354 & 0.354 & 0.866 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.93 \\ -1.07 \\ 0.35 \end{bmatrix}
\end{aligned}
$$

If we pay attention to the obtained result, it can be observed that the magnitude of the vector $\boldsymbol{P}$ in both coordinate systems, which is determined by the Euclidean norm $\| \boldsymbol{P} \|_2$, is equal.

$$
\| \boldsymbol{P} \|_2 = \sqrt{(2^2 + (-1)^2)} = \sqrt{(1.93^2 + (-1.07)^2 + 0.35^2)} =
$$

This occurs because, in a pure rotation mapping the magnitude or length of vectors does not change, and only their components change with respect to the coordinate system under consideration.

**Property V)** Determinant

Determinant of a rotation matrix is equal to one.

$$\det\left({}^{A}\boldsymbol{R}_{B}\right)=1$$

(2.17)

This property can also be proven both mathematically and interpreted physically. To mathematically prove this property, by expanding the determinant of the rotation matrix in terms of its components, we can write:

$$\begin{aligned}\det\left({}^{A}\boldsymbol{R}_{B}\right) &= w_x(u_y v_z - v_y u_z) - w_y(u_x v_z - v_x u_z) + w_x(u\\ &= \boldsymbol{w}\cdot(\boldsymbol{u}\times\boldsymbol{v}) = \boldsymbol{w}\cdot\boldsymbol{w} = 1\end{aligned}$$

The physical interpretation of this relationship is consistent with the previous property and, in a sense, expresses the constancy of vector magnitudes in this transformation.

**Property VI)** Eigenvalues

The eigenvalues of the rotation matrix are equal to 1, $e^{i\theta}$, and $e^{-i\theta}$, where $\theta$ is calculated from the following relationship.

$$\theta=\cos^{-1}\frac{\operatorname{tr}\left({}^{A}\boldsymbol{R}_{B}\right)-1}{2}$$

(2.18)

In this relation, $\cos^{-1}$ represents the inverse cosine function, and $\text{tr}\left({}^{A}\boldsymbol{R}_{B}\right)$ denotes the trace of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$.

To prove this property, calculating the eigenvalues of the rotation matrix $(\lambda)$ using the components provided in Equation 2.4, one can write:

$$\det\left({}^{A}\boldsymbol{R}_{B}-\lambda\boldsymbol{I}\right)=\begin{vmatrix}u_x-\lambda & v_x & w_x\\ u_y & v_y-\lambda & w_y\\ u_z & v_z & w_z-\lambda\end{vmatrix}=0$$

(2.19)

By taking the determinant with respect to the first column and using the definition of the trace of a matrix, along with Equations 2.8 and 2.17, simplify this determinant as follows.

$$
\begin{aligned}
\det\left({}^{A}\boldsymbol{R}_{B}-\lambda\boldsymbol{I}\right) &= (u_x-\lambda)\left[(v_y-\lambda)(w_z-\lambda)-v_z w_y\right] \\
&\quad -u_y\left[v_x(w_z-\lambda)-v_z w_x\right]+u_z\left[v_x w_y-\right. \\
&= -\lambda^3+(u_x+v_y+w_z)\lambda^2 \\
&\quad -\left[(v_y w_z-v_z w_y)+(u_x w_z-u_z w_x)+(u_x\right. \\
&\quad +\det\left({}^{A}\boldsymbol{R}_{B}\right) \\
&= -\lambda^3+\operatorname{tr}\left({}^{A}\boldsymbol{R}_{B}\right)\lambda^2 \\
&\quad -\left[(\boldsymbol{v}\times\boldsymbol{w})_x+(\boldsymbol{w}\times\boldsymbol{u})_y+(\boldsymbol{u}\times\boldsymbol{v})_z\right]\lambda+ \\
&= -\lambda^3+\operatorname{tr}\left({}^{A}\boldsymbol{R}_{B}\right)\lambda^2-(u_x+v_y+w_z)\lambda
\end{aligned}
$$

As a result, the characteristic equation of the rotation matrix simplifies to

$$
\lambda^3-\operatorname{tr}\left({}^{A}\boldsymbol{R}_{B}\right)\lambda^2+\operatorname{tr}\left({}^{A}\boldsymbol{R}_{B}\right)\lambda-1=0 \tag{2.20}
$$

or,

$$
(\lambda-1)\left(\lambda^2+\left(1-\operatorname{tr}\left({}^{A}\boldsymbol{R}_{B}\right)\right)\lambda+1\right)=0 \tag{2.21}
$$

With the help of this equation, the eigenvalues of the rotation matrix can be expressed as:

$$\lambda = 1, e^{i\theta}, \text{ and } e^{-i\theta}$$

(2.22)

where $i = \sqrt{-1}$ and $e^{\pm i\theta} \triangleq \cos\theta \pm i \sin\theta$. Furthermore, by using Equation 2.21, the angle $\theta$ is calculated from the following relation, and the proof is completed.

$$\cos\theta = \frac{\text{tr}\left({}^{A}\boldsymbol{R}_{B}\right) - 1}{2}$$

**Example 2.3 (Rotation matrix eigenvalues)** ⏎ *Find the eigenvalues and eigenvectors of the rotation matrix provided in Example 2.2 and verify Property IV.*

$${}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.933 & -0.067 & -0.354 \\ -0.067 & 0.933 & -0.354 \\ 0.354 & 0.354 & 0.866 \end{bmatrix}$$

**Solution:** Eigenvalues and eigenvectors of the rotation matrix are determined as follows:

$$\lambda_1 = 1 \;\; ; \;\; v_1 = \begin{bmatrix} -0.707 \\ +0.707 \\ 0 \end{bmatrix}$$

$$\lambda_{2,3} = 0.866 \pm 0.5i = e^{\pm \pi/6} \;\; ; \;\; v_{2,3} = \begin{bmatrix} \pm 0.5i \\ \pm 0.5i \\ 0.707 \end{bmatrix}$$

It can easily be shown that the angle $\theta$, using Equation 2.18, is exactly $\theta = \pi/6$.

The angle $\theta$ is a fundamental characteristic of the rotation matrix, signifying the rotation angle in this context. Additionally, the eigenvector corresponding to the real eigenvalue 1 within the rotation matrix delineates the axis around which the rotation occurs. This representation of rotation, capturing its invariant physical properties, will be further explored in the subsequent chapter, particularly in the context of describing the orientation and pose of a rigid body using “screw theory.”

### 2.2.4 Euler Angles

As previously noted, the representation of a rigid body's orientation in space through a rotation matrix encompasses nine interdependent parameters. Despite this, given the inherent simplicity and intuitive understanding of rotational motion within three-dimensional space, it is more

appropriate to employ only three independent parameters for such representations. Extensive research and numerous methodologies documented in the existing literature strive to characterize orientation in space using just these three independent parameters. These methods typically rely on the specific type of orientation measurement employed by rotational instrumentation systems. Among these various representations, the use of Euler angles is the most prevalent, adhering to either fixed-axes or moving-axes conventions. This approach to representing orientation through Euler angles, along with its nuances and applications, will be examined in greater detail in the following sections.

Euler angles comprise a set of angles that describe the orientation of a rigid body in three-dimensional space. They represent a sequence of three fundamental rotations, each performed around one of the coordinate system axes. The specific convention for Euler angles is determined by the order of these rotations and whether the reference frame for the rotation is fixed or moving. Consequently, two distinct sets of Euler angles are identified in the literature. In the first set, the rotations occur around the axes of a stationary coordinate system. In contrast, in the second set, the rotations are performed around the axes of a moving coordinate system. Regardless of the convention, any desired orientation in three-dimensional space can be achieved by combining these fundamental rotations around the principal axes, as illustrated in Example 2.1.

Euler angles offer an intuitive method for visualizing and comprehending rotations, yet they are not without challenges, such as gimbal lock, which can constrain their applicability in certain scenarios. Gimbal lock occurs when one rotational axis becomes aligned with another, resulting in the loss of one degree of freedom and potential difficulties in representing specific orientations. Despite these limitations, Euler angles remain widely utilized in various fields, including aerospace, robotics, and computer graphics. In the ensuing sections, we will introduce a fixed-axis Euler angles representation, specifically with respect to the $\boldsymbol{x} - \boldsymbol{y} - \boldsymbol{z}$ axes. Additionally, we will explore three alternative representations of Euler angles concerning different combinations of axes in the moving frame. Furthermore, we will address the characteristics and implications of gimbal lock for each representation.

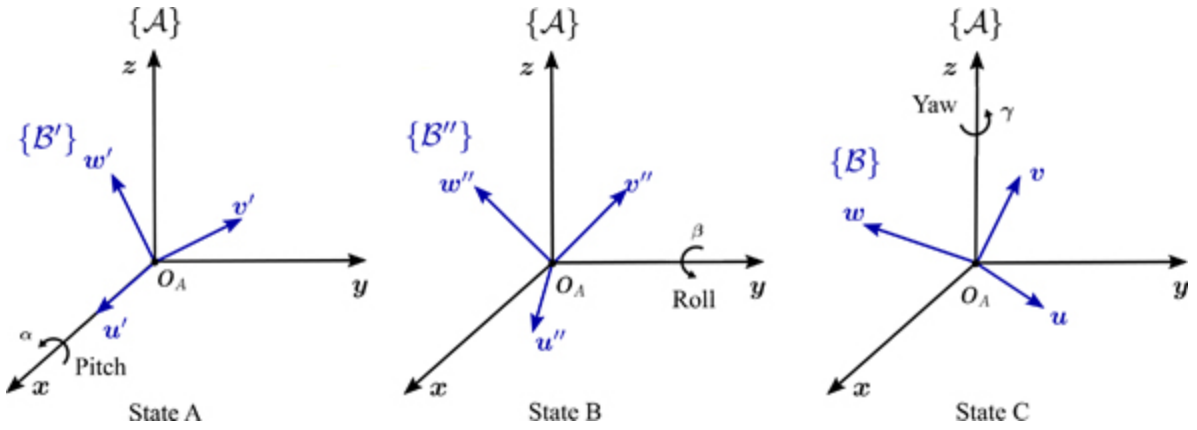
## Fixed-Axes Euler Angles

In this specific orientation representation of a rigid body in space, rotations are executed around the axes of the fixed coordinate system. As illustrated in Figure 2.5, among the twelve possible combinations [3, Appendix B], rotations about the lateral, longitudinal, and vertical axes of the fixed coordinate system are utilized to represent the orientation of the rigid body. As depicted in Figure 2.5, the rotation around the lateral axis is termed *pitch*, the rotation around the longitudinal axis is called *roll*, and the rotation around the vertical axis is referred to as *yaw*. Consequently, this

spatial orientation representation is commonly known as the “pitch–roll–yaw Euler angles.”

**FIGURE 2.5**
Fixed-axis pitch–roll–yaw representation in the motion of a vehicle. ⏎

To precisely define the fixed-axis rotation angles in this case and establish their relationship with the rotation matrix, consider that the orientation of the rigid body in space involves three consecutive rotations around the fixed axes $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ of the fixed coordinate system $\{\mathscr{A}\}$, as depicted in Figure 2.6. The sequence of rotations is outlined in detail as follows.




▶ Long Description for Figure 2.6

**FIGURE 2.6**

Sequential rotation about the fixed coordinate axes by the pitch angle $\alpha$ around the $\boldsymbol{x}$ axis, then by the roll angle $\beta$ around the $\boldsymbol{y}$ axis, and finally by the yaw angle $\gamma$ around the $\boldsymbol{z}$ axis. ⏎

First, consider the moving coordinate system aligned with the fixed coordinate system $\{\mathscr{A}\}$. Then, sequentially rotate the moving coordinate system about the $\boldsymbol{x}$ axis by the angle $\alpha$ to reach the intermediate coordinate system $\{\mathscr{B}'\}$. Next, rotate this intermediate coordinate system about the $\boldsymbol{y}$ axis of the reference coordinate system $\{\mathscr{A}\}$ by the angle $\beta$ to reach the second intermediate coordinate system $\{\mathscr{B}''\}$. Finally, rotate this second intermediate coordinate system about the $\boldsymbol{z}$ axis of the reference coordinate system $\{\mathscr{A}\}$ by the angle $\gamma$ to reach the final coordinate system $\{\mathscr{B}\}$.

In this representation, an effort has been made to break down the final orientation of the rigid body, described with the coordinate system $\{\mathscr{B}\}$ attached to it, relative to the reference coordinate system $\{\mathscr{A}\}$ using three consecutive rotations as previously explained. From a mathematical perspective, understanding the composite rotation matrix as the outcome of combining these three sequential rotational transformations makes this concept easily understandable. In this scenario, the rotational transformations are sequentially applied first around the $\boldsymbol{x}$ axis, then the $\boldsymbol{y}$ axis, and finally the $\boldsymbol{z}$ axis. This approach simplifies the computation of the rotation matrix formed by the composition of these three transformations through the successive pre-multiplication of fundamental rotation matrices about the $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ axes.

$$
\begin{aligned}
\boldsymbol{R}_{pry}(\alpha, \beta, \gamma) &= \boldsymbol{R}_z(\gamma)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha) \\
&= \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{bmatrix}
\end{aligned}
\tag{2.23}
$$

In this expression, the notation $c\alpha$ denotes $\cos\alpha$, and $s\alpha$ denotes $\sin\alpha$, and so on. It is important to note that when rotations are performed about fixed axes, rotation matrices are pre-multiplied together. Because matrix multiplication

is non-commutative, the sequence of rotation angles becomes significant. Mathematically, pre-multiplication involves multiplying rotation matrices from the left. Thus, the initial rotation corresponding to the primary rotation matrix $\boldsymbol{R}_x(\alpha)$ is positioned on the right, while the final rotation corresponding to the primary rotation matrix $\boldsymbol{R}_z(\gamma)$ is positioned on the left. Following this order of matrix multiplication, as described by Equation 2.23, the resulting rotation matrix from the pitch–roll–yaw Euler angles, denoted as $\boldsymbol{R}_{pry}$, is obtained as follows.

$$\boldsymbol{R}_{pry}(\alpha,\beta,\gamma) = \begin{bmatrix} c\beta c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ c\beta s\gamma & s\alpha s\beta s\gamma + c\alpha c\gamma & c\alpha s\beta s\gamma - s\alpha c\gamma \\ -s\beta & s\alpha c\beta & c\alpha c\beta \end{bmatrix} \tag{2.24}$$

The inverse problem in this representation becomes crucial for analyzing the rotational motion of a rigid body. Given the rotation matrix that describes the orientation of the body in space, the computation of pitch–roll–yaw Euler angles relies on solving the inverse problem. This inverse solution involves solving three trigonometric equations with three unknowns and incorporates six implicit algebraic constraints derived from the inherent properties of the rotation matrix. Consequently, the inverse problem may yield multiple solutions.

Here, we present a straightforward solution approach where only elements of reduced complexity from the

rotation matrix are utilized instead of all nine elements. Consider the first column of rotation matrix 2.24. By summing the squares of elements $r_{11}$ and $r_{21}$, $c^2\beta$ can be derived. This approach yields two valid solutions for $\beta$, obtained from $\pm\sqrt{r_{11}^2 + r_{21}^2}$ and $r_{31}$. Subsequently, $\gamma$ is determined from elements $r_{11}$ and $r_{21}$, and $\alpha$ from elements $r_{32}$ and $r_{33}$ of the rotation matrix, under the condition that $c\beta \neq 0$. Assuming this condition holds, the solution can be expressed as follows.

$$\begin{aligned}
\beta &= \operatorname{atan2}(-r_{31}, \pm\sqrt{r_{11}^2 + r_{21}^2}) \\
\gamma &= \operatorname{atan2}(r_{21}/c\beta, r_{11}/c\beta) \\
\alpha &= \operatorname{atan2}(r_{32}/c\beta, r_{33}/c\beta)
\end{aligned} \tag{2.25}$$

These equations utilize the four-quadrant inverse tangent function, denoted as $\operatorname{atan2}(y, x)$, instead of the standard inverse tangent. Similar to the regular inverse tangent function $\operatorname{atan}(y/x)$, $\operatorname{atan2}(y, x)$ employs this relationship. However, a crucial distinction lies in how the signs of both arguments, $x$ and $y$, influence the sign of the resulting angle. For example, whereas $\operatorname{atan}(-1/-1) = \pi/4$, $\operatorname{atan2}(-1, -1)$ yields $-3\pi/4$. This function facilitates the determination of rotation angles across all quadrants of the trigonometric circle, within the interval $[-\pi, \pi]$. Notably, *atan2* effectively manages scenarios where $x$ equals zero,

avoiding division by zero issues encountered with the *atan* function, with the result contingent on the sign of $y$. Consequently, *atan2* offers numerous advantages over *atan* for performing calculations.

When $\beta = \pm\pi/2$, the rotated $\boldsymbol{x}$ axis will be aligned with the $\boldsymbol{z}$ axis. Consequently, the inverse formulation degenerates, and the rotation matrix will be simplified to:

$$\boldsymbol{R}_{pry}(\alpha, \gamma) = \begin{bmatrix} 0 & s(\alpha - \gamma) & c(\alpha - \gamma) \\ 0 & c(\alpha - \gamma) & -s(\alpha - \gamma) \\ -1 & 0 & 0 \end{bmatrix} \tag{2.26}$$

Keep in mind that this case constitutes the gimbal lock state associated with this Euler angle representation. In this case only the algebraic sum of the two angles $\alpha$ and $\gamma$ can be calculated as follows:

$$\begin{cases} \beta = \pi/2 \\ \alpha - \gamma = \operatorname{atan2}(r_{12}, r_{22}) \end{cases} \quad \text{or} \quad \begin{cases} \beta = -\pi/2 \\ \alpha + \gamma = \operatorname{atan2}(-r_{12}, r_{22}) \end{cases} \tag{2.27}$$

**Example 2.4 (Pitch-Roll-Yaw Euler Angles)** ⏎
*Consider the rotation of a rigid body represented using the rotation matrix given in Example 2.2. Represent this*

*rotation using the fixed-angle conventions of pitch-roll-yaw Euler angles.*

**Solution:** For this rotation matrix, the pitch–roll–yaw Euler angles are calculated from Equation 2.25 as follows.

$$\alpha = 22.23^\circ \;\; ; \;\; \beta = -20.73^\circ \;\; ; \;\; \gamma = -4.11^\circ$$

**Example 2.5** *Represent the rotation of a rigid body, given by the following rotation matrix, using pitch–roll–yaw Euler angles.*

$$\boldsymbol{R} = \begin{bmatrix} 0 & 0.5 & 0.866 \\ 0 & 0.866 & -0.5 \\ -1 & 0 & 0 \end{bmatrix}$$

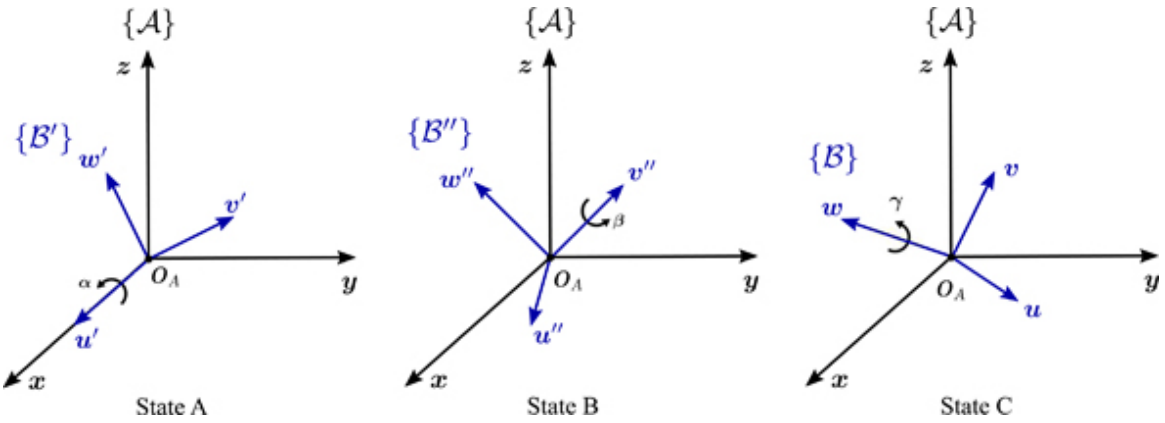
**Solution:** For this rotation matrix, computing the pitch–roll–yaw angles using the relation provided in Equation 2.25 is not feasible. This is due to the fact that $r_{11} = r_{21} = 0$, indicating that the angle $\beta$ can only be either positive or negative 90 degrees. In these specific scenarios, only the algebraic sum of the two angles $\alpha$ and $\gamma$ can be determined using Equation 2.27.

$$\begin{cases} \beta = \pi/2 \\ \alpha - \gamma = \pi/6 \end{cases} \quad \text{or} \quad \begin{cases} \beta = -\pi/2 \\ \alpha + \gamma = -\pi/6 \end{cases}$$

## $\boldsymbol{u - v - w}$ Euler Angles

As previously noted, another widely used approach for describing the rotation of a rigid body entails rotations around the axes of a coordinate system that moves with the body, referred to as the moving-axes Euler angles representation. For instance, one of the twelve configurations of this representation, as detailed in [3, Appendix B], can be illustrated by the $\boldsymbol{u - v - w}$ orientation representation:

As shown in Figure 2.7, consider initially aligning the moving coordinate system with the fixed coordinate system attached to the body $\{\mathscr{A}\}$. Rotate this moving coordinate system first around the $\boldsymbol{u}$ axis by an angle $\alpha$ to reach the intermediate coordinate system $\{\mathscr{B}'\}$. Then, rotate this intermediate coordinate system around the $\boldsymbol{v}'$ axis of the moving frame $\{\mathscr{B}'\}$ by an angle $\beta$ to reach the second intermediate coordinate system $\{\mathscr{B}''\}$. Finally, rotate this intermediate coordinate system around the $\boldsymbol{w}''$ axis of the moving coordinate system $\{\mathscr{B}''\}$ by an angle $\gamma$ to reach the final coordinate system $\{\mathscr{B}\}$.




▶ Long Description for Figure 2.7

**FIGURE 2.7**

Sequential rotation around the moving coordinate axes by an angle $\alpha$ around the $\boldsymbol{u}$ axis, then by an angle $\beta$ around the $\boldsymbol{v}'$ axis, and finally by an angle $\gamma$ around the $\boldsymbol{w}''$ axis. ⏎

In this representation, the orientation of the rigid body is defined with respect to a coordinate system attached to it, employing three successive rotations in a specified sequence relative to the reference coordinate system $\{\mathscr{A}\}$. It's important to note that in this approach, rotations are performed around the axes of the moving coordinate system. Therefore, the conventional pre-multiplication rule for calculating the final rotation matrix does not apply. Instead, the rotation matrix resulting from the post-multiplication of three consecutive rotation matrices around the axes $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ is utilized, as described in [8].

This mapping can be interpreted as the inverse rotational transformation from frame $\{\mathscr{A}\}$ to $\{\mathscr{B}\}$. In this context,

frame $\{\mathscr{A}\}$ remains fixed while rotations occur around its axes. Initially, frame $\{\mathscr{A}\}$ rotates about $\boldsymbol{u}$ by an angle of $-\alpha$, followed by a rotation about $\boldsymbol{v}$ by an angle of $-\beta$, and lastly a rotation about $\boldsymbol{z}$ by an angle of $-\gamma$. This sequence can be expressed as follows:

$$ {}^{B}\boldsymbol{R}_{A}(-\alpha,-\beta,-\gamma) = \boldsymbol{R}_{w}(-\gamma)\boldsymbol{R}_{v}(-\beta)\boldsymbol{R}_{u}(-\alpha) \tag{2.28} $$

Equation 2.28 can be written as:

$$ \begin{aligned} {}^{A}\boldsymbol{R}_{B}(\alpha,\beta,\gamma) &= {}^{B}\boldsymbol{R}_{A}{}^{-1}(-\alpha,-\beta,-\gamma) \\ &= [\boldsymbol{R}_{w}(-\gamma)\boldsymbol{R}_{v}(-\beta)\boldsymbol{R}_{u}(-\alpha)]^{-1} \\ &= \boldsymbol{R}_{u}^{-1}(-\alpha)\boldsymbol{R}_{v}^{-1}(-\beta)\boldsymbol{R}_{w}^{-1}(-\gamma) \end{aligned} $$

and finally as:

$$ {}^{A}\boldsymbol{R}_{B}(\alpha,\beta,\gamma) = \boldsymbol{R}_{u}(\alpha)\boldsymbol{R}_{v}(\beta)\boldsymbol{R}_{w}(\gamma) \tag{2.29} $$

This equation offers a general guideline for calculating the resultant rotation matrix when finite rotations are executed around the axes of a moving frame. Comparing this equation to relation 2.23, it becomes evident that when rotations are performed around the axes of a fixed coordinate system, pre-multiplication is used. In contrast, rotations around the axes of a moving coordinate frame

necessitate post-multiplication, which arises from the inverse rotational mapping.

Simplify relation 2.29, to reach to the final rotation matrix.

$$\begin{aligned}\boldsymbol{R}_{uvw}(\alpha,\beta,\gamma) &= \begin{bmatrix}1 & 0 & 0\\ 0 & c\alpha & -s\alpha\\ 0 & s\alpha & c\alpha\end{bmatrix}\begin{bmatrix}c\beta & 0 & s\beta\\ 0 & 1 & 0\\ -s\beta & 0 & c\beta\end{bmatrix}\begin{bmatrix}c\gamma & -s\gamma & 0\\ s\gamma & c\gamma & 0\\ 0 & 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}c\beta c\gamma & -c\beta s\gamma & s\beta\\ s\alpha s\beta c\gamma + c\alpha s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta\\ -c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma & c\alpha c\beta\end{bmatrix}\end{aligned} \tag{2.30}$$

Similar to the previous case, the inverse problem can be solved by following relations, provided that $c\beta \neq 0$.

$$\begin{aligned}\beta &= \operatorname{atan2}(r_{13}, \pm\sqrt{r_{11}^2 + r_{12}^2})\\ \alpha &= \operatorname{atan2}(-r_{23}/c\beta, r_{33}/c\beta)\\ \gamma &= \operatorname{atan2}(-r_{12}/c\beta, r_{11}/c\beta)\end{aligned} \tag{2.31}$$

In the event that $c\beta = 0$, this scenario corresponds to the gimbal lock condition characteristic of this Euler angle representation. When $\beta = \pm\pi/2$, the rotated axis $\boldsymbol{u}$ becomes aligned with the $\boldsymbol{w}$ axis. As a result, the inverse formulation becomes degenerate, and only the sum of the

two angles $\alpha$ and $\gamma$ can be algebraically computed as follows:

$$\begin{cases} \beta = \pi/2 \\ \alpha + \gamma = \operatorname{atan2}(r_{32}, r_{22}) \end{cases} \quad \text{or} \quad \begin{cases} \beta = -\pi/2 \\ \alpha - \gamma = \operatorname{atan2}(r_{32}, r_{22}) \end{cases} \tag{2.32}$$

**Example 2.6 ($\boldsymbol{u-v-w}$ Euler Angles)** *Consider the rotation matrix introduced in Example 2.2, and analyzed in Examples 2.3 and 2.4. Represent this rotation with $\boldsymbol{u-v-w}$ Euler angles.*

**Solution:** Considering this rotation matrix, $\boldsymbol{u-v-w}$ Euler angles can be derived by using Equation 2.31, as follows.

$$\alpha = 22.23^\circ \;\; ; \;\; \beta = -20.73^\circ \;\; ; \;\; \gamma = 4.11^\circ$$

**Example 2.7** *Consider the rotation of a rigid body, which is represented by the following rotation matrix.*

$$\boldsymbol{R} = \begin{bmatrix} 0 & 0 & 1 \\ 0.966 & -0.259 & 0 \\ 0.259 & 0.966 & 0 \end{bmatrix}$$

*Represent this rotation with $\boldsymbol{u-v-w}$ Euler angles.*

**Solution:** For this rotation matrix, the $\boldsymbol{u}-\boldsymbol{v}-\boldsymbol{w}$ Euler angles cannot be computed using Equation 2.31, as $r_{11}=r_{12}=0$, leading to a gimbal lock in this representation. Therefore, the inverse formulation degenerates, and only the algebraic sum of the two angles $\alpha$ and $\gamma$ can be determined from Equation 2.32:

$$\begin{cases}\beta=\pi/2\\ \alpha+\gamma=5\pi/12\end{cases}\quad\text{or}\quad\begin{cases}\beta=-\pi/2\\ \alpha-\gamma=5\pi/12\end{cases}$$

## $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{w}$ Euler Angles

Another representation of Euler angles involves rotating by an angle $\alpha$ around the axis $\boldsymbol{w}$, followed by rotating by an angle $\beta$ around the axis $\boldsymbol{v}$, and finally rotating by an angle $\gamma$ around the axis $\boldsymbol{w}$. In this scenario, the final rotation matrix is obtained by post-multiplying the individual rotation matrices as follows:

$$\begin{aligned}\boldsymbol{R}_{wvw}(\alpha,\beta,\gamma) &= \boldsymbol{R}_w(\alpha)\boldsymbol{R}_v(\beta)\boldsymbol{R}_w(\gamma)\\ &= \begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta\\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta\\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix}.\end{aligned} \tag{2.33}$$

Similarly to the previous methods, when the angle $s\beta \neq 0$, the solution to the inverse problem can be obtained from the following relationships:

$$\begin{aligned} \beta &= \operatorname{atan2}(\pm\sqrt{r_{31}^2 + r_{32}^2}, r_{33}) \\ \alpha &= \operatorname{atan2}(r_{23}/s\beta, r_{13}/s\beta) \\ \gamma &= \operatorname{atan2}(r_{32}/s\beta, -r_{31}/s\beta) \end{aligned} \tag{2.34}$$

Note that if $s\beta = 0$, this corresponds to the gimbal lock state in this type of Euler angle representation. When $\beta = 0$ or $\pi$, the inverse formulation degenerates, and only the algebraic sum of the two angles $\alpha$ and $\gamma$ can be calculated as follows:

$$\begin{cases} \beta = 0 \\ \gamma + \alpha = \operatorname{atan2}(-r_{12}, r_{11}) \end{cases} \quad \text{or} \quad \begin{cases} \beta = \pi \\ \gamma - \alpha = \operatorname{atan2}(r_{12}, -r_{11} \end{cases} \tag{2.35}$$

**Example 2.8 ($\boldsymbol{w-v-w}$ Euler Angles)** *Consider the rotational transformation of a rigid body using the rotation matrix discussed in Example 2.2. Express this transformation using Euler angles $\boldsymbol{w-v-w}$.*

**Solution:** For this rotation matrix, the $\boldsymbol{w-v-w}$ Euler angles can be calculated from Equation 2.34 as follows.

$$\alpha = -135^\circ \;\; ; \;\; \beta = 30^\circ \;\; ; \;\; \gamma = 135^\circ$$

In this representation, the Euler angles obtained are notably more regular compared to the previous angles, facilitating a clearer visualization and understanding of the physical concept of rotation.

## $w - u - w$ Euler Angles

Another approach using Euler angles includes rotating by an angle $\alpha$ around the axis $\boldsymbol{w}$, followed by a rotation of angle $\beta$ around the axis $\boldsymbol{u}$ (substituting the $\boldsymbol{v}$ frame from the previous case), and concluding with a rotation of angle $\gamma$ around the axis $\boldsymbol{w}$. Here, the resultant rotation matrix is computed by performing post-multiplication of the respective rotation matrices, depicted below.

$$\begin{aligned} \boldsymbol{R}_{wuw}(\alpha, \beta, \gamma) &= \boldsymbol{R}_w(\alpha)\boldsymbol{R}_u(\beta)\boldsymbol{R}_w(\gamma) \\ &= \begin{bmatrix} c\alpha c\gamma - s\alpha c\beta s\gamma & -c\alpha s\gamma - s\alpha c\beta c\gamma & s\alpha s\beta \\ s\alpha c\gamma + c\alpha c\beta s\gamma & -s\alpha s\gamma + c\alpha c\beta c\gamma & -c\alpha s\beta \\ s\beta s\gamma & s\beta c\gamma & c\beta \end{bmatrix} \end{aligned} \tag{2.36}$$

The solution to the inverse problem in this representation, in the case where the angle $s\beta \neq 0$, is obtained from the following equations.

$$
\begin{aligned}
\beta &= \operatorname{atan2}(\pm\sqrt{r_{31}^2 + r_{32}^2}, r_{33}) \\
\alpha &= \operatorname{atan2}(r_{13}/s\beta, -r_{23}/s\beta) \\
\gamma &= \operatorname{atan2}(r_{31}/s\beta, r_{32}/s\beta)
\end{aligned}
\tag{2.37}
$$

Furthermore, if the angle $s\beta = 0$, the inverse problem equations will degenerate and this will result into a gimbal lock of this representation. Consequently, only the algebraic sum of the two angles $\alpha$ and $\gamma$ can be calculated from Equation 2.38 as follows:

$$
\begin{cases} \beta = 0 \\ \alpha + \gamma = \operatorname{atan2}(-r_{12}, r_{11}) \end{cases}
\quad \text{or} \quad
\begin{cases} \beta = \pi \\ \alpha - \gamma = \operatorname{atan2}(r_{12}, r_{11}) \end{cases}
\tag{2.38}
$$

**Example 2.9 ($w - u - w$ Euler Angles)** *Consider the rotation of a rigid body using the rotation matrix introduced in Example 2.2. Represent this rotation using the $w - u - w$ Euler angles.*

**Solution:** For this rotation matrix, the Euler angles $w - u - w$ are calculated as follows using the relation in Equation 2.37:

$$
\alpha = -45^\circ \;\; ; \;\; \beta = -30^\circ \;\; ; \;\; \gamma = 45^\circ
$$

It is apparent that the Euler angles obtained in this representation, similar to the previous example, exhibit a notably more regular pattern. These angles facilitate a clearer visualization, offering a more intuitive understanding of the fundamental physical concept of rotation.

**Example 2.10** *Consider the rotation of a rigid body depicted in Figure 2.8. Suppose the body initially adopts the position with the coordinate system* $\{\mathscr{I}\}$ *and transitions to the final state illustrated by the coordinate system* $\{\mathscr{F}\}$*. Derive the rotation matrix and Euler angles* $\boldsymbol{w-u-w}$ *that represent the rotation of this rigid body using the geometric aspects of the motion.*

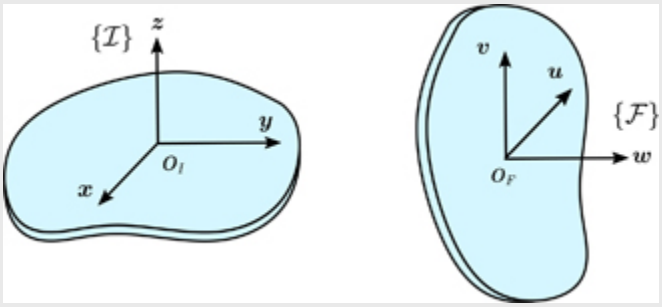



**FIGURE 2.8**
An example of the rotation of a rigid body with specified rotation angles. ⏎

**Solution:** The rotation matrix can be easily derived by using its definition given in the relation 2.2.

$$ {}^{I}\boldsymbol{R}_{F} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} $$

Based on the geometric motion depicted in Figure 2.8, this rotation can be described with two sequential rotations. Initially, a rotation around the moving $\boldsymbol{w}$ axis by an angle of 180 degrees, followed by a rotation around the moving $\boldsymbol{u}$ axis by an angle of 90 degrees. Therefore, the $\boldsymbol{w}-\boldsymbol{u}-\boldsymbol{w}$ Euler angles can be determined as:

$$ \alpha = \pi \;\; ; \;\; \beta = \pi/2 \;\; ; \;\; \gamma = 0 $$

These angles may be derived from the inverse problem solution given in Equation 2.37, as well.

It's crucial to emphasize that despite different Euler angle descriptions for a body's rotation, a single rotation matrix is derived for each representation. Moreover, the inverse problem typically yields multiple solutions, resulting in at least two sets of corresponding angles for each representation. In scenarios of gimbal lock, where the inverse problem equation breaks down, the representation of the body's rotation with Euler angles becomes ambiguous, offering an infinite number of solutions. This makes it challenging to uniquely determine independent

Euler angles along all three axes. Therefore, when using Euler angles to analyze the motion of robotic arm movements, careful consideration is required to avoid gimbal lock.

To address this challenge, continuity constraints in motion are commonly applied to prevent abrupt shifts between potential solutions. Additionally, to mitigate gimbal lock when representing rotations with Euler angles, two sets of these angles are often employed simultaneously in the representation of rotational motion. This approach allows the second set of angles to describe the motion when the first set encounters a gimbal lock. Despite the prevalent use of Euler angles in describing the motion of objects in three-dimensional space, there is a preference for using rotation matrices due to their non-singular nature in representation. In the upcoming chapter, alternative approaches to orientation representation will be explored.

## 2.3 Rigid Body's Pose

In the previous section, we explored the orientation of a rigid body undergoing pure rotational motion using two methods. When the rigid body also undergoes translational motion alongside rotation, analyzing its overall motion requires considering the combination of these two types of motion. For this purpose, we examine the general motion of a body in three-dimensional space, as depicted in Figure

2.1. The orientation of the body in space is represented by a coordinate system attached to the body, denoted as $\{\mathscr{B}\}$, relative to a reference coordinate system.

The translational motion of the body is described by the vector distance between the origin of the moving coordinate system $O_B$ and the origin of the reference coordinate system $O_A$. This motion is illustrated in the figure using the position vector ${}^{A}\boldsymbol{P}_{O_B}$. It's important to note that translational motion is uniquely characterized by the position vector ${}^{A}\boldsymbol{P}_{O_B}$. On the other hand, there are various methods to represent the orientation of the body in space. Fortunately, as demonstrated in the preceding section, diverse representations of a rigid body's orientation in space can all be encapsulated by a single rotation matrix denoted as ${}^{A}\boldsymbol{R}_{B}$. Consequently, the *pose* of a rigid body in space can be precisely determined by knowing the following details.

**1)** The position vector from the origin of the moving coordinate system $O_B$ to the origin of the reference coordinate system $\{\mathscr{A}\}$, represented by the vector ${}^{A}\boldsymbol{P}_{O_B}$ .

**2)** The orientation of a rigid body or a coordinate system attached to it, denoted as $\{\mathscr{B}\}$, relative to the reference coordinate system $\{\mathscr{A}\}$ is represented by the rotation matrix ${}^{A}\boldsymbol{R}_{B}$. This rotation matrix can be obtained from any type of orientation representation of the rigid body in space.

In this way, the exact position vector of any point $P$ on the body, in relation to the reference coordinate system $\{\mathscr{A}\}$, can be found using the translational motion of the body denoted by ${}^{A}\boldsymbol{P}_{O_B}$, as well as the rotational motion and orientation of the body described by the rotation matrix ${}^{A}\boldsymbol{R}_{B}$. This determination can be accomplished using the following relationship:

$$ {}^{A}\boldsymbol{P} = {}^{A}\boldsymbol{P}_{O_B} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} \tag{2.39} $$

Here, ${}^{A}\boldsymbol{P}_{O_B}$ represents the translation vector, ${}^{A}\boldsymbol{R}_{B}$ is the rotation matrix indicating orientation, and ${}^{B}\boldsymbol{P}$ is the position vector of point $P$ relative to the body-fixed coordinate system $\{\mathscr{B}\}$. With this understanding of the body's position in space, we can visualize its initial state where the body-fixed coordinate system $\{\mathscr{B}\}$ aligns with the reference coordinate system $\{\mathscr{A}\}$. Initially, the body moves translationally along the vector ${}^{A}\boldsymbol{P}_{O_B}$ without rotation, positioning $\{\mathscr{B}\}$ parallel to $\{\mathscr{A}\}$ at a distance of ${}^{A}\boldsymbol{P}_{O_B}$. Subsequently, it undergoes rotational motion governed by the rotation matrix ${}^{A}\boldsymbol{R}_{B}$ to reach its final orientation.

The separation of general motion into translational and rotational components is a fundamental concept in classical mechanics and is formally expressed in Mozzi-Chasles' Theorem [58]. This theorem addresses the decomposition of general motion into translational and rotational

elements, emphasizing the instantaneous axis of rotation or the screw axis. Further insights into this topic will be provided in the upcoming chapter and in Section 3.3.

**Example 2.11 (Pose of a rigid body)** *Consider the movement of a rigid body as depicted in Figure 2.9. In this scenario, the rigid body has moved from its initial state with the coordinate system* $\{0\}$ *to a new state represented by the coordinate system* $\{1\}$*, experiencing both translational and rotational motions. Using the given data in the figure, find the position vector of point P relative to the reference frame* $\{0\}$.

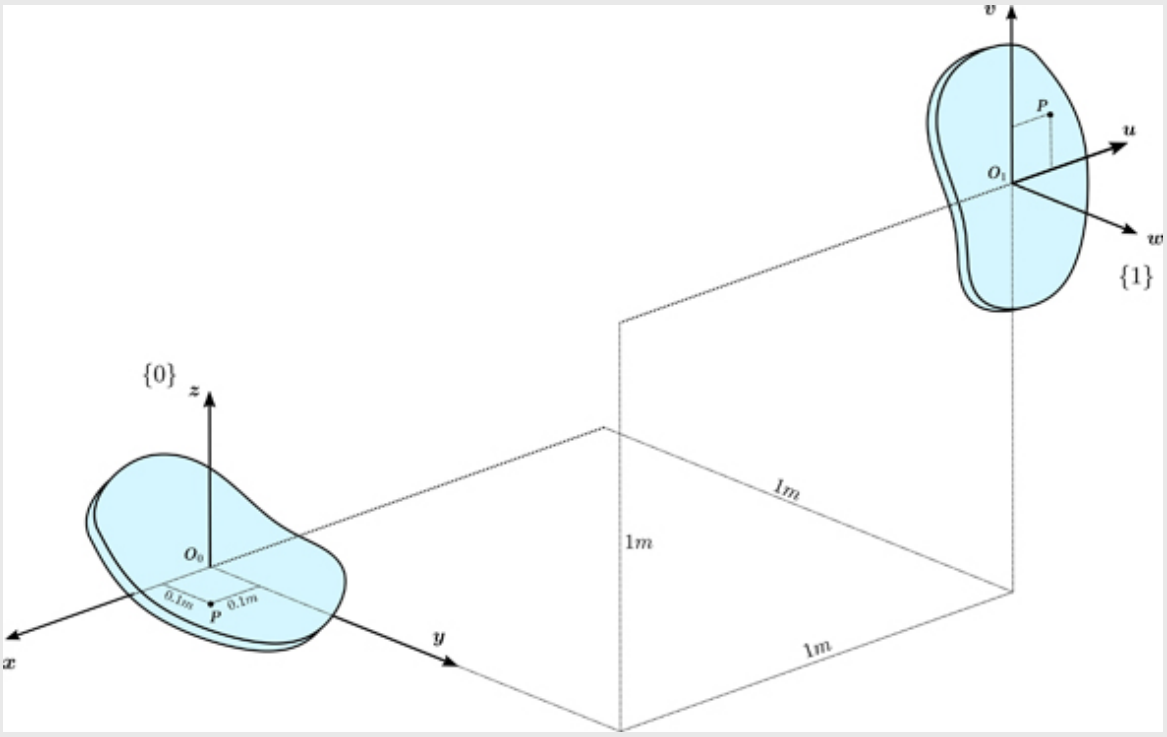



**FIGURE 2.9**

An example of general motion of a rigid body in 3-dimensional space. ⏎

**Solution:** In Figure 2.9, the position of point $P$ relative to the body's coordinate system is described by the vector ${}^{1}\boldsymbol{P} = [0.1 \quad 0.1 \quad 0]^{T}$. Furthermore, the body's translational motion is defined by the vector ${}^{0}\boldsymbol{P}_{o_1}$, and its rotational motion is represented by the rotation matrix ${}^{0}\boldsymbol{R}_{1}$ as follows:

$$ {}^{0}\boldsymbol{P}_{o_1} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \quad ; \quad {}^{0}\boldsymbol{R}_{1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} $$

This allows us to derive the absolute position vector of point $P$ relative to the reference coordinate system $\{0\}$ using the following relationship:

$$ \begin{aligned} {}^{0}\boldsymbol{P} &= {}^{0}\boldsymbol{R}_{1}\,{}^{1}\boldsymbol{P} + {}^{0}\boldsymbol{P}_{o_1} \\ &= \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.1 \\ 1.0 \\ 1.1 \end{bmatrix} \end{aligned} $$

The accuracy of these computations can be readily confirmed by referring to the accompanying figure.

## 2.4 Homogeneous Transformation

In earlier sections, we introduced pure rotational motion in Equation 2.16 and detailed the mapping of general motion in Equation 2.39. Rotational motion mapping involves multiplying the rotation matrix by the relative position vector, symbolically representing the position vector in the reference coordinate system. For general body motion mapping, we combine matrix multiplication and vector addition, incorporating the position vector of the moving coordinate system relative to the reference.

However, using only the $3 \times 3$ rotation matrix for this mapping is insufficient. To address this limitation, we introduce the concept of *homogeneous transformation* in analyzing rigid body motion in three-dimensional space. Here, vectors are extended to homogeneous coordinates of dimensions $4 \times 1$, and the transformation matrix expands to $4 \times 4$. With this approach, Equation 2.39 resembles Equation 2.16, simplifying the relationship to a single matrix multiplication operation.

$$^{A}\boldsymbol{P} = {}^{A}\boldsymbol{T}_{B}\, {}^{B}\boldsymbol{P} \tag{2.40}$$

in which ${}^{A}\boldsymbol{T}_{B}$ denotes the homogeneous transformation matrix with dimensions $4 \times 4$.

### 2.4.1 Homogeneous Coordinates

Up to this point in the chapter, we have focused solely on position vectors. In upcoming chapters, we will introduce and utilize other vectors such as linear and angular velocity, acceleration, force, and torque in robotics analysis. These vectors can be categorized into two groups: line vectors and free vectors.

Line vectors are tied to specific points of application, making the precise location crucial for their analysis. The position vector of a point on a rigid body falls into this category because it varies with even minor shifts in the point's position. Similarly, other quantities like linear velocity of a point on the body and forces acting upon it also fall under line vectors.

Conversely, free vectors represent physical quantities that are independent of their point of application. The orientation of a body exemplifies this category, as it describes the body's overall positioning. Vectors such as angular velocity, angular acceleration, and torque applied to a rigid body also belong to free vectors because they are not contingent on specific points of application.

When transforming coordinates for line vectors, both translation and rotation of the coordinate system significantly impact their values. Therefore, to accommodate these transformations using homogeneous coordinates and the relationship in Equation 2.40, we extend regular vectors to dimensions of $4 \times 1$ by appending a fourth component set to one.

$$\text{For line vectors:} \quad \boldsymbol{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \hline 1 \end{bmatrix} \tag{2.41}$$

This adaptation enables us to employ the homogeneous transformation matrix to unify the representation of both translation and rotation.

For free vectors, only the orientation changes with coordinate transformations. Therefore, homogeneous coordinates for these vectors are defined by appending zero to the fourth component of the regular vector, as follows:

$$\text{For free vectors:} \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \hline 0 \end{bmatrix} \tag{2.42}$$

With the introduction of homogeneous coordinates, it's crucial to emphasize their use specifically in conjunction with homogeneous transformations for changing the coordinate system of a vector quantity.

### 2.4.2 Homogeneous Transformation Matrix

The *homogeneous transformation matrix* is a $4 \times 4$ matrix used to convert a vector quantity represented in homogeneous coordinates from one coordinate system to another. This transformation is expressed in a concise matrix format, as depicted in Equation 2.40. In light of the relationship defined in Equation 2.39 and the simplified matrix structure given in Equation 2.40, the homogeneous transformation matrix consists of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$ and the translation vector ${}^{A}\boldsymbol{P}_{O_B}$, outlined below.

$$
{}^{A}\boldsymbol{T}_{B} = \left[\begin{array}{ccc|c} & {}^{A}\boldsymbol{R}_{B} & & {}^{A}\boldsymbol{P}_{O_B} \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.43}
$$

The transformation matrix ${}^{A}\boldsymbol{T}_{B}$ defines the mapping for line vectors represented in homogeneous coordinates, which are depicted as $4 \times 1$ matrices. This mapping incorporates both translational and rotational motions.

$$\begin{bmatrix} {}^{A}\boldsymbol{P} \\ \hline 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}\boldsymbol{R}_{B} & & {}^{A}\boldsymbol{P}_{O_B} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}\boldsymbol{P} \\ \hline 1 \end{bmatrix} \tag{2.44}$$

$$\begin{aligned} {}^{A}\boldsymbol{P} &= {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} + {}^{A}\boldsymbol{P}_{O_B} \\ 1 &= 1 \end{aligned} \tag{2.45}$$

For the transformation of free vectors, where only the rotational matrix affects the transformation, Equation 2.40 accurately performs the coordinate transformation.

$$\begin{bmatrix} {}^{A}\boldsymbol{\omega} \\ \hline 0 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}\boldsymbol{R}_{B} & & {}^{A}\boldsymbol{P}_{O_B} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}\boldsymbol{\omega} \\ \hline 0 \end{bmatrix} \tag{2.46}$$

$$
\begin{aligned}
{}^{A}\boldsymbol{\omega} &= {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{\omega} \\
0 &= 0
\end{aligned}
\tag{2.47}
$$

**Example 2.12 (Homogeneous Transformations)** *Consider the overall motion of a rigid body in space, depicted in Figure 2.9. Suppose the relative position vector of point P, noted as* ${}^{1}\boldsymbol{P}$*, and the angular velocity vector of the body in the moving coordinate system* $\{1\}$*, denoted by* ${}^{1}\boldsymbol{\omega}$*, are provided as follows:*

$$
{}^{1}\boldsymbol{P} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0 \end{bmatrix} \;\; ; \;\; {}^{1}\boldsymbol{\omega} = \begin{bmatrix} 1 \\ 0 \\ 1.2 \end{bmatrix}
$$

*Find the position vector of point P in the absolute coordinates, and determine the angular velocity of the object relative to the reference coordinate system* $\{0\}$*.*

**Solution:** To ascertain the absolute position vector of point *P*, one can employ the homogeneous transformation and the relationship specified by Equation 2.40. In this scenario, the homogeneous transformation matrix can be formulated as follows.

$$
{}^{0}\boldsymbol{T}_{1} = \left[\begin{array}{ccc|c} & {}^{0}\boldsymbol{R}_{1} & & {}^{0}\boldsymbol{P}_{O_1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

Moreover, because the position vector of point $P$ is a column vector and the angular velocity of the body $\boldsymbol{\omega}$ is a vector that can translate freely, the following homogeneous coordinates should be applied for their transformation.

$$
{}^{1}\boldsymbol{P} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0 \\ \hline 1 \end{bmatrix} \quad ; \quad {}^{1}\boldsymbol{\omega} = \begin{bmatrix} 1 \\ 0 \\ 1.2 \\ \hline 0 \end{bmatrix}
$$

Thus, it becomes straightforward to compute the homogeneous transformation of ${}^{1}\boldsymbol{P}$ and ${}^{1}\boldsymbol{\omega}$ relative to the reference coordinate system as follows.

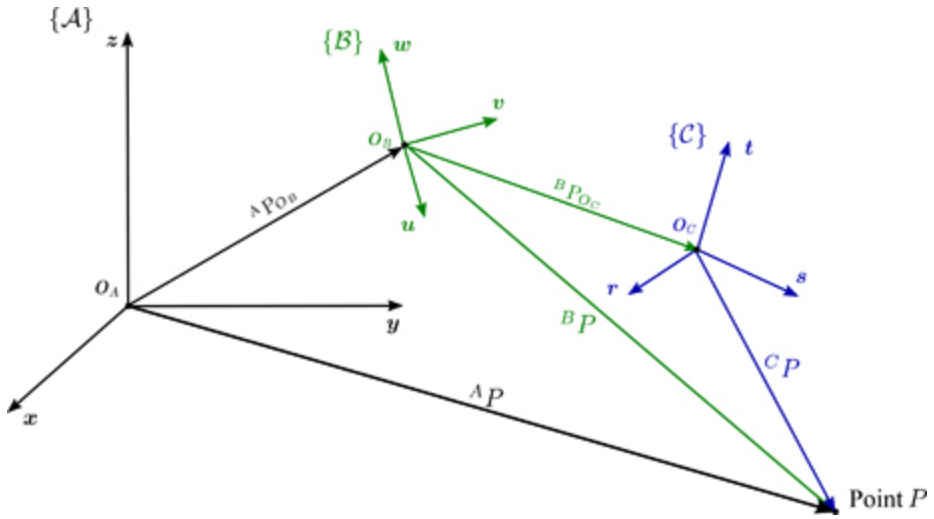
$$ {}^0\boldsymbol{P} = {}^0\boldsymbol{T}_1\,{}^1\boldsymbol{P} = \begin{bmatrix} -1.1 \\ 1.0 \\ 1.1 \\ \hline 1 \end{bmatrix} \quad ; \quad {}^0\boldsymbol{\omega} = {}^0\boldsymbol{T}_1\,{}^1\boldsymbol{\omega} = \begin{bmatrix} -1 \\ 1.2 \\ 0 \\ \hline 0 \end{bmatrix} $$

### 2.4.3 Homogeneous Transformation Arithmetic

In this section, we will outline the essential mathematical procedures for homogeneous transformations. Specifically, we will discuss the methodology for executing successive coordinate transformations and deriving the inverse of a homogeneous transformation.

#### Consecutive Transformations

Consider the successive motion of a rigid body from its initial state in coordinate system $\{\mathscr{A}\}$ to $\{\mathscr{B}\}$ and then to $\{\mathscr{C}\}$, depicted in Figure 2.10. Suppose the position vector of point $P$ in the final state of the body is denoted as ${}^C\boldsymbol{P}$ in the $\{\mathscr{C}\}$ coordinate system. The objective is to determine this vector relative to the reference coordinate system $\{\mathscr{A}\}$ .




▶ Long Description for Figure 2.10

**FIGURE 2.10**

Successive motion of a rigid body from the depicted state in coordinate systems $\{\mathscr{A}\}$, $\{\mathscr{B}\}$, and finally in coordinate system $\{\mathscr{C}\}$. ⏎

Given the known states of the rigid body in two successive coordinate systems, vector ${}^{C}\boldsymbol{P}$ can first be determined in the intermediate coordinate system $\{\mathscr{B}\}$ and subsequently transformed into the reference coordinate system $\{\mathscr{A}\}$ using the relation provided by Equation 2.40. To achieve this, the transformation matrix ${}^{B}\boldsymbol{T}_{C}$ is utilized to establish the position vector relative to the $\{\mathscr{B}\}$ coordinate system.

$$ {}^{B}\boldsymbol{P} = {}^{B}\boldsymbol{T}_{C}\, {}^{C}\boldsymbol{P} \tag{2.48} $$

Furthermore, the transformation matrix ${}^{A}\boldsymbol{T}_{B}$ is then used to obtain the position vector relative to the reference coordinate system $\{\mathscr{A}\}$ as follows.

$$ {}^{A}\boldsymbol{P} = {}^{A}\boldsymbol{T}_{B}\, {}^{B}\boldsymbol{P} \tag{2.49} $$

By combining the Equations 2.48 and 2.49, one may write:

$$ {}^{A}\boldsymbol{P} = {}^{A}\boldsymbol{T}_{B}\ {}^{B}\boldsymbol{T}_{C}\, {}^{C}\boldsymbol{P} \tag{2.50} $$

The combined transformation matrix, derived as the product of two homogeneous transformation matrices in a sequential motion, is expressed as follows:

$$ {}^{A}\boldsymbol{T}_{C} = {}^{A}\boldsymbol{T}_{B}\ {}^{B}\boldsymbol{T}_{C} \tag{2.51} $$

If we aim to express the transformation matrix ${}^{A}\boldsymbol{T}_{C}$ analytically using its constituent transformation matrices, namely ${}^{A}\boldsymbol{T}_{B}$ and ${}^{B}\boldsymbol{T}_{C}$, we can utilize the general form of the homogeneous transformation matrix, denoted as 2.43. It can be formulated as follows:

$$
{}^{A}\boldsymbol{T}_{C} = \left[\begin{array}{ccc|c} & {}^{A}\boldsymbol{R}_{B}\ {}^{B}\boldsymbol{R}_{C} & & {}^{A}\boldsymbol{R}_{B}\ {}^{B}\boldsymbol{P}_{O_C} + {}^{A}\boldsymbol{P}_{O_B} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
\tag{2.52}
$$

If a composite transformation involves more than two consecutive transformations, the relationship 2.51 can be generalized to encompass a sequence of movements across various states. This is represented by the following equation:

$$
{}^{A}\boldsymbol{T}_{Z} = {}^{A}\boldsymbol{T}_{B}\ {}^{B}\boldsymbol{T}_{C}\ {}^{C}\boldsymbol{T}_{D} \cdots\ {}^{Y}\boldsymbol{T}_{Z}
\tag{2.53}
$$

**Example 2.13 (Consecutive Homogeneous Transformation)** *Consider the consecutive movement of a rigid body across three different positions defined by coordinate systems* $\{0\}$, $\{1\}$, *and* $\{2\}$ *shown in Figure 2.11. Calculate the homogeneous transformation matrices* ${}^{0}\boldsymbol{T}_{1}$, ${}^{1}\boldsymbol{T}_{2}$, *and the composite transformation matrix* ${}^{0}\boldsymbol{T}_{2}$. *Validate the results using the definition criteria for homogeneous matrices.*

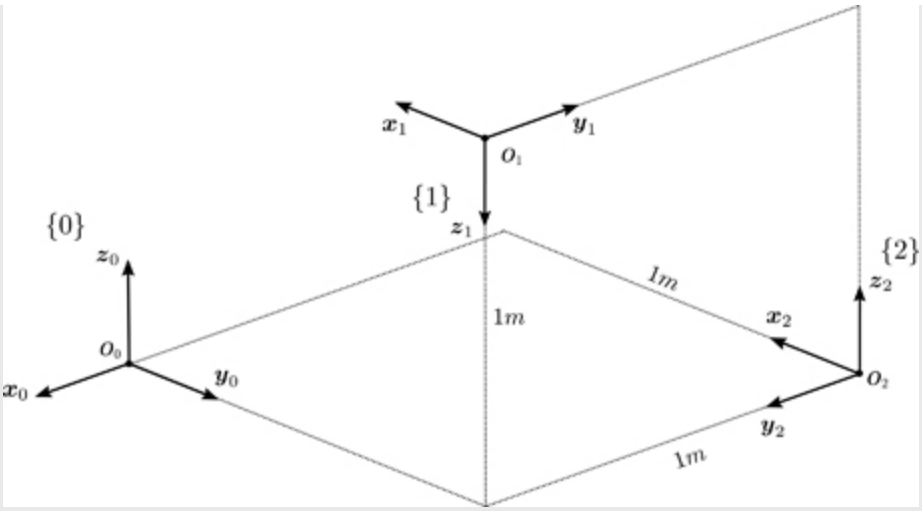



**FIGURE 2.11**
An example of the consecutive motion of a rigid body in three different states is illustrated with coordinate systems $\{0\}, \{1\}, \{2\}$. ⏎

**Solution:** Based on the definitions of rotation matrices and homogeneous transformation matrices, and considering the illustrated motion of the rigid body in space in Figure 2.11, the transformation matrices ${}^0\boldsymbol{T}_1$ and ${}^0\boldsymbol{T}_2$ can be formulated as follows:

$$ {}^0\boldsymbol{T}_1 = \left[\begin{array}{ccc|c} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] , {}^1\boldsymbol{T}_2 = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] $$

By directly multiplying the two matrices above as per Equation 2.53, or alternatively, utilizing the analytical

relationship described in Equation 2.43, the transformation matrix ${}^0\boldsymbol{T}_2$ can be derived as follows:

$$
{}^0\boldsymbol{T}_2 = \left[\begin{array}{ccc|c} 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

The accuracy of the aforementioned calculations can be easily verified with the assistance of Figure 2.11 and the definition of the homogeneous transformation matrix.

**Example 2.14 (Consecutive rotation with a finite angle)** *In this example, it is shown that consecutive rotations with finite angles are not commutative. However, if the rotation angles are infinitesimally small, the rotations become commutative. To illustrate this, consider two consecutive rotations of a rigid body. First, rotate around the $\hat{\boldsymbol{x}}$ axis by a finite angle $\theta_1$. Then, rotate around the $\hat{\boldsymbol{z}}$ axis by a finite angle $\theta_2$. Find the resulting rotation matrix from these two rotations. Next, calculate these consecutive rotations using infinitesimally small angles $\delta\theta_1$ and $\delta\theta_2$, and compare the results.*

**Solution:** First, let's examine infinitesimal rotations around two axes: $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{z}}$.

$$\boldsymbol{R}_x(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \quad , \boldsymbol{R}_z(\theta_2) = \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Based on the matrix multiplication expression provided in Equation 2.53, the combination of rotations in the initial scenario is derived as follows.

$$\boldsymbol{R}_1 = \boldsymbol{R}_x(\theta_1)\, \boldsymbol{R}_z(\theta_2) \quad = \begin{bmatrix} c_2 & -s_2 & 0 \\ c_1 s_2 & c_1 c_2 & -s_1 \\ s_1 s_2 & s_1 c_2 & c_1 \end{bmatrix}$$

Here, shorthand notations $c_1 = \cos(\theta_1)$, $s_1 = \sin(\theta_1)$, $c_2 = \cos(\theta_2)$, and $s_2 = \sin(\theta_2)$ are employed. Finite rotations are represented by multiplying rotation matrices, and since matrix multiplication is not commutative, changing the order of rotation sequences results in a completely different representation, as shown below.

$$\boldsymbol{R}_2 = \boldsymbol{R}_z(\theta_2)\, \boldsymbol{R}_x(\theta_1) \quad = \begin{bmatrix} c_2 & -c_1 s_2 & s_1 s_2 \\ s_2 & c_1 c_2 & -s_1 c_2 \\ 0 & s_1 & c_1 \end{bmatrix}$$

This will result in rotations that are not equal, i.e., $\boldsymbol{R}_1 \neq \boldsymbol{R}_2$. In other words, if the order of rotations in the

sequence changes, the resulting rotation will be completely different.

However, if we consider the consecutive rotations to be infinitesimally small, the situation will be different. By substituting $\theta_1 = \delta\theta_1$ and $\theta_2 = \delta\theta_2$ into the above equations and assuming the angles to be infinitesimally small, we can use the following simplifications:

$$s_1 \simeq \delta\theta_1, \;\; c_1 \simeq 1, \;\; s_2 \simeq \delta\theta_2, \;\; c_2 \simeq 1$$

By this means:

$$\begin{aligned}
\boldsymbol{R}_1 = \boldsymbol{R}_x(\theta_1)\, \boldsymbol{R}_z(\theta_2) &= \begin{bmatrix} 1 & -\delta\theta_2 & 0 \\ \delta\theta_2 & 1 & -\delta\theta_1 \\ \delta\theta_1\delta\theta_2 & \delta\theta_1 & 1 \end{bmatrix} \\
\boldsymbol{R}_2 = \boldsymbol{R}_z(\theta_2)\, \boldsymbol{R}_x(\theta_1) &= \begin{bmatrix} 1 & -\delta\theta_2 & \delta\theta_1\delta\theta_2 \\ \delta\theta_2 & 1 & -\delta\theta_1 \\ 0 & \delta\theta_1 & 1 \end{bmatrix}
\end{aligned}$$

When the rotation angles $\delta\theta_i$ are infinitesimally small, their product $\delta\theta_1\delta\theta_2$ is negligible and approximated as $\delta\theta_1\delta\theta_2 \simeq 0$. Consequently, the result of two consecutive rotations will be equivalent in both scenarios when considering infinitesimally small rotation angles.

$$\boldsymbol{R}_1 = \boldsymbol{R}_2 = \begin{bmatrix} 1 & -\delta\theta_2 & 0 \\ \delta\theta_2 & 1 & -\delta\theta_1 \\ 0 & \delta\theta_1 & 1 \end{bmatrix}$$

This reasoning illustrates that rotations with infinitesimally small angles exhibit the property of commutativity. Additionally, these rotations can be effectively represented using vectors. This concept will be further explored in the upcoming chapter and in Section 3.3. However, this argument primarily serves to establish the vector representation of angular velocity and its derivatives.

## Inverse of a Homogeneous Transformation

In this section, calculating the inverse of a $4 \times 4$ matrix directly may not be computationally efficient. Instead, we utilize the concept of mapping in homogeneous transformations. Given that a significant part of the homogeneous transformation matrix consists of the rotation matrix, its inverse can be obtained simply by transposing it, which involves minimal computation.

On the other hand, another part of the homogeneous transformation matrix corresponds to its translation vector, indicating the displacement of the moving coordinate system's origin relative to the reference coordinate system $\{\mathscr{A}\}$. For the inverse mapping, we must use a vector of the same magnitude but opposite direction, denoted as ${}^{B}\boldsymbol{P}_{OA}$.

Therefore, the inverse of the homogeneous transformation matrix ${}^{A}\boldsymbol{T}_{B}$, denoted as ${}^{B}\boldsymbol{T}_{A}$, can be computed from these two components as follows.

$$ {}^{B}\boldsymbol{T}_{A} = \left[\begin{array}{ccc|c} & {}^{B}\boldsymbol{R}_{A} & & {}^{B}\boldsymbol{P}_{O_A} \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.54} $$

However, as mentioned earlier, ${}^{B}\boldsymbol{R}_{A} = {}^{A}\boldsymbol{R}_{B}^{T}$. To compute the vector ${}^{B}\boldsymbol{P}_{O_A}$ according to the discussion, we can express it as follows:

$$ {}^{B}\boldsymbol{P}_{OB} = {}^{B}\boldsymbol{R}_{A}\ {}^{A}\boldsymbol{P}_{OB} + {}^{B}\boldsymbol{P}_{OA} $$

And since the left-hand side of the above equation equals zero, we can write:

$$ {}^{B}\boldsymbol{P}_{OA} = -\,{}^{A}\boldsymbol{R}_{B}^{T}\ {}^{A}\boldsymbol{P}_{OB} \tag{2.55} $$

In this manner, the inverse of the homogeneous transformation matrix is obtained as follows:

$$
{}^{B}\boldsymbol{T}_{A} = {}^{A}\boldsymbol{T}_{B}^{-1} = \left[\begin{array}{ccc|c} & {}^{A}\boldsymbol{R}_{B}^{T} & & -{}^{A}\boldsymbol{R}_{B}^{T}\,{}^{A}\boldsymbol{P}_{OB} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.56}
$$

The Equation 2.56 calculates the inverse matrix for any transformation matrix with optimized computation.

**Example 2.15** *Assume the homogeneous transformation matrix of a body is represented by*

$$
{}^{0}\boldsymbol{T}_{1} = \begin{bmatrix} 0.933 & 0.167 & 0.354 & 2 \\ 0.067 & 0.933 & -0.354 & 1 \\ -0.354 & 0.354 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

*It seems that there might be some mistakes in reporting certain elements of its rotation matrix. Obtain the correct representation of the homogeneous transformation matrix and then calculate its inverse. Furthermore, if the position vector of point P with respect to the* $\{0\}$ *coordinate system is given as*

${}^0\boldsymbol{P} = [0.25 \quad 0.43 \quad 0.86]^T$ *Find the representation of this vector in frame* $\{1\}$.

**Solution:** By examining the columns of the rotation matrix, it is easily understood that the Euclidean norm of the second column is not equal to one. Also, by inspecting the row norms of the rotation matrix, it becomes evident that the element $r_{12}$ was mistakenly written as 0.167 and should be 0.067. The inverse of the corrected transformation matrix can be obtained using Equation 2.56 with the following calculations.

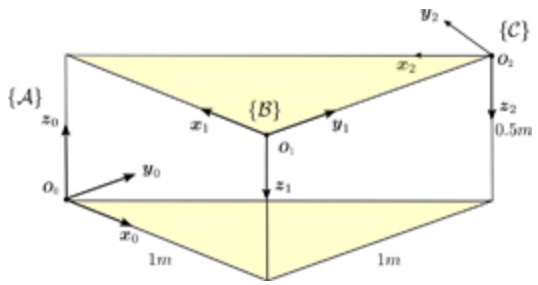
$$
\begin{aligned}
{}^1\boldsymbol{T}_0 = {}^0\boldsymbol{T}_1^{-1} &= \left[\begin{array}{ccc|c} & {}^0\boldsymbol{R}_1^T & & -{}^0\boldsymbol{R}_1^T\,{}^0\boldsymbol{P}_{O_1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \\
&= \left[\begin{array}{rrr|r} 0.933 & 0.067 & -0.354 & -1.80 \\ 0.067 & 0.933 & 0.354 & 0.80 \\ 0.354 & -0.354 & 0.866 & -1.06 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
\end{aligned}
$$

Using this transformation matrix, the vector ${}^1\boldsymbol{P}$ can be calculated from the following relation.

$$^{1}\boldsymbol{P} = {}^{1}\boldsymbol{T}_{0}\,{}^{0}\boldsymbol{P} = \begin{bmatrix} -1.84 & 1.52 & -0.38 \end{bmatrix}^{T}$$

## Problems

2.1 Consider coordinate systems $\{\mathscr{A}\}, \{\mathscr{B}\}, \{\mathscr{C}\}$ in the depicted configuration in Figure 2.12. Obtain the rotation matrices ${}^{A}\boldsymbol{R}_{B}$, ${}^{A}\boldsymbol{R}_{C}$, and ${}^{B}\boldsymbol{R}_{C}$, by using the definition.

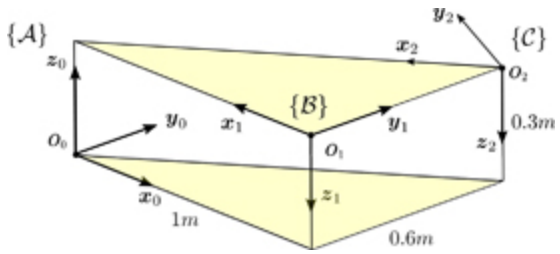



▶ Long Description for Figure 2.12

**FIGURE 2.12**
Illustration of coordinate systems $\{\mathscr{A}\}$, $\{\mathscr{B}\}$, and $\{\mathscr{C}\}$ placed at the corners of a wedge. ⏎

2.2 For the coordinate systems shown in Figure 2.12, obtain the rotation matrices ${}^{B}\boldsymbol{R}_{A}$, ${}^{C}\boldsymbol{R}_{A}$ and ${}^{C}\boldsymbol{R}_{B}$ by using the definition. Then, calculate the inverse rotation matrix obtained in Problem 2.1, and verify your results.

2.3 Repeat Problem 2.1 for the coordinate systems shown in Figure 2.13.



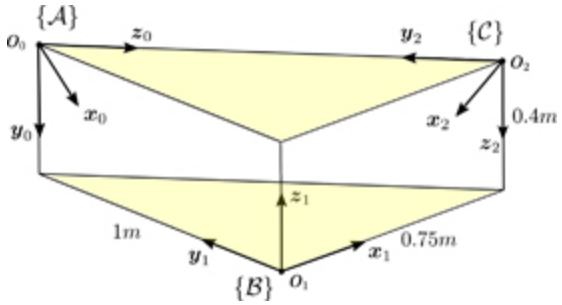

▶ Long Description for Figure 2.13

**FIGURE 2.13**

Representation of coordinate systems $\{\mathscr{A}\}$, $\{\mathscr{B}\}$, and $\{\mathscr{C}\}$, which are positioned at the corners of a wedge. ⏎

2.4 Reiterate Problem 2.2 for the coordinate systems depicted in Figure 2.13.

2.5 Duplicate the exercise outlined in Problem 2.1 for the coordinate systems illustrated in Figure 2.14.




▶ Long Description for Figure 2.14

**FIGURE 2.14**

Representation of coordinate systems $\{\mathscr{A}\}$, $\{\mathscr{B}\}$, and $\{\mathscr{C}\}$, which are positioned at the corners of a wedge. ⏎

2.6 Redo the task presented in Problem 2.2 for the coordinate systems depicted in Figure 2.14.

2.7 The rotation of a rigid body is given by the following rotation matrix:

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.500 & 0.080 & 0.862 \\ 0.500 & 0.787 & -0.362 \\ -0.707 & 0.612 & 0.354 \end{bmatrix} $$

a. If the relative position vector of point $P$ is given as: ${}^{B}\boldsymbol{P} = [[0.707 \quad -0.707 \quad 0]^{T}$ find the absolute position vector of this point, denoted by ${}^{A}\boldsymbol{P}$.

b. If the absolute position vector of point $P$ is given as: ${}^{A}\boldsymbol{P} = [0.577 \quad -0.577 \quad -0.577]^{T}$ find the relative position vector of point $P$, denoted by ${}^{B}\boldsymbol{P}$.

c. Find the eigenvalues and eigenvectors of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$.

d. Calculate the pitch-roll-yaw Euler angles of this rotation.

e. Calculate the $\boldsymbol{w} - \boldsymbol{v} - \boldsymbol{u}$ Euler angles of this rotation.

2.8 The rotation of a rigid body is given by the following rotation matrix:

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} -0.146 & -0.854 & 0.500 \\ 0.854 & 0.146 & 0.500 \\ -0.500 & 0.500 & 0.707 \end{bmatrix} $$

a. Find the eigenvalues and eigenvectors of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$.

b. Calculate the pitch-roll-yaw Euler angles of this rotation.

c. Calculate the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{u}$ Euler angles of this rotation.

d. Calculate the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{w}$ Euler angles of this rotation.

e. Calculate the $\boldsymbol{w}-\boldsymbol{u}-\boldsymbol{w}$ Euler angles of this rotation.

2.9 The rotational motion of a rigid body is represented by the following rotation matrix. It seems that there are errors in recording the elements of this matrix. Correct it using the properties of rotation matrices. Then, repeat Problem 2.7 for the corrected rotation matrix.

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.099 & -0.370 & 0.924 \\ 0.984 & -0.112 & -0.146 \\ 0.149 & 0.924 & 0.354 \end{bmatrix} $$

2.10 Repeat Problem 2.7 for the following rotation matrix.

$$
{}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.933 & 0.067 & 0.354 \\ 0.067 & 0.933 & -0.354 \\ -0.354 & 0.354 & 0.866 \end{bmatrix}
$$

2.11 Reiterate the exercise in Problem 2.8 using the rotation matrix from Problem 2.10.

2.12 Duplicate the task outlined in Problem 2.7 for the given rotation matrix.

$$
{}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0 & 0.259 & 0.966 \\ 0 & 0.966 & -0.250 \\ -1 & 0 & 0 \end{bmatrix}
$$

2.13 Redo the exercise in Problem 2.8 using the rotation matrix provided in Problem 2.12.

2.14 Reiterate the task described in Problem 2.7 for the provided rotation matrix.

$$
{}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} -0.800 & -0.498 & 0.335 \\ 0.462 & 0.155 & 0.873 \\ -0.383 & 0.654 & 0.354 \end{bmatrix}
$$

2.15 Repeat the activity described in Problem 2.8 with the rotation matrix given in Problem 2.14.

2.16 In this exercise, we want to prove that the rotation matrix is an orthogonal matrix, and the inverse of such matrix is equal to its transpose. For this purpose, consider two unit vectors, $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{r}}$, located on a rigid body. Since the angle between these two vectors remains constant for any arbitrary rotation, provide proof for the above-mentioned properties of the rotation matrix.

2.17 Assume a rotational motion of a rigid body is represented by the following pitch-roll-yaw Euler angles.

$$\boldsymbol{R}_x(\pi/6), \boldsymbol{R}_y(\pi/4), \boldsymbol{R}_z(\pi/4)$$

a. Find the rotation matrix ${}^{A}\boldsymbol{R}_B$ representing the motion of this body.

b. Find the eigenvalues and eigenvectors of the rotation matrix ${}^{A}\boldsymbol{R}_B$.

c. Calculate the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{u}$ Euler angles of this rotation.

d. Find the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{w}$ Euler angles of this rotation.

e. Calculate the $\boldsymbol{w}-\boldsymbol{u}-\boldsymbol{w}$ Euler angles of this rotation.

2.18 Assume that the rotation of a rigid body is given with the following angles relative to the following fixed coordinate system:

$$\boldsymbol{R}_z(\alpha), \boldsymbol{R}_y(\beta), \boldsymbol{R}_z(\gamma).$$

a. Find the rotation matrix ${}^{A}\boldsymbol{R}_B$ representing the motion of this body.

b. Evaluate the rotation matrix for $\alpha = \pi/2, \beta = -\pi/4, \gamma = \pi/6$.

c. Find the eigenvalues and eigenvectors of the rotation matrix ${}^{A}\boldsymbol{R}_B$.

d. Calculate the pitch-roll-yaw Euler angles of this rotation.

e. Calculate the $\boldsymbol{w} - \boldsymbol{u} - \boldsymbol{w}$ Euler angles of this rotation.

f. Find the $\boldsymbol{w} - \boldsymbol{v} - \boldsymbol{w}$ Euler angles of this rotation.

2.19 Assume that the rotation of a rigid body is given with constant angles relative to the following moving coordinate system:

$$\boldsymbol{R}_v(\alpha), \boldsymbol{R}_w(\beta), \boldsymbol{R}_u(\gamma).$$

a. Find the rotation matrix ${}^{A}\boldsymbol{R}_B$ representing the motion of this body.

b. Evaluate the rotation matrix for $\alpha = \pi/4, \beta = -\pi/2, \gamma = \pi/3$.

c. Find the eigenvalues and eigenvectors of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$.

d. Calculate the pitch-roll-yaw Euler angles of this rotation.

e. Find the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{u}$ Euler angles of this rotation.

f. Calculate the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{w}$ Euler angles of this rotation.

2.20 Consider the coordinate systems shown in Figure 2.12 and obtain the homogeneous transformation matrices ${}^{A}\boldsymbol{T}_{B}$, ${}^{A}\boldsymbol{T}_{C}$, ${}^{B}\boldsymbol{T}_{C}$ for this configuration.

2.21 Consider the coordinate systems shown in Figure 2.12 and obtain the transformation matrices ${}^{B}\boldsymbol{T}_{A}$, ${}^{C}\boldsymbol{T}_{A}$, ${}^{C}\boldsymbol{T}_{B}$ using the definition. Then, validate your calculations using the inverse transformation matrix relationships obtained in Problem 2.20 and with the help of Equation 2.56.

2.22 Repeat Problem 2.20 for the coordinate systems shown in Figure 2.13.

2.23 Redo the task described in Problem 2.21 for the coordinate systems illustrated in Figure 2.13.

2.24 Duplicate the exercise outlined in Problem 2.20 for the coordinate systems depicted in Figure 2.14.

2.25 Reiterate the task presented in Problem 2.21 for the coordinate systems shown in Figure 2.14.

2.26 If the homogeneous transformation matrix ${}^{A}\boldsymbol{T}_{B}$ is given, obtain the matrix ${}^{B}\boldsymbol{T}_{A}$.

$$
{}^{A}\boldsymbol{T}_{B} = \left[\begin{array}{rrr|r}
0.250 & 0.612 & 0.750 & 0.422 \\
0.612 & 0.500 & -0.612 & 1.673 \\
-0.750 & 0.612 & -0.250 & -0.172 \\
\hline
0 & 0 & 0 & 1
\end{array}\right]
$$

2.27 Consider the coordinate systems shown in Figure 2.12 and assume that the vectors ${}^{C}\boldsymbol{P}$ and ${}^{C}\boldsymbol{\omega}$ are represented as follows:

$$
{}^{C}\boldsymbol{P} = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}^{T} \; ; \; {}^{C}\boldsymbol{\omega} = \begin{bmatrix} 1 & 1.75 & -1.2 \end{bmatrix}^{T}
$$

Using the consecutive homogeneous transformation matrices, obtain the values of ${}^{B}\boldsymbol{P}$, ${}^{B}\boldsymbol{\omega}$, ${}^{A}\boldsymbol{P}$, ${}^{A}\boldsymbol{\omega}$. Then, directly calculate the values of ${}^{A}\boldsymbol{P}$,${}^{A}\boldsymbol{\omega}$ from the homogeneous matrix ${}^{A}\boldsymbol{T}_{C}$ and confirm the accuracy of your calculations by comparing the results of the two methods.

2.28 Repeat Problem 2.27 for the coordinate systems shown in Figure 2.13.

2.29 Redo the task outlined in Problem 2.27 for the coordinate systems depicted in Figure 2.14.

2.30 Consider the coordinate systems shown in Figure 2.12 and assume that the vectors ${}^{A}\boldsymbol{P}$ and ${}^{A}\boldsymbol{\omega}$ are

represented as follows:

$$^{A}\boldsymbol{P} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^{T} \; ; \; ^{A}\boldsymbol{\omega} = \begin{bmatrix} 1 & -1.25 & -2.15 \end{bmatrix}^{T}$$

Using the consecutive homogeneous transformation matrices, obtain the values of $^{B}\boldsymbol{P}, ^{B}\boldsymbol{\omega}, ^{C}\boldsymbol{P}, ^{C}\boldsymbol{\omega}$. Subsequently, compute the values of $^{C}\boldsymbol{P}$ and $^{C}\boldsymbol{\omega}$ directly from the homogeneous matrix $^{C}\boldsymbol{T}_{A}$. Validate your calculations by comparing the results obtained through the two methods.

2.31 Repeat Problem 2.30 for the coordinate systems shown in Figure 2.13.

2.32 Reiterate the task presented in Problem 2.30 for the coordinate systems depicted in Figure 2.14.

# 3

# *Advanced Representations*

DOI: 10.1201/9781003590408-3

The chapter explores the advanced methods for representing spatial motion through the lens of screw theory. The chapter covers an introduction to “screw theory,” providing a mathematical framework for representing rigid body motions. This theory proves beneficial in articulating displacements, velocities, forces, and torques in three-dimensional space, rooted in the idea that rigid body motion combines rotation about a specific axis and translation along that axis.

Furthermore, this chapter addresses the representation of orientation, including screw and quaternion, and the application of screw representation in describing the motion of rigid bodies. A detailed examination of linear and angular velocities is presented next, covering aspects such as the angular velocity of a rigid body, angular velocity and its relation to the rate of a rotation matrix, rate of Euler angles, quaternion rates, and furthermore, the linear velocity of a point, and screw coordinates.

The discussion extends to linear and angular accelerations, encompassing topics like the angular acceleration of a rigid body and the linear acceleration of a point. In essence, this chapter provides a comprehensive foundation in advanced representations, shedding light on the intricacies of motion, orientation, velocity, and acceleration within the context of rigid body dynamics.

## 3.1 Introduction to Screw Theory

The exploration of the kinematics and dynamics of a rigid body takes on a new dimension when approached through the lens of screw theory. Within this framework, the principle of transference emerges as a conspicuous outcome, as highlighted by McCarthy and Soh in [59]. This principle yields notable advantages, particularly in the establishment of an analogy between geometric computations for points utilizing vectors and the analogous calculations for lines achieved through the substitution of vectors with screws.

An inherent advantage of screw theory lies in its capacity to articulate displacements, velocities, forces, and torques in three-dimensional space. This capability is rooted in the foundational idea that any motion of a rigid body can be elegantly expressed as the inseparable fusion of about a specific axis and a translation along that very axis. This axis coincides with the object or particle undergoing

displacement, providing a unifying and intuitive framework for understanding rigid body general motion. The following results in different wordings are among the highlights of such representation of rigid body motion.

> Any motion of a rigid body can be achieved by combining a rotation around a specific axis with a parallel translation along that same axis.
>
> The most general motion of a rigid body can be accomplished through a translation along a line, either followed or preceded by a rotation around that particular line.
>
> In the Euclidean space, the displacement of a rigid body can be effectively described as a rotation around a distinct axis coupled with a parallel translation along that axis.

While Mozzi and Cauchy receive credit for attaining comparable results, the recognition of the renowned theorem is specifically attributed to Michel Chasles in numerous books. To appropriately acknowledge both scientists, we denote the theorem as Mozzi–Chasles' theorem in this book.

**Theorem 3.1 (Mozzi–Chasles' Theorem (1830))** *The most general displacement for a rigid body can be generated by executing a translation along a line, known as its screw axis or Mozzi axis, either followed or preceded by a rotation about an axis parallel to that specific line.*

The decomposition of a general motion into a pure rotation and a translation predates Mozzi and Chasles. However, it seems that Mozzi was the first to conceptualize that both the rotation and the translation occurring along the same line [60]. This line, initially referred to as “asse di Mozzi” in Italian, or the Mozzi axis in early references, is now adopted as the “screw axis” in this book, aligning with the terminology widely recognized in the literature. Even today, about two hundred years later, the notion that the translation occurs along a line while the rotation is about the same line remains somewhat challenging to grasp.

In his contribution Mozzi certainly describes the general motion of a rigid body based on the concept of screw axis. Several decades later, Giorgini proved that although the results reported by Mozzi were in general accurate, the procedures Mozzi used had been partially incorrect. On the contrary, Chasles accurately states the theorem and provides a rigorous proof for it [61]. Another important contribution to this field was given by Plücker in 1866. Plücker or line coordinates, are a way to assign six homogeneous coordinates to each line in projective 3D-space denoted by $\mathbb{P}^3$. This representation proves to be very useful for the screws, twists and wrenches in the theory of kinematics used for robotics. However, despite these pioneering contributions the framework of the theory of screws is credited to Ball based on his book “A treatise on the theory of screws” [62].

As briefly outlined, screw theory boasts a rich history spanning more than two centuries. Presently, it stands as an integral component of the theoretical framework employed in robotics research, playing a pivotal role in the evolution of sophisticated robotic systems capable of intricate and precise movements. This book adeptly utilizes this compelling concept for the representation of motion, kinematics analysis, and the control of robot manipulators. However, a comprehensive exploration of the concepts and methodologies within screw theory necessitates a dedicated investment of time and energy, surpassing the scope of this book [63, 64, 65]. In this book, fundamental concepts and crucial methodologies essential for the analysis and design of serial robots are presented.

## 3.2 Orientation Representation

In the preceding chapter, the definition and characteristics of the orientation of a rigid body are introduced. The representation of orientation using a rotation matrix and Euler angles are also discussed. Although the rotation matrix is preferred for its unique features, it requires nine parameters to represent the orientation. On the other hand, Euler angles only need three parameters, but they are susceptible to gimbal lock situations and suffer from the regeneracy of their inverse problem solution.

In this chapter, we will explore two other well-known representations of orientation. The first representation employs the theory of screws to uniquely represent the orientation of a rigid body with only four parameters. The second representation also utilizes four parameters but employs the concept of unit quaternions to represent the motion. Subsequent sections will provide detailed explanations of these representations.

### 3.2.1 Screw Representation

As expressed in the properties of the rotation matrix, the eigenvalues of the rotation matrix are 1 and $e^{\pm i\theta}$, where $\theta$ is one of the intrinsic properties of the rotation matrix that indirectly represents the amount of rotation of the rigid body. Moreover, the eigenvector corresponding to the real eigenvalue $\lambda = 1$ of the rotation matrix is also a vector with real coefficients that specifies a defined direction in three-dimensional space. This direction can be introduced as the instantaneous axis of rotation of the rigid body.

Consequently, instead of interpreting the rotation of a rigid body by a matrix or about three (fixed or moving) axes, the rotation of the body at any moment can be interpreted an instantaneous axis with an equivalent rotation angle. This representation of rotation is referred to in the robotics literature as the *equivalent angle and axis of rotation*, while the instantaneous axis of rotation is recognized as the *screw axis*. The use of the screw axis to represent the pure rotation of a rigid body has the feature

that it can be generalized to the general motion of a rigid body, which includes translational and rotational motion, using the theory of screws. This will be discussed later in Section 3.3.

The description of rotation using the equivalent angle and axes of rotation takes advantage of the physical concept of the rotation of a rigid body through a rotation matrix. If we consider the rotation matrix as a mapping of a vector from the reference coordinate system $\{\mathscr{A}\}$ to the moving coordinate system $\{\mathscr{B}\}$, rotation only involves a change in the viewing angle in this coordinate transformation and does not introduce a change in its magnitude. This property is manifested in the unit magnitude of all eigenvectors of the rotation matrix and its unit determinant. It can be demonstrated that the instantaneous axis of rotation or the screw axis is another intrinsic characteristic of rotation. For example, if we consider a rotation from one coordinate system to another around this instantaneous axis, and by an angle $\theta$, the inverse rotation mapping from the second coordinate system to the first is also expressible along this same axis but by an angle of $-\theta$.

With this introduction, the representation of the orientation of a rigid body using the rotation angle of the screw axis can be described by four parameters. In this description, the rotation angle $\theta$ is the first parameter, and the screw axis, represented by the unit vector

$$^{A}\hat{\boldsymbol{s}} = [s_x, s_y, s_z]^T$$

encompasses the remaining three parameters. It should be noted, however, that due to the unit magnitude of this vector, its three constituent parameters satisfy the following algebraic constraint:

$$\hat{\boldsymbol{s}} \cdot \hat{\boldsymbol{s}} = 1 \text{or} s_x^2 + s_y^2 + s_z^2 = 1$$

(3.1)

This constraint arises from the unit length requirement of the screw axis vector. Therefore, this representation can be expressed with the following four parameters placed in an array.

$$\{\theta, s_x, s_y, s_z\} \text{ subject to} s_x^2 + s_y^2 + s_z^2 = 1$$

(3.2)

Now, if the vector symbol $\boldsymbol{\theta}$ is used as the “screw vector” in the following form, this representation can also be interpreted with the following three parameters.

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \quad , \quad \theta_x = \theta s_x, \quad \theta_y = \theta s_y, \quad \theta_z = \theta s_z$$

(3.3)

If a rotation matrix describing the orientation of a rigid body is given, it is straightforward to obtain the four

parameters of the screw representation by calculating its eigenvalues and the corresponding unit eigenvector. To calculate the rotation angle $\theta$, it is sufficient to use the relationship given in Equation 2.18,

Although calculating the screw axis can be easily done by finding the eigenvector of the rotation matrix, we will use the *Rodrigues' Rotation Formula* here to obtain the analytical expression for the screw axis. In this formula, the relationship between the vector ${}^{A}\boldsymbol{P}$ in the reference coordinate system $\{\mathscr{A}\}$ and the vector ${}^{B}\boldsymbol{P}$ in the rotated coordinate system $\{\mathscr{B}\}$ is found by multiplying it by the vector of the equivalent rotation axis $\hat{\boldsymbol{s}}$ and the trigonometric functions of the rotation angle $\theta$ as follows [66].

$$ {}^{A}\boldsymbol{P} = (\cos\theta)\,{}^{B}\boldsymbol{P} + (\sin\theta)(\hat{\boldsymbol{s}} \times {}^{B}\boldsymbol{P}) + (1-\cos\theta)({}^{B}\boldsymbol{P} \cdot \hat{\boldsymbol{s}})\hat{\boldsymbol{s}} \tag{3.4} $$

If we represent this relationship in a conventional matrix form,

$$ {}^{A}\boldsymbol{P} = {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} \tag{3.5} $$

the rotation matrix in terms of the rotation angle $\theta$ and the screw axis ${}^{A}\hat{\boldsymbol{s}} = [s_x, s_y, s_z]^T$ is determined by the following equation [67].

$$
{}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} s_x^2 v\theta + c\theta & s_x s_y v\theta - s_z s\theta & s_x s_z v\theta + s_y s\theta \\ s_y s_x v\theta + s_z s\theta & s_y^2 v\theta + c\theta & s_y s_z v\theta - s_x s\theta \\ s_z s_x v\theta - s_y s\theta & s_z s_y v\theta + s_x s\theta & s_z^2 v\theta + c\theta \end{bmatrix}
$$

(3.6)

In which notations $s\theta = \sin\theta$, $c\theta = \cos\theta$, and $v\theta = 1 - \cos\theta$ have been used. Equation 3.6 interprets the rotation of a rigid body with the parameters of the rotation angle and the screw axis. If the rotation angle and the screw axis are given, the rotation matrix is easily and uniquely calculated from this equation. The inverse problem is also not as challenging as the Euler angles. If the rotation matrix is given, the rotation angle is determined using Equation 2.18, which is restated here for reference.

$$
\theta = \cos^{-1} \frac{\mathrm{tr}\left({}^{A}\boldsymbol{R}_{B}\right) - 1}{2}
$$

(3.7)

If the rotation angle is not zero, the components of the screw axis can also be easily calculated using the following equation.

$$
\begin{aligned}
s_x &= \frac{r_{32} - r_{23}}{2s\theta} \\
s_y &= \frac{r_{13} - r_{31}}{2s\theta} \\
s_z &= \frac{r_{21} - r_{12}}{2s\theta}
\end{aligned}
$$

(3.8)

From Equations 3.7 and 3.8, it can be understood that the rotation of a body can be represented by two sets, namely $\theta, \hat{s}$ and $-\theta, -\hat{s}$, which have the same physical nature represented by the unique screw vector $\boldsymbol{\theta}$. Furthermore, if the rotation angle becomes zero, it is natural that there is no rotation axis, and Equation 3.8 cannot be used. However, if the rotation angle becomes $\pm\pi$ radians, a singular case occurs, and it is not possible to obtain the rotation axis from Equation 3.8. Similar to a gimbal lock situation in Euler angle representation, in this case, the rotation matrix simplifies, and the components of the screw vector can be obtained using the diagonal elements of the matrix with the following equation.

$$
s_x^2 = 1 + r_{11} \ , s_y^2 = 1 + r_{22} \ , s_z^2 = 1 + r_{33}.
$$

(3.9)

**Example 3.1 (Screw Representation)** ⏎ *Assume, similar to what we had in Example 2.1, the moving*

*coordinate system $\{\mathscr{B}\}$ attached to a rigid body rotates only around one of the principal axes of the reference coordinate system $\{\mathscr{A}\}$ with the following angles, as illustrated in Figure 2.4:*

***A)*** *Rotation around the x-axis by an angle $\alpha$.*

***B)*** *Rotation around the y-axis by an angle $\beta$.*

***C)*** *Rotation around the z-axis by an angle $\gamma$.*

*Represent these rotations using the screw vector $\boldsymbol{\theta}$.*

**Solution:** The screw parameters in these three cases can be obtained using Equation 3.7 and 3.8 in the following form.

**A)** For the first case, we have:

$$\begin{aligned}
&\theta = \cos^{-1}\left((1 + 2\cos\alpha - 1)/2\right) = \alpha \\
&s_x = (s\alpha - (-s\alpha))/2s\theta = 1 \\
&s_y = s_z = 0
\end{aligned}$$

Therefore, as expected,

$$\boldsymbol{\theta}_A = \alpha[1 \quad 0 \quad 0]^T$$

**B)** Similarly, for the second case, we can derive:

$$\boldsymbol{\theta}_B = \beta[0 \quad 1 \quad 0]^T$$

**C)** And for the third case:

$$\boldsymbol{\theta}_C = \gamma[0 \quad 0 \quad 1]^T$$

The correctness of the calculated results in this example can easily be verified and confirmed with the intuitive understanding of the rotation angle and screw axis.

**Example 3.2** ⏎ *Consider the rotation of a rigid body around the axis* $\hat{\boldsymbol{s}} = \frac{1}{\sqrt{3}}[1 \quad 1 \quad 1]^T$ *with an angle* $\theta = \pi/6$. *Find the rotation matrix in this case.*

**Solution:** The rotation matrix in this case is obtained using Equation 3.6 as follows.

$$^A\boldsymbol{R}_B = \begin{bmatrix} 0.911 & -0.244 & 0.333 \\ 0.333 & 0.911 & -0.244 \\ -0.244 & 0.333 & 0.911 \end{bmatrix}$$

To validate the calculations, we utilize the eigenvalues and real eigenvectors of the rotation matrix. The eigenvalues of the rotation matrix $^A\boldsymbol{R}_B$ are $\lambda_{1,2,3} = 1, e^{\pm i\pi/6}$, and its real eigenvector is $\boldsymbol{v}_1 = [0.577 \quad 0.577 \quad 0.577]^T$ which precisely corresponds to the screw axis. Thus, the accuracy of the calculations is verified.

**Example 3.3** *Consider the rotation matrix of Example 2.2, which is in the form of*

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.933 & -0.067 & -0.354 \\ -0.067 & 0.933 & -0.354 \\ 0.354 & 0.354 & 0.866 \end{bmatrix} $$

*Obtain the screw representation of this rotation.*

**Solution:** The rotation of this matrix, using Equation 3.7, is equal to $\theta = \pi/6$. Additionally, the components of the equivalent rotation axis, using Equation 3.8, are equal to:

$$ \hat{\boldsymbol{s}} = \begin{bmatrix} -0.707 & 0.707 & 0 \end{bmatrix}^{T} $$

Thus, the screw vector in this rotation is equal to:

$$ \boldsymbol{\theta} = \begin{bmatrix} -0.37 & 0.37 & 0 \end{bmatrix}^{T}. $$

### 3.2.2 Quaternion

One of the applications of complex numbers is the representation of planar rotations on a unit circle. The unit quaternion, by extending this concept and using complex numbers, can similarly represent rotations in three-

dimensional space. Quaternions were first introduced by Hamilton in 1843. Hamilton's primary motivation was to find an algebraic representation for points $(x, y, z)$ in three-dimensional space through extending complex numbers. Initially, he sought a three-dimensional hyper-complex system with suitable properties [68]. For over a decade, Hamilton attempted to extend this concept by defining a triad consisting of a real part and two imaginary parts. Eventually, he realized that, for this purpose, he needed three imaginary units with the following properties instead of two [69].

$$i^2 = j^2 = k^2 = -1 \tag{3.10}$$

$$ij = k \qquad ji = -k \tag{3.11}$$

$$i \to j \to k \to i \tag{3.12}$$

where, Equation (3.12) describes a rotational cyclic permutation. Hamilton referred to the resulting four-element structure as a Quaternion.

$$Q = q_1 i + q_2 j + q_3 k + q_4$$

(3.13)

In the representation of Quaternion, unlike Euler angle representation, four numbers are used instead of three to represent rotation. The use of Quaternion in this context is one of the most common methods for representing the orientation of a rigid body in space.

Assume that the rotation of a rigid body is described by four parameters introduced in the screw representation by the rotation angle and screw axis $\{\theta,\ s_x,\ s_y,\ s_z\}$. With this assumption, by using the quaternion definition given in Equation 3.13, the *Euler Parameters* are defined by the following four parameters.

$$\epsilon_1 = s_x \sin(\theta/2);\ \ \epsilon_2 = s_y \sin(\theta/2);\ \ \epsilon_3 = s_z \sin(\theta/2);\ \ \epsilon_4 = \cos$$

(3.14)

If the Euler parameters are represented in a $4 \times 1$ array as follows,

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 & \epsilon_2 & \epsilon_3 & \epsilon_4 \end{bmatrix}^T$$

(3.15)

This array forms a unit quaternion because its Euclidean norm is equal to one.

$$\| \boldsymbol{\epsilon} \|_2^2 = \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \epsilon_4^2 = 1$$

(3.16)

The rotation matrix can be calculated using the components of the unit quaternion as follows [70].

$$
{}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 1-2\epsilon_2^2-2\epsilon_3^2 & 2(\epsilon_1\epsilon_2-\epsilon_3\epsilon_4) & 2(\epsilon_1\epsilon_3+\epsilon_2\epsilon_4) \\ 2(\epsilon_1\epsilon_2+\epsilon_3\epsilon_4) & 1-2\epsilon_1^2-2\epsilon_3^2 & 2(\epsilon_2\epsilon_3-\epsilon_1\epsilon_4) \\ 2(\epsilon_1\epsilon_3-\epsilon_2\epsilon_4) & 2(\epsilon_2\epsilon_3+\epsilon_1\epsilon_4) & 1-2\epsilon_1^2-2\epsilon_2^2 \end{bmatrix}
$$

(3.17)

Furthermore, it can be shown that by solving the inverse problem, if the rotation matrix is given, the components of the unit quaternion can be determined as follows.

$$
\begin{aligned}
\epsilon_1 &= \frac{(r_{32}-r_{23})}{4\epsilon_4} \\
\epsilon_2 &= \frac{(r_{13}-r_{31})}{4\epsilon_4} \\
\epsilon_3 &= \frac{(r_{21}-r_{12})}{4\epsilon_4} \\
\epsilon_4 &= \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}
\end{aligned}
$$

(3.18)

Although the unit quaternion representation does not directly inherit the physical concept of rotation as seen in the screw representation, it is less singular compared to it. In a sense, if the rotation angle is zero, the unit quaternion

will be as expected, equal to $\boldsymbol{\epsilon} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$. The singular case in quaternion only occurs when the rotation angle is $\pm\pi$, a situation that is avoided in quaternion-based measurement systems.

**Example 3.4 (Rotation around principal axes)** *Represent the rotation around the principal axes, as introduced in Example 2.1, and further analyzed in Example 3.1, using quaternions.*

**Solution:** The representation of quaternions in three rotation cases around the principal axes can be obtained using Equation 3.14 as follows.

**a)** For the first case, we have,

$$\boldsymbol{\epsilon} = \begin{bmatrix} \sin(\alpha/2) & 0 & 0 & \cos(\alpha/2) \end{bmatrix}^T \tag{3.19}$$

**b)** Similarly, for the second case, we can write,

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0 & \sin(\beta/2) & 0 & \cos(\beta/2) \end{bmatrix}^T \tag{3.20}$$

**c.** Finally, for the third case:

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0 & 0 & \sin(\gamma/2) & \cos(\gamma/2) \end{bmatrix}^T$$

(3.21)

**Example 3.5** *Consider the rotation of a rigid body using the quaternion representation below.*

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.149 & 0.149 & 0.149 & 0.966 \end{bmatrix}^T \tag{3.22}$$

*Find the rotation matrix in this case.*

**Solution:** The rotation matrix in this case is calculated as:

$$^A\boldsymbol{R}_B = \begin{bmatrix} 0.911 & -0.244 & 0.333 \\ 0.333 & 0.911 & -0.244 \\ -0.244 & 0.333 & 0.911 \end{bmatrix}$$

It is observed that this rotation matrix is equal to what has been calculated in Example 3.2. To validate the calculations, compare the quaternion in this example with the screw representation in Example 3.2. Evaluating these two representations reveals that both depict the same rotation angle and the direction of the screw axis, confirming the accuracy of the calculations.

**Example 3.6** *Consider the rotation matrix given in Example 2.2. Find the quaternion representation of this rotation.*

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.933 & -0.067 & -0.354 \\ -0.067 & 0.933 & -0.354 \\ 0.354 & 0.354 & 0.866 \end{bmatrix} $$

**Solution:** The quaternion corresponding to this matrix can be obtained using the Equation 3.18 as follows.

$$ \boldsymbol{\epsilon} = \begin{bmatrix} -0.183 & 0.183 & 0 & 0.966 \end{bmatrix}^{T} $$

## 3.3 Screw Representation for Motion

As explained earlier, the Mozzi–Chasles's theorem states that the general motion of a rigid body can be obtained by combining its instantaneous rotational motion about an axis and its translational motion along the same axis. If we accept this claim, given that the direction of translational motion is aligned with the rotational axis, the use of the term “Screw” becomes more natural. This is because, as seen in Figure 3.2, considering the motion of a screw, depending on the pitch of the screw, the rotational motion

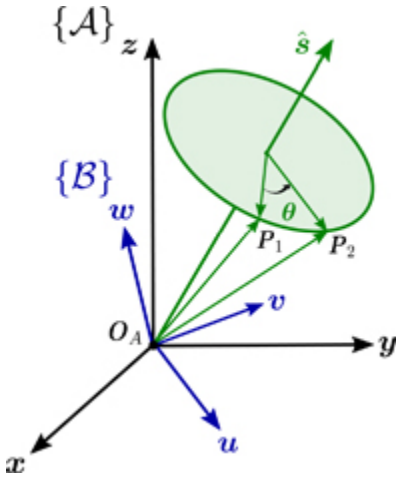
of the screw around its axis induces translational motion along the same axis on the nut.

The first part of this theorem, which deals with the pure rotational motion of a body, is discussed in Section 3.2.1 with the introduction of the rotation angle and the screw axis. However, the second part of this theorem, which interprets translational motion along the same axis, is by no means intuitive and requires further investigation, which we address in this section.

Consider the general motion of a rigid body in space as shown in Figure 3.2. As observed in this figure, the rigid body has undergone a translational motion represented by the vector ${}^{A}\boldsymbol{P}_{O_B}$ and a pure rotational motion described by the angle $\theta$ and the screw axis $\hat{\boldsymbol{s}}$, starting from its initial state represented by the coordinate system $\{\mathscr{A}\}$. The final state of the rigid body is depicted with the coordinate system $\{\mathscr{B}\}$, and an intermediate coordinate system $\{\mathscr{A}'\}$, shown with the dashed line, is used to separate translational and rotational motions from each other. As seen in the figure, the origin of the intermediate coordinate system which is parallel to $\{\mathscr{A}\}$, coincides with the origin of the $\{\mathscr{B}\}$ coordinate system. Thus, this coordinate system only represents the translational motion of the body along the vector ${}^{A}\boldsymbol{P}_{O_B}$.

First, consider only the rotational motion of the rigid body, which is described by two coordinate systems $\{\mathscr{A}'\}$ and $\{\mathscr{B}\}$, respectively. As detailed in Section 3.2.1 and depicted in Figure 3.1, pure rotational motion of a rigid

body can be represented by the rotation angle $\theta$ about the screw axis $\hat{\boldsymbol{s}}$. In this representation, the rotation angle is denoted by $\theta$, and the screw axis is denoted by $\hat{\boldsymbol{s}}$, while the screw vector has been interpreted as $\boldsymbol{\theta} = \theta\hat{\boldsymbol{s}}$. If the rotation matrix is given, the rotational motion of the rigid body can be easily represented with four parameters $\{\theta, s_x, s_y, s_z\}$, calculated from Equations 3.7 and 3.8.

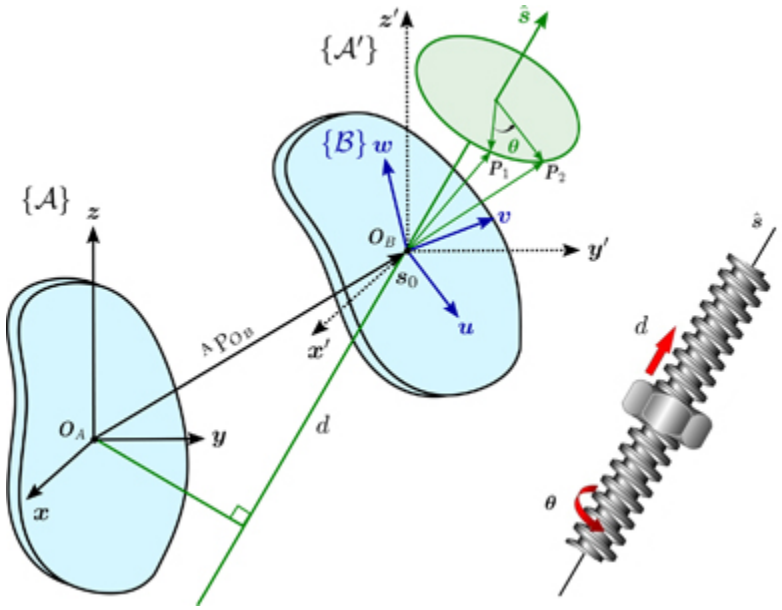



▶ Long Description for Figure 3.1

**FIGURE 3.1**
The pure rotational motion of a rigid body represented about a screw axis. ⏎

According to the Mozzi–Chasles's theorem, this representation can be extended to general motion in space to interpret translational motion along the same axis. Unlike pure rotational motion where the origins of the two coordinate systems coincide, in the case of translational motion, the origins of the $\{\mathscr{A}\}$ and $\{\mathscr{B}\}$ coordinate systems do not coincide and have a distance. This distance can be

expressed using the *position vector of the origin of the moving coordinate system* $O_B$ *with respect to the reference coordinates* $\{\mathscr{A}\}$, represented with the vector $\boldsymbol{s}_o$ in the figure. In the homogeneous transformation matrix representation, this distance is interpreted by the vector ${}^{A}\boldsymbol{P}_{O_B}$. However, considering what is shown in Figure 3.2, if we project this vector onto the screw axis with a scalar parameter $d$, it can be interpreted by the screw pitch as the physical interpretation of the motion by the screw.




▶ Long Description for Figure 3.2

**FIGURE 3.2**
The general motion of a rigid body represented by a screw.
⏎

In this way, the motion of a rigid body in space, consisting of rotational and translational motion, can be

described with a set of two scalars $\theta$ and $d$, and two vectors $\hat{\boldsymbol{s}}$ and $\boldsymbol{s}_o$.

$$\left\{\theta \ , \ \hat{\boldsymbol{s}} = [s_x, s_y, s_z]^T \quad ; \quad d \ , \ \boldsymbol{s}_o = [s_{o_x}, s_{o_y}, s_{o_z}]^T\right\} \tag{3.23}$$

Despite the use of eight parameters in this representation, considering the two algebraic constraints:

$$\hat{\boldsymbol{s}}^T \hat{\boldsymbol{s}} = 1 \quad ; \quad \boldsymbol{s}_o^T \hat{\boldsymbol{s}} = d \tag{3.24}$$

as expected, six independent parameters are sufficient to represent motion in three-dimensional space.

If the homogeneous transformation matrix is given, one can easily obtain all eight parameters of the screw representation. The rotation parameters $\theta$ and $\hat{\boldsymbol{s}}$ can be calculated using the relationships 3.7 and 3.8, which are reiterated here for reference.

$$\theta = \cos^{-1} \frac{\operatorname{tr}\left({}^A\boldsymbol{R}_B\right) - 1}{2}$$

$$s_x = \frac{r_{32} - r_{23}}{2s\theta}$$

$$s_y = \frac{r_{13} - r_{31}}{2s\theta}$$

$$s_z = \frac{r_{21} - r_{12}}{2s\theta}$$

(3.25)

The translational motion parameters can also be easily obtained using the following relationship.

$$\boldsymbol{s}_o = {}^{A}\boldsymbol{P}_{O_B} \quad ; \quad d = \boldsymbol{s}_o^T \hat{\boldsymbol{s}}$$

(3.26)

The solution to the inverse problem can also be presented by performing the following calculations [8]. If the eight parameters represent the screw representation are given,

$$\left\{\theta, \hat{\boldsymbol{s}} = [s_x, s_y, s_z]^T \quad ; \quad d, \boldsymbol{s}_o = [s_{o_x}, s_{o_y}, s_{o_z}]^T\right\}$$

(3.27)

then the homogeneous transformation matrix is obtained using the following relationship.

$${}^{A}\boldsymbol{T}_B = \left[\begin{array}{ccc|c} s_x^2 v\theta + c\theta & s_x s_y v\theta - s_z s\theta & s_x s_z v\theta + s_y s\theta & p_x \\ s_y s_x v\theta + s_z s\theta & s_y^2 v\theta + c\theta & s_y s_z v\theta - s_x s\theta & p_y \\ s_z s_x v\theta - s_y s\theta & s_z s_y v\theta + s_x s\theta & s_z^2 v\theta + c\theta & p_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

(3.28)

which in this case

$$
\begin{aligned}
p_x &= d\,s_x - s_{o_x}(s_x^2 - 1)v\theta - s_{o_y}(s_x s_y v\theta - s_z s\theta) - s_{o_z}(s_x s_z v\theta - \\
p_y &= d\,s_y - s_{o_x}(s_y s_x v\theta + s_z s\theta) - s_{o_y}(s_y^2 - 1)v\theta - s_{o_z}(s_y s_z v\theta - \\
p_z &= d\,s_z - s_{o_x}(s_z s_x v\theta - s_y s\theta) - s_{o_y}(s_z s_y v\theta + s_x s\theta) - s_{o_z}(s_z^2
\end{aligned}
\tag{3.29}
$$

Similarly, here, the shorthand notations $s\theta = \sin\theta$, $c\theta = \cos\theta$, and $v\theta = 1 - \cos\theta$ have been used.

**Example 3.7 (Motion Representation with Screw)** *Assume that the motion of a rigid body is represented by the following homogeneous transformation matrix. Obtain the parameters of this motion based on the screw representation.*

$$
{}^{A}\boldsymbol{T}_{B} = \left[\begin{array}{ccc|c}
0.750 & 0.612 & -0.250 & 1.414 \\
-0.612 & 0.500 & -0.612 & -1.576 \\
-0.250 & 0.612 & 0.750 & -1.414 \\
\hline
0 & 0 & 0 & 1
\end{array}\right]
$$

**Solution:** The parameters of rotational motion, $\theta$ and $\hat{\boldsymbol{s}}$, in the screw representation are obtained using the relationship 3.25.

$$
\left\{\theta = \frac{\pi}{3} \quad ; \quad \hat{\boldsymbol{s}} = \frac{1}{\sqrt{2}}[1, 0, -1]^T\right\}
$$

The translational motion parameters of the screw, represented by the symbols $d$ and $\boldsymbol{s}_o$, can also be calculated using the relationship 3.26 as follows.

$$\left\{d = 2 \quad ; \quad \boldsymbol{s}_o = [1.414, -1.576, -1.414]^T\right\}$$

**Example 3.8** *Suppose the general motion of a rigid body is represented by the screw representation with the following parameters.*

$$\left\{\theta = \frac{\pi}{4}, \hat{\boldsymbol{s}} = \frac{1}{\sqrt{6}}[1, -2, -1]^T \quad ; \quad d = 0.5, \boldsymbol{s}_o = [0, 0, 1]^T\right\}$$

*Represent this motion in space using the homogeneous transformation matrix.*

**Solution:** In this example, Equation 3.28 is used for determining the homogeneous transformation matrix.

$${}^A\boldsymbol{T}_B = \left[\begin{array}{ccc|c} 0.756 & 0.191 & -0.626 & 0.830 \\ -0.386 & 0.902 & -0.191 & -0.217 \\ 0.529 & 0.386 & 0.756 & 0.040 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

## 3.4 Linear and Angular Velocities

In this section, we elaborate the linear and angular velocity quantities in the motion of a rigid body. The linear velocity vector of a point on a rigid body is determined by the derivative of its position vector with respect to time. If this differentiation is performed with respect to the reference coordinate system, the absolute velocity of that point is obtained. This is in contrast to deriving the relative position vector of a point on the body, which is measured with respect to the moving coordinate system, to determine its relative velocity. In robotics, establishing the relationship between the absolute velocity of a point on a body in space and its relative velocity holds special significance, and this is the focus of this section.

On the other hand, the angular velocity quantity falls into the category of free vectors, which is defined for the entire body. Angular velocity represents the rate of change of the rotation vector with respect to time. Considering that the orientation of a body is represented in various ways, determining the angular velocity vector of a body and its relationship with the time derivatives of variables defining the orientation in a rigid body, such as the rotation matrix, Euler angles, screw axis, and quaternion, is a topic that we will explore further. It is natural that representing the angular velocity vector of a body in certain representations offers more convenience, and using them in the analysis of robotic systems is preferable.

### 3.4.1 Angular Velocity of a Rigid Body

In order to define the angular velocity of a rigid body, as before, we use a coordinate system attached to it, denoted as the moving frame $\{\mathscr{B}\}$. The analysis of the body's angular velocity, as shown in Figure 3.3, is performed by examining the relative angular motion of this moving coordinate system with respect to the reference coordinate system $\{\mathscr{A}\}$. The angular velocity, denoted by $\boldsymbol{\Omega}$, is a quantity related to the entire rigid body, and its magnitude is the same for different points on the body, making it a free vector. The direction of the angular velocity vector at any time indicates the instantaneous axis of rotation of the moving coordinate system relative to the fixed coordinate system.

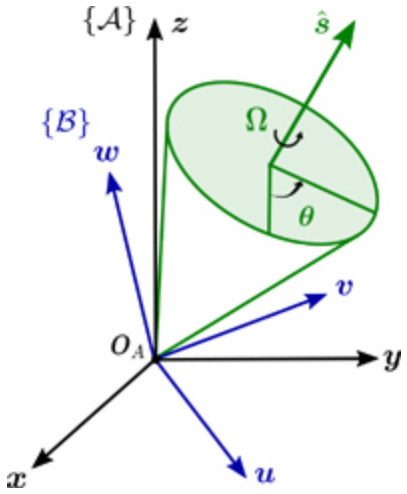



**FIGURE 3.3**

The representation of the instantaneous axis of rotation of a rigid body and the angular velocity vector. ⏎

If you recall the definition of the screw axis, it is evident that the instantaneous rotation of the rigid body at each instance occurs around the screw axis. Additionally, the magnitude of the angular velocity vector at any moment could be determined by the time rate of change of the rotation angle $\theta$. With this explanation, the angular velocity vector of a rigid body can be defined as follows.

$$\boldsymbol{\Omega} \triangleq \dot{\theta}\,\hat{\boldsymbol{s}}$$

(3.30)

Given this definition, it becomes clear that employing the screw vector $\boldsymbol{\theta}$ in the orientation representation results in the body's angular velocity precisely corresponding to the time derivative of this vector, expressed as $\boldsymbol{\Omega} = \dot{\boldsymbol{\theta}}$. This is the reason why utilizing the screw representation in the analysis of robotic systems is more straightforward, and therefore preferable, compared to alternative approaches.

Note that the angular velocity is a vector that can be represented in any coordinate frame. As before, the superscript before the vector is used to indicate the coordinate system in which the angular velocity is represented. For example, the vector ${}^{B}\boldsymbol{\Omega}$ represents the angular velocity of the body represented in the $\{\mathscr{B}\}$ coordinate system. The angular velocity vector can be represented in any coordinate system with its components as follows.

$$
\begin{aligned}
{}^{A}\boldsymbol{\Omega} &= \Omega_x \hat{\boldsymbol{x}} + \Omega_y \hat{\boldsymbol{y}} + \Omega_z \hat{\boldsymbol{z}} \\
&= \dot{\theta} \left( s_x \hat{\boldsymbol{x}} + s_y \hat{\boldsymbol{y}} + s_z \hat{\boldsymbol{z}} \right)
\end{aligned}
\tag{3.31}
$$

In this representation, $\Omega_x, \Omega_y, \Omega_z$ are the respective components of the angular velocity vector of the body, expressed in the $\{\mathscr{A}\}$ coordinate system and along the axes $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$.

### 3.4.2 Angular Velocity and the Rate of Rotation Matrix

As previously discussed, the rotation matrix serves as one of the most versatile approaches to represent the orientation of a body, facilitating the establishment of connections between diverse representations of the body's orientation. In this section, we will explore the connection between the derivative of the rotation matrix and the angular velocity vector of the body. Unfortunately, unlike the screw representation, where rotational motion is depicted with the vector $\boldsymbol{\theta}$, the rotation matrix representation conveys this concept using a matrix. It is evident that its time derivative will not be directly equivalent to the angular velocity of the body. Within this section, we will derive the mathematical relationship between the derivative of the rotation matrix and the angular velocity vector of the body.

The rotation matrix is an orthogonal matrix, meaning its transpose is equal to its inverse. Therefore,

$$ {}^{A}\boldsymbol{R}_{B}\ {}^{A}\boldsymbol{R}_{B}^{T} = \boldsymbol{I} \tag{3.32} $$

in which, $\boldsymbol{I}$ is the $3 \times 3$ identity matrix. By differentiating both sides of this equation with respect to time,

$$ \dot{}{}^{A}\boldsymbol{R}_{B}\ {}^{A}\boldsymbol{R}_{B}^{T} + {}^{A}\boldsymbol{R}_{B}\ {}^{A}\dot{\boldsymbol{R}}_{B}^{T} = \boldsymbol{0} \tag{3.33} $$

and by substituting ${}^{A}\boldsymbol{R}_{B}^{T} = {}^{A}\boldsymbol{R}_{B}^{-1}$ and ${}^{A}\boldsymbol{R}_{B} = ({}^{A}\boldsymbol{R}_{B}^{-1})^{T}$, we can arrive at the following relationship.

$$ \left(\dot{}{}^{A}\boldsymbol{R}_{B}\ {}^{A}\boldsymbol{R}_{B}^{-1}\right) + \left(\dot{}{}^{A}\boldsymbol{R}_{B}\ {}^{A}\boldsymbol{R}_{B}^{-1}\right)^{T} = \boldsymbol{0} \tag{3.34} $$

From this relationship, it can be concluded that the matrix $\dot{}{}^{A}\boldsymbol{R}_{B}\ {}^{A}\boldsymbol{R}_{B}^{-1}$ is a $3 \times 3$ skew-symmetric matrix. Therefore, it can be represented in the form of a general skew-symmetric matrix denoted as $\boldsymbol{\Omega}^{\times}$ as follows.

$$ \boldsymbol{\Omega}^{\times} \equiv \dot{}{}^{A}\boldsymbol{R}_{B}\ {}^{A}\boldsymbol{R}_{B}^{-1} = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} $$

(3.35)

It can be shown that the components used in the representation of this skew-symmetric matrix are the same as the components of the angular velocity vector of the body, as introduced in Equation 3.4.1 [71]. If we simplify this relationship in terms of the derivative of the rotation matrix, we will arrive at the following final equation.

$$ {}^{A}\dot{\boldsymbol{R}}_{B} = \boldsymbol{\Omega}^{\times} \ {}^{A}\boldsymbol{R}_{B} $$

(3.36)

### 3.4.3 Angular Velocity and Rate of Euler Angles

In this section, we will explore the correlation between the angular velocity vector of a rigid body and the time derivatives of its Euler angle components. If we express an array containing each type of Euler angle as $[\alpha, \beta, \gamma]^{T}$, this array does not qualify as a vector due to its failure to adhere to the properties of vector addition and commutativity on finite rotation angles. It is worth noting that for none of the Euler angle representations, including the ones introduced in Section 2.2.4, the time rates of change of these angles are not equal to the angular velocity vector.

$$\boldsymbol{\Omega} \neq \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \tag{3.37}$$

It can be demonstrated that the relationship between the time derivative of this array and the angular velocity vector of the body is determined by the matrix $\boldsymbol{E}(\alpha, \beta, \gamma)$, which needs to be specified in the motion analysis for each type of Euler angle.

$$\boldsymbol{\Omega} = \boldsymbol{E}(\alpha, \beta, \gamma) \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \tag{3.38}$$

To determine this matrix, one can use the relationship given in Equation 3.35. For this purpose, it is sufficient to obtain the rotation matrix associated with the desired Euler angles and differentiate it with respect to time. Subsequently, Equation 3.35 can be simplified using the components of the rotation matrix and its derivatives.

$$\begin{aligned} \Omega_x &= \dot{r}_{31} r_{21} + \dot{r}_{32} r_{22} + \dot{r}_{33} r_{23} \\ \Omega_y &= \dot{r}_{11} r_{31} + \dot{r}_{12} r_{32} + \dot{r}_{13} r_{33} \\ \Omega_z &= \dot{r}_{21} r_{11} + \dot{r}_{22} r_{12} + \dot{r}_{23} r_{13} \end{aligned}$$

(3.39)

Here, $r_{ij}$ represents the $(i, j)$ component of the rotation matrix associated with the Euler angles of interest. By performing these calculations and simplifying them, the matrix $\boldsymbol{E}(\alpha, \beta, \gamma)$ can be determined. As an example, consider the $\boldsymbol{w} - \boldsymbol{v} - \boldsymbol{w}$ Euler angles, whose rotation matrix is given in Equation 2.33 as follows.

$$\boldsymbol{R}_{wvw}(\alpha, \beta, \gamma) = \begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix}$$

Taking the derivative of this matrix with respect to time, employing the relationship in 3.39, and simplifying and categorizing the result in terms of the variables $\dot{\alpha}, \dot{\beta}, \dot{\gamma}$, we obtain the following result.

$$\boldsymbol{\Omega} = \boldsymbol{E}_{wvw} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \quad ; \boldsymbol{E}_{wvw} = \begin{bmatrix} 0 & -s\alpha & c\alpha s\beta \\ 0 & c\alpha & s\alpha s\beta \\ 1 & 0 & c\beta \end{bmatrix}$$

(3.40)

### 3.4.4 Angular Velocity and Rate of Quaternion

In this section, we will explore the connection between the angular velocity vector of a rigid body and the time

derivatives of its unit quaternion components. While quaternions utilize parameters associated with the screw vector, they do not directly incorporate the physical concept of the rotation vector. This is due to the specific arrangement chosen in their definition, as introduced in Equation 3.14. Consequently, the relationship between the angular velocity and the time derivatives of the quaternion components, represented by the four-dimensional array $\dot{\boldsymbol{\epsilon}} = \left[\dot{\epsilon}_1, \dot{\epsilon}_2, \dot{\epsilon}_3, \dot{\epsilon}_4\right]^T$, is not as straightforward as the screw representation.

For an in-depth examination of this relationship, quaternion algebra must be employed. Here, we introduce some key relationships commonly used in the literature on this subject. However, for further details, it is recommended to review specialized references in this field such as [72]. The relationship between the body's angular velocity and the derivative of its quaternion components can be obtained as follows [73]:

$$\boldsymbol{\Omega} = 2\boldsymbol{\epsilon}^{\dagger} \odot \dot{\boldsymbol{\epsilon}} \tag{3.41}$$

In this relation homogeneous coordinates $\boldsymbol{\Omega} = \begin{bmatrix}\Omega_x & \Omega_y & \Omega_z & 0\end{bmatrix}^T$ are used to represent the vector of angular velocity. Furthermore, the symbol $\odot$ is employed in this equation to denote quaternion multiplication. To define this multiplication, consider two quaternions $\boldsymbol{a}, \boldsymbol{b}$

decomposed into its vector and scalar components as follows:

$$\boldsymbol{a} = \begin{bmatrix} \mathbf{a} \\ a \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \mathbf{b} \\ b \end{bmatrix} \tag{3.42}$$

In this representation $\mathbf{a}, \mathbf{b}$ are the first three-dimensional arrays of unit quaternions, while $a$,$b$ correspond to the last scalar components. Then, according to the definition, the product of these two quaternions can be obtained from the following equation [73]:

$$\boldsymbol{a} \odot \boldsymbol{b} = \begin{bmatrix} a\boldsymbol{I} + \mathbf{a}^{\times} & \mathbf{a} \\ -\mathbf{a}^{T} & a \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ b \end{bmatrix} \tag{3.43}$$

Furthermore, in Equation 3.41, the notation $(\cdot)^{\dagger}$ is also used to represent the quaternion conjugate, which is defined by the following equation

$$\boldsymbol{a}^{\dagger} = \begin{bmatrix} \mathbf{a} \\ a \end{bmatrix}^{\dagger} = \begin{bmatrix} -\mathbf{a} \\ a \end{bmatrix} \tag{3.44}$$

When utilizing unit quaternions to denote the rotation of a body, the quaternion conjugate signifies a rotation around

the principal quaternion axis (or the screw axis) with a negative rotation angle ($-\theta$). Consequently, it can serve as the inverse quaternion in the representation of the pure rotation of a body.

With this concise overview of quaternion algebra, the connection between the angular velocity of a body and the time derivative of quaternion components, as defined by Equation 3.41, can be streamlined through some calculations as follows.

$$\begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} = 2\,\boldsymbol{E}(\boldsymbol{\epsilon})\,\dot{\boldsymbol{\epsilon}} \tag{3.45}$$

in which,

$$\boldsymbol{E}(\boldsymbol{\epsilon}) = \begin{bmatrix} \epsilon_4 & \epsilon_3 & -\epsilon_2 & -\epsilon_1 \\ -\epsilon_3 & \epsilon_4 & \epsilon_1 & -\epsilon_2 \\ \epsilon_2 & -\epsilon_1 & \epsilon_4 & -\epsilon_3 \end{bmatrix} \tag{3.46}$$

In this equation, $\boldsymbol{\epsilon} = \left[\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\right]^T$ denotes the quaternion for rotational motion, and $\dot{\boldsymbol{\epsilon}}$ represents its time derivative. As evident, the connection between the angular velocity of the body and the time derivative of the quaternion is established using a $3 \times 4$ matrix. Although this introduces

additional complexity compared to the screw representation, it is considerably less complex than the rates in Euler angles.

The solution to the inverse of this representation can be achieved using the following equation.

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\boldsymbol{\epsilon} \odot \boldsymbol{\Omega}$$

(3.47)

In this equation, $\boldsymbol{\Omega} = \begin{bmatrix}\Omega_x & \Omega_y & \Omega_z & 0\end{bmatrix}^T$ signifies the angular velocity represented in homogeneous coordinates. Utilizing quaternion algebra, the equation can be alternatively expressed in terms of the components of the quaternion array as follows.

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\,\boldsymbol{E}^T(\boldsymbol{\epsilon}) \begin{bmatrix}\Omega_x \\ \Omega_y \\ \Omega_z\end{bmatrix}$$

(3.48)

In this equation, the relationship between the rate of change of quaternions and the angular velocity of the body is established using a $4 \times 3$ matrix, namely $\boldsymbol{E}^T(\boldsymbol{\epsilon})$. Remarkably, this matrix is the transpose of the one introduced in Equation 3.46.

### 3.4.5 Linear Velocity of a Point

The linear velocity vector of point $P$ is easily obtained from the time derivative of the position vector $\boldsymbol{P}$. A noteworthy point in this equation is that if the representation of the position vector is in the fixed reference frame, then its time derivative determines the absolute linear velocity vector of that point.

$$\boldsymbol{v}_p = \dot{\boldsymbol{P}} = \left( \frac{d\boldsymbol{P}}{dt} \right)_{fix} \tag{3.49}$$

To emphasize the differentiation of the position vector with respect to the fixed coordinate system, the notation $\left(d(\,\cdot\,)/dt\right)_{fix}$ is used in this equation. If the changes in the position vector are measured with respect to the moving coordinate system, then the relative linear velocity vector will be obtained. To differentiate this type of derivation, the notation $\left(\partial(\,\cdot\,)/\partial t\right)_{mov}$ is used intentionally to differentiate it from absolute derivatives.

$$\boldsymbol{v}_{rel} = \left( \frac{\partial \boldsymbol{P}}{\partial t} \right)_{mov} \tag{3.50}$$

In this equation, $\boldsymbol{v}_{rel}$ represents the relative velocity of point $P$ as observed from the moving coordinate system

$\{\mathscr{B}\}$.

In classical mechanics, the relationship between the derivative of a quantity in the fixed coordinate system $\{\mathscr{A}\}$, denoted by $(d(\,\cdot\,)/dt)_{fix}$, and its derivative in the moving coordinate system $\{\mathscr{B}\}$, denoted by $(\partial(\,\cdot\,)/\partial t)_{mov}$, is given by the following equation [74].

$$\left(\frac{d(\,\cdot\,)}{dt}\right)_{fix} = \left(\frac{\partial(\cdot)}{\partial t}\right)_{mov} + \ \boldsymbol{\Omega} \times (\,\cdot\,) \tag{3.51}$$

The link between these two types of vector derivatives is provided by the vector $\boldsymbol{\Omega}$. This vector represents the angular velocity of the moving coordinate system $\{\mathscr{B}\}$ with respect to the fixed coordinate system $\{\mathscr{A}\}$. Additionally, the notation $\boldsymbol{\Omega} \times (\,\cdot\,)$ denotes the cross product of the angular velocity vector $\boldsymbol{\Omega}$ with the desired quantity $(\,\cdot\,)$. Since matrix multiplication is commonly used in robotics literature, the cross-product term in Equation 3.51 can be expressed using the skew-symmetric matrix $\boldsymbol{\Omega}^{\times}$.

$$\boldsymbol{\Omega}^{\times} = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \tag{3.52}$$

This notation was previously defined in Equation 3.35. By substituting it, the matrix form of Equation 3.51 can be written as follows:

$$\left( \frac{d( \cdot )}{dt} \right)_{fix} = \left( \frac{\partial (\cdot)}{\partial t} \right)_{mov} + \boldsymbol{\Omega}^{\times} ( \cdot )$$

(3.53)

Using this notation, it is straightforward to obtain the relationship between the absolute and relative derivatives of matrix quantities as well. For example, the derivative of the rotation matrix can be easily expressed using Equation 3.53 as follows:

$$\left( \frac{d( {}^{A}\boldsymbol{R}_{B})}{dt} \right)_{fix} = \left( \frac{\partial ( {}^{A}\boldsymbol{R}_{B})}{\partial t} \right)_{mov} + \boldsymbol{\Omega}^{\times} ( {}^{A}\boldsymbol{R}_{B})$$

(3.54)

From the perspective of the moving coordinate system, no change is observed in the rotation matrix ${}^{A}\boldsymbol{R}_{B}$, Hence

$$\left( \frac{\partial ( {}^{A}\boldsymbol{R}_{B})}{\partial t} \right)_{mov} = 0$$

Therefore,

$$.{}^{A}\boldsymbol{R}_{B} = \boldsymbol{\Omega}^{\times} ( {}^{A}\boldsymbol{R}_{B})$$

(3.55)

This approach, in addition to providing a quick proof for the relation 3.35, demonstrates that the skew-symmetric matrix used in this equation is derived from the components of the angular velocity vector $\boldsymbol{\Omega}$.

Now, consider the general motion of a rigid body, including both translational and rotational motion, as shown in Figure 3.4. In this figure, by attaching the coordinate system $\{\mathscr{B}\}$ to the rigid body, we aim to find the absolute velocity vector of point $P$ with respect to the reference coordinate system $\{\mathscr{A}\}$. In this scenario, the general motion of the rigid body is described by the combination of its translational motion, represented by the vector ${}^{A}\boldsymbol{P}_{O_B}$, and its rotational motion, represented by the screw vector of rotation.

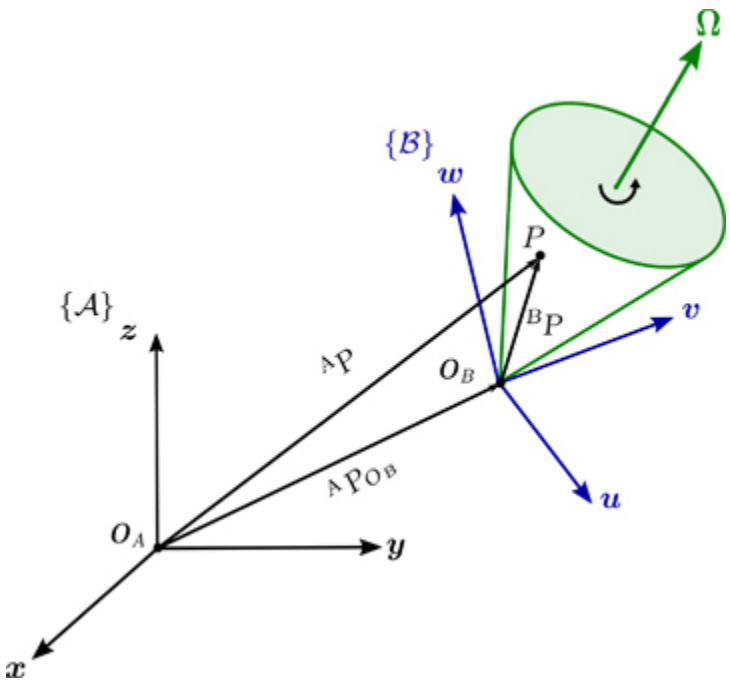


**FIGURE 3.4**
Instantaneous velocity of point $P$ relative to the reference coordinate system $\{\mathscr{A}\}$. ⏎

In order to determine the absolute linear velocity vector of point $P$, rewrite the vector describing the absolute position of this point provided in Equation 2.39.

$$^{A}\boldsymbol{P} = {}^{A}\boldsymbol{P}_{O_B} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} \tag{3.56}$$

To establish the relationship between the absolute and relative linear velocities of point $P$, differentiate both sides of the Equation 3.56 with respect to time.

$$\begin{aligned} {}^{A}\dot{\boldsymbol{P}} &= {}^{A}\dot{\boldsymbol{P}}_{O_B} + \dot{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\dot{\boldsymbol{P}} \\ {}^{A}\boldsymbol{v}_{p} &= {}^{A}\boldsymbol{v}_{O_B} + \dot{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{v}_{p} \end{aligned} \tag{3.57}$$

In this equation, ${}^{B}\boldsymbol{v}_{p} = v_{rel}$ represents the relative velocity vector of point $P$ as observed from the perspective of the moving coordinate system $\{\mathscr{B}\}$. The derivative of the rotation matrix $\dot{}^{A}\boldsymbol{R}_{B}$ can also be obtained using the verified Equation 3.55 as follows.

$$\dot{}^{A}\boldsymbol{R}_{B} = {}^{A}\boldsymbol{\Omega}^{\times}\ {}^{A}\boldsymbol{R}_{B}$$

(3.58)

Hence,

$$ {}^{A}\boldsymbol{v}_{p} = {}^{A}\boldsymbol{v}_{O_B} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{v}_{p} + {}^{A}\boldsymbol{\Omega}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} $$

(3.59)

In the case where $P$ is a point on a rigid body, this point does not have relative motion with respect to the body, and ${}^{B}\boldsymbol{v}_{p}$ will be equal to zero. In this situation,

$$ {}^{A}\boldsymbol{v}_{p} = {}^{A}\boldsymbol{v}_{O_B} + {}^{A}\boldsymbol{\Omega}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} $$

(3.60)

This relationship will be used in Chapter 6, in the analysis of the end-effector velocity in terms of its joint variables for the robot.

## 3.5 Screw Coordinates

We have seen that the finite rotation of a rigid body can be described by the angle of rotation $\theta$ about the screw axis $\hat{\boldsymbol{s}}$. Additionally, we mentioned that the angular velocity vector of the body can also be represented along this axis with a magnitude of $\dot{\theta}$. In Section 3.3, using the Mozzi–Chasles's theorem, it was demonstrated that the translational motion of a body can be represented along the screw axis with a

pitch parameter $d$. Considering that the general motion of a rigid body is described by decomposing it into two parts: rotation about the screw axis and translational motion along that axis, the screw representation bears a resemblance to the motion of a screw and nut, justifying the use of the term “screw” for this representation.

In this section, we introduce “Screw Coordinates” that effectively describe simultaneous rotational and translational motion. For this purpose, consider the general motion of a rigid body, a combination of rotational motion around the screw axis and translational motion along this axis. Furthermore, similar to the screw pitch in the motion of a screw and nut, define the screw pitch $\lambda$ as the ratio of translational motion to rotational motion, given by $\lambda = d/\theta$, where the unit of pitch measurement is $m/rad$. The ratio of translational velocity to angular velocity can also be used to define the screw pitch using the following relationship.

$$\lambda = \frac{\dot{d}}{\dot{\theta}} \tag{3.61}$$

The general motion of a rigid body, which consists of a combination of rotational and translational motion, can be represented by a $6 \times 1$ array composed of two corresponding vectors representing rotational and translational motion. This array, known by various names in robotics literature, is introduced in this book as *Screw*

*Coordinates* [62, 75]. Thus, the unit screw coordinates, denoted by $\hat{\$}$, are defined as a $6 \times 1$ array, combining the following two vectors:

$$\hat{\$} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} + \lambda \hat{\boldsymbol{s}} \end{bmatrix} = [\$_1, \$_2, \$_3, : \$_4, \$_5, \$_6]^T \tag{3.62}$$

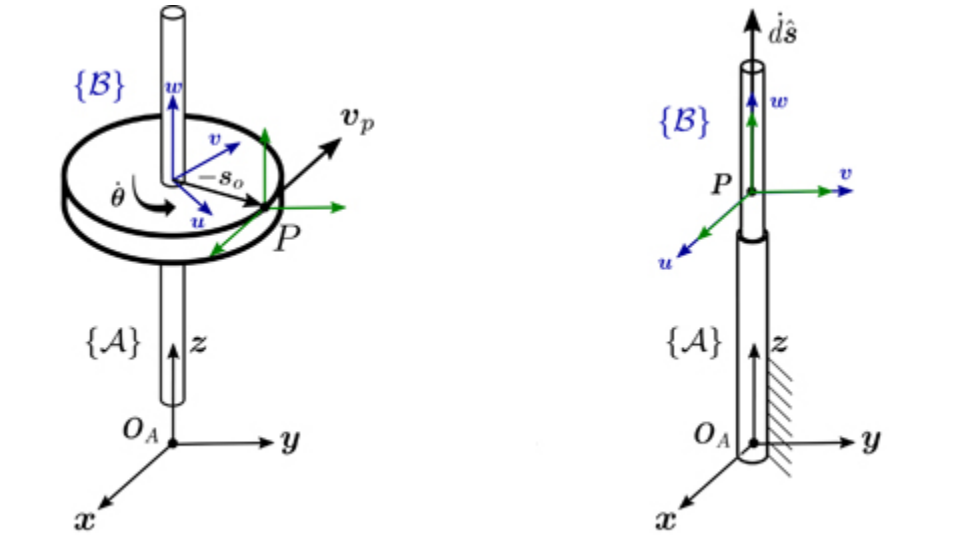
In this definition, the unit vector $\hat{\boldsymbol{s}}$ indicates the axis of rotation or the screw axis, and the parameter $\lambda$, from the relationship 3.61, represents the ratio of translational to rotational motion. Additionally, $\boldsymbol{s}_o$ is a vector specifying the distance from the point of interest for velocity analysis to the screw axis. To determine the characteristics of the rotational and translational motion of a body, it is sufficient to multiply the unit screw by the motion intensity $\dot{q}$. The motion intensity in rotational motion is the angular velocity $\dot{q} = \dot{\theta}$, and in translational motion, it is the linear translational velocity $\dot{q} = \dot{d}$. This multiplication results in a unique quantity called the *twist*.

$$\$ = \dot{q}\,\hat{\$} \tag{3.63}$$

In order to grasp the physical concept of the “twist,” let us consider two specific cases initially. Later, we will examine the general motion of a rigid body by referring to

these two specific cases. To this end, suppose that, according to Figure 3.5(a), the rigid body has only rotational motion, and we want to examine the characteristics of both rotational and translational motion of point $P$. In this case, $\lambda = 0$, and the screw coordinates in this specific case simplify to the following form.

$$\hat{\$} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} \end{bmatrix} \tag{3.64}$$




(a) Pure Rotational Motion. (b) Pure Translational Motion.

**FIGURE 3.5**
Representation of screw coordinates in rotational and translational motions. ⏎

The six-dimensional quantity of twist is obtained by multiplying the rotational rate $\dot{\theta}$ by the screw of unit pitch,

as follows:

$$\$ = \hat{\$}\,\dot{\theta} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} \end{bmatrix} \dot{\theta}$$

(3.65)

If we examine the vectors forming the twist in this case, we will find that the first vector determines the angular velocity of the body.

$$\hat{\boldsymbol{s}}\dot{\theta} = \boldsymbol{\Omega}$$

(3.66)

If we consider the vector $\boldsymbol{s}_o$ according to Figure 3.5(a) as the vector from the point $P$ to the screw axis with the direction indicated in the figure, then the second vector forming the twist determines the linear velocity of this point.

$$\left(\boldsymbol{s}_o \times \hat{\boldsymbol{s}}\right)\dot{\theta} = \boldsymbol{s}_o \times \boldsymbol{\Omega} = \boldsymbol{\Omega} \times \left(-\boldsymbol{s}_o\right) = \boldsymbol{v}_p$$

(3.67)

According to the twist definition, the vector $\boldsymbol{s}_o$ is equal to the vectorial distance from the origin of the moving coordinate system to the fixed coordinate system, i.e., ${}^{A}\boldsymbol{P}_{O_B} = \boldsymbol{s}_o$. However, this interpretation cannot be directly used to determine the translational motion characteristics

of point $P$ that is moving with a rigid body. To determine the translational motion characteristics of point $P$, assume that the coordinate system $\{\mathscr{A}\}$ is instantly placed on the point of interest without changing the orientation of its principal axes. In this way, the vector $-\boldsymbol{s}_o$ can be determined by using the inverse of this vector, i.e., ${}^{B}\boldsymbol{P}_{O_A}$, and with this arrangement, the translational motion characteristics of any point on the body can be easily obtained by simply placing the origin of the $\{\mathscr{A}\}$ coordinate system instantaneously on it.

Similarly, pure translational motion in a prismatic joint can be examined as another special case of this representation. As shown in Figure 3.5(b), consider a point $P$ on a rigid body with only translational motion. In this case, the pitch of motion is calculated asymptotically, since $\dot{\theta}$ is infinitesimally small, therefore, $\lambda = \infty$, and the velocity of motion is considered to be $\dot{q} = \dot{d}$ in the limit. The screw coordinates in this special case simplify to the following form.

$$\$\ \dot{d} = \begin{bmatrix} \mathbf{0} \\ \hat{\boldsymbol{s}} \end{bmatrix} \dot{d} \tag{3.68}$$

In this case, the first vector, which is zero, indicates that the body does not have any rotational motion. However, the second vector precisely determines the translational velocity of point $P$, if the origin of the $\{\mathscr{A}\}$ coordinate

system is instantaneously placed on the point $P$ according to the convention:

$$\boldsymbol{v}_p = \hat{\boldsymbol{s}}\,\dot{d}$$

(3.69)

This condition similarly holds for the general motion of a rigid body, as illustrated in Figure 3.6. In this case, the six-dimensional screw coordinate is written as follows.

$$\$ = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} + \lambda \hat{\boldsymbol{s}} \end{bmatrix} \dot{\theta}$$

(3.70)

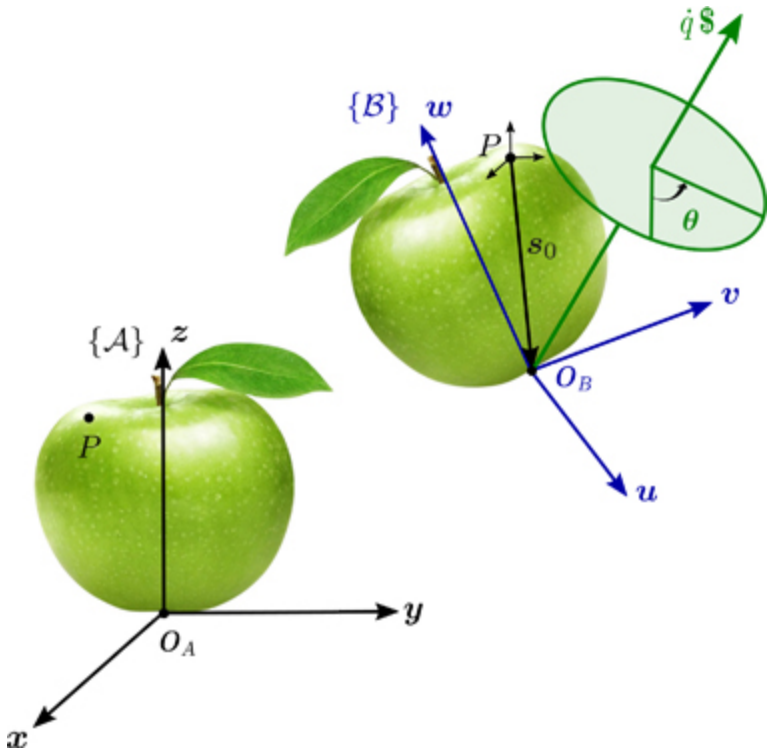


**FIGURE 3.6**
Instantaneous placement of the origin of the $\{\mathscr{A}\}$ coordinate system on point $P$ to determine the screw coordinates. ⏎

Note that the first three components of this array determine the angular velocity of the body.

$$\hat{\boldsymbol{s}}\,\dot{\theta} = \boldsymbol{\Omega}$$

(3.71)

The other three components of this array also determine the velocity vector of point $P$ on the rigid body if we place the origin of coordinate system $\{\mathscr{A}\}$ instantly on point $P$ according to the convention.

$$\begin{aligned}(\boldsymbol{s}_o \times \hat{\boldsymbol{s}} + \lambda\hat{\boldsymbol{s}})\,\dot{\theta} &= \boldsymbol{s}_o \times \dot{\theta}\hat{\boldsymbol{s}} + \lambda\,\dot{\theta}\,\hat{\boldsymbol{s}} \\ &= \boldsymbol{s}_o \times \boldsymbol{\Omega} + \lambda\,\dot{\theta}\,\hat{\boldsymbol{s}} \\ &= \boldsymbol{\Omega} \times (-\boldsymbol{s}_o) + \dot{d}\,\hat{\boldsymbol{s}} \\ &= \boldsymbol{\Omega} \times {}^{B}\boldsymbol{P}_{O_A} + \dot{d}\,\hat{\boldsymbol{s}}\end{aligned}$$

(3.72)

Note that the final relation in this equation, obtained by placing the origin of coordinate system $\{\mathscr{A}\}$ instantly on the point $P$ according to the convention. This results in ${}^{A}\boldsymbol{P}_{O_B} = \boldsymbol{s}_o$, and for this reason, ${}^{B}\boldsymbol{P}_{O_A} = -\boldsymbol{s}_o$.

By comparing this result with the linear velocity vector representation of point $P$, as expressed in Equation 3.60, it becomes clear that the second term in Equation 3.72 is a result of the translational motion of the body, corresponding to ${}^{A}\boldsymbol{v}_{O_B}$ in Equation 3.60. The first term in Equation 3.72 is a result of the rotational motion of the rigid body, contributing to a portion of the linear velocity vector of point $P$.

In the field of robotics, the use of the six-dimensional array in screw coordinates is not only used for interpreting the general motion of a point on a rigid body but can also be employed to represent the six-dimensional force/torque applied to a rigid body at a specific point. The term “wrench” is used to describe this six-dimensional array, and we will explore the algebra governing it and its relationship with screw coordinates in the subsequent chapters.

## 3.6 Linear and Angular Accelerations

In this section, we explore the definition of linear and angular accelerations in the motion of a rigid body. The acceleration vector of a point on the body is determined by the derivative of its velocity vector with respect to time. If the differentiation is done with respect to a fixed coordinate system, the absolute acceleration of that point is obtained. This is in contrast to differentiating the relative

velocity vector of a point on the body with respect to the moving coordinate system, which determines the relative acceleration of that point.

On the other hand, the quantity of angular acceleration falls into the category of free vectors, defined for the entire rigid body. Angular acceleration determines the rate of change of angular velocity of the body with respect to time. By using the method introduced in the Section 3.4.5, which relates the absolute derivative of a quantity to its relative derivative, one can easily obtain the vectors of the angular acceleration of the body and the linear acceleration of a point.

### 3.6.1 Angular Acceleration of a Rigid Body

To define the angular acceleration of a rigid body, as illustrated in Figure 3.7, we examine its motion by installing a moving coordinate system $\{\mathscr{B}\}$ relative to the reference coordinate system $\{\mathscr{A}\}$. The absolute angular acceleration of the body is a characteristic of its overall motion, obtained by differentiating the angular velocity vector with respect to the reference coordinate system. Thus, the *angular acceleration vector*, denoted by $\dot{\boldsymbol{\Omega}}$, is obtained by differentiating the relation in Equation 3.30 as follows:

$$\begin{aligned}\dot{\boldsymbol{\Omega}} = \frac{d\boldsymbol{\Omega}}{dt} &= \ddot{\theta}\,\hat{\boldsymbol{s}} + \dot{\theta}\dot{\hat{\boldsymbol{s}}} \\ &= \ddot{\theta}\,\hat{\boldsymbol{s}} + \dot{\theta}\,(\boldsymbol{\Omega} \times \boldsymbol{s}) \\ &= \ddot{\theta}\,\hat{\boldsymbol{s}}\end{aligned} \tag{3.73}$$

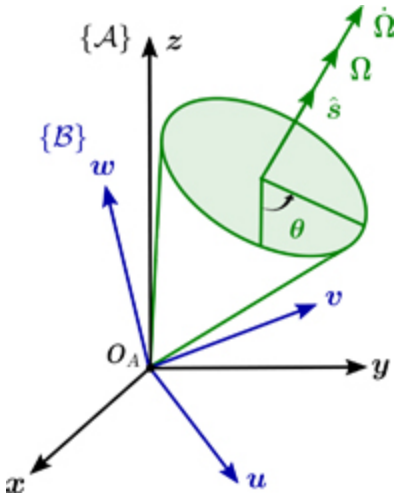



**FIGURE 3.7**
Angular acceleration of a rigid body in pure rotational motion. ⏎

Note that in this relation, since the angular velocity vector of the body lies along the screw axis, the cross product $\boldsymbol{\Omega} \times \boldsymbol{s}$ equals zero. Consequently, it is demonstrated that not only is the instantaneous angular velocity of the body aligned with the screw axis at any given moment, but its angular acceleration also remains in the same direction as shown in Figure 3.7. This crucial characteristic only arises

when representing the rotational motion of the body using a screw representation.

### 3.6.2 Linear Acceleration of a Point

Linear acceleration of point $P$ can easily be obtained from the derivative of its linear velocity vector $\boldsymbol{v}_p$ and is given by:

$$\boldsymbol{a}_p = \dot{\boldsymbol{v}}_p = \left( \frac{d\boldsymbol{v}_p}{dt} \right)_{fix} \tag{3.74}$$

Note that this differentiation must be taken with respect to the fixed coordinate system to accurately calculate the absolute acceleration of point $P$. This importance is emphasized in this equation by representing absolute differentiation with the notation $\left(d(\,\cdot\,)/dt\right)_{fix}$.

If the linear velocity vector of point $P$ is differentiated with respect to the moving coordinate system, the result will be the calculation of the *relative acceleration* of that point, denoted as $\boldsymbol{a}_{rel}$ and calculated as follows:

$$\boldsymbol{a}_{rel} = \left( \frac{\partial \boldsymbol{v}_p}{\partial t} \right)_{mov} \tag{3.75}$$

In this equation, differentiation with respect to the moving coordinate system, as before, is represented by $(\partial(\,\cdot\,)/\partial t)_{mov}$. Fortunately, the relationship between absolute and relative differentiation of a physical quantity can be determined using Equations 3.51 or 3.53.

Now, consider the general motion of a rigid body in space, as shown in Figure 3.4. Let's analyze the linear acceleration of point $P$ by attaching the moving coordinate system $\{\mathscr{B}\}$ to the body, as before, to find the linear acceleration of point $P$ in the reference coordinate system $\{\mathscr{A}\}$. For this purpose, by rewriting the relation 3.59 in the interpretation of absolute and relative velocities of point $P$, we have:

$$^{A}\boldsymbol{v}_p = {}^{A}\boldsymbol{v}_{O_B} + {}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{v}_p + {}^{A}\boldsymbol{\Omega}^{\times}\;{}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{P}$$

Differentiate this relation with respect to time to determine the linear acceleration of point $P$:

$$\begin{aligned}
{}^{A}\dot{\boldsymbol{v}}_p &= {}^{A}\dot{\boldsymbol{v}}_{O_B} + \frac{d}{dt}\left({}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{v}_p\right) + \frac{d}{dt}\left({}^{A}\boldsymbol{\Omega}^{\times}\;{}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{P}\right)\\
{}^{A}\boldsymbol{a}_p &= {}^{A}\boldsymbol{a}_{O_B} + \left({}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{a}_p + {}^{A}\boldsymbol{\Omega}^{\times}\;{}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{v}_P\right)\\
&\quad + {}^{A}\dot{\boldsymbol{\Omega}}^{\times}\;{}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{P} + {}^{A}\boldsymbol{\Omega}^{\times}\;\left({}^{A}\boldsymbol{\Omega}^{\times}\;{}^{A}\boldsymbol{R}_B\,{}^{B}\boldsymbol{P}\right) + {}^{A}\boldsymbol{\Omega}^{\times}\;{}^{A}\boldsymbol{I}
\end{aligned}$$

(3.76)

By simplifying this equation, the absolute acceleration of point $P$ is expressed in terms of five terms as follows:

$$\begin{aligned} {}^{A}\boldsymbol{a}_{p} &= {}^{A}\boldsymbol{a}_{O_B} + {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{a}_{p} + \\ &\quad {}^{A}\dot{\boldsymbol{\Omega}}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} + {}^{A}\boldsymbol{\Omega}^{\times}\left({}^{A}\boldsymbol{\Omega}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P}\right) + 2\,{}^{A}\boldsymbol{\Omega}^{\times}\,{}^{A} \end{aligned} \tag{3.77}$$

In this equation, ${}^{A}\boldsymbol{a}_{O_B}$ represents the linear acceleration of the origin of the moving coordinate system $\{\mathscr{B}\}$. Furthermore, ${}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{a}_{p}$ represents the relative acceleration of point $P$ with respect to the moving coordinate system $\{\mathscr{B}\}$. The term ${}^{A}\dot{\boldsymbol{\Omega}}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P}$ indicates the contribution of the angular acceleration of the moving coordinate system $\{\mathscr{B}\}$ to the linear acceleration of point $P$, and the two other terms determine the centrifugal and Coriolis accelerations of point $P$, respectively.

If point $P$ belongs to the rigid body, its relative velocity and acceleration with respect to the moving coordinate system will be zero: ${}^{B}\boldsymbol{v}_{p} = {}^{B}\boldsymbol{a}_{p} = 0$. Consequently, the expression for the linear acceleration of point $P$ simplifies to:

$${}^{A}\boldsymbol{a}_{p} = {}^{A}\boldsymbol{a}_{O_B} + {}^{A}\dot{\boldsymbol{\Omega}}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P} + {}^{A}\boldsymbol{\Omega}^{\times}\left({}^{A}\boldsymbol{\Omega}^{\times}\,{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P}\right) \tag{3.78}$$

This equation will be used in Chapter 8 for the dynamic analysis of robot motion.

## Problems

3.1 Rotation of a rigid body is given by the following rotation matrix:

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.500 & 0.080 & 0.862 \\ 0.500 & 0.787 & -0.362 \\ -0.707 & 0.612 & 0.354 \end{bmatrix} $$

a. Find the eigenvalues and eigenvectors of the rotation matrix ${}^{A}\boldsymbol{R}_{B}$.

b. Obtain the representation of this rotation with a screw axis, determine the rotation angle $\theta$, and find the axis of rotation ${}^{A}\hat{\boldsymbol{s}} = [s_x \quad s_y \quad s_z]^T$. Verify your calculations with the eigenvalues and eigenvectors obtained in the previous part.

c. Represent this rotational motion with its corresponding quaternion.

3.2 Revisit Problem 3.1 using the given rotation matrix.

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} -0.146 & -0.854 & 0.500 \\ 0.854 & 0.146 & 0.500 \\ -0.500 & 0.500 & 0.707 \end{bmatrix}. $$

3.3 Reiterate Problem 3.1 with the provided rotation matrix.

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.0990 & -0.3696 & 0.9239 \\ 0.9839 & -0.1024 & -0.1464 \\ 0.1487 & 0.9235 & 0.3536 \end{bmatrix} $$

3.4 Reattempt Problem 3.1 using the given rotation matrix.

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0.933 & 0.067 & 0.354 \\ 0.067 & 0.933 & -0.354 \\ -0.354 & 0.354 & 0.866 \end{bmatrix}. $$

3.5 Repeat Problem 3.1 with the provided rotation matrix.

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} 0 & 0.259 & 0.966 \\ 0 & 0.966 & -0.250 \\ -1 & 0 & 0 \end{bmatrix}. $$

3.6 Redo Problem 3.1 using the given rotation matrix.

$$
{}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} -0.8001 & -0.4975 & 0.3351 \\ 0.4619 & -0.1546 & 0.8733 \\ -0.3827 & 0.8536 & 0.3536 \end{bmatrix}
$$

3.7 Assume that the rotation of a rigid body is given with constant angles relative to the following coordinate systems:

$$
\boldsymbol{R}_z(\pi/2), \boldsymbol{R}_y(-\pi/4), \boldsymbol{R}_z(\pi/6).
$$

a. Obtain the representation of this rotation with a screw axis, determine the rotation angle $\theta$, and find the axis of rotation ${}^{A}\hat{\boldsymbol{s}} = [s_x \quad s_y \quad s_z]^T$.

b. Represent this rotational motion with its corresponding quaternion.

3.8 Revisit Problem 3.7 for the rotation of a rigid body with the following pitch-roll-yaw Euler angles.

$$
\boldsymbol{R}_x(\pi/6), \boldsymbol{R}_y(\pi/4), \boldsymbol{R}_z(\pi/4)
$$

.

3.9 Reiterate Problem 3.7 for the rotation of a rigid body relative to the following moving coordinate systems:

$$
\boldsymbol{R}_v(\pi/4), \boldsymbol{R}_w(-\pi/2), \boldsymbol{R}_u(\pi/3)
$$

.

3.10 Redo Problem 3.7 for the rotation of a rigid body relative to the following moving coordinate systems:

$$\boldsymbol{R}_v(\pi/4), \boldsymbol{R}_w(-\pi/2), \boldsymbol{R}_u(\pi/3)$$

.

3.11 Assume that the rotation of a rigid body is represented by a screw axis with the following parameters:

$$\left\{\theta = \pi/2, \hat{\boldsymbol{s}} = [-0.577 \quad 0.577 \quad 0.577]^T\right\}.$$

a. Determine the rotation matrix ${}^A\boldsymbol{R}_B$ for this body.

b. Calculate the pitch-roll-yaw Euler angles in this rotation.

c. Calculate the $\boldsymbol{w}-\boldsymbol{v}-\boldsymbol{w}$ Euler angles in this rotation.

d. Calculate the $\boldsymbol{w}-\boldsymbol{u}-\boldsymbol{w}$ Euler angles in this rotation.

e. Represent this rotational motion with its corresponding quaternion.

3.12 Repeat Problem 3.11 for the rotation of a rigid body represented by a screw axis with the following parameters:

$$\left\{\theta = \pi/4, \hat{\boldsymbol{s}} = \begin{bmatrix} 0.707 & -0.707 & 0 \end{bmatrix}^T\right\}.$$

3.13 Revisit Problem 3.11 for the rotation of a rigid body represented by the following screw vector:

$$\boldsymbol{\theta} = \frac{\pi}{6}\begin{bmatrix} 0.447 & 0 & -0.894 \end{bmatrix}^T.$$

3.14 Reiterate Problem 3.11 for the rotation of a rigid body represented by a screw vector:

$$\boldsymbol{\theta} = \frac{\pi}{12}\begin{bmatrix} 2 & -1 & -2 \end{bmatrix}^T.$$

3.15 Assume that the rotation of a rigid body is represented by the following quaternion:

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.707 & -0.707 & 0 & 0 \end{bmatrix}^T$$

a. Determine the rotation matrix ${}^A\boldsymbol{R}_B$ for this body.

b. Calculate the pitch-roll-yaw Euler angles in this rotation.

c. Calculate the $\boldsymbol{w} - \boldsymbol{v} - \boldsymbol{w}$ Euler angles in this rotation.

d. Calculate the $\boldsymbol{w} - \boldsymbol{u} - \boldsymbol{w}$ Euler angles in this rotation.

e. Represent this rotational motion with a screw vector $\boldsymbol{\theta}$.

3.16 Repeat Problem 3.15 for the rotation of a rigid body represented by the following quaternion:

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.408 & 0.408 & 0.408 & 0.707 \end{bmatrix}^T$$

3.17 Reiterate Problem 3.15 for the rotation of a rigid body represented by the following quaternion:

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.204 & 0.408 & -0.204 & 0.866 \end{bmatrix}^T$$

3.18 Revisit Problem 3.15 for the rotation of a rigid body represented by the following quaternion:

$$\boldsymbol{\epsilon} = \begin{bmatrix} -0.204 & 0.204 & 0.408 & 0.866 \end{bmatrix}^T$$

3.19 Redo Problem 3.15 for the rotation of a rigid body represented by the following quaternion:

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0.289 & 0.577 & 0.289 & 0.707 \end{bmatrix}^T$$

3.20 Assume that the rotation of a rigid body is represented by a screw axis with the following parameters:

$$\left\{\theta, \hat{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s}_x & \boldsymbol{s}_y & \boldsymbol{s}_z \end{bmatrix}^T\right\}.$$

Show that the rotation matrix can be obtained using the exponential function as follows:

$$^A\boldsymbol{R}_B = e^{\hat{\boldsymbol{s}}^\times \theta},$$

(3.79)

where the matrix $\hat{\boldsymbol{s}}^\times$ is defined as:

$$\hat{\boldsymbol{s}}^\times = \begin{bmatrix} 0 & -\boldsymbol{s}_z & \boldsymbol{s}_y \\ \boldsymbol{s}_z & 0 & -\boldsymbol{s}_x \\ -\boldsymbol{s}_y & \boldsymbol{s}_x & 0 \end{bmatrix}.$$

(3.80)

3.21 Verify that if the motion of a rigid body is described by a quaternion, the rotation matrix can be determined from the following equation:

$$^A\boldsymbol{R}_B = \begin{bmatrix} 1-2\epsilon_2^2-2\epsilon_3^2 & 2(\epsilon_1\epsilon_2-\epsilon_3\epsilon_4) & 2(\epsilon_1\epsilon_3+\epsilon_2\epsilon_4) \\ 2(\epsilon_1\epsilon_2+\epsilon_3\epsilon_4) & 1-2\epsilon_1^2-2\epsilon_3^2 & 2(\epsilon_2\epsilon_3-\epsilon_1\epsilon_4) \\ 2(\epsilon_1\epsilon_3-\epsilon_2\epsilon_4) & 2(\epsilon_2\epsilon_3+\epsilon_1\epsilon_4) & 1-2\epsilon_1^2-2\epsilon_2^2 \end{bmatrix}.$$

3.22 For a general motion in space that includes rotational and translational motion, derive the Rodrigues formula

as given in Equation 3.4:

$$^{A}\boldsymbol{P} = \boldsymbol{s}_o + d\,\hat{\boldsymbol{s}} + (^{B}\boldsymbol{P} - \boldsymbol{s}_o)c\theta + \hat{\boldsymbol{s}} \times (^{B}\boldsymbol{P} - \boldsymbol{s}_o)s\theta + \left[(^{B}\boldsymbol{P} - \boldsymbol{s}\right.$$

where notations $c\theta = \cos\theta, s\theta = \sin\theta, v\theta = 1 - \cos\theta$ are used. Furthermore, prove the relationship 3.29.

3.23 Show how to determine the rotation matrix of a rigid body, as given in Equation 3.6, using the Rodrigues' formula.

3.24 Assume that the motion of a rigid body is represented by the following homogeneous transformation matrix. Obtain the parameters of this motion based on the screw representation.

$$^{A}\boldsymbol{T}_B = \left[\begin{array}{ccc|c} 0.5 & -0.75 & 0.433 & 0.5 \\ 0.866 & 0.433 & -0.25 & 0.866 \\ 0 & 0.5 & 0.866 & 0.1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

3.25 Repeat Problem 3.24 for the motion represented by the following homogeneous transformation matrix.

$$
{}^{A}\boldsymbol{T}_{B} = \left[\begin{array}{ccc|c} +0.808 & -0.575 & -0.130 & 0.991 \\ +0.534 & +0.620 & +0.575 & 1.083 \\ -0.250 & -0.534 & +0.808 & 0.148 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

3.26 Revisit Problem 3.24 for the motion represented by the following homogeneous transformation matrix.

$$
{}^{A}\boldsymbol{T}_{B} = \left[\begin{array}{ccc|c} +0.250 & -0.967 & -0.058 & 0.916 \\ +0.433 & +0.058 & +0.900 & 0.587 \\ -0.866 & -0.250 & +0.433 & 0.527 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

3.27 Reiterate Problem 3.24 for the motion represented by the following homogeneous transformation matrix.

$$
{}^{A}\boldsymbol{T}_{B} = \left[\begin{array}{ccc|c} -0.575 & -0.635 & -0.516 & 0.746 \\ +0.621 & -0.750 & +0.231 & 1.782 \\ -0.534 & -0.188 & +0.826 & 0.850 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

3.28 Suppose the general motion of a rigid body is represented by the screw representation with the following parameters. Represent this motion in space using the homogeneous transformation matrix.

$$\left\{\theta = \frac{\pi}{4}, \hat{\boldsymbol{s}} = \frac{1}{\sqrt{6}}[1, -2, -1]^T \quad ; \quad d = 0.5, \boldsymbol{s}_o = [0, 0, 1]^T\right\}$$

3.29 Repeat Problem 3.28 for the motion described by the following screw representation.

$$\left\{\theta = \frac{-\pi}{4}, \hat{\boldsymbol{s}} = \frac{1}{\sqrt{3}}[1, -1, -1]^T \quad ; \quad d = 0.75, \boldsymbol{s}_o = [0, -1, 1]^T\right.$$

3.30 Revisit Problem 3.28 for the motion described by the subsequent screw representation.

$$\left\{\theta = \frac{\pi}{2}, \hat{\boldsymbol{s}} = \frac{1}{\sqrt{2}}[1, 0, -1]^T \quad ; \quad d = 1.5, \boldsymbol{s}_o = [-1, 0, -1]^T\right\}$$

3.31 Reiterate Problem 3.28 for the motion described by the following screw representation.

$$\left\{\theta = \frac{-\pi}{6}, \hat{\boldsymbol{s}} = \frac{1}{3}[-2, -2, -1]^T \quad ; \quad d = 1, \boldsymbol{s}_o = [1, 1, 1]^T\right\}$$

3.32 Redo Problem 3.28 for the motion described by the subsequent screw representation.

$$\left\{\theta = \frac{\pi}{3}, \hat{\boldsymbol{s}} = \frac{1}{\sqrt{3}}[1, -1, 1]^T \quad ; \quad d = 2, \boldsymbol{s}_o = [1, -1, 1]^T\right\}$$

# Part II

# Kinematics

# 4

# *Kinematic Analysis*

DOI: 10.1201/9781003590408-4

This chapter explores the comprehensive framework for kinematic analysis, focusing on parallel robotic systems. It begins by establishing foundational concepts and categorizing various types of parallel robots, emphasizing their distinct structural and operational characteristics. A detailed examination of the loop closure method as a fundamental approach to kinematic analysis is provided, highlighting its application in solving the interdependencies of robotic mechanisms.

The chapter further explores three case studies, each representing a unique parallel robotic system. The first case focuses on a planar cable robot, where the geometry and operational principles are analyzed, followed by an exploration of its inverse and forward kinematics. Computational simulations validate its behavior and accuracy. The second case examines a shoulder manipulator, integrating detailed geometric descriptions with kinematic models to provide insights into its

functionality and real-world applications. Simulations reinforce the manipulator's performance and highlight design considerations.

The final case study addresses the Stewart–Gough platform, a widely recognized parallel robot often used in advanced applications such as flight simulation. The chapter details its geometric configuration, derives its inverse and forward kinematic formulations, and presents simulation results that underscore the platform's precision and versatility. Across all case studies, simulations play a crucial role in bridging theoretical analyses with practical outcomes, offering a holistic perspective on kinematic design and optimization.

The chapter concludes with a set of challenging problems that encourage critical thinking and further exploration of kinematic principles, ensuring that readers not only grasp the concepts but also gain confidence in their application to complex robotic systems. These tasks are designed to deepen understanding and develop problem-solving skills, for twelve other parallel robots extensively used in the literature, bridging the gap between theoretical knowledge and real-world applications.

## 4.1 Introduction

In examining the kinematics of parallel robots, the focus is on studying the geometry of motion without considering

the forces and torques that cause it. Kinematic analysis assumes that the robot is in a specific static position: the moving platform has a particular pose, and the robot's limb links and joints are each in specific poses within the space. The investigation into the relationship between the position and orientation of the robot's moving platform and the motions of its active limbs constitutes the robot's forward kinematic analysis. The inverse kinematic problem involves analytically determining the robot's motion geometry, where the variables describing the moving platform in the task space, $\boldsymbol{\chi}$, are known, and the goal is to determine the corresponding active limb variables, $\boldsymbol{q}$. Using this description of joint and task space, forward and inverse kinematic analyses can be defined as stated, and as illustrated in Figure 4.1.

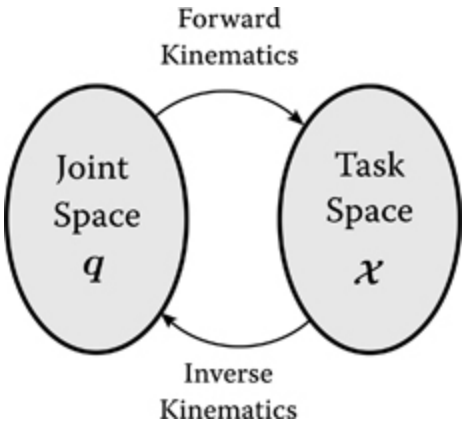



**FIGURE 4.1**

Representation of the joint and task space variables in a parallel robot context, delineating the definitions of forward and inverse kinematics. ⏎

Unlike serial robots, solving the inverse kinematic problem in parallel robots is often straightforward and typically yields a unique solution. Conversely, the forward kinematic analysis of parallel robots is more complex. This complexity has been addressed through various methods that simplify and transform the problem into multiple lower-dimensional problems, depending on the type of robot. Analyzing the robot's speed and acceleration along with its motion state leads to the study of differential kinematics. The analysis of robot speed will be covered in Chapter 5, while the analysis of acceleration and the dynamics of motion will be reviewed in Chapter 8.

## 4.2 Parallel Robot Types

A parallel robot is characterized by multiple closed kinematic chains, with its moving platform connected to the base through several independent kinematic chains. This broad definition includes both redundant parallel manipulators, which have more actuators than controlled degrees of freedom, and underactuated parallel manipulators. Parallel robots where the number of kinematic chains matches the degrees of freedom of the moving platform are known as “fully parallel robots.” If, in addition, the type and number of joints in each limb, as well as the number and placement of actuated joints, are identical across all limbs, the robot is called *symmetric*.

Symmetrical parallel manipulators are often preferred for kinematic analysis and are commonly used in many industrial robots, such as the Stewart–Gough platform and the Delta robot.

There are three main types of fully parallel manipulators: planar robots with two translational and one rotational degree of freedom (2T1R) in a plane; spatial orientation manipulators with three rotational degrees of freedom (3R) in space; and general spatial robots with three translational and three rotational degrees of freedom (3T3R) in space. Figure 4.2 illustrates a symmetrical fully planar parallel robot with three limbs, each consisting of three identical revolute joints, known as a 3RRR manipulator. Here, the number 3 in the notation, indicates the number of limbs, while RRR describes the kinematic structure of each limb, with the underlined joint denoting the actuated one. Another example of a planar robot denoted as 4RPR, is depicted in Figure 1.6. This is a redundantly actuated manipulator with four limbs, each having an RPR structure, with the prismatic joint being actuated. Similarly, Figure 4.31 depicts a redundantly actuated 4RRR manipulator, which has four limbs with an RRR structure, where the first revolute joint is actuated.

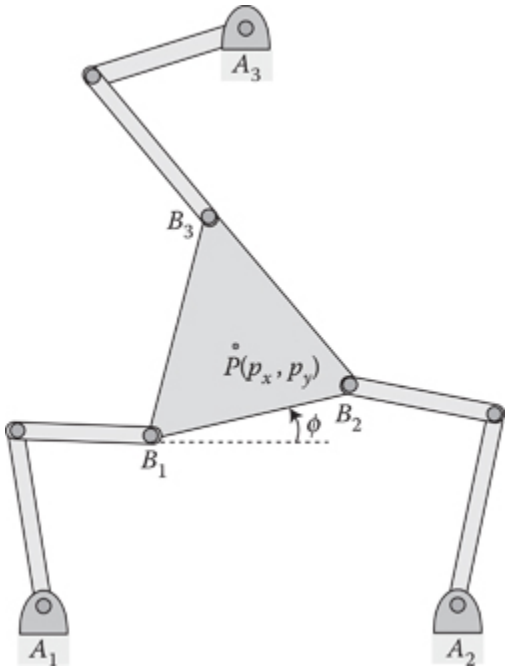



**FIGURE 4.2**

3RR Rrobot: An example of a symmetrical planar parallel robot. ⏎

Examples of spatial orientation manipulators are depicted in Figures 4.32 and 4.11. The first manipulator is a fully parallel platform with three degrees of freedom, while the latter is a redundant parallel manipulator. Both are designed to provide purely orientational motion for their moving platforms. The first manipulator, shown in Figure 4.32, consists of two tetrahedrals connected by a spherical joint, constraining the moving platform to three rotational degrees of freedom. The limbs of this robot have an SPS structure, with the prismatic joint being actuated. Similarly, the redundant manipulator illustrated in Figure 4.11, known as the hydraulic shoulder manipulator, has a passive link connected to the base with a spherical joint

($S$), which restricts the moving platform's motion to three rotational degrees of freedom. This manipulator employs four limbs, each with a U$\underline{P}$U structure, creating a redundant parallel manipulator denoted by 4U$\underline{P}$U. This book provides a comprehensive kinematic analysis of both the redundant planar manipulator and the redundant spatial orientation manipulator.

The kinematic analysis of Stewart–Gough platform is also examined as a representative of fully parallel six degrees of freedom manipulators. This manipulator consists of six identical limbs with an S$\underline{P}$S or U$\underline{P}$S structure. Although the kinematic analysis of the Stewart–Gough platform in its most general form has been extensively studied, yielding fascinating results [76, 77, 76], this book focuses on the closed-form solution of a specific geometry known as the 6-$6^{\mathrm{p}}$ Stewart–Gough platform. This choice is primarily because the analytic forward kinematic solution of this specific geometry is more tractable and can be more easily understood by the reader compared to other geometries. The 6-$6^{\mathrm{p}}$ Stewart–Gough platform is illustrated in Figure 4.18.

It is well-known that, unlike serial manipulators, the inverse kinematic analysis of parallel robots is usually simple and straightforward. In most cases, actuated limb variables $\boldsymbol{q}$, can be computed independently using the given pose of the moving platform $\boldsymbol{\chi}$, and the solution is typically uniquely determined, even for redundant manipulators. However, the forward kinematics of parallel

manipulators is generally very complex, involving systems of nonlinear equations that are highly coupled and generally lack a closed-form and unique solution [78, 79].

Various approaches to solving this problem are discussed in the literature, both in general [80] and in special cases [81, 82]. The analysis of two such special three degrees of freedom-constrained mechanisms has been explored in [51, 83]. Generally, different solutions to the forward kinematic problem of parallel manipulators can be found using either numerical [80] or analytical approaches [81]. In the following sections, a systematic method for performing kinematic analysis of parallel manipulators is presented, and this method is examined through the three selected case studies throughout the chapter.

## 4.3 Loop Closure Method

As illustrated in Figure 4.3, a typical parallel manipulator consists of two main bodies and a number of limbs. Body $A$, arbitrarily designated as fixed, is called the *base*, while Body $B$, designed to be movable, is referred to as the *moving platform* or sometimes the *end-effector*. These two bodies are connected via $n$ limbs, each attached to points $A_i$ and $B_i$, known as the fixed and moving attachment points of limb $i$, respectively.

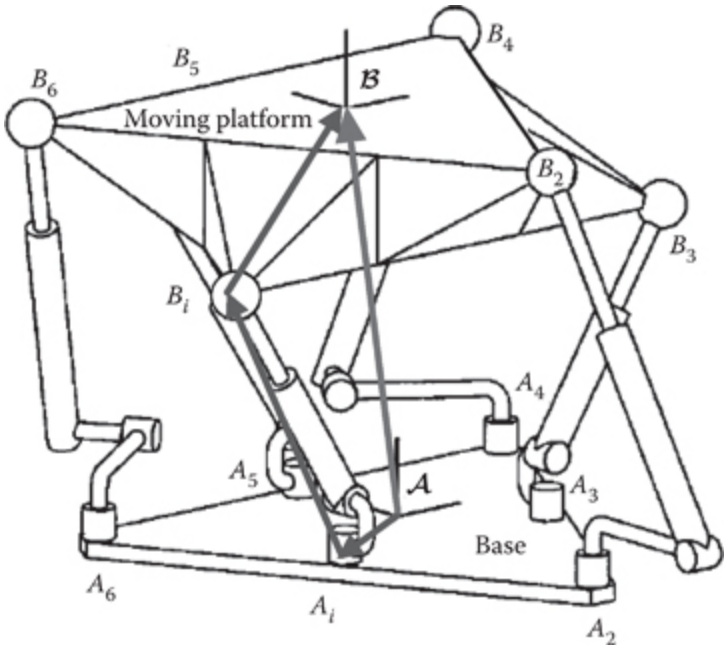



**FIGURE 4.3**
Vector loop closures in a general parallel manipulator. ⏎

At the displacement level, the forward kinematic problem involves determining the actual pose of the moving platform relative to the base from a set of joint-position readouts. At the velocity level, it involves determining the actual twist rate, i.e., the translational and angular velocities of the moving platform relative to the base, from a set of joint-velocity readouts for a known configuration.

Since the characteristics of parallel manipulators are typically complementary to those of serial architectures, they are suitable for applications where the latter do not meet the application's requirements. However, the higher degree of complexity of parallel mechanisms leads to more challenging forward kinematic problems at both the displacement and velocity levels. The complexity of these

problems largely depends on the architecture, geometry, and sensory layouts.

To describe the motion of the moving platform relative to the base, frame $\{\mathscr{A}\}$ is attached to body $A$, and frame $\{\mathscr{B}\}$ is attached to body $B$, as depicted in Figure 1.11. The pose of the moving platform relative to the base is defined by a position vector $\boldsymbol{p}$ and a rotation matrix $\boldsymbol{R}$. Here, $\boldsymbol{p}$ represents the position vector from the origin of $\{\mathscr{A}\}$ to the origin of $\{\mathscr{B}\}$, indicating the absolute position of body $B$ with respect to the fixed frame $\{\mathscr{A}\}$. The orientation of $\{\mathscr{B}\}$ relative to $\{\mathscr{A}\}$ is characterized by a $3 \times 3$ rotation matrix $\boldsymbol{R}$. Each limb of a parallel manipulator forms a *kinematic loop* that passes through the origins of frames $\{\mathscr{A}\}$ and $\{\mathscr{B}\}$, and through the two attachment points $A_i$ and $B_i$ of the limb.

At the displacement level, the closure of each kinematic loop can be formulated in vector form as:

$$\overrightarrow{AB} = \overrightarrow{AA_i} + \overrightarrow{A_iB_i} - \overrightarrow{BB_i} \quad \text{for } i = 1, \ldots, n \tag{4.1}$$

where $\overrightarrow{AA_i}$ and $\overrightarrow{BB_i}$ are straightforwardly determined from the geometric arrangement of attachment points on the base and moving platform. Let $\boldsymbol{a}_i = \overrightarrow{AA_i}$ in the fixed frame $\{\mathscr{A}\}$, and $\boldsymbol{b}_i = \overrightarrow{BB_i}$ in the moving frame $\{\mathscr{B}\}$.

Define $\boldsymbol{f}(\boldsymbol{q}_i) = \overrightarrow{A_i B_i}$ as the kinematic function of the limb variable, encapsulating the geometric configuration of the limb, which typically includes both active and passive segments. If the kinematic structure of the limbs is identical, as in symmetric parallel robots, the function $\boldsymbol{f}$ remains the same across different limbs, while the complexity of this function depends on the number of passive joints incorporated in each limb.

The variation in length and orientation of the vector $\boldsymbol{f}(\boldsymbol{q}_i) = \overrightarrow{A_i B_i}$ is governed by the active joint variable $\boldsymbol{q}_i$. Therefore, the closure of the loop can be expressed in terms of the unknown pose variables $\boldsymbol{p}$ and $\boldsymbol{R}$, which describe the position and orientation of the moving platform relative to the base, as well as the known vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$, representing the geometric configuration of the base and moving platform, respectively.

The limb vector $\boldsymbol{q}_i$ is determined solely by the kinematic equation of the limb and the outputs from position sensors located at different joints along the limb. Thus, the loop closure equations are formulated as follows:

$$\boldsymbol{p} = \boldsymbol{a}_i + \boldsymbol{f}(\boldsymbol{q}_i) - \boldsymbol{R}\,\boldsymbol{b}_i \quad \text{for } \; i = 1, \ldots, n \tag{4.2}$$

These equations form the core of kinematic analysis. In the inverse kinematic problem, the position $\boldsymbol{p}$ and orientation $\boldsymbol{R}$ of the moving platform are given, and the task is to

determine the active limb variables. Therefore, to solve the inverse kinematic problem based on these equations, the passive limb variables need to be eliminated. This analysis is usually straightforward and yields a unique solution for the limb variables, even with redundantly actuated parallel manipulators, assuming the kinematic function $\boldsymbol{f}(\boldsymbol{q}_i)$ remains relatively simple. However, solving the forward kinematic problem is not straightforward and often requires more effort and may need to be solved using numerical methods.

As evidenced in this discussion, the complexity of forward and inverse kinematic analyses in parallel robots varies based on the kinematic structure of the limbs and the intricacy of the parallel configuration. Consequently, there is no universally applicable method that can be prescribed for every robot. In the forthcoming sections, comprehensive examinations of inverse and forward kinematic analyses will be conducted on three distinct case studies featuring varying structures. These discussions aim to provide insights that can guide readers in approaching similar problems encountered in different applications.

## 4.4 Case Study I: Planar Cable Robot

In this section, a thorough investigation into the kinematics of a planar cable-driven parallel manipulator is undertaken. Initially, the kinematic structure of the mechanism is

delineated, followed by a detailed exposition of the manipulator's geometric configuration. Subsequently, comprehensive analyses of both inverse and forward kinematics for the manipulator are meticulously carried out. To validate these formulations, simulation results for computing inverse and forward kinematics are presented in Section 4.4.5, ensuring the accuracy of the numerical solutions for forward kinematics.

### 4.4.1 Mechanism Description

The architecture of the planar 4R$\underline{P}$R parallel manipulator depicted in Figure 4.4 features a moving platform supported by four limbs, each sharing an identical kinematic structure. Each limb connects the fixed base to the manipulator's moving platform through a sequence of a revolute joint ($R$), an actuated prismatic joint ($\underline{P}$), and another revolute joint ($R$), resulting in a total structure denoted as 4R$\underline{P}$R. The prismatic joint's kinematic structure can represent either a piston-cylinder actuator for each limb or a cable-driven mechanism. To prevent singularities at the manipulator's central position, the limbs are arranged in a crossed configuration. A comprehensive singularity analysis of the mechanism is detailed in Section 5.5.2.

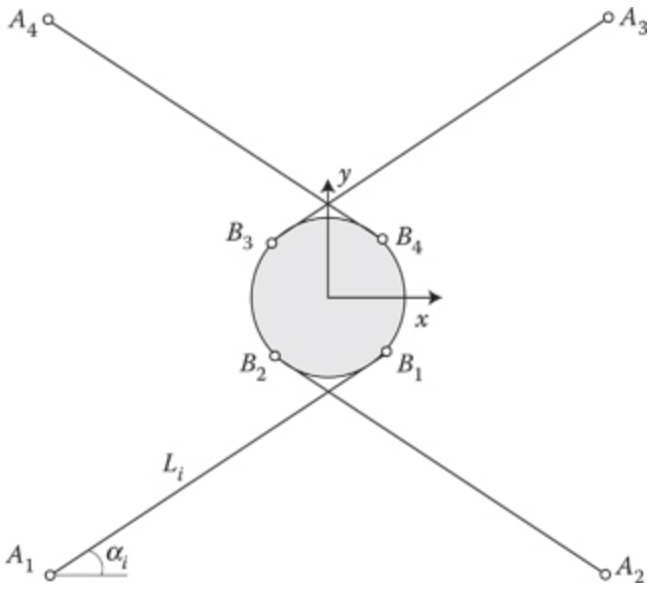



**FIGURE 4.4**
The schematics of a 4RPR cable-driven parallel manipulator. ⏎

As shown in Figure 4.4, $A_i$ denotes the fixed base points of the limbs, $B_i$ denotes the points where the limbs connect to the moving platform, $L_i$ denotes the lengths of the limbs, and $\alpha_i$ denotes the angles of the limbs. The position of the center of the moving platform $G$ is represented as $G = [x_G, y_G]$, and the orientation of the manipulator's moving platform is denoted by $\phi$ with respect to the fixed coordinate frame. Therefore, the manipulator possesses three degrees of freedom in task space denoted by $\boldsymbol{\chi} = [x_G, y_G, \phi]$, with one degree of redundancy in actuators.

### 4.4.2 Geometry of the Manipulator

For analytical purposes, as depicted in Figure 4.5, a fixed frame $O$ is affixed to the fixed base at point $O$, which serves as the center of the base point circle encompassing $A_i$'s. Additionally, a moving coordinate frame $G$ is attached to the manipulator's moving platform at point $G$. It is assumed that each point $A_i$ lies at a radial distance of $R_A$ from point $O$, and each point $B_i$ lies at a radial distance of $R_B$ from point $G$ in the $xy$ plane when the manipulator is positioned centrally.

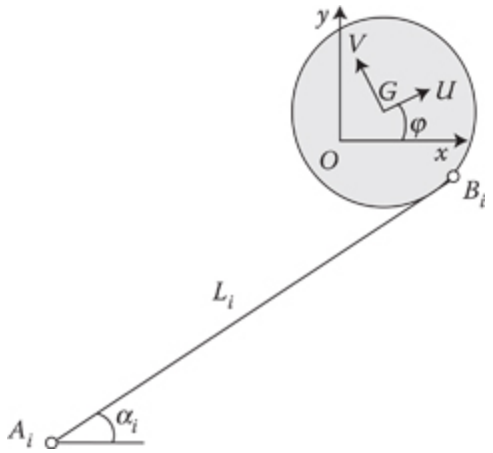



**FIGURE 4.5**
Kinematic configuration of the 4RPR manipulator. ⏎

To characterize the geometry of the manipulator, designate $\theta_{Ai}$ and $\theta_{Bi}$ as the absolute angles of points $A_i$ and $B_i$ at the central configuration of the manipulator, relative to the fixed frame $O$. Let's define the instantaneous orientation angle of $B_i$'s as follows:

$$\phi_i = \phi + \theta_{Bi}$$

(4.3)

Therefore, for each limb, $i = 1, 2, \ldots, 4$, the positions of the base points, $A_i$, are given by:

$$A_i = [R_A \cos(\theta_{Ai}), R_A \sin(\theta_{Ai})]^T.$$

(4.4)

### 4.4.3 Inverse Kinematics

For inverse kinematic analysis, it is assumed that the position and orientation of the moving platform $\boldsymbol{\chi} = [x_G, y_G, \phi]^T$ are given, and the objective is to determine the active joint variables of the manipulator, $\boldsymbol{L} = [L_1, L_2, L_3, L_4]^T$. From the geometry of the manipulator, as depicted in Figure 4.5, the loop closure equation for each limb, $i = 1, 2, \ldots, 4$, can be expressed as

$$\overrightarrow{A_iG} = \overrightarrow{A_iB_i} - \overrightarrow{GB_i}$$

(4.5)

Rewriting the vector loop closure component–wise,

$$\begin{aligned} x_G - x_{Ai} &= L_i \cos(\alpha_i) - R_B \cos(\phi_i) \end{aligned}$$

(4.6)

$$y_G - y_{Ai} \quad = L_i \sin(\alpha_i) - R_B \sin(\phi_i)$$

(4.7)

in which $\alpha_i$'s represent the absolute limb angles. To solve the inverse kinematic problem, the elimination of passive limb variables $\alpha_i$'s, from the above equations is necessary to solve for the active variables $L_i$'s. This can be achieved by reordering the equations as

$$L_i \cos(\alpha_i) \quad = x_G - x_{Ai} + R_B \cos(\phi_i)$$

(4.8)

$$L_i \sin(\alpha_i) \quad = y_G - y_{Ai} + R_B \sin(\phi_i)$$

(4.9)

By adding the squares of both sides of Equations 4.8 and 4.9, the active limb lengths are uniquely determined.

$$L_i = \left[ (x_G - x_{Ai} + R_B \cos(\phi_i))^2 + (y_G - y_{Ai} + R_B \sin(\phi_i))^2 \right]^1$$

(4.10)

Furthermore, the passive limb angles $\alpha_i$'s can be determined from the following equation

$$\alpha_i = \operatorname{atan} 2 \left[ (y_G - y_{Ai} + R_B \sin(\phi_i)), (x_G - x_{Ai} + R_B \cos(\phi_i) \right.$$

(4.11)

Therefore, at each specified moving platform pose $\boldsymbol{\chi} = [x_G, y_G, \phi]^T$, there exists a distinct solution set for the limb lengths $L_i$ and limb angles $\alpha_i$.

### 4.4.4 Forward Kinematics

In the forward kinematic problem, the joint variables $L_i$ are provided, and the goal is to determine the position and orientation of the moving platform $\boldsymbol{\chi} = [x_G, y_G, \phi]^T$. This can be achieved by eliminating $x_G$ and $y_G$ from Equations 4.6 and 4.7 as outlined below. Let's introduce two intermediate variables $x_i$ and $y_i$ defined as follows:

$$\begin{cases} x_i = -x_{Ai} + R_B \cos(\phi_i) \\ y_i = -y_{Ai} + R_B \sin(\phi_i) \end{cases}$$

(4.12)

To streamline the computations, let's consider the square of Equation 4.10:

$$L_i^2 = (x_G + x_i)^2 + (y_G + y_i)^2$$

(4.13)

Initially, we attempt to solve for $x_G$ and $y_G$. This can be achieved by rearranging equation 4.13 as follows:

$$x_G^2 + y_G^2 + r_i x_G + s_i y_G + u_i = 0,$$

(4.14)

in which for $i = 1, 2, \ldots, 4$,

$$r_i = 2x_i, \quad s_i = 2y_i, \quad u_i = x_i^2 + y_i^2 - L_i^2.$$

(4.15)

Equation 4.14 yields four quadratic relationships for $x_G$ and $y_G$ corresponding to $i = 1, 2, \ldots, 4$. Subtracting pairs of these equations from each other leads to linear equations in terms of $x_G$ and $y_G$.

$$\boldsymbol{A} \cdot \begin{bmatrix} x_G \\ y_G \end{bmatrix} = \boldsymbol{b}$$

(4.16)

in which,

$$\boldsymbol{A} = \begin{bmatrix} r_1 - r_2 & s_1 - s_2 \\ r_2 - r_3 & s_2 - s_3 \\ r_3 - r_4 & s_3 - s_4 \\ r_4 - r_1 & s_4 - s_1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} u_2 - u_1 \\ u_3 - u_2 \\ u_4 - u_3 \\ u_1 - u_4 \end{bmatrix}.$$

(4.17)

The components of $\boldsymbol{A}$ and $\boldsymbol{b}$ are functions of $\phi$. It seems that only two of the equations above would suffice to determine $x_G$ and $y_G$ in terms of $\phi$, but all four equations are utilized here to ensure tractable solutions even at singular configurations. As demonstrated in Appendix A, the overdetermined Equation 4.16 can be solved using the left pseudo-inverse.

$$\begin{bmatrix} x_G \\ y_G \end{bmatrix} = \boldsymbol{A}^{\dagger} \cdot \boldsymbol{b}, \tag{4.18}$$

where $A^{\dagger}$ denotes the left pseudo-inverse of $A$, which is computed from

$$\boldsymbol{A}^{\dagger} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T. \tag{4.19}$$

Tractable numerical methods such as Householder reflection or Cholesky decomposition[1] can be employed to compute the pseudo-inverse.

[1] Use `pinv` function in MATLAB® ⏎

It is important to note that the components of $\boldsymbol{A}$ and $\boldsymbol{b}$ are functions of $\phi$, and for a given $\phi$, Equation 4.18 provides the solution with the least square error for $x_G$ and $y_G$. To solve for $\phi$, this solution can be substituted into

Equation 4.14, resulting in a function involving only one unknown variable, $\phi$:

$$f_i(\phi) = x_G^2 + y_G^2 + r_i x_G + s_i y_G + u_i \tag{4.20}$$

Consider,

$$f(\phi) = \sum_{i=1}^{4} f_i(\phi) \tag{4.21}$$

and employ numerical methods with iterative search routines[1] to find the solution of $f(\phi) = 0$. While any function $f_i(\phi) = 0$ could theoretically be used to solve for the forward kinematics $\phi$, all four equations are summed here to ensure a tractable solution even at singular configurations.

The flowchart depicted in Figure 4.6 illustrates the iterative method for solving the forward kinematics. According to this flowchart, given a specific $\phi$, the values of $x_i$, $y_i$, and $\phi_i$ are computed using Equations 4.12 and 4.3, respectively. Subsequently, $r_i$, $s_i$, and $u_i$ are determined using equation 4.15, and matrices $\boldsymbol{A}$ and $\boldsymbol{b}$ are constructed based on equation 4.17. Using these matrices, the least squares solution for $x_i$ and $y_i$ is obtained from equation 4.19. The function $f(\phi)$ is then evaluated using equation

4.21, and if it deviates significantly from zero, an iterative search method is employed to compute a new value for $\phi$. This iterative process continues until a solution for $f(\phi) = 0$ is achieved with an accuracy level $\epsilon \ll 1$.

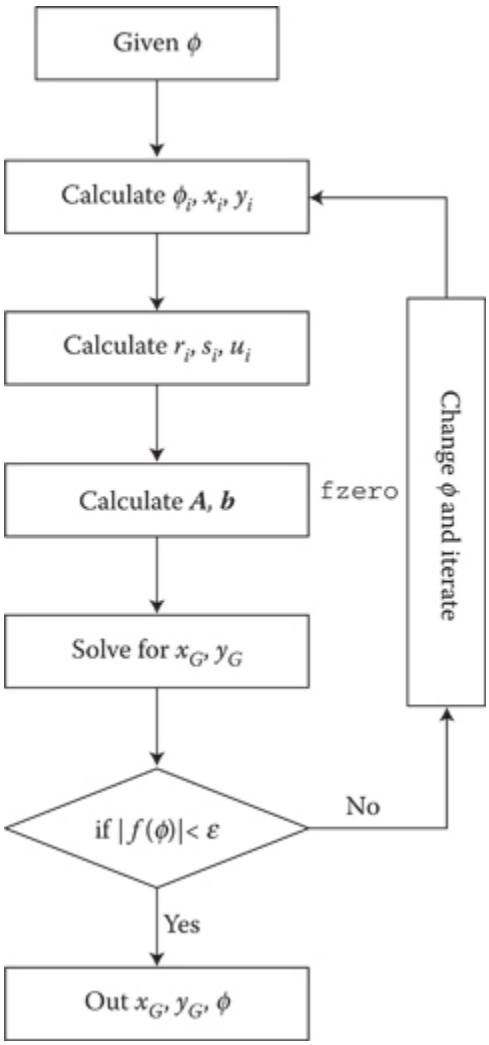



**FIGURE 4.6**

Flowchart of the process for solving the forward kinematics of the 4RPR manipulator. ⏎

[1]Use `fzero` function in MATLAB ⏎

To handle the possibility of multiple solutions for the equation $f(\phi) = 0$ and to ensure smooth transitions in the forward kinematic solutions, the numerical routine utilizes the solution from the previous iteration as the starting point for finding the next solution. The simulation results, detailed in section 4.4.5, demonstrate the effectiveness and accuracy of these numerical methods for solving forward kinematics. If it becomes necessary to determine the limb angles $\alpha_i$, Equation 4.11 can be directly applied by substituting the values of $x_G$, $y_G$, and $\phi$.

### 4.4.5 Simulations

The geometric parameters used in the simulations are adopted from a cable-driven redundant parallel manipulator design [84], detailed in Table 4.1. This manipulator design originates from the Canadian concept of next-generation telescopes [85, 86]. Due to its specific application, the limb lengths of this manipulator are considerably longer than those of conventional manipulators, resulting in a significantly larger workspace.

**TABLE 4.1**

Geometric parameters of the 4RPR cable-driven parallel manipulator ⏎

| Description | Quantity |
|---|---|
| $R_A$: Radius of the fixed base points $A_i$'s | $900m$ |

| Description | Quantity |
|---|---|
| $R_B$: Radius of the moving platform points $B_i$'s | $10m$ |
| $\theta_{A_i}$: Angle of the fixed base points $A_i$'s | $[-\frac{3\pi}{4}, -\frac{\pi}{4}, \frac{\pi}{4}, \frac{3\pi}{4}]$ |
| $\theta_{B_i}$: Angle of the moving platform points $B_i$'s | $[-\frac{\pi}{4}, -\frac{3\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{4}]$ |

For the simulations, a typical trajectory $\boldsymbol{\chi}_d$ for the 4RPR manipulator is considered and depicted in Figure 4.7. This trajectory spans 200 meters in the $x$ direction, 100 meters in the $y$ direction, and includes a 45-degree rotation of the moving platform $\phi$ over ten seconds. Using Equation 4.10, the inverse kinematic solution of the manipulator is computed, leading to the unique determination of the manipulator limb lengths, as illustrated in Figure 4.8. It is evident from this figure that the manipulator limb lengths vary significantly, accommodating the variations in the desired trajectory.

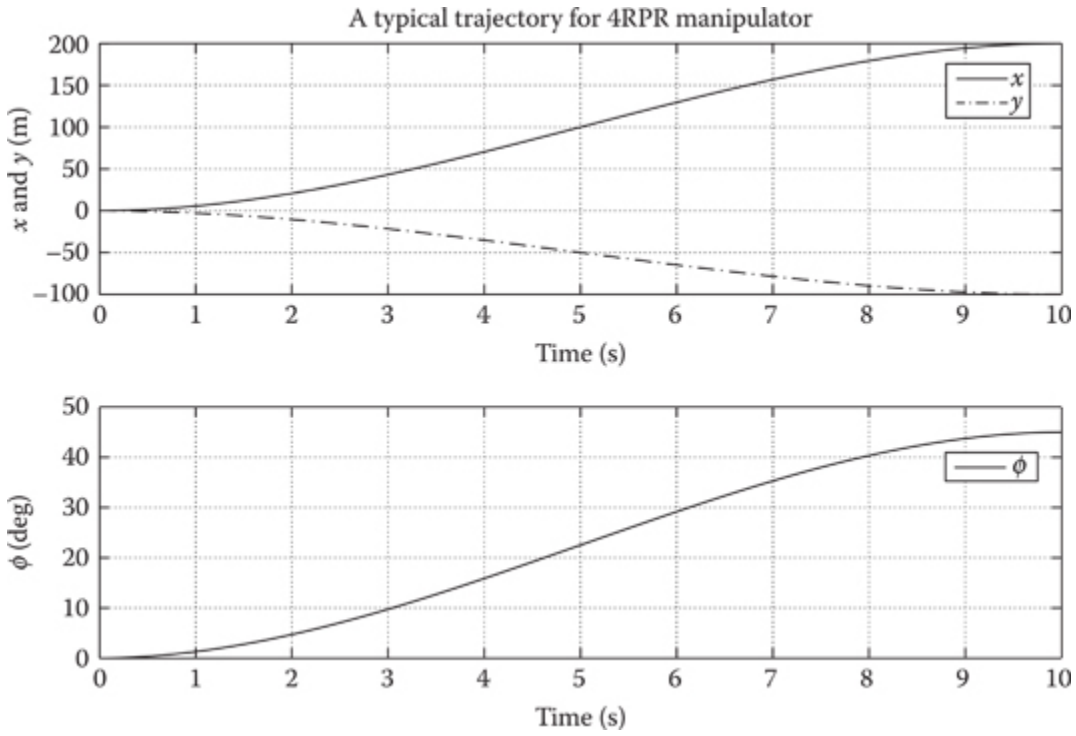
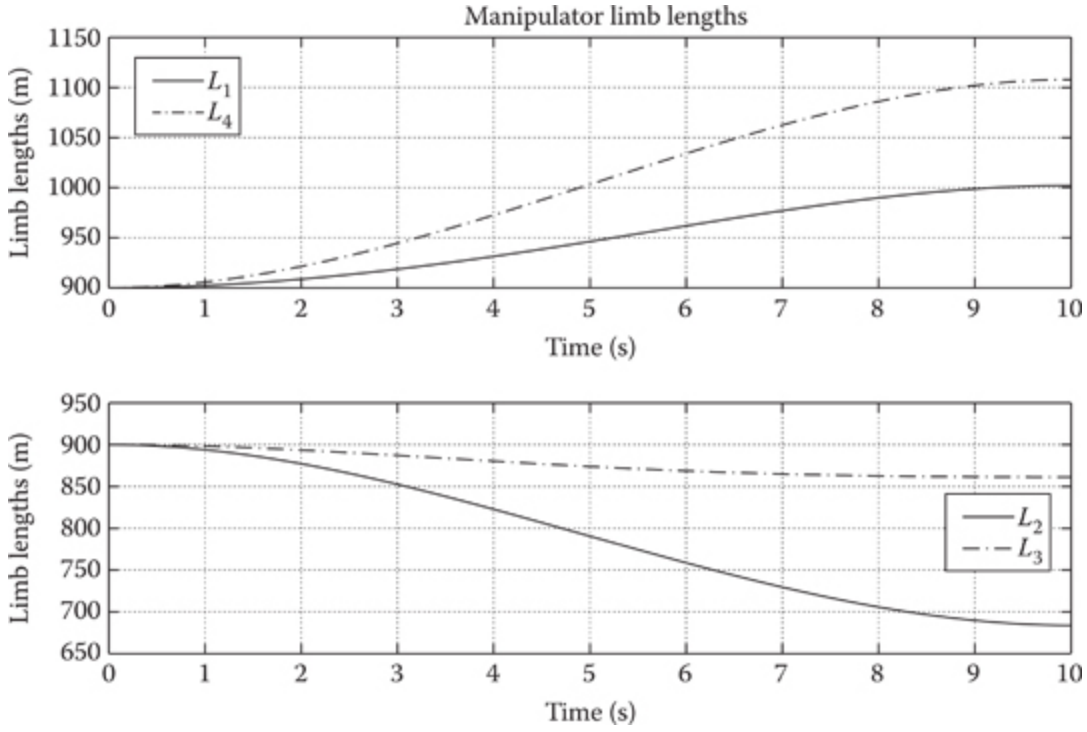



**FIGURE 4.7**

A typical trajectory for the 4RPR parallel manipulator maneuver. ⏎




▶ Long Description for Figure 4.8

**FIGURE 4.8**

Inverse kinematic solution for the given trajectory. ⏎

As depicted in Figure 4.9, the verification process involves determining the forward kinematic solution of the manipulator using the provided limb lengths to assess the accuracy and reliability of numerical solutions. The discrepancies between the computed numerical solutions $\boldsymbol{\chi}_c$ and the desired trajectory $\boldsymbol{\chi}_d$ are illustrated in Figure 4.10. It is observed that the computed numerical solutions from the forward kinematic problem closely match the desired trajectories, achieving an accuracy of $10^{-13}$. This confirms the precision and reliability of the numerical

methods employed to obtain the forward kinematic solution.

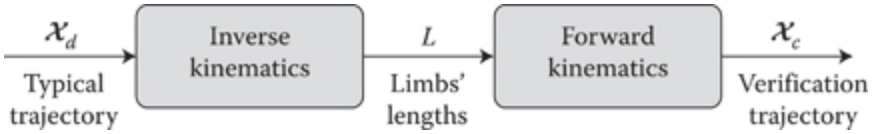



**FIGURE 4.9**
Mechanism of forward kinematics verification for a given trajectory. ⏎

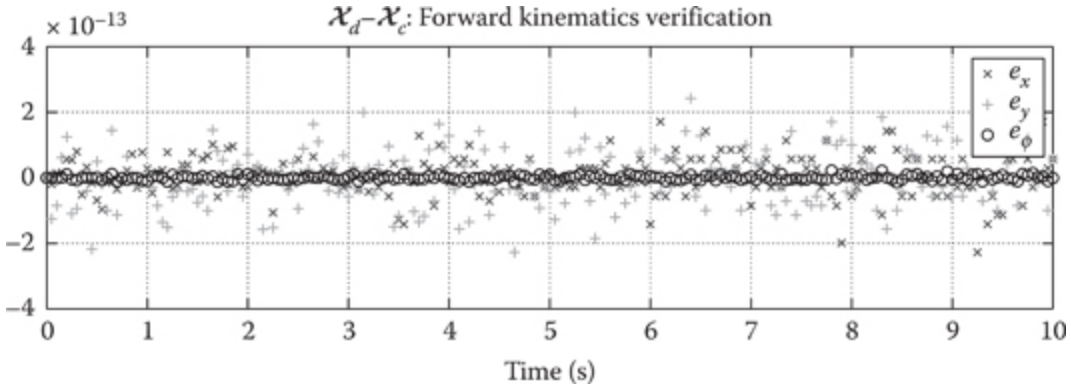



**FIGURE 4.10**
The accuracy and reliability of the forward and inverse kinematic solutions. ⏎

It is important to note that forward kinematic solutions are not always unique. To maintain consistency and avoid convergence to alternative solutions at each iteration, the forward kinematic solution from the previous step is utilized as the initial guess for subsequent iterations. This iterative approach ensures that the numerical solution converges to the correct solution at all examined points.

## 4.5 Case Study II: Shoulder Manipulator

In this section, we elaborate on the detailed kinematic analysis of a three-degrees-of-freedom spatial orientation manipulator. Initially, we describe the kinematic structure of the mechanism, followed by a thorough examination of its geometry. Subsequently, we analyze both the inverse and closed-form forward kinematics of the manipulator. To validate our formulations, simulation results for computing inverse and forward kinematics are presented, where the accuracy of the forward kinematic solution is verified.

### 4.5.1 Mechanism Description

This section is devoted to complete kinematic analysis for a three-degrees-of-freedom parallel manipulator with one degree of actuator redundancy. Known as the “shoulder manipulator,” this mechanism which was designed by late Prof. Vincent Hayward [87, 88] in 1992, draws design inspiration from biological shoulders. Its intriguing features, reminiscent of the human shoulder, contribute to its unique design characteristics and suitability as a proper experimental setup in parallel robotics research.

In the anatomy of a human shoulder, there are at least four muscle groups identified to generate a pure rotational motion for the shoulder [89]. These muscle groups are used to keep the socket and ball structure of the shoulder tight while providing the joint with three independent rotational

movements [90]. The use of at least one redundant muscle group in the human shoulder inspires the engineers to design an artificial shoulder manipulator with one degree of actuation redundancy [88]. This redundancy increases the manipulator stiffness, and dexterity, it enlarges the singular–free workspace of the manipulator [91].

A CAD model of the mechanism and an image of the manipulator in a twisted configuration is shown in Figure 4.11. This figure illustrates how the motion of the moving platform is restricted to three rotational degrees of freedom (3R). The motion is generated by four high-performance hydraulic piston actuators, and therefore, this design includes one degree of actuator redundancy, similar to the structure of a human shoulder. Each actuator is equipped with a Linear Variable Differential Transformer (LVDT) position sensor and an embedded Hall effect force sensor. The manipulator utilizes simple revolute joints to construct spherical and universal joints, and all four limbs have identical kinematic structure. The four active limbs follow a UPU kinematic structure, where each pair of limbs connects to the moving platform via a single connecting link.

**FIGURE 4.11**
A CAD model and an image of the shoulder manipulator in a twisted configuration. ⏎

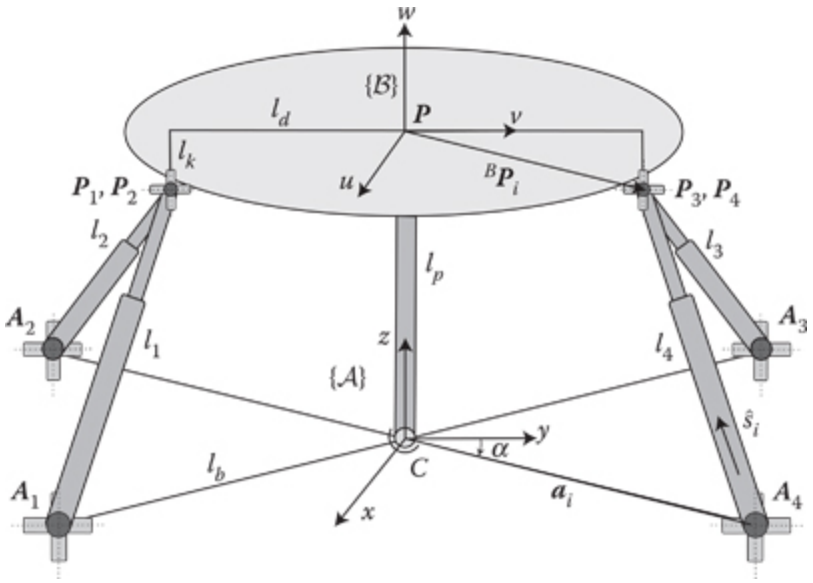
As shown in Figure 4.11, the base link $L_1$ is connected to the cylinder link $L_3$ via a connecting link $L_2$. Consequently, the universal joint in the manipulator can be broken down into two binary revolute joints, $R_1$ and $R_2$. The prismatic joint is labeled $P_3$ in this figure, and the universal joint that connects the piston link $L_4$ to the moving platform $L_6$ is also divided into two binary revolute joints, $R_4$ and $R_5$. Notably, each pair of piston actuators shares a connecting link, designated as $L_5$ in the figure. Additionally, a passive limb connects the moving platform to the fixed base via a spherical joint $S_6$, which prevents pure translations of the moving platform. The spherical joint $S_6$ is constructed from three perpendicular revolute joints, as well.

From a structural perspective, the shoulder manipulator can be viewed as part of a lightweight, seven-degrees-of-freedom robotic arm capable of carrying loads several times its own weight. The workspace of this mechanism resembles a portion of a spherical surface, with orientation angles constrained between $-\pi/6$ and $\pi/6$. There are no sensors available to measure the orientation angles of the moving platform, highlighting the importance of forward kinematic analysis for the position control of the shoulder manipulator. In this control scheme, LVDT position sensors are the sole source of measurement. A comprehensive kinematic analysis of the manipulator is essential to understand the structure's characteristics and to design appropriate control laws for achieving the desired tracking performance.

### 4.5.2 Geometry of the Manipulator

Figure 4.12 depicts a geometric model for the shoulder manipulator. The parameters used in the kinematic analysis can be defined as follows:

$$l_b = \| \overrightarrow{CA_i} \|, \quad l_p = \| \overrightarrow{CP} \|, \quad l_d = \| \overrightarrow{PP_i} \|_v, \quad l_k = \| \overrightarrow{PP_i} \|_w, \tag{4.22}$$




▶ Long Description for Figure 4.12

**FIGURE 4.12**

The Geometry of the shoulder manipulator. ⏎

where the latter two norms respectively denote the components of $\| \overrightarrow{PP_i} \|$ in the $v$ and $w$ directions. Furthermore, $\alpha$ is the angle between $CA_4$ and $y$, $C$ is the center of the reference frame, $P$ is the center of the moving platform, $l_i$ is the actuator length for $i = 1, 2, 3, 4$, $P_i$ represents the moving endpoints of the actuators, and $A_i$ denotes the fixed endpoints of the actuators.

Two coordinate frames are defined for the purpose of analysis. The base coordinate frame $\{\mathscr{A}\} : xyz$ is attached to the fixed base at point $C$ (rotation center) with its $z$-axis perpendicular to the plane defined by the actuator base points $A_1, A_2, A_3, A_4$, and its $x$-axis parallel to the bisector of angle $\angle A_1 C A_4$. The second coordinate frame $\{\mathscr{B}\} : uvw$

is attached to the center of the moving platform $P$ with its $w$-axis perpendicular to the line defined by the actuators' moving endpoints $P_1P_3$ along the passive limb. It is assumed that the actuators' fixed endpoints lie on the same plane as the rotation center $C$. The position of the moving platform center $P$ is defined by:

$$^{A}\boldsymbol{P} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T \tag{4.23}$$

Moreover, a rotation matrix $^{A}\boldsymbol{R}_B$ is used to define the orientation of the moving platform with respect to the base frame, using a pitch-roll-yaw representation:

$$\begin{aligned} ^{A}\boldsymbol{R}_B &= \boldsymbol{R}_z(\theta_z)\boldsymbol{R}_y(\theta_y)\boldsymbol{R}_x(\theta_x) \\ &= \begin{bmatrix} c\theta_z c\theta_y & c\theta_z s\theta_y s\theta_x - s\theta_z c\theta_x & c\theta_z s\theta_y c\theta_x + s\theta_z s\theta_x \\ s\theta_z c\theta_y & s\theta_z s\theta_y s\theta_x + c\theta_z c\theta_x & s\theta_z s\theta_y c\theta_x - c\theta_z s\theta_x \\ -s\theta_y & c\theta_y s\theta_x & c\theta_y c\theta_x \end{bmatrix} \end{aligned} \tag{4.24}$$

where $\theta_x, \theta_y,$ and $\theta_z$ are the orientation angles of the moving platform, denoting rotations of the moving frame about the fixed $x$, $y$, and $z$ axes, respectively. Furthermore, $c\theta$ and $s\theta$ are shorthand notations representing $\cos(\theta)$ and $\sin(\theta)$, respectively.

Although pitch-roll-yaw Euler angles are a convenient way to represent the orientation of the moving platform,

they are not very practical for solving the forward kinematics. A thorough kinematic analysis of this manipulator is reported in [92, 93, 94]. Using the pitch-roll-yaw representation for orientation leads to a complex forward kinematic formulation for the system, which can be solved numerically using neural networks [94], Taylor series expansion, or a quasi-closed solution [92]. However, using an alternative representation for orientation provides a closed-form solution for the forward kinematics [93].

This alternative is based on the kinematic decomposition of the spherical joint into three perpendicular binary joints. As shown in 4.11, the spherical joint $S_6$ is constructed from three perpendicular revolute joints intersecting at a point. These rotations can be represented by *u*-*v*-*w* Euler angles as follows:

$$\begin{aligned} \boldsymbol{R}_{uvw}(\theta_1, \theta_2, \theta_3) &= \boldsymbol{R}_u(\theta_1)\, \boldsymbol{R}_v(\theta_2)\, \boldsymbol{R}_w(\theta_3) \\ &= \begin{bmatrix} c_3c_2 & -s_3c_2 & s_2 \\ s_3c_1 + c_3s_2s_1 & c_3c_1 - s_3s_2s_1 & -c_2s_1 \\ s_3s_1 - c_3s_2c_1 & c_3s_1 + s_3s_2c_1 & c_2c_1 \end{bmatrix} \end{aligned} \tag{4.25}$$

Therefore, the position and orientation of the moving platform are fully defined by six variables, among which only three orientation angles $\theta_1, \theta_2$, and $\theta_3$ are specified as the independent motion variables of the shoulder manipulator.

### 4.5.3 Inverse Kinematics

Although the shoulder manipulator is redundantly actuated, its inverse kinematics can be easily solved given the orientation of the moving platform. In the inverse kinematic solution, the lengths of the actuators $l_i$ must be determined as functions of the manipulator orientation variables $\theta_1, \theta_2$, and $\theta_3$. To find the lengths $l_i$ of the actuators, we write the loop-closure equation for each actuated limb.

$$\boldsymbol{q}_i = l_i\, \hat{\boldsymbol{s}}_i = {}^{A}\boldsymbol{P} + {}^{A}\boldsymbol{R}_B\, {}^{B}\boldsymbol{P}_i - \boldsymbol{a}_i \tag{4.26}$$

where $l_i$ is the length of the $i$'th actuated limb and $\hat{\boldsymbol{s}}_i$ is the unit vector pointing along the direction of the $i$'th actuated limb. Moreover, ${}^{A}\boldsymbol{R}_B$ is the rotation matrix of the moving platform, and ${}^{A}\boldsymbol{P}$ is its center position vector:

$${}^{A}\boldsymbol{P} = l_p[s_2 \quad -c_2 s_1 \quad c_2 c_1]^T. \tag{4.27}$$

Furthermore, as depicted in Figure 4.12, vectors $\boldsymbol{a}_i$ and ${}^{B}\boldsymbol{P}_i$ represent the fixed endpoints of the actuators $(A_i)$ in the base frame and the moving attachment points of the actuators, respectively. Hence, for the given geometry of the manipulator, we have:

$$
\begin{aligned}
\boldsymbol{a}_1 &= l_b[+\sin\alpha \quad -\cos\alpha \quad 0]^T,\\
\boldsymbol{a}_2 &= l_b[-\sin\alpha \quad -\cos\alpha \quad 0]^T,\\
\boldsymbol{a}_3 &= l_b[-\sin\alpha \quad +\cos\alpha \quad 0]^T,\\
\boldsymbol{a}_4 &= l_b[+\sin\alpha \quad +\cos\alpha \quad 0]^T,
\end{aligned}
\tag{4.28}
$$

and

$$
\begin{aligned}
{}^B\boldsymbol{P}_1 = {}^B\boldsymbol{P}_2 &= [0 \quad -l_d \quad l_p - l_k]^T,\\
{}^B\boldsymbol{P}_3 = {}^B\boldsymbol{P}_4 &= [0 \quad +l_d \quad l_p - l_k]^T.
\end{aligned}
\tag{4.29}
$$

The actuator lengths, $l_i$, can be easily computed by taking the dot product of Equation 4.26 with itself.

$$
\boldsymbol{q}_i^T\boldsymbol{q}_i = l_i^2 = \left[{}^A\boldsymbol{P} + {}^A\boldsymbol{R}_B\,{}^B\boldsymbol{P}_i - \boldsymbol{a}_i\right]^T \left[{}^A\boldsymbol{P} + {}^A\boldsymbol{R}_B\,{}^B\boldsymbol{P}_i - \boldsymbol{a}_i\right]
\tag{4.30}
$$

Writing Equation 4.30 four times with the corresponding parameters given in Equations 4.25, 4.27 to 4.29, and simplifying the results yields to:

$$
l_1^2 \quad = k_1 + k_2 s_2 - k_3 c_2 s_1 + k_4 s_3 c_2 + k_5(c_3 c_1 - s_3 s_2 s_1),
$$

(4.31)

$$l_2^2 = k_1 - k_2 s_2 - k_3 c_2 s_1 - k_4 s_3 c_2 + k_5 (c_3 c_1 - s_3 s_2 s_1),$$

(4.32)

$$l_3^2 = k_1 - k_2 s_2 + k_3 c_2 s_1 + k_4 s_3 c_2 + k_5 (c_3 c_1 - s_3 s_2 s_1),$$

(4.33)

$$l_4^2 = k_1 + k_2 s_2 + k_3 c_2 s_1 - k_4 s_3 c_2 + k_5 (c_3 c_1 - s_3 s_2 s_1).$$

(4.34)

in which,

$$\begin{aligned} k_1 &= 4l_p^2 + l_d^2 + l_k^2 + l_b^2 - 4l_p\, l_k \\ k_2 &= 2l_b\,(l_k - 2l_p)\sin\alpha \\ k_3 &= 2l_b\,(2l_p - l_k)\cos\alpha \\ k_4 &= -2l_d\, l_b \sin\alpha \\ k_5 &= -2l_d\, l_b \cos\alpha \end{aligned}$$

(4.35)

Therefore, the actuator lengths are uniquely determined by Equations 4.31 to 4.34 as functions of the moving platform orientation variables.

### 4.5.4 Forward Kinematics

In the forward kinematic analysis of the shoulder manipulator, we aim to determine all possible orientations of the moving platform for a given set of actuated limb lengths. Equations 4.31 to 4.34 can be utilized to find the forward kinematic solution of the hydraulic shoulder, where the actuator lengths serve as the input variables. Given that the manipulator is redundantly actuated, we encounter four nonlinear equations to solve for three unknowns. To begin, let's express the position and orientation of the moving platform in terms of the joint variables $\theta_1, \theta_2,$ and $\theta_3$ using Equations 4.25 and 4.27. Subtract Equation 4.32 from Equation 4.31, and similarly, subtract Equation 4.34 from Equation 4.33, to simplify the equations to:

$$l_1^2 - l_2^2 \quad = +2k_2\, s_2 + 2k_4\, s_3\, c_2, \tag{4.36}$$

$$l_3^2 - l_4^2 \quad = -2k_2\, s_2 + 2k_4\, s_3\, c_2. \tag{4.37}$$

Subtract Equation 4.37 from Equation 4.36, and solve for $s_2$:

$$s_2 = \frac{l_1^2 - l_2^2 - l_3^2 + l_4^2}{4k_2}.$$

(4.38)

Substituting Equation 4.38 into the trigonometric identity $s_2^2 + c_2^2 = 1$ results in two solutions for $c_2$.

$$c_2 = \pm\sqrt{1 - s_2^2}$$

(4.39)

Having $s_2$ and $c_2$ in hand, we can solve for $s_3$ from Equation 4.36 as:

$$s_3 = \frac{l_1^2 - l_2^2 - 2k_2\, s_2}{2k_4\, c_2} \quad \text{or} \quad \frac{l_3^2 - l_4^2 + 2k_2\, s_2}{2k_4\, c_2}.$$

(4.40)

Similarly,

$$c_3 = \pm\sqrt{1 - s_3^2}$$

(4.41)

Up to this point, four possible solutions for the forward kinematics can be generated. To solve for the remaining unknowns, $c_1$ and $s_1$, subtract Equation 4.31 from Equation

4.33, and similarly, subtract Equation 4.32 from Equation 4.34:

$$l_3^2 - l_1^2 \quad = 2k_3\, c_2\, s_1 - 2k_2\, s_2, \tag{4.42}$$

$$l_4^2 - l_2^2 \quad = 2k_3\, c_2\, s_1 + 2k_2\, s_2. \tag{4.43}$$

Hence,

$$s_1 = \frac{l_3^2 - l_1^2 + 2k_2\, s_2}{2k_3\, c_2} \quad \text{or} \quad \frac{l_4^2 - l_2^2 - 2k_2\, s_2}{2k_3\, c_2}. \tag{4.44}$$

and,

$$c_1 = \pm\sqrt{1 - s_1^2} \tag{4.45}$$

Therefore, a total of eight solutions for the forward kinematics can be summarized as follows:

$$\theta_1 = \operatorname{atan2}(s_1, c_1), \quad \theta_2 = \operatorname{atan2}(s_2, c_2), \quad \theta_3 = \operatorname{atan2}(s_3, c_3) \tag{4.46}$$

If $c_2 = 0$, the solutions degenerate because in this case, two independent values for $\theta_1$ and $\theta_3$ cannot be determined uniquely. If $c_2 = 0$ and $s_2 = 1$, then

$$\theta_3 + \theta_1 = \cos^{-1} \frac{l_1^2 - k_1 - k_2}{k_5} \quad \text{or} \quad \cos^{-1} \frac{l_2^2 - k_1 + k_2}{k_5} \tag{4.47}$$

Otherwise, if $c_2 = 0$ and $s_2 = -1$, then

$$\theta_3 - \theta_1 = \cos^{-1} \frac{l_1^2 - k_1 + k_2}{k_5} \quad \text{or} \quad \cos^{-1} \frac{l_2^2 - k_1 - k_2}{k_5} \tag{4.48}$$

Furthermore, the moving platform position ${}^A\boldsymbol{P}$ and orientation ${}^A\boldsymbol{R}_B$ can be found using Equations 4.27 and 4.25. If there is a need to represent the orientation with pitch-roll-yaw Euler angles, we can solve for $\theta_x, \theta_y$, and $\theta_z$ using 4.24, completing the solution process for the forward kinematics of the shoulder manipulator.

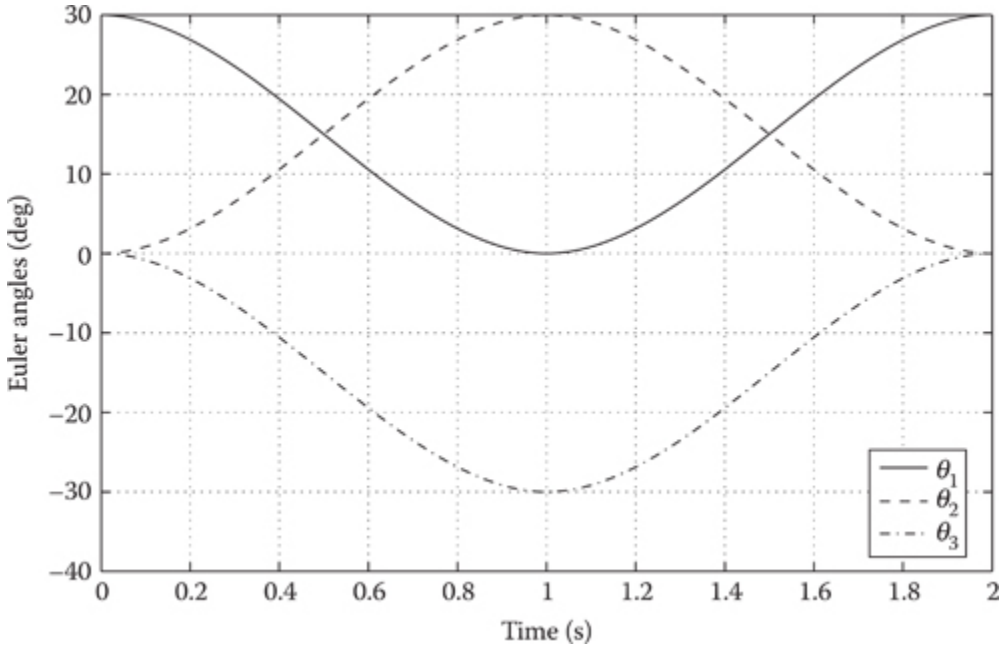
It is important to note that, as mentioned earlier, there are eight solutions for the forward kinematics due to the involvement of several square roots in the process, and undesired solutions must be identified and omitted. However, within the limited workspace of the shoulder manipulator, where the orientation angles are confined to

$\pm\pi/6$, the positive square roots in Equations 4.45, 4.39, and 4.41 yield the correct solutions.

### 4.5.5 Simulations

To verify the results of the forward kinematic solution, a simulation study was conducted using a sample trajectory within the reachable workspace of the manipulator. Both the provided inverse and forward kinematic solutions were tested along this trajectory, and their results were compared. The trajectory was defined as a smooth motion specified by the desired orientation of the moving platform. A sample trajectory was generated using a cubic polynomial based on initial and final Euler angles and the maneuver time of two seconds (refer to Section 10.2 for details on cubic trajectory planning).

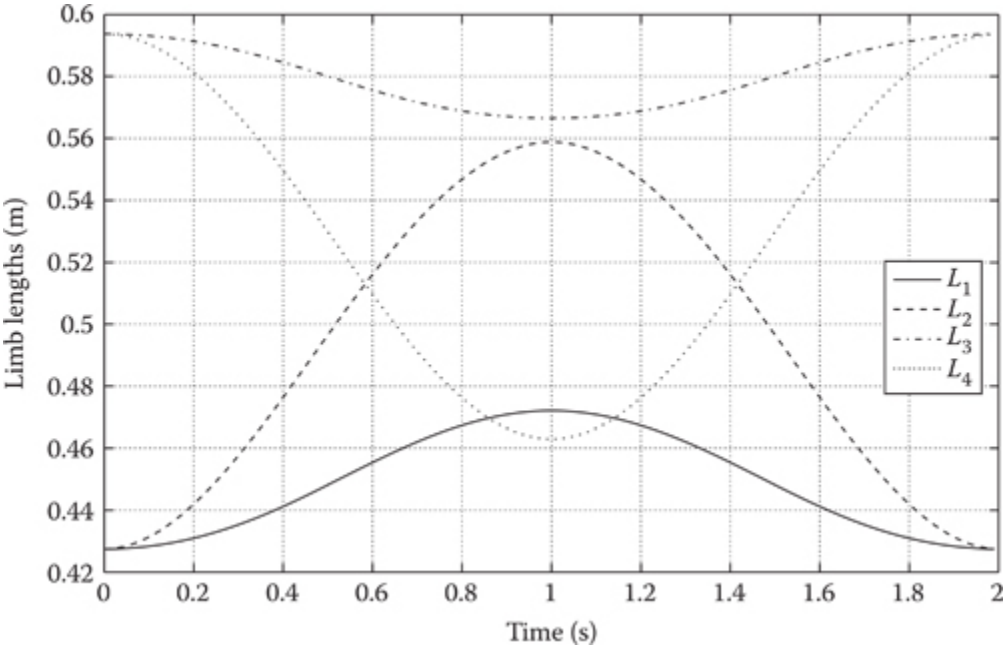
Figure 4.13 displays the sample trajectory for the shoulder manipulator, represented by orientation angles in u-v-w Euler angles. As shown in this figure, the Euler angles vary within $\pm 30$ degrees, staying within the reachable workspace of the shoulder manipulator. For this trajectory, the inverse kinematic solution of the manipulator was computed using Equations 4.31 to 4.34. The lengths of the manipulator limbs were uniquely determined and are illustrated in Figure 4.14. As depicted in this figure, the lengths of the manipulator limbs vary by a few centimeters in response to variations in the desired trajectory




▶ Long Description for Figure 4.13

**FIGURE 4.13**

A typical trajectory of the hydraulic shoulder depicted for u–v–w Euler angles. ⏎




▶ Long Description for Figure 4.14

**FIGURE 4.14**

Inverse kinematic solution of the hydraulic shoulder for the given trajectory. ⏎

To verify the accuracy of the calculations, the forward kinematic solutions of the manipulator were computed using Equation 4.46 for the given limb lengths. The difference between the calculated forward kinematic solutions $\theta_c$ and the desired motion $\theta_d$ is illustrated in Figure 4.15. It is observed that the calculated forward kinematic solutions are virtually identical to the desired trajectories, with an accuracy of up to $10^{-14}$. These simulations confirm the precision and reliability of the forward kinematic formulations.

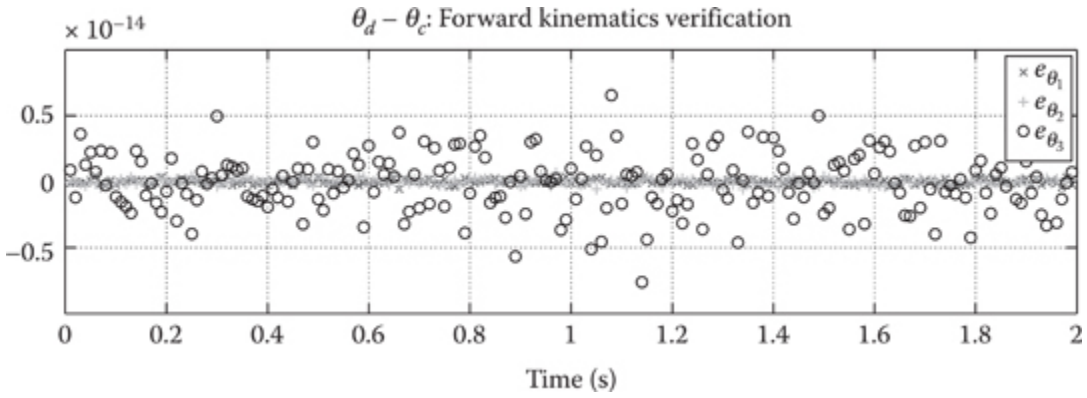



**FIGURE 4.15**

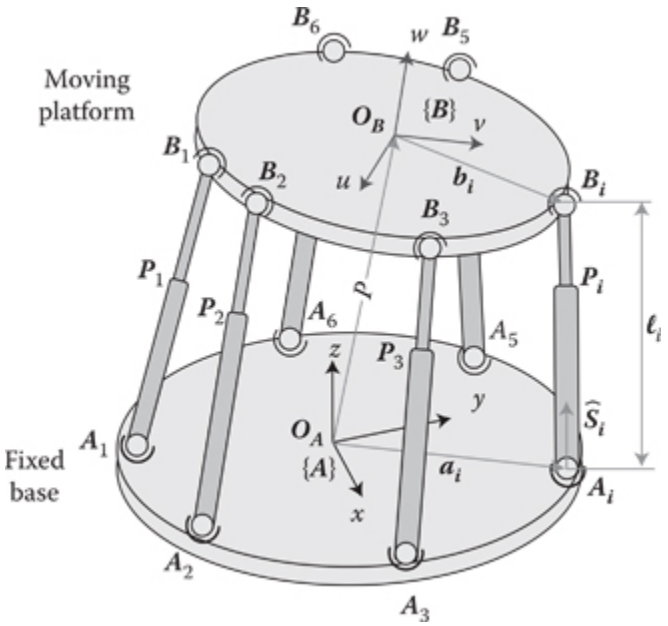
The precision and reliability of the forward and inverse kinematic solutions for hydraulic shoulder manipulator. ⏎

## 4.6 Case Study III: Stewart–Gough Platform

In this section, the kinematic analysis of the Stewart-Gough platform is studied in detail. The analysis begins with an explanation of the manipulator's kinematic structure, followed by a description of the system's geometry. Subsequently, both inverse and forward kinematic analyses of the manipulator are reported and discussed.

### 4.6.1 Mechanism Description

Figure 4.16 illustrates the schematics of a Stewart-Gough platform (SGP). This manipulator achieves spatial motion of the moving platform through six identical piston-cylinder actuators. Each actuator comprises two parts connected by a prismatic joint, typically driven by hydraulic actuators or linear motors to generate prismatic motion in the limbs. The actuators connect the fixed base to the moving platform via spherical joints located at points $A_i$ and $B_i$, where $i = 1, 2, \ldots, 6$, respectively. It's important to note that in Figure 4.16, all attachment points $A_i$ lie in the base plane, and all $B_i$ lie in the moving platform plane. However, in a general Stewart-Gough platform, the attachment points are not necessarily confined to lie in a single plane.




▶ Long Description for Figure 4.16

**FIGURE 4.16**

Schematics of a Stewart-Gough Platform with attached frames to the fixed base and moving platform. ⏎

As depicted in Figure 4.16, frame $\{\mathscr{A}\}$ is attached to the fixed base at point $O_A$, and frame $\{\mathscr{B}\}$ is attached to the moving platform at point $O_B$. The kinematic structure of each limb follows an SPS arrangement. As discussed in Example 1.3 in Chapter 1, this manipulator consists of 14 links connected by 6 prismatic joints and 12 spherical joints.

In this configuration, each SPS kinematic structure introduces one passive degree of freedom. Therefore, this spatial manipulator possesses six degrees of freedom, enabling it to function as a general parallel manipulator for complete spatial movement: three degrees of freedom in

position and three degrees of freedom in orientation. The number of actuators equals the degrees of freedom of the manipulator, confirming its fully parallel nature.

Given that all limbs are connected to both the moving platform and the base via spherical joints, no twisting torques can be transmitted through the limbs. Consequently, the forces transmitted along each limb are directed about its longitudinal axis.

### 4.6.2 Geometry of the Manipulator

As depicted in Figure 4.16, for the purpose of kinematic analysis, the geometry of the manipulator is defined using two coordinate frames: one attached to the fixed base and another attached to the moving platform. The fixed frame $\{\mathscr{A}\}$ is fixed to the base at point $O_A$, and the positions of the attachment points on the fixed base are denoted by vectors $\boldsymbol{a}_i$. Similarly, the moving frame $\{\mathscr{B}\}$ is attached to the moving platform at point $O_B$, and the positions of the moving attachment points are denoted by vectors $\boldsymbol{b}_i$.

Each limb's geometry is characterized by its length $\ell_i$ and its direction, represented by a unit vector $\hat{\boldsymbol{s}}_i$. The position of point $O_B$ on the moving platform is described by the position vector ${}^{A}\boldsymbol{P} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$, and the orientation of the moving platform is represented by the rotation matrix ${}^{A}\boldsymbol{R}_B$. The rotation matrix ${}^{A}\boldsymbol{R}_B$ can be expressed component-wise using the unit vectors $\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}$, and $\hat{\boldsymbol{w}}$, defined as follows:

$$ {}^{A}\boldsymbol{R}_{B} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} $$

(4.49)

Note that the rotation matrix ${}^{A}\boldsymbol{R}_{B}$ is orthonormal and satisfies the conditions detailed in Equations 2.6 to 2.8. The orientation of the moving platform can alternatively be represented using screw axis representation or various Euler angles. In the forward kinematic analysis of the Stewart-Gough platform reported in this book, the following screw representation is utilized.

$$ \left\{ \hat{\boldsymbol{s}} = \begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T \, , \, \theta \right\} $$

(4.50)

The Stewart-Gough platform can be classified into various types based on the geometry of its moving platform and base attachments. A well-studied class in the literature involves platforms where both attachment points of the six limbs, $A_i$ and $B_i$, are coplanar. In such configurations, the centers of the ball joints typically form two hexagonal patterns on both the top platform and the base.

When ball joints overlap with each other, the number of distinct attachment points where the six limbs connect to the moving and base platforms decreases. If the number of

attachment points on the base is denoted as $N$, and on the moving platform as $n$, the Stewart-Gough platform is categorized as a type $N-n$. Lower values of $n$ generally simplify the forward kinematics solutions, but designs with concentric ball joints can introduce complexities.

To mitigate the challenges posed by concentric ball joints, designers can separate the ball joints of the limbs to increase the type number while maintaining the overall geometric shape of the platform. This approach helps in achieving a higher type platform configuration without relying on concentric ball joints, thereby addressing potential design constraints.

### 4.6.3 Inverse Kinematics

For the inverse kinematic analysis, it is assumed that the position ${}^{A}\boldsymbol{P}$ and orientation of the moving platform ${}^{A}\boldsymbol{R}_{B}$ are given, and the objective is to determine the joint variables $\boldsymbol{q}=\left[\ell_1,\ell_2,\ell_3,\ell_4,\ell_5,\ell_6\right]^{T}$. From the manipulator's geometry, as depicted in Figure 4.16, the loop closure equation for each limb, where $i=1,2,\ldots,6$, can be expressed as follows:

$$\begin{aligned}\ell_i\,{}^{A}\hat{\boldsymbol{s}}_i &= {}^{A}\boldsymbol{P}+{}^{A}\boldsymbol{b}_i-{}^{A}\boldsymbol{a}_i\\ &= {}^{A}\boldsymbol{P}+{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{b}_i-{}^{A}\boldsymbol{a}_i\end{aligned} \tag{4.51}$$

To determine the length of each actuator and eliminate $\hat{\boldsymbol{s}}_i$, it is sufficient to dot multiply each side by itself:

$$\ell_i^2 \left[ {}^A\hat{\boldsymbol{s}}_i{}^T {}^A\hat{\boldsymbol{s}}_i \right] = \left[ {}^A\boldsymbol{P} + {}^A\boldsymbol{R}_B {}^B\boldsymbol{b}_i - {}^A\boldsymbol{a}_i \right]^T \left[ {}^A\boldsymbol{P} + {}^A\boldsymbol{R}_B {}^B\boldsymbol{b}_i - \right. \tag{4.52}$$

or,

$$\begin{aligned} \ell_i^2 \quad = {}^A\boldsymbol{P}^T {}^A\boldsymbol{P} + {}^B\boldsymbol{b}_i^T {}^B\boldsymbol{b}_i + {}^A\boldsymbol{a}_i^T {}^A\boldsymbol{a}_i - 2\,{}^A\boldsymbol{P}^T {}^A\boldsymbol{a}_i + \\ 2\,{}^A\boldsymbol{P}^T \left[ {}^A\boldsymbol{R}_B {}^B\boldsymbol{b}_i \right] - 2 \left[ {}^A\boldsymbol{R}_B {}^B\boldsymbol{b}_i \right]^T {}^A\boldsymbol{a}_i. \end{aligned} \tag{4.53}$$

Therefore, for $i = 1, 2, \ldots, 6$, each limb length can be uniquely determined by:

$$\begin{aligned} \ell_i = \Big[ {}^A\boldsymbol{P}^T {}^A\boldsymbol{P} + {}^B\boldsymbol{b}_i^T {}^B\boldsymbol{b}_i + {}^A\boldsymbol{a}_i^T {}^A\boldsymbol{a}_i - 2\,{}^A\boldsymbol{P}^T {}^A\boldsymbol{a}_i \\ + 2\,{}^A\boldsymbol{P}^T \left[ {}^A\boldsymbol{R}_B {}^B\boldsymbol{b}_i \right] - 2 \left[ {}^A\boldsymbol{R}_B {}^B\boldsymbol{b}_i \right]^T {}^A\boldsymbol{a}_i \Big]^{1/2}. \end{aligned} \tag{4.54}$$

If the position and orientation of the moving platform lie within the feasible workspace of the manipulator, a unique solution for the limb lengths is determined by Equation 4.54. However, if the limb lengths derived from Equation

4.54 result in complex numbers, then the position or orientation of the moving platform is unreachable.

### 4.6.4 Forward Kinematics

In forward kinematic analysis, the vector of limb lengths $\boldsymbol{q} = [\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6]^T$ is assumed to be given, and the goal is to find the position ${}^A\boldsymbol{P}$ and orientation of the moving platform ${}^A\boldsymbol{R}_B$. The size of the problem depends on the representation used for orientation. If a rotation matrix is used, the forward kinematic problem involves 12 unknowns: the position vector ${}^A\boldsymbol{P}$ introduces three unknowns $(p_x, p_y, p_z)$, and the rotation matrix introduces nine unknowns.

Equation 4.53 provides six nonlinear equations for each $i = 1, 2, \ldots, 6$, and the orthonormality conditions 2.6 to 2.8 provide the remaining six equations to be solved simultaneously. Therefore, the forward kinematic problem is recast into twelve nonlinear equations with twelve unknowns. This problem is highly nonlinear and exceedingly difficult to solve in general and only a few practical closed-form solutions have been found for the Stewart–Gough platform.

Furthermore, even numerical iterative methods are impractical for the Stewart–Gough platform due to the heavy computational burden and the strong dependence on a good initial guess for convergence to the correct solution. Using screw or quaternion representations for orientation reduces the number of equations and unknowns to seven,

while Euler angle representations reduce them to six. However, the complexity of the equations increases significantly, resulting in higher computational costs

## Background Literature

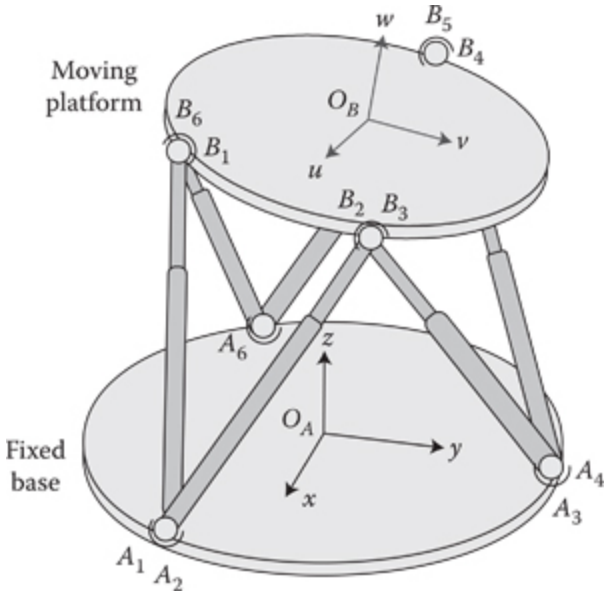
Forward kinematic problems in the context of parallel manipulators have been extensively studied in the literature, focusing primarily on determining all feasible solutions. Various approaches utilize algebraic formulations to derive high-degree polynomials or sets of nonlinear equations [95, 96, 97, 98]. Subsequently, methods such as algebraic elimination [99, 100], interval analysis [98], and continuation methods [101, 102] are employed to compute the roots of these equations.

However, finding all possible solutions does not resolve the forward kinematic problem entirely. Additional schemes are required to determine a unique pose of the platform from these solutions. Common strategies include iterative numerical procedures [103, 104, 80, 105], which can be sensitive to initial values and the nature of constraint equations produced. Another approach involves using auxiliary sensors [106, 107, 108, 109], although this method is constrained by cost considerations and measurement inaccuracies.

Researchers have also explored the use of neural networks as an alternative approach to solving the forward kinematic problem [110, 111, 112]. Regardless of the method employed, achieving a direct and unique solution

remains a challenging task. The complexity of the forward kinematic problem varies significantly depending on factors such as manipulator architecture, geometry, and sensor configurations. While some progress has been made for specific simplified configurations, the development of practical algorithms for the general Stewart–Gough platform remains an ongoing pursuit [113].

Much of the existing research focuses on solving the forward kinematics of specific types of parallel manipulators. Figure 4.17 illustrates a 3–3 Stewart–Gough platform where the six limbs intersect pairwise at three points between the moving platform and the base. This particular configuration allows for a closed-form solution to the forward kinematics. Such platforms are commonly used in flight simulators.




▶ Long Description for Figure 4.17

**FIGURE 4.17**

Schematics of a 3–3 Stewart–Gough Platform. ⏎

The forward kinematics of the 3-3 Stewart–Gough platform has been extensively studied in works such as [114, 115, 116]. It has been demonstrated in [114] and [115] that the 3-3 Stewart–Gough platform is kinematically equivalent to a spatial 3PRS mechanism. As a result, the forward kinematics can be expressed by a 16th-order polynomial equation. Interestingly, it has been shown that to obtain the complete set of 16 solutions, solving an 8th-order polynomial is sufficient.

A more general configuration, the 6–3 Stewart–Gough platform, is analyzed comprehensively in [117]. In this setup, the six limbs intersect pairwise at three points on the moving platform, while their positions at the base are arbitrary within a plane. The forward kinematic solution is derived in [117] using an algorithm that involves numerically solving three simultaneous nonlinear algebraic equations. Notably, this method does not require a specific initial guess for convergence to the correct solution. Once these equations are solved, the unique kinematic solutions are obtained through subsequent algebraic operations.

In the literature, notable studies such as [76], [77], and [113] provide comprehensive analyses of nearly general 6–6 Stewart–Gough platforms. In [76], analysis leads to a 20th-order polynomial equation with one unknown, yielding 40 different platform locations when combined with other equations. [77] introduces a generalized framework for the Stewart–Gough platform by considering all possible

geometric constraints between six pairs of primitives on the base and platform. This results in 3850 types of generalized platforms, encompassing the original Stewart–Gough platform as a specific case. Additionally, closed-form solutions are provided for general Stewart–Gough platforms, demonstrating that the original configuration can have up to 40 real solutions.

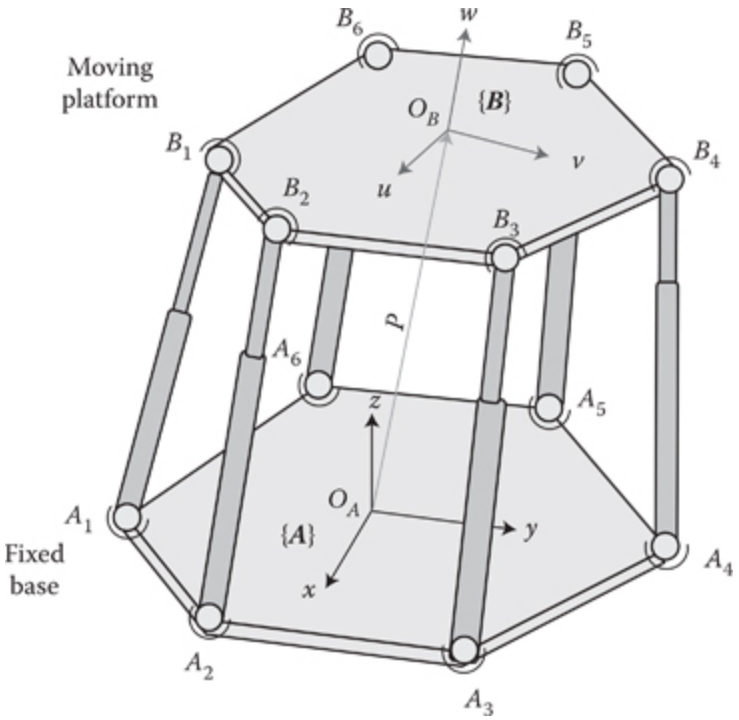
Analytical solutions such as those presented in [76, 77] are highly intricate and challenging to implement. Wang proposed a numerical method for the forward kinematics of nearly general Stewart–Gough platforms, offering a direct path to generating a unique solution [113]. This method capitalizes on the inherent simplicity of inverse kinematics in parallel manipulators, establishing a linear relationship between small changes in joint variables (limb lengths) and resulting small motions of the platform. By iteratively adjusting joint variables, the forward kinematic solution is efficiently computed.

The mathematical complexity associated with the forward kinematics of Stewart–Gough platforms poses a significant barrier, limiting their application in high-speed, real-time, and online scenarios. In the subsequent sections, an analytic solution tailored for a specific structure of the Stewart–Gough platform and a numerical method for the general case are detailed. These methods, representative of extensive research in the field, are assessed to provide a comprehensive understanding of their performance, limitations, and practical implementation characteristics. A

simulation study comparing these methods is conducted in Section 4.6.5.

## Analytical Solution

In this section, we focus on the closed-form solution of a specific configuration of the Stewart–Gough platform known as the 6–6$^{p}$ Stewart–Gough platform. This particular geometry is selected for analysis because its analytic solution is manageable and more straightforward to follow compared to other configurations. The 6–6$^{P}$ Stewart–Gough platform, depicted in Figure 4.18, differs from a general 6–6 Stewart–Gough platform based on the following characteristics [118]:




▶ Long Description for Figure 4.18

**FIGURE 4.18**
Schematics of a 6–6$^{\mathrm{P}}$ Stewart–Gough Platform. ⏎

- **Planar base and moving platforms:** The base platform is constrained such that its six vertices lie in a single plane, forming a hexagon. Similarly, the moving platform also has its six vertices positioned in a plane, forming another hexagon.

- **Similar hexagons:** Both hexagons (on the base and moving platforms) must be geometrically similar. This implies that if the positions of the six vertices on the base platform are known, the corresponding vertices on the moving platform can be found by uniformly scaling the positions of the base platform vertices using a scalar factor.

As shown in Figure 4.18, the two coordinate systems $\{\mathscr{A}\}$ and $\{\mathscr{B}\}$ are affixed to the base and the moving platforms, respectively. As depicted in Figure 4.16, the platform's layout is delineated by vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ for $i = 1, 2, \ldots, 6$, which signify the vertices of the base platform in frame $\{\mathscr{A}\}$ and the vertices of the moving platform in the frame $\{\mathscr{B}\}$, respectively. The six limbs are indicated by vectors $\ell_i \hat{\boldsymbol{s}}_i$, linking vertices $A_i$ and $B_i$ for $i = 1, 2, \ldots, 6$. In this 6–6$^{\mathrm{p}}$ Stewart-Gough platform, because of the similarity of the two hexagons, the following condition prevails:

$$\boldsymbol{b}_i = \mu\, \boldsymbol{a}_i; \qquad i = 1, 2, \ldots, 6$$

(4.55)

where $\mu$ is a specified constant known as the scaling factor [118]. In a general case,

$$\boldsymbol{a}_i = a_{x_i}\hat{\boldsymbol{x}} + a_{y_i}\hat{\boldsymbol{y}} + a_{z_i}\hat{\boldsymbol{z}}$$

(4.56)

where $a_{x_i}, a_{y_i}$, and $a_{z_i}$ represent the three components of $\boldsymbol{a}_i$, and $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ are three unit vectors in the frame $\{\mathscr{A}\}$. In the 6-6$^{\mathrm{p}}$ Stewart-Gough platform, $a_{z_i} = 0$. Therefore,

$$\boldsymbol{a}_i \quad = a_{x_i}\hat{\boldsymbol{x}} + a_{y_i}\hat{\boldsymbol{y}}$$

(4.57)

$$\boldsymbol{b}_i \quad = \mu(a_{x_i}\hat{\boldsymbol{x}} + a_{y_i}\hat{\boldsymbol{y}})$$

(4.58)

Recall the kinematic loop closure Equation 4.51 and substitute Equation 4.56. Hence, for $i = 1, 2, \ldots, 6$,

$$\begin{aligned} \ell_i\, {}^{A}\hat{\boldsymbol{s}}_i \quad &= {}^{A}\boldsymbol{P} + {}^{A}\boldsymbol{R}_B\, {}^{B}\boldsymbol{b}_i - {}^{A}\boldsymbol{a}_i \\ \ell_i\, \hat{\boldsymbol{s}}_i \quad &= \boldsymbol{P} + (\mu\boldsymbol{R} - \boldsymbol{I})\, \boldsymbol{a}_i. \end{aligned}$$

(4.59)

in which $\boldsymbol{I}$ is a $3 \times 3$ identity matrix, $\boldsymbol{R} = {}^{A}\boldsymbol{R}_B$, and for simplicity, the superscript $^{\mathrm{A}}$ is omitted from the notations.

Therefore, for $i = 1, 2, \ldots, 6$, the limb lengths can be obtained from

$$\begin{aligned}\ell_i^2 &= \left[(\mu R - I)\,\boldsymbol{a}_i + \boldsymbol{P}\right]^T \left[(\mu R - I)\,\boldsymbol{a}_i + \boldsymbol{P}\right] \\ &= \boldsymbol{P}^T\boldsymbol{P} + \boldsymbol{a}_i^T\left[(\mu\boldsymbol{R}^T - \boldsymbol{I})(\mu\boldsymbol{R} - \boldsymbol{I})\right]\boldsymbol{a}_i + 2\boldsymbol{a}_i^T(\mu\boldsymbol{R}^T - \boldsymbol{I})\end{aligned} \tag{4.60}$$

By substituting Equation 4.57 into Equation 4.60, and for $i = 1, 2, \ldots, 6$ we have:

$$\begin{aligned}\ell_i^2 &= \boldsymbol{P}^T\boldsymbol{P} + a_{x_i}\left[\hat{\boldsymbol{x}}^T(\mu\boldsymbol{R}^T\,\boldsymbol{P} - \boldsymbol{P})\right] + 2\,a_{y_i}\left[\hat{\boldsymbol{y}}^T(\mu\boldsymbol{R}^T\,\boldsymbol{P} - \boldsymbol{I}\right. \\ &\quad +2\mu\left\{a_{x_i}^2(\hat{\boldsymbol{x}}^T\boldsymbol{R}\hat{\boldsymbol{x}}) + a_{x_i}\,a_{y_i}\,(\hat{\boldsymbol{x}}^T\boldsymbol{R}\hat{\boldsymbol{y}} + \hat{\boldsymbol{y}}^T\boldsymbol{R}\hat{\boldsymbol{x}}) + a_{y_i}^2\,(\hat{\boldsymbol{y}}^T\boldsymbol{I}\right. \\ &\quad +(1+\mu^2)(a_{x_i}^2 + a_{y_i}^2).\end{aligned} \tag{4.61}$$

By defining $\boldsymbol{W} = [W_1, W_2, W_3, W_4, W_5, W_6]^T$ where,

$$\begin{aligned}W_1 &= \boldsymbol{P}^T\boldsymbol{P} \\ W_2 &= \hat{\boldsymbol{x}}^T(\mu\boldsymbol{R}^T\,\boldsymbol{P} - \boldsymbol{P}) \\ W_3 &= \hat{\boldsymbol{y}}^T(\mu\boldsymbol{R}^T\,\boldsymbol{P} - \boldsymbol{P})\end{aligned} \tag{4.62}$$

and,

$$
\begin{aligned}
W_4 &= \hat{\boldsymbol{x}}^T \boldsymbol{R} \hat{\boldsymbol{x}} \\
W_5 &= \hat{\boldsymbol{x}}^T \boldsymbol{R} \hat{\boldsymbol{y}} + \hat{\boldsymbol{y}}^T \boldsymbol{R} \hat{\boldsymbol{x}} \\
W_6 &= \hat{\boldsymbol{y}}^T \boldsymbol{R} \hat{\boldsymbol{y}}
\end{aligned}
\tag{4.63}
$$

Equation 4.61 transforms into a linear algebraic set of equations represented by:

$$
\boldsymbol{Q}\, \boldsymbol{W} = \boldsymbol{d}
\tag{4.64}
$$

The matrix $\boldsymbol{Q} = [\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{Q}_3, \boldsymbol{Q}_4, \boldsymbol{Q}_5, \boldsymbol{Q}_6]^T$ can be derived componentwise by

$$
\boldsymbol{Q}_i = \left[1, 2a_{x_i}, 2a_{y_i}, -2\mu a_{x_i}^2, -2\mu a_{x_i} a_{y_i}, -2\mu a_{x_i}^2\right],
\tag{4.65}
$$

and the vector $\boldsymbol{d} = [d_1, d_2, d_3, d_4, d_5, d_6]^T$ can be found by

$$
d_i = \ell_i^2 - (1 + \mu^2)(a_{x_i}^2 + a_{y_i}^2).
\tag{4.66}
$$

It is noteworthy that, according to this formulation, the position of the moving platform depends solely on $W_1, W_2$, and $W_3$, and is independent of the orientation of the moving platform, which is determined by $W_4, W_5$, and $W_6$. If the

matrix $\boldsymbol{Q}$ is nonsingular, implying that the six vertices of either the base platform or the moving platform are not collinear, $\boldsymbol{W}$ can be determined by

$$\boldsymbol{W} = \boldsymbol{Q}^{-1}\,\boldsymbol{d}. \tag{4.67}$$

Since $\boldsymbol{W}$ is now determined, Equation 4.63 can be used to analyze the orientation, and Equation 4.62 can be used to derive the position of the moving platform.

Let us represent the orientation of the moving platform by the screw axis $\left\{\hat{\boldsymbol{s}} = \begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T,\ \theta\right\}$. Hence, the rotation matrix can be written as

$$\boldsymbol{R} = \begin{bmatrix} s_x^2 v\theta + c\theta & s_x s_y v\theta - s_z s\theta & s_x s_z v\theta + s_y s\theta \\ s_y s_x v\theta + s_z s\theta & s_y^2 v\theta + c\theta & s_y s_z v\theta - s_x s\theta \\ s_z s_x v\theta - s_y s\theta & s_z s_y v\theta + s_x s\theta & s_z^2 v\theta + c\theta \end{bmatrix} \tag{4.68}$$

Notice that,

$$\begin{aligned} W_4 &= \hat{\boldsymbol{x}}^T \boldsymbol{R} \hat{\boldsymbol{x}} = r_{11} \\ W_5 &= \hat{\boldsymbol{x}}^T \boldsymbol{R} \hat{\boldsymbol{y}} + \hat{\boldsymbol{y}}^T \boldsymbol{R} \hat{\boldsymbol{x}} = r_{12} + r_{21} \\ W_6 &= \hat{\boldsymbol{y}}^T \boldsymbol{R} \hat{\boldsymbol{y}} = r_{22} \end{aligned} \tag{4.69}$$

where $r_{ij}$ denotes the components of the rotation matrix $\boldsymbol{R}$, and $s\theta = \sin\theta,\ c\theta = \cos\theta,$ and $v\theta = 1 - \cos\theta$ are used for shorthand notation. Equation 4.69 provides three nonlinear equations for the screw parameters. Considering the unitary property of the screw axis, the following four equations must be solved to find the screw parameters.

$$W_4 = s_x^2 v\theta + c\theta \tag{4.70}$$

$$W_5 = 2 s_x s_y v\theta \tag{4.71}$$

$$W_6 = s_y^2 v\theta + c\theta \tag{4.72}$$

$$1 = s_x^2 + s_y^2 + s_z^2 \tag{4.73}$$

Hence,

$$s_x^2 = \frac{W_4 - c\theta}{v\theta} \quad ; \quad s_y^2 = \frac{W_6 - c\theta}{v\theta}. \tag{4.74}$$

By substituting Equation 4.74 into Equation 4.71, and with some manipulation, a quadratic equation in $\cos\theta$ is obtained,

$$\cos^2\theta - B\cos\theta + C = 0 \tag{4.75}$$

in which,

$$B = W_4 + W_6 \quad ; \quad C = W_4 W_6 - W_5^2/4 \tag{4.76}$$

Therefore, the screw parameters can be established by:

$$\cos\theta = \frac{1}{2}\left(B - \sqrt{B^2 - 4C}\right) \tag{4.77}$$

and if $W_4 \neq c\theta$, then

$$s_x = \sqrt{\frac{W_4 - c\theta}{v\theta}} \quad ; \quad s_y = \frac{W_5}{2 s_x v\theta} \quad ; \quad s_z = \sqrt{1 - s_x^2 - s_y^2} \tag{4.78}$$

otherwise, if $W_6 \neq c\theta$, then

$$s_y = \sqrt{\frac{W_6 - c\theta}{v\theta}} \quad ; \quad s_x = \frac{W_5}{2 s_y v\theta} \quad ; \quad s_z = \sqrt{1 - s_x^2 - s_y^2}$$

(4.79)

and in case $W_4 = W_6 = c\theta$, then

$$s_x = s_y = 0 \quad ; \quad s_z = 1.$$

(4.80)

Here, the positive square root of $B^2 - 4C$ is excluded, ensuring that unnecessary complex roots of $\theta$ are avoided. Additionally, the negative roots of $s_x$ and $s_z$ yield four distinct screw representations for the orientation of the moving platform. These four rotations are represented as follows:

$$\begin{aligned} \boldsymbol{s} &= \begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T \quad , \boldsymbol{s} = \begin{bmatrix} s_x & s_y & -s_z \end{bmatrix}^T, \\ \boldsymbol{s} &= \begin{bmatrix} -s_x & -s_y & s_z \end{bmatrix}^T \quad , \boldsymbol{s} = -\begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T. \end{aligned}$$

By substituting the screw parameters into Equation 4.68, four distinct transformation matrices can be derived as $\boldsymbol{R}_i, i = 1, 2, \ldots, 4$, respectively.

Now that the orientation of the moving platform is determined, the location of its center point is determined using $W_1, W_2$, and $W_3$. Let's define,

$$\boldsymbol{U}^T = \hat{\boldsymbol{x}}^T (\mu \boldsymbol{R}^T - \boldsymbol{I}), \tag{4.81}$$

$$\boldsymbol{V}^T = \hat{\boldsymbol{y}}^T (\mu \boldsymbol{R}^T - \boldsymbol{I}), \tag{4.82}$$

Vectors $\boldsymbol{U}$ and $\boldsymbol{V}$ are derived from each transformation matrix $\boldsymbol{R}_i, i = 1, 2, \ldots, 4$, thus, $\boldsymbol{U}$ and $\boldsymbol{V}$ also have four corresponding cases. Now Equation 4.62 becomes

$$W_1 = \boldsymbol{P}^T \boldsymbol{P} \tag{4.83}$$

$$W_2 = \boldsymbol{U}^T \boldsymbol{P} \tag{4.84}$$

$$W_3 = \boldsymbol{V}^T \boldsymbol{P} \tag{4.85}$$

Obviously, Equation 4.84 and Equation 4.85 represent two planes, and their intersection is a line, denoted as

$$\boldsymbol{P} = \boldsymbol{r}_0 + t\boldsymbol{r}_1 \tag{4.86}$$

where $t$ is the parameter of the line, $r_0$ is a line in the plane spanned by vectors $\boldsymbol{U}$ and $\boldsymbol{V}$.

$$\begin{aligned} r_0 \quad &= \frac{(\boldsymbol{V}^T\boldsymbol{V})W_2 - (\boldsymbol{U}^T\boldsymbol{V})W_3}{(\boldsymbol{U}^T\boldsymbol{U})(\boldsymbol{V}^T\boldsymbol{V}) - (\boldsymbol{U}^T\boldsymbol{V})^2}\boldsymbol{U} \\ &+ \frac{(\boldsymbol{U}^T\boldsymbol{V})W_2 - (\boldsymbol{U}^T\boldsymbol{U})W_3}{(\boldsymbol{U}^T\boldsymbol{U})(\boldsymbol{V}^T\boldsymbol{V}) - (\boldsymbol{U}^T\boldsymbol{V})^2}\boldsymbol{V} \end{aligned} \tag{4.87}$$

and $r_1$ is a unit vector perpendicular to both vectors $\boldsymbol{U}$, and $\boldsymbol{V}$:

$$r_1 = \frac{\boldsymbol{U} \times \boldsymbol{V}}{\| \boldsymbol{U} \times \boldsymbol{V} \|} \tag{4.88}$$

where × denotes vector cross product. Obviously,

$$\boldsymbol{r}_0^T\boldsymbol{r}_1 = \boldsymbol{r}_1^T\boldsymbol{r}_0 = 0 \tag{4.89}$$

The line Equation 4.86, intersects the sphere Equation 4.83 at two points $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, which are the solutions to Equations 4.83 through 4.85. Here,

$$\boldsymbol{P}_1 \quad = \boldsymbol{r}_0 + t\boldsymbol{r}_1$$

(4.90)

$$\boldsymbol{P}_2 \quad = \boldsymbol{r}_0 - t\boldsymbol{r}_1$$

(4.91)

and

$$t = \sqrt{W_1 - \boldsymbol{r}_0^T \boldsymbol{r}_0}$$

(4.92)

considering Equation 4.89. Up to this point, both the orientation matrix $\boldsymbol{R}$ and the position vector $\boldsymbol{P}$ have been successfully determined. For the 6–6$^{\mathrm{p}}$ Stewart platform, the forward kinematics problem typically results in eight possible solutions, provided that the underlying equations do not degenerate.

The analytic solution presented here for the 6–6$^{\mathrm{P}}$ Stewart–Gough platform offers significant advantages over other solutions found in the literature for a nearly general Stewart–Gough platform, as seen in [76, 119]. This analytic solution is computationally more efficient and more manageable than others. However, like any analytical solution, specific shortcomings exist, which are less frequently discussed in the literature. The analytic solution for the 6–6$^{\mathrm{P}}$ Stewart–Gough platform is not exempt from these considerations, and two primary deficiencies have been identified through simulations of this solution.

The first deficiency is the dependency of the orientation parameters of the solution, namely $W_4, W_5$, and $W_6$ to the value of $c\theta$. Reconsider Equations 4.70 to 4.72 for the case where $\theta = 0$ or $\theta = \pi$, by which $c\theta = 1$ and $v\theta = 0$. In this case, $W_4 = W_6 = 1$, and $W_5$ becomes zero, and these equations fail to generate any solution to $s_x, s_y$ and $s_z$. Unfortunately, there is no means to overcome this problem in the solution, and this is a singular orientation for the given analytic solution.

The second deficiency is a broad issue that pertains universally to all analytic solutions. As demonstrated in the present analysis, the solution framework typically yields eight potential outcomes for a given configuration of limb lengths. Further methodologies are essential to ascertain a definitive position and orientation of the platform from the array of possible solutions. Commonly employed strategies include iterative numerical techniques [103, 104, 80, 105] and the use of auxiliary sensors [106, 107, 108, 109]. Numerical iteration methods are often sensitive to initial values and the inherent characteristics of the constraint equations they attempt to resolve. Conversely, the auxiliary sensors approach is constrained by practical limitations such as cost and potential measurement errors. Despite the approach used to solve the forward kinematic problem, achieving a definitive determination of a unique solution remains a formidable challenge.

The approach employed here to identify a unique solution from the array of possible solutions is based on comparing

the solution at each iteration with the previous one. The solution with a smaller error norm relative to the previous iteration, below a specified threshold, is selected as the correct solution. However, this methodology reveals two deficiencies inherent in the analytical solution.

The first deficiency arises from the requirement for an initial guess of position and orientation to converge to a unique solution initially. While the analytical method itself does not inherently demand an initial guess to determine possible solutions, achieving convergence to the desired solution heavily relies on the accuracy of this initial guess.

The second deficiency becomes apparent in simulations where either $s_x, s_y$ or particularly $s_z$ approaches zero. Although the method accurately identifies the solution when any of these screw axis components reach zero, it struggles immediately afterward to distinguish between positive and negative values of the solutions. This ambiguity may lead to failure in providing the correct solution. The problem is exaggerated when two of the screw axis components approach zero simultaneously.

Based on the observations above, it can be concluded that the analytic solution generally provides the correct solution in cases where none of the screw axis parameters are zero or approaching zero. However, when these parameters do approach zero, the analytic solution may either degenerate or diverge from the correct solution. In the next section, a numerical solution to the problem is presented. Despite requiring ten times more computational

time than the analytical solution, the numerical approach is less sensitive to special cases of orientation and tends to converge reliably to the correct solution.

## Numerical Solution

To generate a numerical solution for the forward kinematic problem, assume that the limb lengths $\boldsymbol{q} = [\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6]^T$ are given, and the objective is to determine the position ${}^A\boldsymbol{P} = [p_x, p_y, p_z]^T$ and the orientation of the moving platform, which is represented by the screw $\left\{\hat{\boldsymbol{s}} = [s_x, s_y, s_z]^T, \theta\right\}$. Consequently, the seven unknown variables can be encapsulated in the vector $\boldsymbol{x}$ as follows:

$$\boldsymbol{x} = [p_x \quad p_y \quad p_z \quad s_x \quad s_y \quad s_z \quad \theta]^T \tag{4.93}$$

$$= [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7]^T \tag{4.94}$$

The unknown position vector $\boldsymbol{P}$ and the orientation matrix $\boldsymbol{R}(\boldsymbol{x})$ can be expressed as functions of $\boldsymbol{x}$ by the following equations:

$$\boldsymbol{P}(\boldsymbol{x}) = [x_1 \quad x_2 \quad x_3]^T, \tag{4.95}$$

and,

$$\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} x_4^2\, vx_7 + cx_7 & x_4 x_5\, vx_7 - x_6\, sx_7 & x_4 x_6\, vx_7 + \\ x_5 x_4\, vx_7 + x_6\, sx_7 & x_5^2\, vx_7 + cx_7 & x_5 x_6\, vx_7 - \\ x_6 x_4\, vx_7 - x_5\, sx_7 & x_6 x_5\, vx_7 + x_4\, sx_7 & x_6^2\, vx_7 + \end{bmatrix} \tag{4.96}$$

Assume that the geometry of the base and moving platform attachment points $A_i$, and $B_i$ are given by vectors $\boldsymbol{a}_i$, and $\boldsymbol{b}_i$, respectively. The numerical solution consists of iteratively finding the solution to the following seven nonlinear equations:

$$\begin{cases} F_i(\boldsymbol{x}) &= -\ell_i^2 + \left[\boldsymbol{P}(\boldsymbol{x}) + \boldsymbol{R}(\boldsymbol{x})\, \boldsymbol{b}_i - \boldsymbol{a}_i\right]^T \left[\boldsymbol{P}(\boldsymbol{x}) + \boldsymbol{R}(\boldsymbol{x})\, \boldsymbol{b}_i \right. \\ & \quad \text{for} \;\; i = 1, 2, \ldots, 6 \\ \\ F_7(\boldsymbol{x}) &= x_4^2 + x_5^2 + x_6^2 - 1 \end{cases} \tag{4.97}$$

Various numerical methods employing nonlinear least-square optimization routines [120] can be used to find a solution for $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$[1]. In a least-square problem, the function $f(\boldsymbol{x}) = \frac{1}{2}\sum_i F_i(\boldsymbol{x})^2$ is minimized over $\boldsymbol{x} \in \mathbb{R}^n$ [121]. In these methods, the direction of changing the components of the solution is determined in a way that all components of the residual function decrease. Some of

these methods are generalizations of the conventional Newton-Raphson method, in which the appropriate direction for changing the joint variables is determined using the analytical or numerical solution of the Jacobian matrix of the residual function $\boldsymbol{f}$. This method uses the inverse of the Jacobian matrix, but due to the computational complexity required to determine the inverse of the Jacobian matrix, the transpose of the Jacobian matrix is used as an alternative in the "Steepest Descent" method [2]. Another efficient method used for this purpose is the Levenberg-Marquardt method [122, 123, 124]. This method provides a compromise between the above two methods, using an interpolation of the results of these two methods at each iteration.

The flowchart in Figure 4.19 details the iterative method used to find the forward kinematic solution. As shown in this flowchart, for a given $\boldsymbol{x}$, the values of $\boldsymbol{P}(\boldsymbol{x})$ and $\boldsymbol{R}(\boldsymbol{x})$ are calculated using Equations 4.95 and 4.96, respectively. Next, $F_i(\boldsymbol{x})$ for $i = 1, 2, \ldots, 7$ are determined using equation 4.97. Then, the value of $f(\boldsymbol{x}) = \frac{1}{2}\sum_i F_i(\boldsymbol{x})^2$ is computed. If this value is not sufficiently close to zero, an optimal search routine recalculates a new value for $\boldsymbol{x}$. This iterative process continues until a solution for $f(\boldsymbol{x}) = 0$ with a high accuracy of $\epsilon \ll 1$ is achieved. Since multiple solutions may exist for the equation $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$, the solution from the previous iteration is used to guide the search for the next step, and to avoid abrupt changes in the forward kinematic solutions. Simulation results detailed in Section

4.6.5 demonstrate the effectiveness and accuracy of the numerical routines used to solve the forward kinematics.

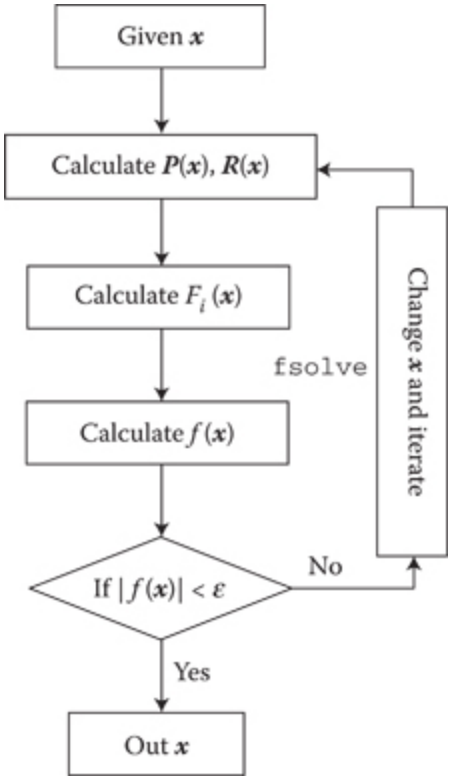



**FIGURE 4.19**
Flowchart of iterative routine used to solve the forward kinematics of Stewart–Gough platform. ⏎

## 4.6.5 Simulations

To verify the precision and reliability of the solutions, a simulation study is performed on a standard 6–6$^{p}$ Stewart-Gough platform in this section. The forward kinematic solution results are presented for both the analytical and numerical approaches for comparison. The geometric parameters utilized in the simulations are detailed in Table

4.2, and the base platform geometry is depicted in Figure 4.20.

**TABLE 4.2**

Geometric parameters of the 6–6$^{\mathrm{P}}$ Stewart–Gough platform ⏎

| Description | Quantity | Unit |
|---|---|---|
| $\boldsymbol{a}_1$: Position vector of point $A_1$ | $[+0.8, +0.0, 0]^T$ | $m$ |
| $\boldsymbol{a}_2$: Position vector of point $A_2$ | $[+0.5, +1.2, 0]^T$ | $m$ |
| $\boldsymbol{a}_3$: Position vector of point $A_3$ | $[-0.2, +1.0, 0]^T$ | $m$ |
| $\boldsymbol{a}_4$: Position vector of point $A_4$ | $[-0.6, +0.4, 0]^T$ | $m$ |
| $\boldsymbol{a}_5$: Position vector of point $A_5$ | $[-0.4, -0.8, 0]^T$ | $m$ |
| $\boldsymbol{a}_6$: Position vector of point $A_6$ | $[+0.4, -1.2, 0]^T$ | $m$ |
| $\mu$: The moving platform size ratio $\boldsymbol{b}_i = \mu \boldsymbol{a}_i$ | 0.6 | $-$ |

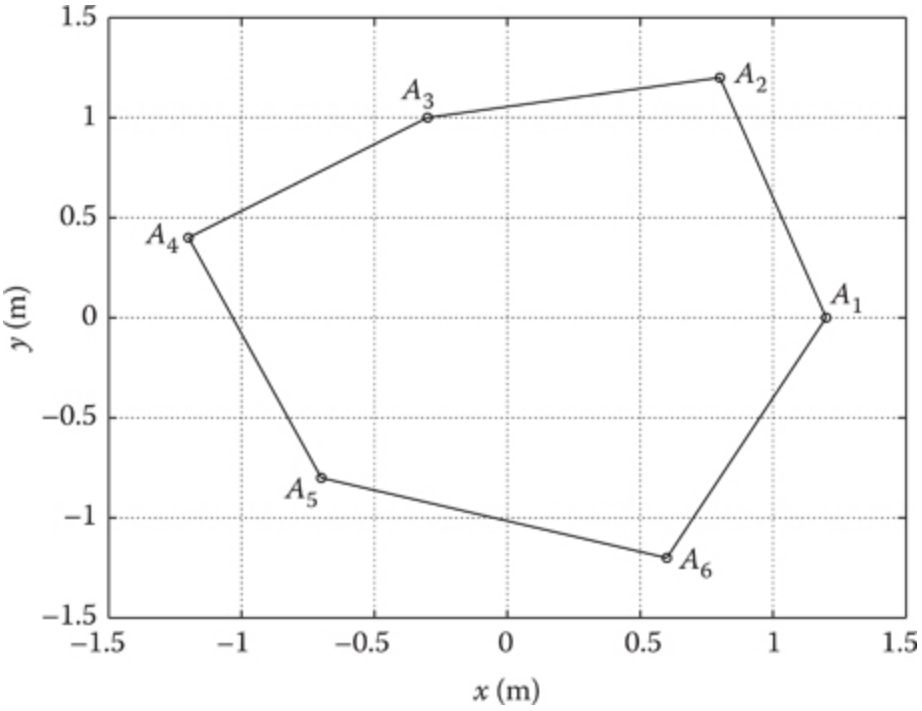
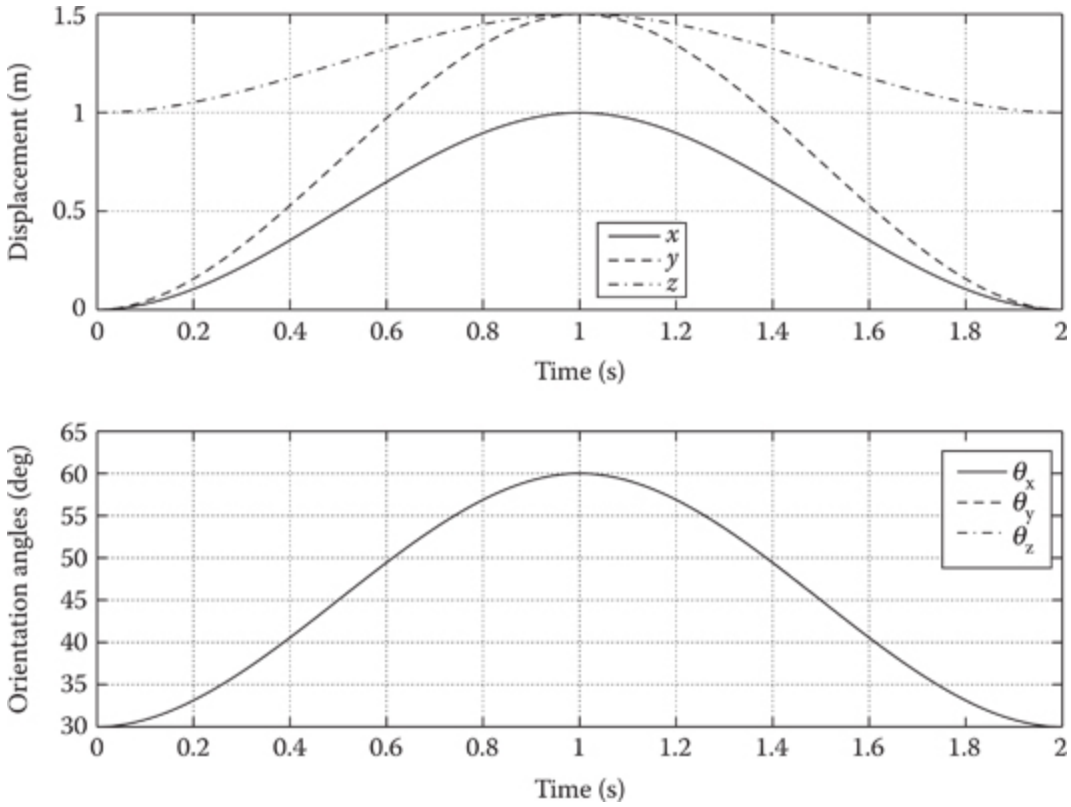



**FIGURE 4.20**
The base geometry of the examined 6–6$^{P}$ Stewart–Gough platform. ⏎

## Analytical Solution

For the simulations, a typical spatial trajectory for the 6–6$^{p}$ Stewart-Gough platform is considered. The trajectories include three-position trajectories and three orientation trajectories represented by screw parameters, as illustrated in Figure 4.21. As shown in this figure, the movement in the $x$ and $y$ directions is 1 and 1.5 meters, respectively, while in the $z$ direction, the platform starts and ends at a height of 1 meter, moving 0.5 meters upward. The orientation of the moving platform is generated by considering a spatial rotation about a constant screw axis

$\hat{\boldsymbol{s}} = [1, 1, 1]^T/\sqrt{3}$, with an initial angle of $\pi/6$ and a maximum rotation angle of $\pi/3$.




▶ Long Description for Figure 4.21

**FIGURE 4.21**

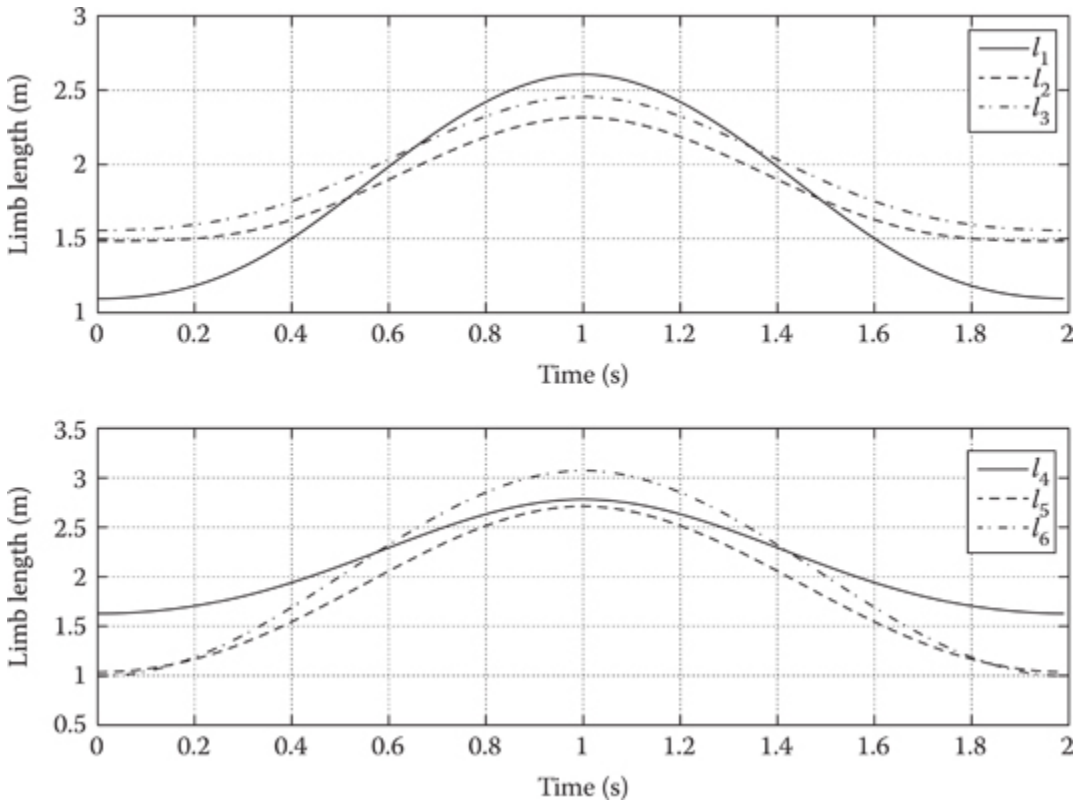
A typical trajectory examined for the 6–6$^{\mathrm{P}}$ Stewart–Gough platform. ⏎

---

[1]Use the `fsolve` function in MATLAB ⏎

For the given trajectory, the inverse kinematic solution of the manipulator is obtained from Equation 4.54, and the lengths of the manipulator limbs are uniquely determined and depicted in Figure 4.22. As shown in this figure, the

lengths of the manipulator limbs range from 1 meter to approximately 3 meters, accommodating variations in the desired trajectory.

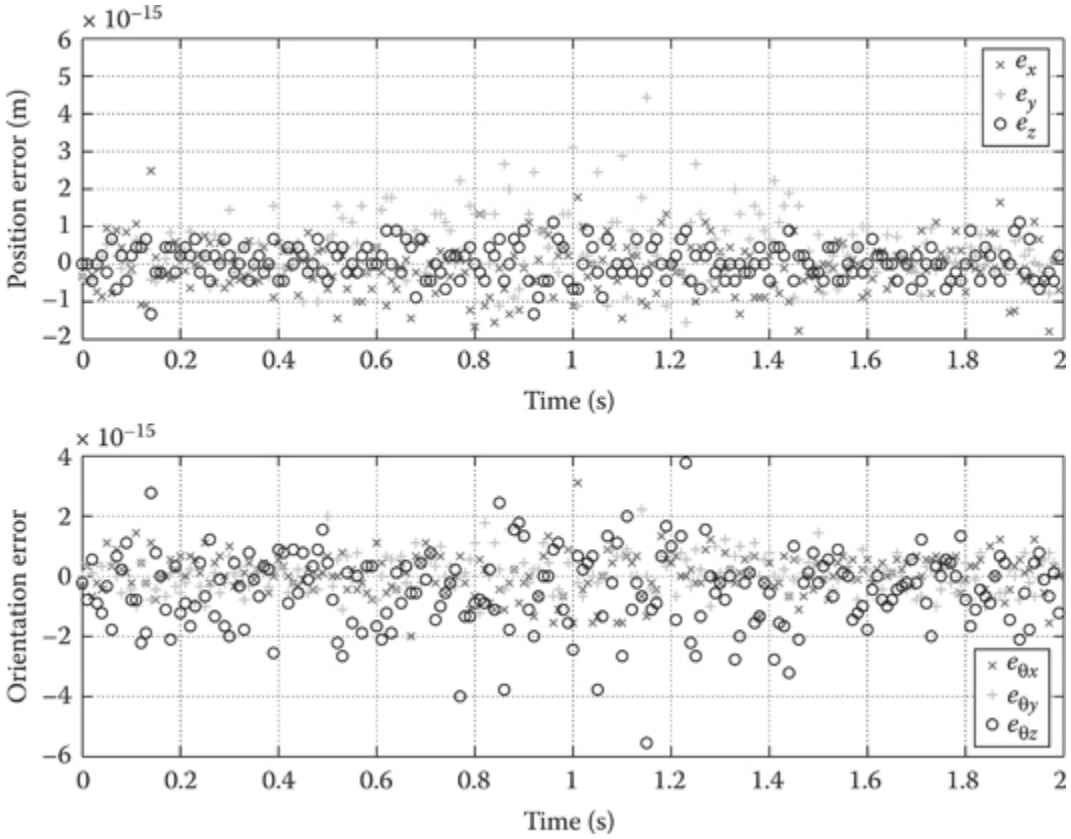



▶ Long Description for Figure 4.22

**FIGURE 4.22**
Inverse kinematic solution of a 6–6$^{P}$ Stewart–Gough platform for the given trajectory. ⏎

To verify the accuracy and reliability of the solutions, the forward kinematic solution for the manipulator is computed using the determined limb lengths from the inverse kinematics, as illustrated in Figure 4.22. The difference between the forward kinematic solutions $\boldsymbol{\chi}_c$ and the desired motion $\boldsymbol{\chi}_d$ is visualized in Figure 4.23. It is

observed that the calculated analytic solutions to the forward kinematic problem match the desired trajectories precisely, with an accuracy of up to $10^{-14}$.




▶ Long Description for Figure 4.23

**FIGURE 4.23**

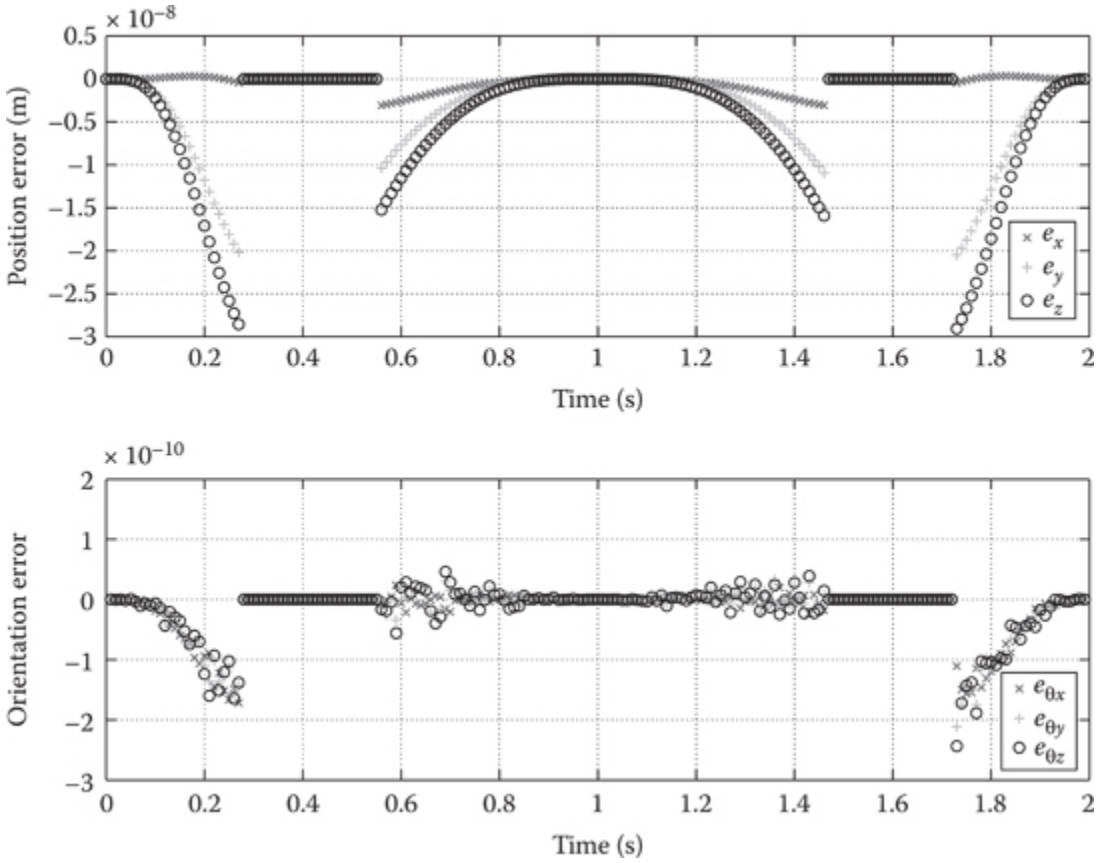
The precision and reliability of the forward and inverse kinematic solutions: Analytical approach. ⏎

It is important to note that the forward kinematic solutions are not unique. To ensure convergence to the correct solution at each step, the forward kinematic solution from the previous step is used as the initial guess for the next iteration. This approach guarantees that the

numerical solution converges correctly at all examined points. Similar results are observed for other trajectories that satisfy the conditions preventing the degeneration of the analytical solutions. As previously explained, cases where $\theta_d = 0$ or $s_z = 0$ in the desired trajectories can cause the analytical solution to fail in providing the correct solutions.

## Numerical Solution

To compare the accuracy and computational effort between the analytical and numerical solutions, we consider the same trajectory depicted in Figure 4.21 for simulation. Numerical methods are employed to find the forward kinematic solution, and the differences between the numerical solutions $\boldsymbol{\chi}_c$ and the desired motion $\boldsymbol{\chi}_d$ are illustrated in Figure 4.24.




▶ Long Description for Figure 4.24

**FIGURE 4.24**

The precision and reliability of the forward and inverse kinematic solutions: Numerical approach. ⏎

It is observed that the numerical solutions to the forward kinematic problem closely match the desired trajectories with an accuracy of $10^{-8}$ in positions and $10^{-10}$ in orientation. While these accuracies are slightly lower than those obtained with the analytical solutions, they are still highly acceptable and meet the required precision.

To compare the computation time for each method, simulations were conducted on a Sony Laptop, VGN-TX2

Series, with a 1.20 GHz Intel Pentium processor over 200 time steps. The total simulation time for two seconds indicates that the elapsed time for the numerical solution is approximately 5.89 seconds, while for the analytical solution, it is about 0.65 seconds. Additionally, the average computation time for each forward kinematic solution is approximately 26.33 milliseconds for the numerical method and about 0.86 milliseconds for the analytical method. This clearly demonstrates that the numerical solution is significantly slower in computational speed compared to the analytical solution.

In summary, while the numerical solution achieves slightly lower accuracy than the analytical solution, it still meets required precision standards. However, it comes at the cost of significantly higher computational time compared to the more efficient analytical method.

The numerical solution is advantageous in scenarios where the analytical solution may fail, particularly in handling orientation parameters that can lead to degeneration in the analytical approach. Unlike the analytical method, numerical methods can effectively solve the forward kinematics of a general Stewart-Gough platform without requiring a specific structural constraint.

Furthermore, numerical solutions can be found in cases where the constraints of the analytical solution are not met. For instance, consider a general Stewart-Gough platform with geometric parameters provided in Table 4.3. In this configuration, the six vertices of the base and moving

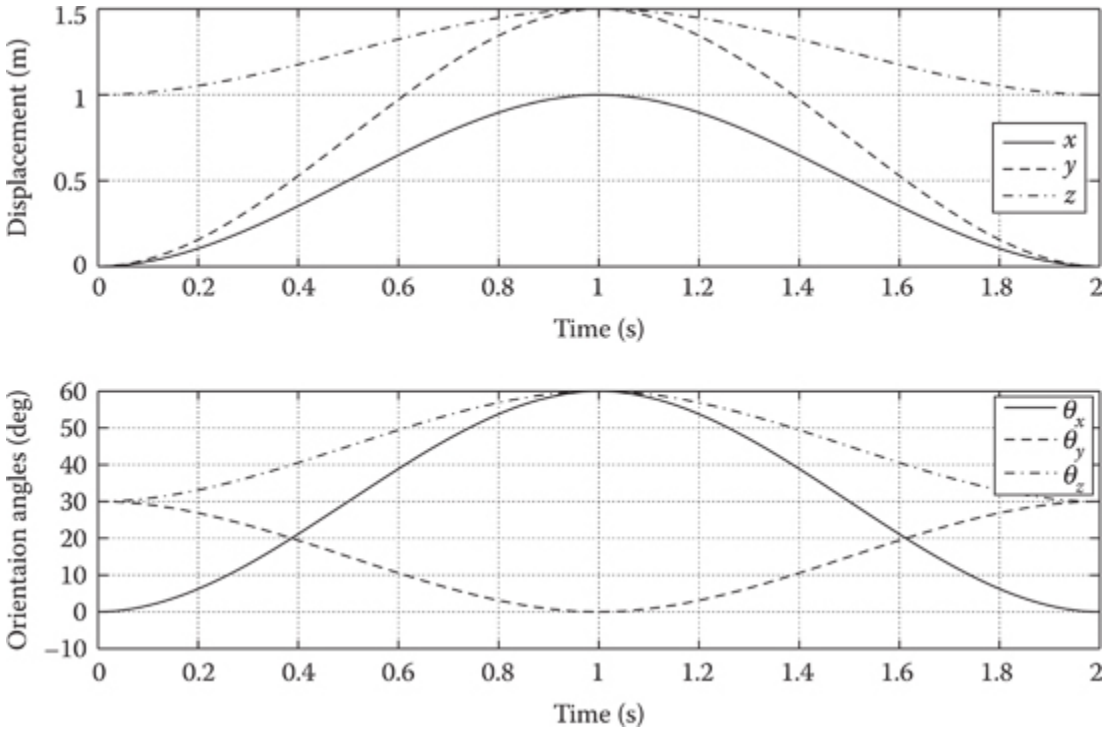
platform do not lie in a plane and they are not similar in geometry, representing a general case for the Stewart-Gough platform that does not conform to the assumptions of the analytical solution. The position trajectory of the moving platform is set similarly to previous examples. However, the orientation of the moving platform is adjusted such that the screw parameters become zero at certain points in time, as depicted in Figure 4.25, making the condition for finding the solutions even more challenging.

**TABLE 4.3**

Geometric parameters of a general Stewart–Gough platform ⏎

| Description | Quantity | Unit |
|---|---|---|
| $\boldsymbol{a}_1$: Position vector of point $A_1$ | $[+0.8, +0.0, +0.1]^T$ | $m$ |
| $\boldsymbol{a}_2$: Position vector of point $A_2$ | $[+0.5, +1.2, -0.2]^T$ | $m$ |
| $\boldsymbol{a}_3$: Position vector of point $A_3$ | $[-0.2, +1.0, -0.1]^T$ | $m$ |
| $\boldsymbol{a}_4$: Position vector of point $A_4$ | $[-0.6, +0.4, +0.2]^T$ | $m$ |
| $\boldsymbol{a}_5$: Position vector of point $A_5$ | $[-0.4, -0.8, -0.1]^T$ | $m$ |
| $\boldsymbol{a}_6$: Position vector of point $A_6$ | $[+0.4, -1.2, +0.1]^T$ | $m$ |
| $\boldsymbol{b}_1$: Position vector of point $B_1$ | $[+0.6, +0.2, -0.1]^T$ | $m$ |

| Description | Quantity | Unit |
|---|---|---|
| $\boldsymbol{b}_2$: Position vector of point $B_2$ | $[+0.3, +1.0, +0.2]^T$ | $m$ |
| $\boldsymbol{b}_3$: Position vector of point $B_3$ | $[-0.1, +1.2, -0.2]^T$ | $m$ |
| $\boldsymbol{b}_4$: Position vector of point $B_4$ | $[-0.4, +0.2, -0.1]^T$ | $m$ |
| $\boldsymbol{b}_5$: Position vector of point $B_5$ | $[-0.2, -1.0, +0.1]^T$ | $m$ |
| $\boldsymbol{b}_6$: Position vector of point $B_6$ | $[+0.3, -1.0, -0.1]^T$ | $m$ |

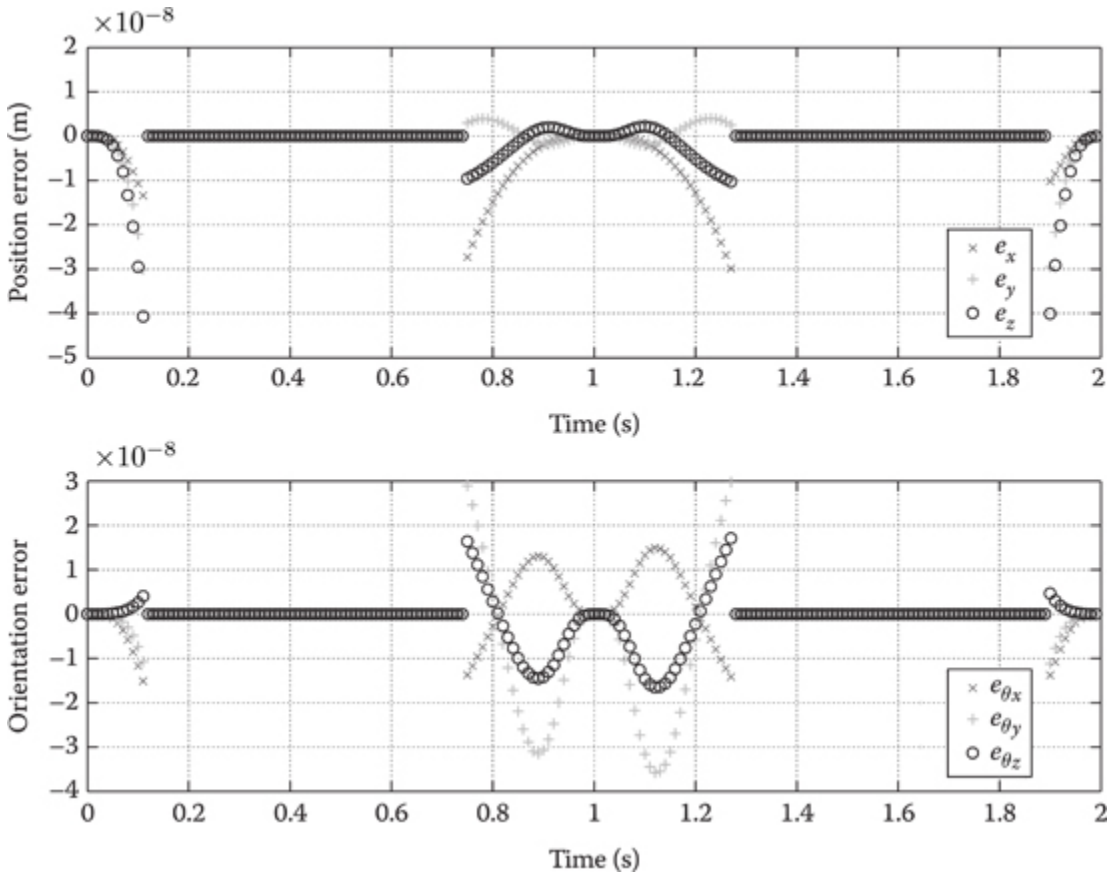



▶ Long Description for Figure 4.25

**FIGURE 4.25**

A typical trajectory examined for the general Stewart–Gough platform. ⏎

Due to the complex structure of the manipulator and the nature of the trajectory, the forward kinematic problem becomes unsolvable using the analytical method. However, the numerical method remains capable of addressing this challenge and provides a viable solution. Numerical methods are employed to find the forward kinematic solution, and the differences between the numerical solutions $\boldsymbol{\chi}_c$ and the desired motion $\boldsymbol{\chi}_d$ are illustrated in Figure 4.26. It is observed that the numerical solutions to the forward kinematic problem closely match the desired trajectories with an accuracy of $10^{-8}$. This flexibility highlights the robustness and applicability of numerical methods in handling general Stewart-Gough platforms under varied conditions and structural configurations.




▶ Long Description for Figure 4.26

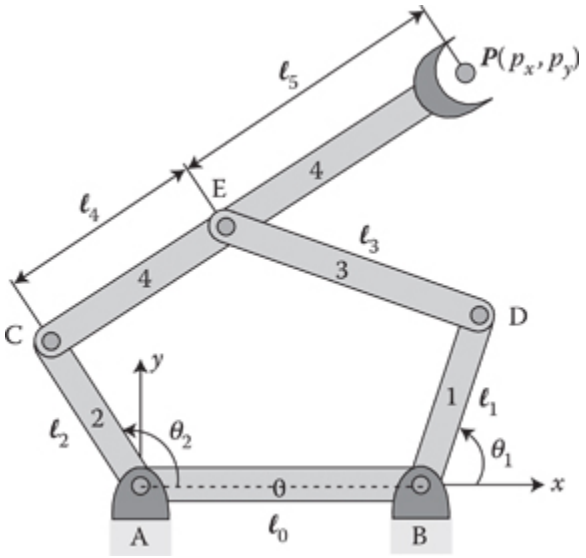
**FIGURE 4.26**

The precision and reliability of the forward and inverse kinematic solutions for the general Stewart–Gough platform in numerical approach. ⏎

## Problems

4.1 Consider the five-bar mechanism depicted in Figure 4.27, which consists of five revolute joints. For this mechanism, solve the forward kinematic problem to

determine the end-effector position vector $\boldsymbol{P} = [p_x, p_y]^T$ as a function of the two input angles $\boldsymbol{\chi} = [\theta_1, \theta_2]^T$.

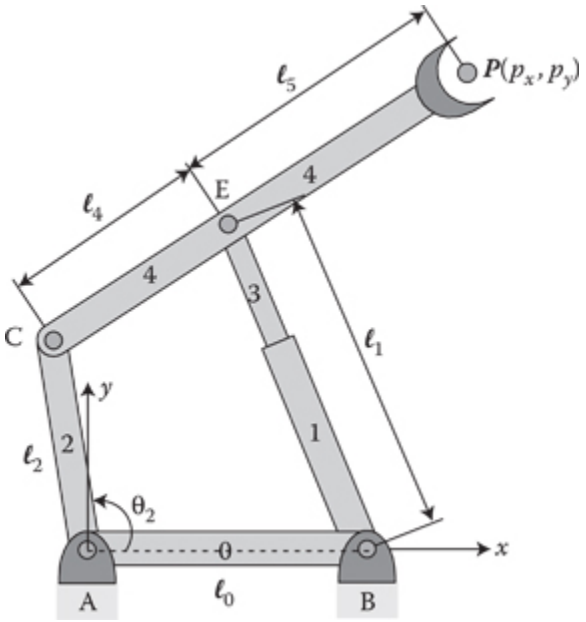



▶ Long Description for Figure 4.27

**FIGURE 4.27**

Five bar mechanism of Problem 4.1. ⏎

4.2 Consider the five-bar mechanism shown in Figure 4.28, which consists of four revolute joints and one prismatic joint. For this mechanism, analyze the forward kinematic problem and determine the end-effector position vector $\boldsymbol{\chi} = [p_x, p_y]^T$ as a function of the two joint input variables $\boldsymbol{q} = [\ell_1, \theta_2]^T$.

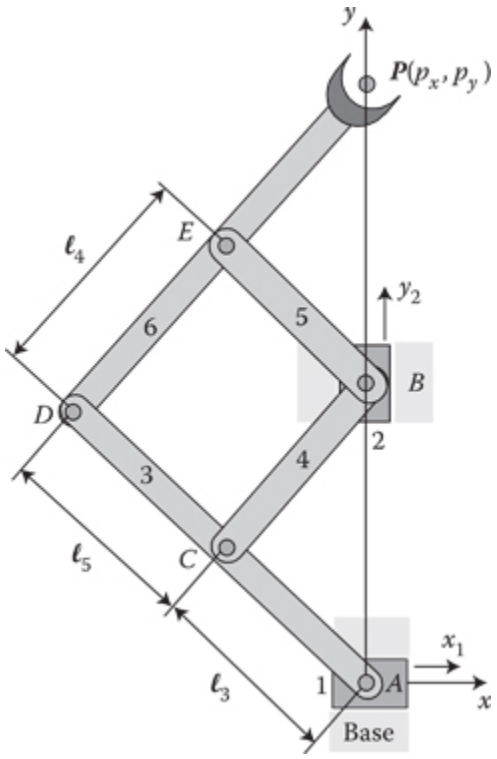



▶ Long Description for Figure 4.28

**FIGURE 4.28**

Five bar, 4P1R mechanism. ⏎

4.3 Figure 4.29 shows a planar pantograph mechanism in which the ground-connected joints are prismatic, and the remaining joints are revolute. The structure of this mechanism is based on a parallelogram, with links 3 and 5 parallel to each other, and links 4 and 6 also parallel. For this mechanism, solve the forward kinematics to find the position vector of the end-effector $\boldsymbol{\chi} = [p_x, p_y]^T$ as a function of the two joint input linear displacements $\boldsymbol{q} = [x_1, x_2]^T$.



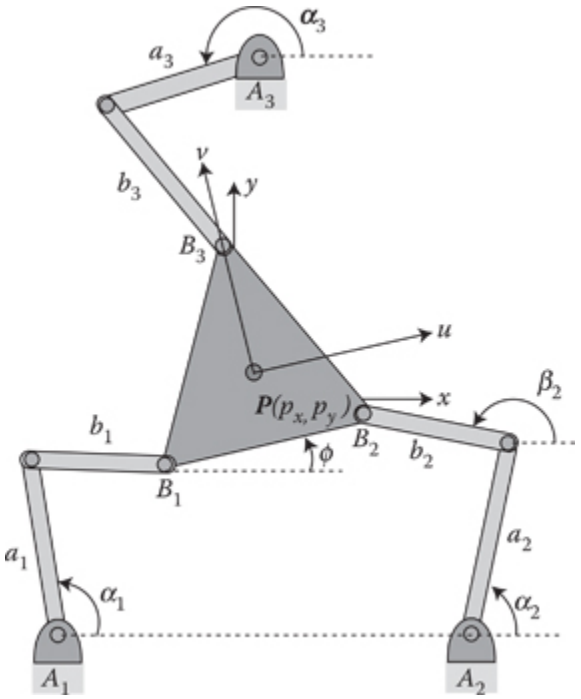

▶ Long Description for Figure 4.29

**FIGURE 4.29**

Pantograph mechanism. ⏎

4.4 A planar three degrees of freedom 3RRR parallel manipulator is illustrated in Figure 4.30. In this manipulator, three fixed pivots, $A_1, A_2$, and $A_3$ establish the geometry of the base, and three moving pivots, $B_1, B_2$, and $B_3$, establish the geometry of the moving platform. A fixed frame $\mathscr{A} : xy$ is defined at the central position of the moving plate $B_1B_2B_3$, with its $x$ direction aligned with the line connecting $A_1$ to $A_2$, while a moving frame $\mathscr{B} : uv$ is attached to the moving plate at point $\boldsymbol{P} = [p_x, p_y]^T$, the center of the equilateral triangle $\triangle B_1B_2B_3$ with the link length of $\ell$.

The fixed base attachment points $A_1, A_2$, and $A_3$ form another equilateral triangle with link length $L$.




▶ Long Description for Figure 4.30

**FIGURE 4.30**

The planar three degrees of freedom 3RRR parallel manipulator of Problem 4.4. ⏎

The positions of the fixed base attachment points $A_i$'s are indicated by $\boldsymbol{A}_i = [A_{x_i}, A_{y_i}]^T$, and those for the moving platform are indicated by $\boldsymbol{B}_i = [B_{x_i}, B_{y_i}]^T$. Frame $\{\mathscr{B}\}$ coincides with the frame $\{\mathscr{A}\}$ when the moving platform is at the central position, and the orientation angle $\phi$ denotes the angle of the moving platform with respect to the fixed frame. The limb lengths of the manipulator are denoted by $a_i$'s and $b_i$'s

for $i = 1, 2, 3$. The input limb angles are denoted by the vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$, while the position and orientation of the moving platform are represented by $\boldsymbol{P} = [p_x, p_y]^T$ and $\phi$, respectively.

a. From the geometry of the moving platform, determine the position of point $\boldsymbol{P}$ in terms of the position vectors of the moving platform vertices $\boldsymbol{B}_i = [B_{i_x}, B_{i_y}]^T$.

b. Formulate the vector closure equations. Assuming the given position and orientation of the moving platform $\boldsymbol{\chi} = [p_x, p_y, \phi]^T$, derive expressions for the limb angles $\boldsymbol{\alpha}$ as functions of $\boldsymbol{\chi}$. Eliminate the passive limb angles $\beta_i$ from these equations and provide expressions for the inverse kinematics of the manipulator.

c. Given the limb angles $\boldsymbol{q} = [\alpha_1, \alpha_2, \alpha_3]^T$, solve for the forward kinematics of the manipulator. Find expressions for the position and orientation of the moving platform $\boldsymbol{\chi} = [p_x, p_y, \phi]^T$.

d. Implement a program in MATLAB or Python using the following specific numerical values for the manipulator parameters. Given a trajectory $\boldsymbol{\chi}(t)$ for the moving platform, compute the corresponding limb angles $\boldsymbol{q}(t)$ using inverse kinematics. Then, using the time trajectories of $\boldsymbol{q}(t)$ obtained from inverse kinematic simulations, solve for the forward kinematics of the manipulator and compare the

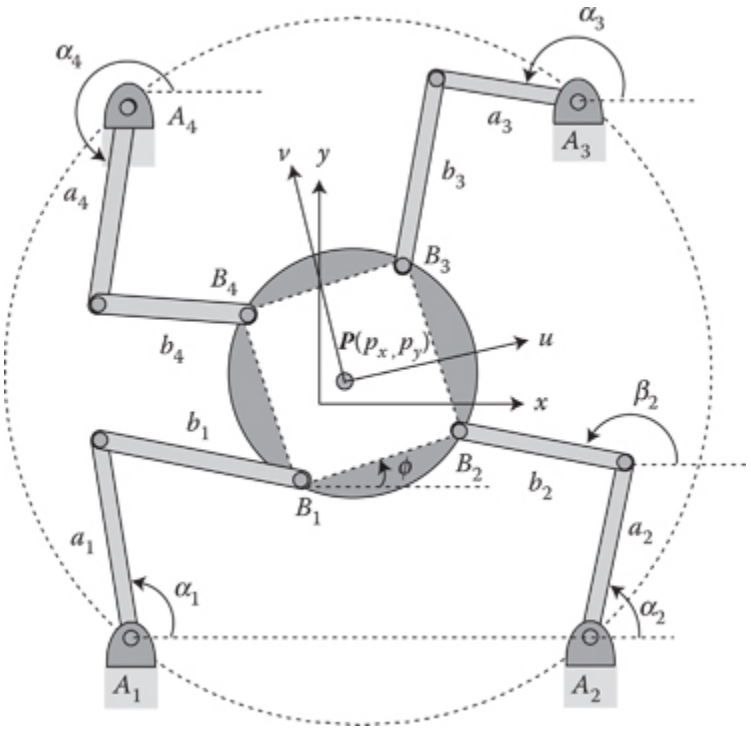
resulting $\boldsymbol{\chi}(t)$ with the original trajectory $\boldsymbol{\chi}_d(t)$. Notice that, due to multiple solutions in inverse kinematics, the solution must be found in such a way as to avoid discontinuities in manipulator trajectories. Provide illustrative plots of $\alpha_i(t)$ and possibly $\boldsymbol{\chi}_d(t) - \boldsymbol{\chi}(t)$ for verification purposes.

$$a_i = b_i = 1(m) \;\; ; \;\; L = 5(m) \;\; ; \;\; \ell = 1.5(m) \;\; ;$$

Consider cubic polynomial trajectories with the following parameters:

$$\boldsymbol{x}_o = [0, 0, 0]^T \;\; ; \;\; \boldsymbol{x}_f = [0.2, -0.1, \pi/3]^T \;\; ; \;\; t_o = 0 \;\; ; \;\; t$$

4.5 A planar redundant parallel manipulator with three degrees of freedom is depicted in Figure 4.31. This manipulator features four fixed pivots, $A_1, A_2, A_3$, and $A_4$, defining the base geometry, and four moving pivots, $B_1, B_2, B_3$, and $B_4$, defining the moving platform geometry. A fixed frame $\mathscr{A} : xy$ is positioned at the central location of the moving plate $B_1B_2B_3B_4$, with its $x$ direction aligned along the line connecting $A_1$ to $A_2$. A moving frame $\mathscr{B} : uv$ is attached to the moving plate at point $\boldsymbol{P} = [p_x, p_y]^T$, which is the center of the circle passing through the moving platform attachment points $B_i$'s.




▶ Long Description for Figure 4.31

**FIGURE 4.31**

The planar three degrees of freedom 4RRR redundantly actuated parallel manipulator of Problem 4.5. ⏎

The positions of the fixed base attachment points $A_i$'s are denoted by $\boldsymbol{A}_i = [A_{x_i}, A_{y_i}]^T$, and those for the moving platform are denoted by $\boldsymbol{B}_i = [B_{x_i}, B_{y_i}]^T$. Frame $\{\mathscr{B}\}$ coincides with frame $\{\mathscr{A}\}$ when the moving plate is at the central position, and the orientation angle $\phi$ of the moving platform represents the rotation angle between the moving frame and the fixed frame. Additionally, the limb lengths of the manipulator are denoted by $a_i$'s and $b_i$'s for $i = 1, 2, 3, 4$. The input limb angles are represented by the vector $\boldsymbol{q} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]^T$, while the pose of the moving

platform are given by $\boldsymbol{\chi} = [p_x, p_y, \phi]^T$ in the task space.

a. From the moving platform's geometry, determine the position vector $\boldsymbol{P} = [p_x, p_y]^T$ as a function of the position vectors of the moving platform vertices $\boldsymbol{B}_i = [B_{i_x}, B_{i_y}]^T$.

b. From the vector-loop equations, and under the assumption that the given moving platform position and orientation $\boldsymbol{\chi} = [p_x, p_y, \phi]^T$, derive expressions for the limb angles $\boldsymbol{q}$ as a function of $\boldsymbol{\chi}$. Remove the passive limb angles $\beta_i$'s from these equations and provide expressions for the inverse kinematics of the manipulator.

c. Given the limb angles $\boldsymbol{q} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]^T$, solve the forward kinematics to derive expressions for the position and orientation of the moving platform $\boldsymbol{\chi} = [p_x, p_y, \phi]^T$.

d. Develop a program in MATLAB or Python using specified numerical values for the manipulator parameters. This program should compute the limb angles $\boldsymbol{q}(t)$ for a given moving platform trajectory $\boldsymbol{\chi}(t)$. Then, using the trajectories of limb angles $\boldsymbol{q}(t)$ obtained from inverse kinematic simulations, solve for the forward kinematics of the manipulator and compare the calculated $\boldsymbol{\chi}(t)$ with the original trajectory $\boldsymbol{\chi}_d(t)$. Notice that, due to multiple solutions in inverse kinematics, the solution must be

found in such a way as to avoid discontinuities in manipulator trajectories. Provide illustrative plots of $\boldsymbol{q}(t)$ and possibly $\boldsymbol{\chi}_d(t) - \boldsymbol{\chi}(t)$ for verification purposes.

For simulations, assume the base and moving platform attachment points lie on circles with radii $R_A = 3m$ and $R_B = 1m$, respectively. The positions of these attachment points are determined by their angles of placement on the circles, as follows:

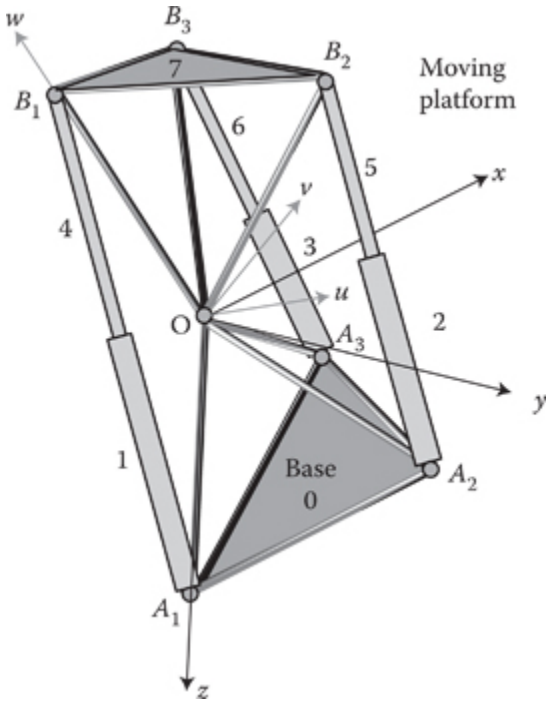
$$\theta_{A_i} = \theta_{B_i} = [-3\pi/4, -\pi/4, \pi/4, 3\pi/4]^T \ ;$$
$$a_i = b_i = 1.5m.$$

Finally, consider cubic polynomial trajectories with the following parameters:

$$\boldsymbol{x}_o = [0, 0, 0]^T \ ; \ \boldsymbol{x}_f = [0.2, -0.1, \pi/3]^T \ ; \ t_o = 0 \ ; \ t$$

4.6 Figure 4.32 illustrates a spatial orientation manipulator composed of two tetrahedrons pivoted at point $O$ with a spherical joint. The fixed base tetrahedron is denoted by $OA_1A_2A_3$, and the moving platform is represented by the $OB_1B_2B_3$ tetrahedron. Three extensible limbs connect the moving platform at points $B_i$ to the fixed base at points $A_i$ via spherical joints. Each limb consists of cylinder-piston members connected by a prismatic joint. This manipulator can

generate pure orientation for the moving platform relative to the base.




▶ Long Description for Figure 4.32

**FIGURE 4.32**

The two tetrahedral orientation manipulator of Problem 4.6. ⏎

For analysis, consider a fixed frame $\{\mathscr{A}\}$ attached to the base at point $O$, with the $z$-axis along $OA_1$ and the x-axis parallel to $A_1A_2$. Similarly, a moving frame $\{\mathscr{B}\}$ is attached to the moving platform at point $O$, with the $w$-axis along $OB_1$ and the $v$-axis parallel to $B_1B_2$. The positions of the fixed base attachment points $A_i$ are denoted by ${}^A\boldsymbol{A}_i = [A_{x_i}, A_{y_i}, A_{z_i}]^T$, and the positions for the moving platform are denoted by

${}^{B}\boldsymbol{B}_i = [B_{u_i}, B_{v_i}, B_{w_i}]^T$. The limb lengths of the manipulator are denoted by $\ell_i$ for $i = 1, 2, 3$. The input limb lengths are represented by the vector $\boldsymbol{q} = [\ell_1, \ell_2, \ell_3]^T$, while the orientation of the moving platform is given by the resulting rotation matrix ${}^{A}\boldsymbol{R}_B$.

a. Utilize a $w - u - w$ Euler angle representation for orientation, and express the position vector ${}^{A}\boldsymbol{B}$ as a function of the corresponding Euler angles $[\alpha, \beta, \gamma]^T$.

b. Assume the orientation of the moving platform is given. Using the vector-loop equations, solve the inverse kinematics and provide expressions for the limb lengths of the manipulator.

c. Assume the manipulator limb lengths $\boldsymbol{q} = [\ell_1, \ell_2, \ell_3]^T$ are given. Solve the forward kinematics of the manipulator to determine the orientation of the moving platform represented by $w - u - w$ Euler angles $[\alpha, \beta, \gamma]^T$ (Refer to [125]).

d. Assign some parametric values to the geometry of the manipulator, and develop a program in MATLAB or Python that, for a given moving platform orientation trajectory $[\alpha(t), \beta(t), \gamma(t)]^T$, computes the limb lengths $\boldsymbol{q}(t) = [\ell_1(t), \ell_2(t), \ell_3(t)]^T$. Then, using the given limb lengths trajectories, solve the forward kinematics to calculate the Euler angles over time. Ensure that due to multiple solutions in forward kinematics, the solution is found without any jumps in the manipulator trajectories. Compare

the calculated Euler angles with the original ones to verify your calculations. Provide illustrative plots of $\ell_i(t)$ and possibly the difference between the calculated and given Euler angles over time for verification purposes.

4.7 The architecture of a 3UPU parallel manipulator is depicted in Figure 4.33. This manipulator consists of a moving platform, a fixed base, and three supporting limbs with identical kinematic structures [52]. Each limb connects the fixed base to the moving platform via a universal joint, followed by an actuated prismatic joint, and another universal joint. Each prismatic joint is driven by a linear actuator. Since each limb has five joint degrees of freedom, each limb provides one constraint to the moving platform. It has been shown that the universal joints in each limb can be configured such that to provide one rotational constraint to the moving platform. Therefore, if the constraints are independent of one another, the moving platform exhibits pure translational motion [50].

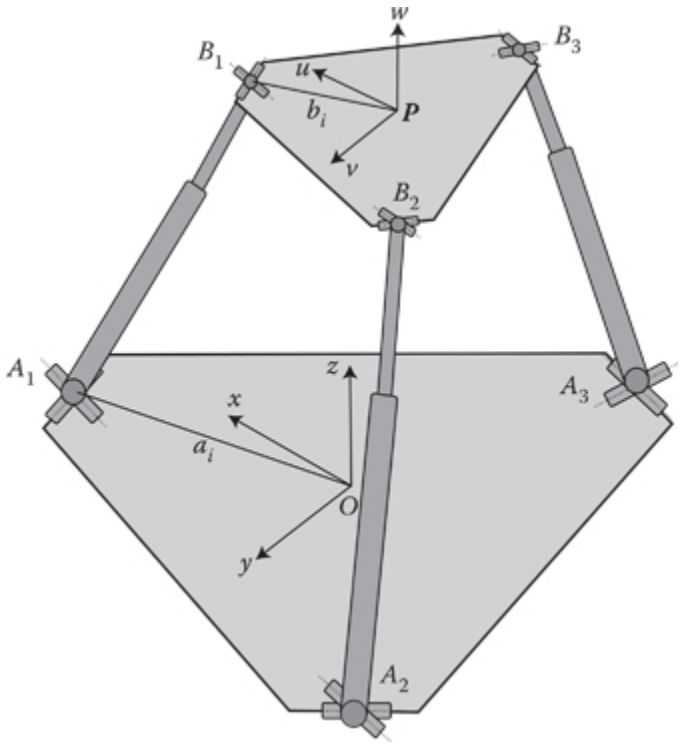



**FIGURE 4.33**

A three degrees of freedom 3UPU parallel manipulator of Problem 4.7, [50]. ⏎

For inverse kinematics analysis, the position vector $\boldsymbol{P} = [p_x, p_y, p_z]^T$ of a reference point $P$ on the moving platform is given, and the task is to determine the active joint variables. For analysis, a fixed coordinate frame is attached to the fixed base at point $O$, and another coordinate frame is attached to the moving platform at point $P$. The $i$'th actuated limb is connected to the moving platform at point $B_i$ and to the fixed base at point $A_i$. Additionally, assume that points $B_1, B_2, B_3$ lie on the $u - v$ plane at a radial distance $r_b$ from point $P$. Points $A_1, A_2, A_3$ lie on the $x - y$ plane at a radial

distance $r_a$ from point $O$. Since the orientation of the moving platform relative to the fixed base remains constant in all positions, the rotation matrix ${}^{A}\boldsymbol{R}_{B} = \boldsymbol{I}$, and hence, ${}^{A}\boldsymbol{b}_i = {}^{B}\boldsymbol{b}_i$. The angle $\theta$ is measured from line $PB_1$ to the $u$ axis.

a. Given $\boldsymbol{p}$, solve the inverse kinematics to determine the limb lengths $q_i$.

b. Given the limb positions $q_i$, solve the forward kinematics to determine the position vector $\boldsymbol{P}$ of the moving platform.

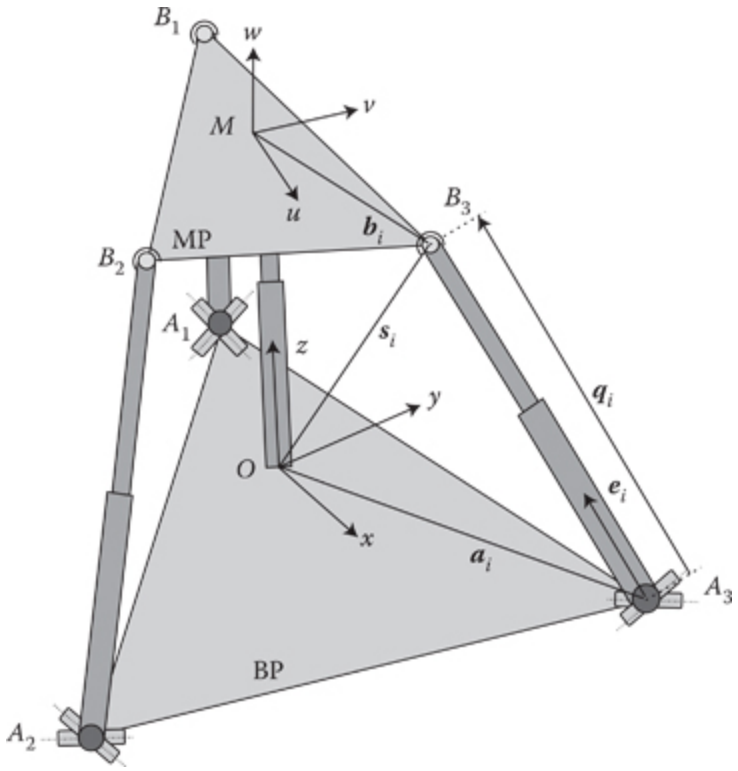
c. The geometric properties of the manipulator are as follows: the moving and base platforms are equilateral triangles with side lengths of 1 meter and 2 meters, respectively. For these parametric values, develop a program in MATLAB or Python to find the limb lengths $\boldsymbol{q}(t) = [q_1(t), q_2(t), q_3(t)]^T$ for the following moving platform trajectories. Consider cubic polynomial trajectories with the following parameters:

$$\boldsymbol{p}_o = [0, 0, 1]^T \;\; ; \;\; \boldsymbol{p}_f = [0.2, -0.3, 1.5]^T \;\; ; \;\; t_o = 0 \;\; ; \;\; t_f$$

d. For the determined limb length trajectories in part (c), solve the forward kinematics to calculate the position and orientation of the moving platform over time. Ensure that due to multiple solutions in

forward kinematics, the solution is found in such a way that no jumps are observed in the manipulator trajectories. Compare the calculated position and orientation of the moving platform with the original one to verify your programs. Provide illustrative plots of $q_i(t)$, and possibly the difference between the calculated and given position components over time for verification purposes.

4.8 Consider a 3UPS parallel manipulator introduced in [51], consisting of a moving platform (MP), a base platform (BP), and four limbs, as depicted in Figure 4.34. Three limbs are connected to the MP via spherical joints and to the BP via universal joints. The fourth limb, or central limb, is connected to the centroid of the MP via a universal joint and fixed to the BP. Each limb comprises two links connected by a prismatic joint. The manipulator has 3 degrees of freedom (DOF) according to the Chebychev–Grübler–Kutzbach formula. Therefore, three linear hydraulic actuators drive the prismatic joints of three limbs, while an idle prismatic joint is used for the central limb. This design, including the central limb, provides three independent degrees of freedom for the MP, namely, heave $h$: vertical displacement of the MP along the $z$ axis; pitch $\psi$: rotation of the MP about the $v$ axis; and roll $\phi$: rotation of the MP about the $u$ axis, as illustrated in Figure 4.34.




▶ Long Description for Figure 4.34

**FIGURE 4.34**

The four–legged three degrees of freedom 3UPS parallel manipulator of Problem 4.8, [51]. ⏎

With reference to Figure 4.34, the geometric parameters of the manipulator are defined as follows: The reference frame $\mathscr{A}$ : $xyz$ is located at $O$, the origin; the coordinate frame $\mathscr{B}$ : $uvw$ is located at $M$ and attached to the moving platform (MP). Each limb is connected to the MP at $B_i$ and to the base platform (BP) at $A_i$. The position vector of $A_i$ with respect to frame $\{\mathscr{A}\}$ is denoted by $\boldsymbol{a}_i$, and the length of the $i$'th limb is denoted by $q_i$. The unit vector along the $i$'th limb is represented by $\hat{\boldsymbol{e}}_i$, and the position vector of $B_i$

with respect to the coordinate frame $\mathscr{B}$ is represented by $\boldsymbol{b}_i$. The angle of the limb $i$ with respect to the $z$ axis is denoted by $\beta_i$, and the angle of the projection of limb $i$ on BP with respect to the $x$ axis is denoted by $\alpha_i$. The position vector of $A_i$ with respect to $M$, expressed in the reference frame $\{\mathscr{A}\}$ and frame $\{\mathscr{B}\}$, is represented by $\boldsymbol{b}_i$ and ${}^{B}\boldsymbol{b}_i$, respectively (for $i = 1, 2, 3$). Furthermore, the position vector of $M$ with respect to $O$ is defined by $\boldsymbol{p}$, and the rotation matrix of the MP with respect to the reference frame $\{\mathscr{A}\}$ is denoted by $\boldsymbol{R}$. The rotation matrix $\boldsymbol{R}$ can be determined directly from the roll and pitch angles of the MP.

a. Given $\boldsymbol{p}$ and $\boldsymbol{R}$, determine the motion of the actuators, $q_i$, and the unit vectors along each limb, $\hat{\boldsymbol{e}}_i$.

b. Given the limb lengths $q_i$, determine the motion of the moving platform (MP), i.e., the position vector $\boldsymbol{P}$ and the orientation of the MP denoted by $\boldsymbol{R}$.

c. The geometric properties of the manipulator are as follows: The moving and base platforms are equilateral triangles with sides of 1 meter and 2 meters, respectively. The minimum length of the central limb is 1 meter, and its stroke, the heave of the moving platform (MP), is 0.4 meters. The pitch and roll angles of the MP vary from $-\pi/6$ to $\pi/6$. For these parameter values, develop a program in MATLAB or Python where, for a given moving

platform position and orientation, the limb lengths $\boldsymbol{q}(t) = [q_1(t), q_2(t), q_3(t)]^T$ are determined.
To solve the inverse position kinematics, the position vector of point $M$ on the MP, $\boldsymbol{p}$, with respect to the reference frame is defined by a prescribed cycloidal maneuver as:

$$\begin{aligned} \boldsymbol{p} &= [0 \quad 0 \quad h(t)]^T; \\ h(t) &= 1 + 0.4\left[\frac{t}{T} - \frac{1}{2\pi}\sin\left(\frac{2\pi t}{T}\right)\right], (m) \quad 0 \leq t \leq \end{aligned}$$

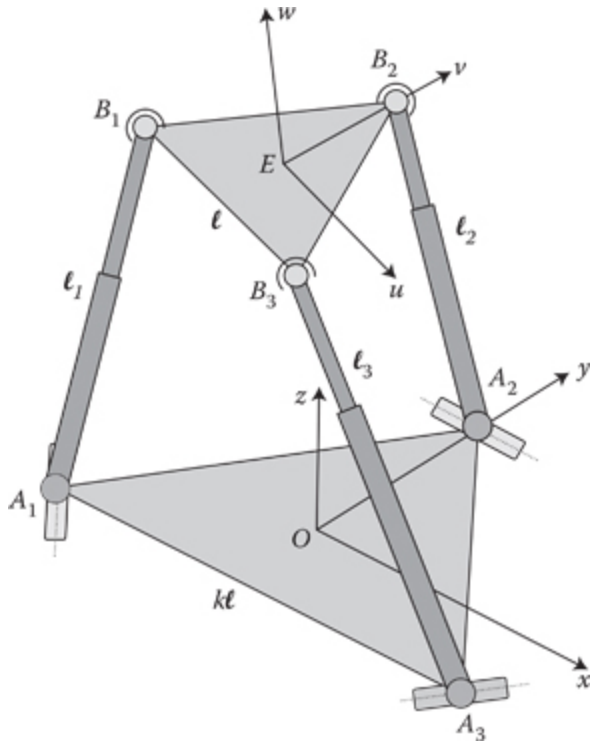
where $h(t)$ represents the vertical movement of MP and $T$ denotes the period of the maneuver in seconds. The rotation matrix $\boldsymbol{R}$ for MP relative to the reference frame can be defined as

$$\begin{aligned} \boldsymbol{R} &= \begin{bmatrix} \cos\psi(t) & 0 & \sin\psi(t) \\ 0 & 1 & 0 \\ -\sin\psi(t) & 0 & \cos\psi(t) \end{bmatrix}; \\ \psi(t) &= \frac{\pi}{6}\left[\frac{2t}{T} - \frac{1}{\pi}\sin(\frac{2\pi t}{T}) - 1\right] \quad 0 \leq t \leq T. \end{aligned}$$

d. For the specified trajectories of limb lengths in part (c), perform forward kinematics to determine the position and orientation of the moving platform over time. Given the multiple solutions in forward kinematics, ensure continuity in manipulator

trajectories during solution. Subsequently, compare the computed position and orientation of the moving platform with the original values to validate your algorithms. Include graphical plots of $q_i(t)$ 's and possibly show the discrepancies between the computed and provided trajectory components over time for verification.

4.9 Consider a three-degrees-of-freedom parallel manipulator depicted in Figure 4.35. This manipulator comprises two platforms: the moving platform $(B_1, B_2, B_3)$ and the base platform $(A_1, A_2, A_3)$, both equilateral triangles with side lengths $\ell$ and $k\ell$, respectively. Three extendable prismatic limbs connect the moving platform to the base platform via spherical joints and rotational joints; each base joint's revolute axis runs parallel to the opposite edge. The manipulator's 3RPS joint configuration includes revolute joints between the limbs and the base, actuated prismatic joints within the limbs, and spherical joints between the limbs and the moving platform. The manipulator is driven by the actuated prismatic joints within the limbs.




▶ Long Description for Figure 4.35

**FIGURE 4.35**

A three degrees of freedom 3RPS structure parallel manipulator of Problem 4.9, [53]. ⏎

Assume the end-effector of the manipulator coincides with the center point $E$ of the moving platform. Let frame $\mathscr{A} : (x, y, z)$ denote its coordinates in the base coordinate system centered at point $O$ of the base platform, with the $x - y$ plane aligned with the base platform. The joint coordinates are the lengths $\ell_1, \ell_2$, and $\ell_3$ of the platform limbs, as depicted in Figure 4.35. Frame $\mathscr{B} : (u, v, w)$ represents the coordinate system attached to the moving platform, with its origin at point $E$. Consider the coordinates of the vertices

$B_1, B_2, B_3$ of the moving platform. The coordinates of these points on the moving platform are derived based on the constraints imposed by the rotational joints at points $A_1, A_2, A_3$ as follows:

$$y_{B_1} = x_{B_1}/\sqrt{3} \;\; ; \;\; y_{B_3} = -x_{B_3}/\sqrt{3} \;\; ; \;\; x_{B_2} = 0.$$

From the equilateral triangular shape of the upper platform, it is evident that:

$$x = (x_{B_1} + x_{B_2} + x_{B_3})/3 \;\; ; \;\; y = (y_{B_1} + y_{B_2} + y_{B_3})/3\sqrt{3} \;\; ; \;\; z$$

Therefore, through substitution and manipulation, we arrive at the following constraint on the positions of the end effector.

$$x_{B_1} = (3x + 3\sqrt{3}y - y_{B_2}\sqrt{3})/2 \;\; ; \;\; x_{B_3} = (3x - 3\sqrt{3}y + y_{B_2}\sqrt{}$$

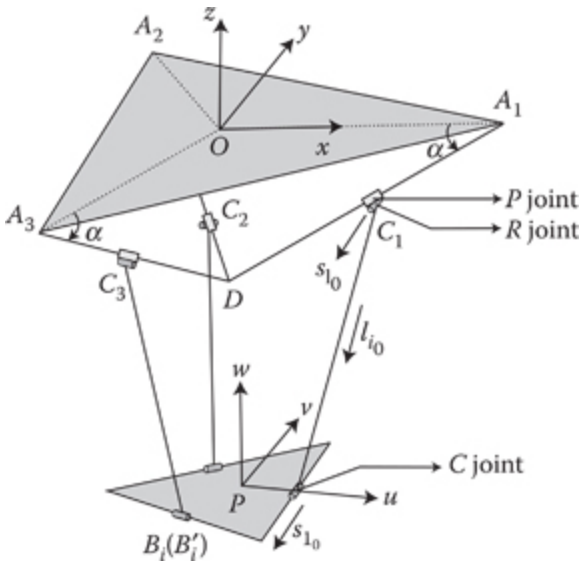
**a.** From the set of four equations of the distances:

$$B_1E^2 = \ell^2/3 \;\; ; \;\; B_1B_2^2 = \ell^2 \;\; ; \;\; B_2B_3^2 = \ell^2 \;\; ; \;\; B_1B_3^2 =$$

Derive the coordinates of points $B_i$ in terms of the end-effector $E$ coordinates $[x, y, z]^T$ and length $\ell$.

**b.** These equations illustrate that the inverse kinematic problem has multiple solutions. Determine the correct solutions with real values for the coordinates of points $B_i$ and ensure continuous curves for the joint coordinates as functions of the end-effector trajectories [53].

4.10 The schematic diagram of a 3PRC translational parallel manipulator is illustrated in Figure 4.36. It consists of a moving platform, a fixed base, and three limbs with identical kinematic structures. Each limb connects the fixed base to the moving platform via an actuated $P$ joint, an $R$ joint, and a $C$ joint in sequence. Hence, the moving platform is connected to the base by three identical $\underline{P}RC$ linkages. The mobility analysis of this manipulator [54] indicates that to prevent any change in the orientation of the moving platform, it is sufficient for the three joint axes within the same limb to meet specific geometric conditions. Referring to Figure 4.36, these conditions are:




▶ Long Description for Figure 4.36

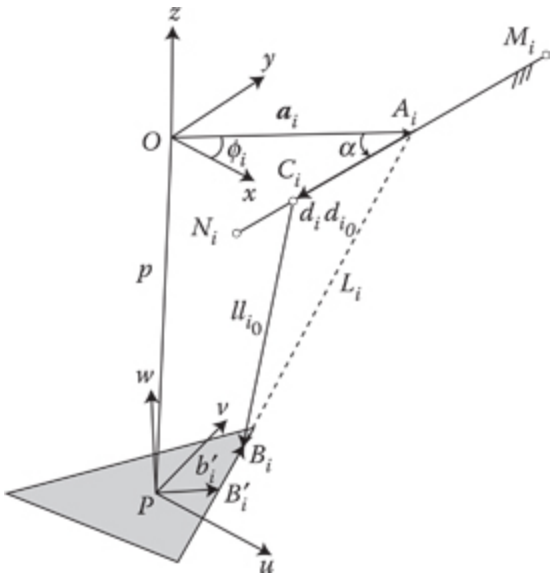
**FIGURE 4.36**

The schematics of a three degrees of freedom 3PRC translational parallel manipulator of Problem 4.10, [54]. ⏎

- The $R$ joint axis $(\boldsymbol{r}_i)$ and the $C$ joint axis $(\boldsymbol{c}_i)$ within the $i$'th limb, for $i = 1, 2, 3$, are parallel to the same unit vector $\boldsymbol{s}_{i_0}$.
- The limbs are arranged so that $\boldsymbol{s}_{i_0} \neq \boldsymbol{s}_{j_0}$ for $i \neq j$, and $i, j = 1, 2, 3$.

To facilitate the kinematic analysis, as shown in Figures 4.36 and 4.37, assign a fixed Cartesian frame $\mathscr{A} : (x, y, z)$ at the center point $O$ of the fixed base, and a moving Cartesian frame $\mathscr{B} : (u, v, w)$ on the triangular moving platform at the center point $P$, with the $z$ and $w$ axes perpendicular to the platform, and the $x$ and $y$ axes parallel to the $u$ and $v$ axes, respectively. Additionally, the $i$'th limb $C_iB_i$ for

$(i = 1, 2, 3)$ with a length of $\ell$ is connected to the moving platform at point $B_i$, which is located on the axis of the $i$'th $C$ joint.




▶ Long Description for Figure 4.37

**FIGURE 4.37**

A schematics for the 3PRC translational parallel manipulator used for kinematic analysis of the manipulator in Problem 4.10. ⏎

where $B'_i$ denotes the point on the moving platform that coincides with the initial position of $B_i$, and the three points $B'_i$ for $i = 1, 2, 3$ lie on a circle of radius $b$. The three rails $M_iN_i$ intersect at point $D$ and intersect the $x - y$ plane at points $A_1$, $A_2$, and $A_3$, which lie on a circle of radius $a$. The sliders of $P$ joints $C_i$ are constrained to move along the rails between $M_i$ and $N_i$. Additionally, the axis of the $P$ joint is perpendicular to

the axes of the $R$ and $C$ joints within the $i$'th limb. Angle $\alpha$ is measured from the fixed base to rails $M_iN_i$ and is defined as the layout angle of actuators. To achieve a compact architecture, $\alpha$ is designed within the range of $0 \leq \alpha \leq \pi/2$. Angle $\phi_i$ is defined from the $x$ axis to $OA_i$ in the fixed frame, and also from the $u$ axis to $PB'_i$ in the moving frame. Without loss of generality, let the $x$ axis align with $OA_1$, and the $u$ axis align with $PB'_1$. Therefore, $\phi_1 = 0$.

a. The purpose of inverse kinematics is to determine the actuated variables from a given position of the moving platform. Write the vector loop closures, and assuming that the moving platform position $\boldsymbol{x} = [p_x, p_y, \phi]^T$ is given, while its orientation is ${}^A\boldsymbol{R}_B = \boldsymbol{I}$, provide the expressions for the inverse kinematics of the manipulator.

b. Given a set of actuated inputs, the position of the moving platform can be determined through forward kinematic analysis. Using the inverse kinematic equations, derive three second-degree algebraic equations in the unknowns $p_x$, $p_y$, and $p_z$.

c. Apply the Sylvester dialytic elimination method [54] to reduce the above system of equations to an eighth-degree polynomial in a single variable.

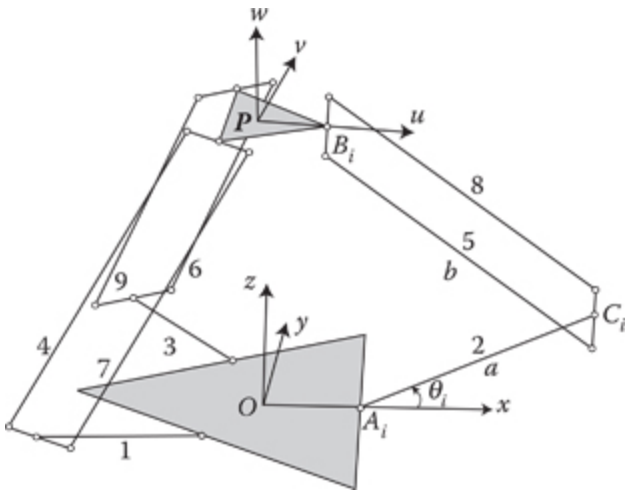
d. Assign some parametric values to the geometry of the manipulator and develop a program in MATLAB or Python to find the actuator inputs $d_i$'s for a given

moving platform position trajectory $[p_x(t), p_y(t), p_z(t)]^T$ (for example, the one given in problem 4.7). For the determined actuator input trajectories, solve forward kinematics and calculate the position vector of the moving platform over time. In forward kinematic simulations, use two methods: first, a numerical solution to part (b), and then numerical solutions to part (c). Ensure the solutions are found without jumps in the manipulator trajectories due to multiple solutions in forward kinematics.

Compare the calculated position vectors from each solution to the original to verify your programs. Provide illustrative plots for actuator inputs and possibly the difference between the calculated and given position vectors over time for verification purposes. Analyze the computational effort and convergence of the forward kinematic solutions in each method.

4.11 The Delta robot was developed in the early 1980s by Reymond Clavel at EPFL in Switzerland [24]. This innovative robot was designed to pick and place light and small objects at extremely high speeds. As illustrated in Figure 4.38, the Delta robot is a parallel robot comprising three main components: a fixed base, a moving platform, and three arms. Each arm is divided into two sections: the actuated lower arm and the upper arm. The lower arm is connected to the fixed

base via revolute joints, allowing rotational movement. A distinctive feature of the upper arm is the use of parallelograms, which are crucial for maintaining the orientation of the end effector. These parallelograms are connected to both the lower arm and the moving platform through spherical joints. This configuration enables the moving platform to achieve three degrees of translational freedom, allowing precise and rapid movement in three-dimensional space.




▶ Long Description for Figure 4.38

**FIGURE 4.38**

The schematics of a three degrees of freedom Delta robot of Problem 4.11. ⏎

For the kinematic analysis, consider the Delta robot without the gripper actuator, depicted in Figure 4.38. In this figure, a fixed Cartesian frame $\mathscr{A} : (x, y, z)$ is positioned at the center point $O$ of the fixed base. A moving Cartesian frame $\mathscr{B} : (u, v, w)$ is located on the

triangular moving platform at the center point $P$. The $x$ axis is aligned in the direction of $OA_1$, and the $u$ axis is aligned along $PB_1$. The lower arms are uniform in length and denoted by $a$, while the upper arms also have equal lengths denoted by $b$.

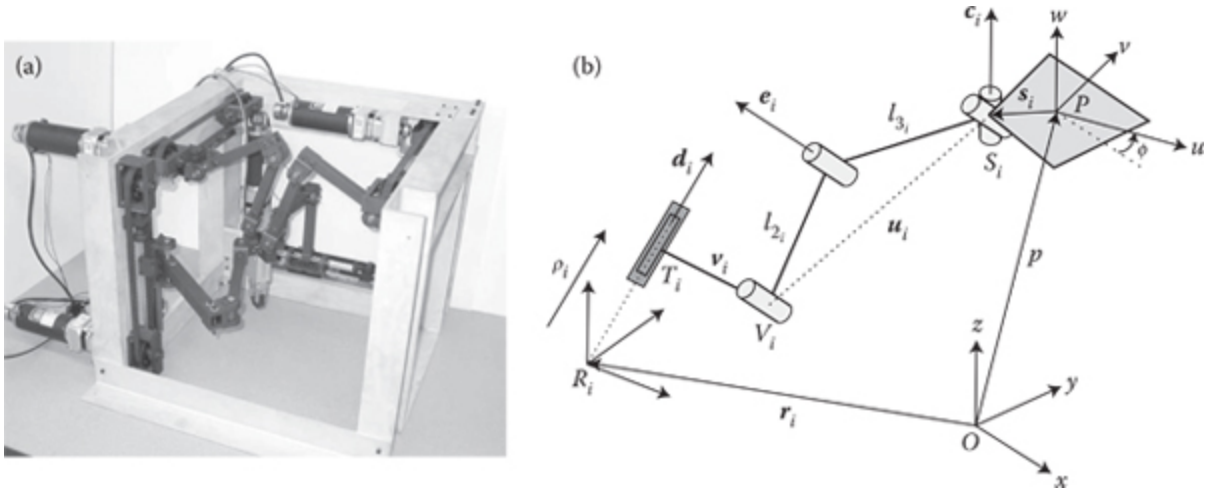
a. Compute the inverse kinematics of this manipulator to obtain expressions for the lower arm angles $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^T$ as functions of the given moving platform position $\boldsymbol{P} = [p_x, p_y, p_z]^T$.

b. Employ the inverse kinematic formulations to establish a forward kinematic solution for the manipulator through numerical methods.

c. Assign specific geometric values to the manipulator's parameters and develop a MATLAB or Python program. This program should compute the actuator inputs $\theta_i$'s for a given trajectory of the moving platform position $[p_x(t), p_y(t), p_z(t)]^T$ (for example, as provided in problem 4.7). Subsequently, using the determined actuator input trajectories, solve the forward kinematics and compute the position vector of the moving platform over time. Ensure that the correct solution is identified in forward kinematics to avoid discontinuities in the manipulator trajectories due to multiple solutions. Compare the computed position vectors with the original ones to validate your programs. Include illustrative plots of the actuator inputs and possibly

the differences between the calculated and given position vectors over time for verification purposes.

4.12 A picture of a Quadrupteron robot and the schematic of one of its limb kinematics are depicted in Figure 4.39 [55]. This robot comprises a moving platform connected to a fixed base through four limbs, each equipped with five joints. These joints include, from the base to the moving platform, a *P* joint followed by four *R* joints labeled sequentially from 1 to 5. To ensure practicality, the axes of the last two *R* joints within each limb are arranged to be intersected. The *P* joints, which are mounted directly on the base, are the only actuated ones. To enable the architecture to achieve three translational and one rotational motion, driven by the four actuated *P* joints, the following geometric constraints must be met:




▶ Long Description for Figure 4.39

**FIGURE 4.39**

The Quadrupteron: a parallel manipulator with 3T1R degrees-of-freedom (Courtesy of Clément Gosselin, Laval University, Canada.) (a) and a schematic of the kinematic structure of one of its limbs (b) (Adapted from P. L. Richard and C. M. Gosselin. *ASME Journal of Mechanical Design*, 129(6):611–616, 2007.) ⏎

**a.** The axes of the four $R$ joints connected to the moving platform are all parallel.

**b.** The axes of joints 2, 3, and 4 within the same limb are parallel.

**c.** The axes of joints 1 and 2 within the same limb are not perpendicular to each other.

**d.** The axes of joint 2 in all four limbs are not all parallel to a plane parallel to the axes of the $R$ joints attached to the moving platform.

**e.** In at most one of the four limbs, all axes of joints 2 to 5 within the limb are parallel. In this scenario, joints 4 and 5 merge into a single joint. If such a limb exists, it is referred to as limb 1 for convenience.

**f.** The axes of joint 2 in all four limbs are not all parallel to a plane. This condition ensures that the axes of joints 4 and 5 within a limb are not perpendicular to each other in at least one of the four limbs.

Conditions (a) to (c) ensure that the moving platform can achieve a minimum of four degrees of freedom, specifically a 3T1R (three translational and one rotational) motion. Condition (d), along with conditions (a) to (c), guarantees that the moving platform indeed possesses four degrees of freedom. Conditions (e) and (f) further ensure that the motion of the moving platform can be controlled effectively using the four $P$ joints.

In Figure 4.39, two unit vectors $\boldsymbol{c}_i$ and $\boldsymbol{e}_i$ represent the directions of joint 5 and joints 2, 3, and 4, respectively, within the $i$-th limb (assuming $i = 1, 2, \ldots, 4$ unless specified otherwise). A $\underline{P}4R$ limb kinematics imposes either one constraint on the moving platform, preventing rotation about any axis parallel to $\boldsymbol{c}_i \times \boldsymbol{e}_i$ if $\boldsymbol{c}_i \neq \boldsymbol{e}_i$, or two constraints if $\boldsymbol{c}_1 = \boldsymbol{e}_1$, preventing rotation about any axis perpendicular to $\boldsymbol{c}_1$. Since all $\boldsymbol{c}_i$ vectors are parallel, the directions in which rotations are constrained are all parallel to a specific plane. Consequently, the moving platform can only rotate in a direction orthogonal to that plane, specifically along the direction of $\boldsymbol{c}_i$, thus facilitating the 3T1R motions.

Referring to Figure 4.39, the geometry of the manipulator is described using the following quantities. A fixed reference frame $O - xyz$ is located on the base, and a moving reference frame $P - uvw$ is attached to the moving platform. The position of the

moving platform is represented by the vector $\boldsymbol{P} = [p_x, p_y, p_z]^T$, which connects point $O$ to point $P$. The orientation of the moving platform relative to the base is defined by the rotation matrix $\boldsymbol{R}$. Without loss of generality, both the $z$ and $w$ axes are aligned parallel to $\boldsymbol{c}_i$. Therefore, $\boldsymbol{c}_i = [0, 0, 1]^T$, and the rotation matrix $\boldsymbol{R}$ can be expressed as:

$$\boldsymbol{R} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The direction of the $i$'th fixed $P$ joint is represented by the unit vector $\boldsymbol{d}_i$. A reference point $R_i$ is positioned on the axis of the $i$'th fixed $P$ joint, as illustrated in Figure 4.39. The motion of the $P$ joint is measured relative to this reference point and denoted by the signed distance $\rho_i$ between $R_i$ and a point $T_i$ on the sliding body. Point $S_i$ marks the intersection of the axes of the last two joints in limb $i$. Additionally, the position vector of point $R_i$, connecting $O$ to $R_i$ in the fixed frame, is denoted by $\boldsymbol{r}_i = [r_{x_i}, r_{y_i}, r_{z_i}]^T$. The position vector of point $S_i$ in the moving frame is denoted by ${}^P\boldsymbol{s}_i = [s_{x_i}, s_{y_i}, s_{z_i}]^T$, and in the fixed frame, it is represented by $\boldsymbol{s}_i = \boldsymbol{R}, {}^P\boldsymbol{s}_i$. Since the axes of joints 2, 3, and 4 within a limb are parallel, a point $V_i$ can be defined on the axis of joint 2 such that the vector $\boldsymbol{u}_i$,

connecting $V_i$ to $S_i$, is orthogonal to the unit vector $\boldsymbol{e}_i = [e_{x_i}, e_{y_i}, e_{z_i}]^T$.

a. Find the inverse kinematic solutions for this manipulator to derive equations describing the limb lengths $\rho_i$. These equations will be formulated based on the given position $\boldsymbol{P} = [p_x, p_y, p_z]^T$ of the moving platform and its orientation $\phi$.

b. Utilize the inverse kinematic equations to compute a forward kinematic solution for the manipulator employing numerical techniques.

c. Assign specific geometric parameters to the manipulator and develop a MATLAB program. This program will compute the actuator inputs $\rho_i(t)$ for a given trajectory of the moving platform position $[p_x(t), p_y(t), p_z(t)]^T$ (such as that provided in problem 4.7) and orientation $\phi(t)$. Subsequently, use these determined actuator input trajectories to solve the forward kinematics and compute the moving platform's position vector over time. Ensure that the solution method avoids jumps in the manipulator trajectories, given the possibility of multiple solutions in forward kinematics. Validate your programs by comparing the calculated position vectors with the original ones. Present illustrative plots of the actuator inputs and, if applicable, the differences between the calculated and given position vectors over time for verification purposes.

# 5

# *Jacobian Matrix and Singularity*

DOI: 10.1201/9781003590408-5

This chapter explores the Jacobian matrix and its critical role in analyzing the kinematics and dynamics of parallel manipulators, with particular attention to singularities. It begins with a foundational discussion on the Jacobian matrices of parallel manipulators, outlining their importance in understanding motion and force transmission within robotic systems. A systematic approach to Jacobian matrix derivation is presented, emphasizing its utility in connecting joint velocities and actuator torques to moving platform twist rate and wrenches.

The chapter then examines singularity analysis, categorizing singularities into three main types: inverse kinematic singularity, forward kinematic singularity, and combined kinematic singularity. Each type is explained with clarity, highlighting the implications of these conditions on manipulator performance, such as loss of

control or force transmission. Strategies for avoiding singularities are further explored, offering practical insights for improving system design and operation.

Three case studies are introduced to demonstrate the application of Jacobian analysis and singularity evaluation in real-world scenarios. The planar cable robot serves as the first example, where velocity loop closure equations are developed, and singularity and sensitivity analyses are performed to assess its operational robustness. The second case, focusing on the shoulder manipulator, provides insights into its velocity relationships and singular configurations, reinforcing the theoretical principles. The final case study examines the Stewart–Gough platform, showcasing advanced singularity analysis and its implications for high-precision applications.

The chapter concludes with a collection of challenging problems, encouraging readers to refine their understanding of Jacobian matrices, singularities, and their impact on manipulator performance. These exercises bridge the gap between theory and practice, equipping readers with essential analytical and design tools.

## 5.1 Introduction

In the preceding chapter, we explored the forward and inverse kinematics of parallel robots, establishing a solid foundation for understanding their basic operational

principles. In this chapter, we venture further into the intricate world of kinematic analysis of robot manipulators, shifting our focus from static positioning to differential kinematics. Here, we emphasize the examination of both linear and angular velocities of the robot's moving platform, exploring their intricate relationships with the angular velocities of the active limbs.

To achieve this, we employ the fundamental concepts of linear velocity of a point and angular velocity of a rigid body. These concepts serve as the cornerstone for our detailed differential kinematic analysis of parallel manipulators. Our exploration naturally culminates in the introduction of the parallel robot's Jacobian matrix. The Jacobian matrix plays a pivotal role in robotics. It not only elucidates the relationship between the joint variable velocities of a parallel manipulator and the linear and angular velocities of the moving platform, but it also provides the necessary transformations to determine the actuator forces from the forces and moments acting on the moving platform. This dual functionality of the Jacobian matrix underscores its importance in both differential kinematics and static force analyses, making it an indispensable tool for anyone seeking to master the complexities of parallel robotic systems.

The optimization of a robot's kinematic structure is a fascinating and critical area within robotics research, driving many technological advancements [126, 127]. In this domain, many studies have concentrated on enhancing

criteria such as the dexterity and manipulability of robots. These aspects are intricately connected to the fundamental properties of the Jacobian matrix [128, 129]. Recognizing the pivotal role the Jacobian matrix plays in robot design, this chapter dives into the detailed analysis of a robot's dexterous motion and the identification of singular configurations. To achieve this, we will thoroughly explore the properties of the Jacobian matrix. We begin by examining the robot's configurations in singular states, offering a clear definition of singular configurations and investigating their correlation with the characteristics of the Jacobian matrix [130, 131]. Singular configurations are crucial to understand as they represent positions where the robot may lose certain degrees of freedom, leading to a loss of control or precision.

In the context of parallel manipulators, maintaining kinematic accuracy is of paramount importance. It is well established that these manipulators can experience a specific type of local degeneracy in their motion [132]. This chapter will provide a comprehensive overview of these phenomena, discussing how singular configurations can impact the performance and accuracy of parallel manipulators. By understanding and addressing these issues, we can enhance the design and functionality of robotic systems, ensuring they perform optimally even in complex tasks. Physically, singular configurations lead to:

a. An instantaneous change in the degrees-of-freedom of the system, resulting in a loss of controllability.

b. A significant degradation of the natural stiffness, which may cause very high joint forces or torques.

Therefore, it is crucial to identify singular configurations at the design stage to enhance the robot's performance. Singularities in parallel manipulators have been extensively studied by numerous researchers [133, 134, 135]. In the context of Jacobian analysis of parallel manipulators, Gosselin and Angeles [132] were among the first to propose the use of two Jacobian matrices to specify inverse and forward kinematics singularities. They demonstrated that by expressing the velocity equations of closed-loop chains with two Jacobian matrices, singularities could be classified into three distinct types, each with unique physical interpretations.

In [136], Merlet utilized Grassmann geometry to investigate the singularities of spatial six-degrees-of-freedom parallel manipulators, providing a comprehensive list of possible singular configurations. This method is versatile and can be applied to various mechanisms with different degrees of freedom. Additionally, screw theory has been extensively employed to derive Jacobian matrices in parallel structures [134, 137], further contributing to our understanding of these complex systems.

What follows is a systematic approach to analyze the differential kinematics of parallel robots. This approach not only aids in identifying singular configurations but also provides insights into optimizing the robot's design for improved performance and reliability. By understanding

and addressing these singularities, we can ensure that parallel manipulators operate efficiently and effectively in various applications.

## 5.2 Jacobian Matrices of a Parallel Manipulator

A general parallel manipulator, such as the one illustrated in Figure 1.11, comprises a moving platform connected to a fixed base by multiple limbs. In a fully parallel manipulator, the number of limbs corresponds to the number of degrees of freedom of the moving platform. Each limb consists of several joints, with only one being actuated. Consequently, some joints are powered by actuators, while others remain passive. Typically, the number of actuated joints is greater than or equal to the number of degrees of freedom.

In scenarios where the number of actuators surpasses the number of degrees of freedom, the manipulator is classified as redundantly actuated. This redundancy can enhance the manipulator's performance by increasing its stiffness and load-carrying capacity, as well as improving fault tolerance. The design of such manipulators requires careful consideration of the kinematic and dynamic interactions between the limbs and joints to optimize their efficiency and functionality.

In general, let $\boldsymbol{q}$ denote the vector of actuated joint coordinates, and $\boldsymbol{\chi}$ represent the vector of moving platform motion variables, which include both position and orientation variables. In general, we have:

$$\boldsymbol{q} = [q_1, q_2, \ldots, q_m]^T \tag{5.1}$$

and

$$\boldsymbol{\chi} = [x_1, x_2, \ldots, x_n]^T, \tag{5.2}$$

where $q_i$ signifies either the angular rotation of an actuated revolute joint or the linear displacement of an actuated prismatic joint, and $x_i$ represents the position or orientation variable of the moving platform. Here, $m$ denotes the number of actuated joints in the manipulator, while $n$ represents the number of degrees of freedom of the manipulator.

Typically, $m \geq n$; however, for fully parallel manipulators, $m = n$, whereas for redundant manipulators, $m > n$. The vectors $\boldsymbol{q}$ and $\boldsymbol{\chi}$ are commonly referred to as the active joint variables of the limbs and the variables of the moving platform pose, respectively. These vectors are connected through a system of nonlinear algebraic equations that represent the kinematic formulation of the

manipulator. This relationship can generally be expressed as:

$$\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\mathcal{X}}) = \mathbf{0}.$$

(5.3)

This equation constitutes a set of $m$ nonlinear algebraic equations, where $\mathbf{0}$ represents an $m$-dimensional zero vector. To derive a relationship between the input joint rates and the moving platform output twist rate, we differentiate Equation 5.3 with respect to time, yielding:

$$\boldsymbol{J}_q \dot{\boldsymbol{q}} = \boldsymbol{J}_x \dot{\boldsymbol{\mathcal{X}}},$$

(5.4)

where

$$\boldsymbol{J}_x = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mathcal{X}}} \quad \text{and} \quad \boldsymbol{J}_q = -\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}.$$

(5.5)

These matrices, $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$, are known as the Jacobian matrices of the parallel manipulator. As proposed by Gosselin and Angeles [132], this differentiation results in two distinct Jacobian matrices, with $\boldsymbol{J}_q$ being a square matrix of dimension $m$ and $\boldsymbol{J}_x$ being an $m \times n$ matrix. For

fully parallel manipulators, when $m = n$, both Jacobian matrices are square.

As shown in Figure 5.1, the general Jacobian matrix, $\boldsymbol{J}$, of a parallel manipulator is defined as:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}\dot{\boldsymbol{\chi}}, \tag{5.6}$$

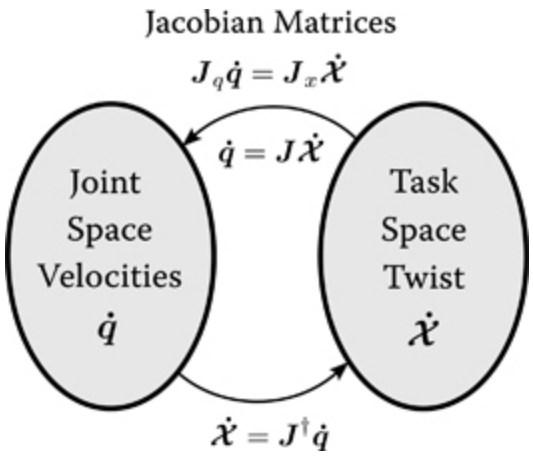



**FIGURE 5.1**
Representation of the active joint and task space variables delineating different Jacobian matrix definitions of parallel robots. ⏎

where, assuming $\boldsymbol{J}_q$ is nonsingular,

$$\boldsymbol{J} = \boldsymbol{J}_q^{-1}\boldsymbol{J}_x. \tag{5.7}$$

It is essential to recognize that the general Jacobian $\boldsymbol{J}$ of a parallel manipulator, as described in Equation 5.6 and illustrated in Figure 5.1, aligns with the direction in which inverse kinematics is defined. Thus, it corresponds to the inverse Jacobian definition applied to serial manipulators. This differentiation is vital for comprehending the kinematic behavior and control of parallel robots and their connection to serial robots, offering an understanding of how joint velocities convert into platform movements.

## 5.3 Jacobian Matrix Derivation

To derive the Jacobian matrices of a parallel manipulator and establish the relationship between the moving platform twist rate, $\dot{\boldsymbol{\chi}}$, and the active joint rates, $\dot{\boldsymbol{q}}$, two primary methods are discussed in the literature. The first approach involves using screw coordinates to represent both the moving platform twist and the joint rates. This method, known as the *screw-based Jacobian*, leverages the mathematical framework of screw theory to derive the Jacobian matrices, providing a comprehensive and systematic way to capture the motion and constraints of the manipulator.

The second, more commonly employed method, involves using the conventional *velocity loop closures*. This approach is often preferred due to its relative simplicity and straightforward application. In this method, the

kinematic loop closures of the manipulator are further analyzed to form velocity loop closures that relate the active joint rates of the limbs to the platform twist rate. This technique is examined in detail in this section to highlight its practical application in deriving the Jacobian matrices of parallel manipulators.

Generally, velocity loop closures are derived by directly differentiating kinematic loop closures. As detailed in Section 4.3, at the displacement level, the closure of each kinematic loop can be expressed in vector form, as shown in Equation 4.2. To facilitate further analysis, let's rewrite this equation for the scenario where the number of limbs, and consequently, the number of loop closures, is $m$. This allows us to systematically examine the relationships between the velocities of different components in different types of parallel manipulators:

$$\boldsymbol{p} = \boldsymbol{a}_i + \boldsymbol{f}(\boldsymbol{q}_i) - \boldsymbol{R}\,\boldsymbol{b}_i \quad \text{for} \;\; i = 1, \ldots, n \tag{5.8}$$

This loop closure is expressed as a function of the position and orientation variables of the moving platform, $\boldsymbol{p}$ and $\boldsymbol{R}$, respectively. It also incorporates the position vectors that describe the known geometry of the base and the moving platform, denoted by $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$. Additionally, the limb vector, denoted by $\boldsymbol{d}_i = \boldsymbol{f}(\boldsymbol{q}_i)$, depends on the active joint variables $\boldsymbol{q}_i$, and possibly some other passive joint variables corresponding to the unactuated joints. By differentiating

this equation with respect to time, we can derive the velocity loop closures, which are essential for forming the Jacobian matrices. This process will enable us to relate the moving platform twist rate to the joint rates, providing a clear understanding of the kinematic behavior of parallel manipulators.

Let us take the time derivative of both sides of Equation 5.8, resulting in:

$$\dot{\boldsymbol{p}} = \dot{\boldsymbol{d}}_i - \boldsymbol{\omega} \times \boldsymbol{R}\,\boldsymbol{b}_i \quad \text{for} \;\; i = 1, \ldots, m. \tag{5.9}$$

Note that the time derivatives of the $\boldsymbol{a}_i$ vectors with respect to the fixed frame $\{\mathscr{A}\}$, and the $\boldsymbol{b}_i$ vectors with respect to the moving frame $\{\mathscr{B}\}$, are all zero, while the last term in these equations comes from the relation 3.51, which provides the relationship between the derivative of $\boldsymbol{b}_i$ in the fixed coordinate system and its derivative in the moving coordinate system. This equation establishes a relationship between the active and passive joint rates, $\dot{\boldsymbol{d}}_i$, and the moving platform's linear and angular velocities, $\dot{\boldsymbol{p}}$ and $\boldsymbol{\omega}$, respectively.

To eliminate the effects of unactuated joint rates within each limb, the velocity loop closures are projected onto a suitable vector orthogonal to all vectors representing passive joint rates. This projection isolates only the active joint rates $\dot{\boldsymbol{q}}_i$ within the vector loop closures. While this step is a standard part of Jacobian derivation for all parallel

manipulators, selecting the correct orthogonal vector depends on the specific structure and geometry of each manipulator. The process and nuances of this derivation are exemplified through various examples in subsequent sections.

Subsequently, the resulting equations are organized into matrix form, yielding the Jacobian matrices of the manipulator as defined in Equation 5.6. To maintain consistency, the conventional Jacobian defines the rate of moving platform twist rate $\dot{\boldsymbol{\chi}}$ as a six-dimensional vector comprising the linear velocity of a point on the moving platform followed by the angular velocity of the platform, expressed as:

$$\dot{\boldsymbol{\chi}} = \begin{bmatrix} \boldsymbol{v}_p \\ \boldsymbol{\omega} \end{bmatrix}. \tag{5.10}$$

This method is best illustrated through case studies, which will be elaborated upon in Sections 5.5 through 5.7.

## 5.4 Singularity Analysis

Singularities of parallel mechanisms have been extensively studied by researchers. Gosselin and Angeles [132] categorized three types of singularities for closed-loop mechanisms based on the roots of the determinant of

the Jacobian matrices. Tsai [8] adopted similar definitions, referring to these as inverse, forward, and combined singularities, corresponding to the first, second, and third kinds, respectively.

Further advancements were made by Zlatanov, Fenton, and Benhabib [138], who introduced six types of singular configurations with detailed physical interpretations. Additionally, Zlatanov, Bonev, and Gosselin [139] identified another type known as constraint singularity. In this discussion, we adhere to the general classification given in [132], as further detailed in [8], for categorizing singularities. These singularities occur when

**a.** Matrix $\boldsymbol{J}_q$ is rank deficient, or

**b.** Matrix $\boldsymbol{J}_x$ is rank deficient, or

**c.** The positioning equations degenerate.

As noted in [132], only the first type of singularity is possible for serial manipulators. The physical interpretations of these singularities are briefly described below.

### 5.4.1 Inverse Kinematic Singularity (Type I)

Referring to Equation 5.4, the inverse kinematic singularity occurs when the matrix $\boldsymbol{J}_q$ becomes rank deficient. Since for all parallel manipulators, $\boldsymbol{J}_q$ is an $m \times m$ square matrix; such a singularity occurs only when

$$\det\left(\boldsymbol{J}_q\right) = 0.$$

(5.11)

This occurs when the end-effector's configuration leads to a loss of control over one or more actuators, making it impossible to determine the necessary joint velocities for a given end-effector velocity. These configurations typically occur at the boundaries of the manipulator's workspace or on the internal boundaries between subregions of the workspace where the number of solutions of the inverse kinematic problem is not the same. In such scenarios, because the null space of $\boldsymbol{J}_q$ is non-empty, there exist non-zero vectors $\dot{\boldsymbol{q}}$ that correspond to a zero Cartesian twist rate vector $\dot{\boldsymbol{\chi}}$. Essentially, this means that infinitesimal movements of the moving platform along certain directions become impossible. Consequently, the manipulator loses one or more degrees-of-freedom when an inverse kinematic singularity occurs, leading to uncontrolled movements of the end-effector.

Furthermore, due to the kinematic-static duality [140], there exist wrenches that can be applied to the moving platform without necessitating actuator forces or torques. In other words, at an inverse kinematic singular configuration, the manipulator can resist forces or moments in specific directions with zero actuator effort. This characteristic parallels singularities observed in serial manipulators [1]. As an example, for a planar 3RRR

manipulator depicted in Figure 4.2, this might occur when the lines connecting the centers of the base and the moving platform become collinear.

### 5.4.2 Forward Kinematic Singularity (Type II)

Referring to Equation 5.4, forward kinematic singularity occurs when the matrix $\boldsymbol{J}_x$ becomes rank deficient. In the context of parallel manipulators, $\boldsymbol{J}_x$ is an $m \times n$ matrix where typically $m \geq n$. This situation arises when

$$\det\left(\boldsymbol{J}_x^T \boldsymbol{J}_x\right) = 0. \tag{5.12}$$

If the manipulator is not redundantly actuated, then $m = n$, and the Jacobian matrix $\boldsymbol{J}_x$ becomes a square matrix. Hence, in this case, forward kinematic singularity occurs when

$$\det\left(\boldsymbol{J}_x\right) = 0. \tag{5.13}$$

This occurs when the moving platform reaches a configuration where the mapping from joint velocities to moving platform velocities becomes ambiguous or non-unique. Unlike inverse kinematic singularities, forward kinematic singularities can occur within the manipulator's Cartesian workspace and correspond to configurations

where two different branches of the forward kinematic solution converge to a repeated solution. This causes multiple joint configurations to correspond to the same moving platform position and orientation, and making it difficult to determine the joint angles required to achieve a desired moving platform position. This type of singularity is unique to parallel manipulators because the forward kinematic problem in serial manipulators always yields a unique solution.

In such singularity configuration, consider Equations 5.4 and 5.12. Since the null space of the matrix $\boldsymbol{J}_x$ is not empty, there exist non-zero Cartesian twist rate $\dot{\boldsymbol{\chi}}$ that result in a zero actuator velocity vector $\dot{\boldsymbol{q}}$. In such configurations, infinitesimal motion of the end effector is possible even when the actuators are locked. Essentially, this means that the end effector gains additional degrees-of-freedom, causing the manipulator's stiffness to decrease or vanish in the corresponding direction(s). As an example, in a Stewart-Gough platform, this can happen when the platform becomes coplanar with the base, causing ambiguity in joint configuration.

### 5.4.3 Combined Kinematic Singularity (Type III)

This type of singularity differs slightly from the first two and occurs when both the inverse and direct kinematic singularities coincide, resulting in a complete loss of

control and rigidity of the manipulator. A combined singularity occurs when both $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ become rank deficient simultaneously. It signifies a degeneracy in the position or orientation equations of the manipulator. Specific conditions on the geometric parameters of the robot must be met for this type of singularity to manifest. Only certain special architectures exhibit this type of singularity, which justifies the term “architecture singularity” proposed in [135].

In practice, the geometric conditions under which manipulators experience combined singularities are detailed in [132] and are generally avoidable with careful design. Such singularities result in configurations where the end effector can undergo finite motion even when the actuators are immobilized. Alternatively, they can lead to scenarios where the motion of the actuators does not translate into the motion of the end effector. In both cases, the manipulator loses its complete controllability and rigidity.

### 5.4.4 Singularity Avoidance

To avoid singularities in parallel manipulators, several effective methods can be employed. Firstly, careful design and optimization of the manipulator's geometry and workspace are crucial, ensuring that the operational range avoids configurations prone to singularities. Advanced control algorithms can be implemented to detect and steer clear of singular configurations in real-time, enhancing the

manipulator's robustness. Additionally, redundancy in the actuation system, such as adding extra actuators, can provide alternative control paths, allowing the manipulator to maintain stable operation even near singularities. Regular monitoring and updating of the manipulator's parameters can also help in predicting and mitigating the risk of encountering singularities, ensuring smooth and reliable performance.

Deriving Jacobian matrices in parallel manipulators and performing singularity analysis are highly dependent on the type and structure of the manipulator. The process involves calculating the Jacobian matrices by eliminating the effects of unactuated joint rates within each limb, relating the active joint velocities of the limbs to the moving platform velocities, and capturing the manipulator's differential kinematic behavior. The complexity of this task varies with different manipulator architectures and the intricacy of the limbs, each presenting unique challenges in the formulation of the Jacobian. To comprehensively understand the underlying concepts and methods, it is beneficial to conduct Jacobian derivation and singularity analysis through diverse case studies. This approach allows for the exploration of various configurations and their associated singularities, providing insights into the practical implications of Jacobian singularities and helping to develop tailored strategies for different manipulator designs. In what follows, three different case studies are

analyzed to identify critical patterns and devise robust solutions to effectively mitigate singularity issues.

## 5.5 Case Study I: Planar Cable Robot

Jacobian analysis is crucial in the study of robotic manipulators. The Jacobian matrix not only elucidates the relationship between active joint variable velocities $\dot{\boldsymbol{q}}$ and the velocities of the moving platform $\dot{\boldsymbol{x}}$, but also facilitates the transformation necessary to derive actuator forces $\boldsymbol{\tau}$ from the wrench applied on the moving platform $\boldsymbol{\mathcal{F}}$.

In this section, we conduct Jacobian analysis specifically for the planar 4RPR manipulator. Subsequently, the manipulator's Jacobian is utilized for singularity and sensitivity analyses in the upcoming sections. These analyses provide insights into critical aspects of the manipulator's behavior and performance.

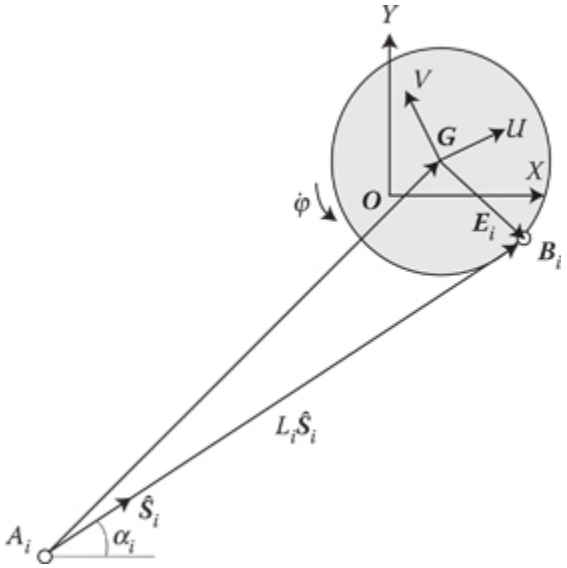
### 5.5.1 Velocity Loop Closure

The architecture of the planar 4RPR parallel cable robot discussed here is detailed in Section 4.4.1 and depicted in Figure 4.4. In this manipulator, the moving platform is supported by four cable-driven limbs, each following an identical kinematic structure. Each limb connects the fixed base to the moving platform through a sequence of joints: a revolute joint (R), an actuated prismatic joint (P) that

represents the elongation/contraction of the cables, and another revolute joint (R). The prismatic joint models can either represent piston-cylinder actuators as in regular manipulators or cable-driven mechanisms in this specific case. To prevent singularities at the central position of the manipulator at each operational level, the cables are configured in a crossed arrangement. A comprehensive analysis of singularities within the mechanism is presented in Section 5.5.2.

As depicted in Figure 4.4, $A_i$ denotes the fixed base points of the limbs, $B_i$ denotes the connection points of the limbs on the moving platform, $L_i$ denotes the lengths of the limbs, and $\alpha_i$ denotes the angles of the limbs. The position of the center of the moving platform $G$ is described by $G = [x_G, y_G]$, and the orientation of the manipulator's moving platform is denoted by $\phi$ with respect to the fixed coordinate frame. Therefore, the manipulator possesses three degrees-of-freedom represented by the task space variable $\boldsymbol{\chi} = [x_G, y_G, \phi]$, along with one degree of actuation redundancy.

For analytical purposes, as illustrated in Figure 5.2, a fixed frame $O$ is affixed to the fixed base at point $O$, which serves as the center of the base point circle passing through $A_i$ points. Additionally, a moving coordinate frame $G$ is attached to the moving platform at point $G$. Detailed geometric parameters of the manipulator are provided in Section 4.4.2.




▶ Long Description for Figure 5.2

**FIGURE 5.2**
Definition of vectors used in deriving the Jacobian for the planar 4RPR manipulator. ⏎

Unlike serial manipulators, the Jacobian matrix of a parallel manipulator is defined as the transformation matrix that relates the moving platform twist rate to the joint variable velocities:

$$\dot{\boldsymbol{q}} = \boldsymbol{J} \cdot \dot{\boldsymbol{\chi}}, \tag{5.14}$$

where, for the planar 4RPR manipulator, $\dot{\boldsymbol{q}} = [\dot{L}_1, \dot{L}_2, \dot{L}_3, \dot{L}_4]$ represents the $4 \times 1$ limb active velocity vector, and $\dot{\boldsymbol{\chi}} = [\dot{x}_G, \dot{y}_G, \dot{\phi}]$ denotes the $3 \times 1$ moving platform twist rate. Due to the manipulator's redundant

actuation, the Jacobian matrix $\boldsymbol{J}$ is a non-square $4 \times 3$ matrix.

From the geometric configuration of the manipulator, as depicted in Figure 4.5, the loop closure Equation 4.5 for each limb is reformulated here for detailed analysis:

$$A_i G = A_i B_i + B_i G. \tag{5.15}$$

Considering the vector definitions $\hat{\boldsymbol{S}}_i$ and $\boldsymbol{E}_i$ depicted in Figure 5.2, the vector loop closure simplifies to the following equation

$$\boldsymbol{G} = L_i \hat{\boldsymbol{S}}_i - \boldsymbol{E}_i \quad \text{or} \quad \boldsymbol{G} + \boldsymbol{E}_i = L_i \hat{\boldsymbol{S}}_i, \tag{5.16}$$

where $\boldsymbol{G}$ represents the position vector of point $G$.

To derive the Jacobian matrix, let us differentiate the vector loop Equation 5.16 with respect to time. The time derivative of the position vector $\boldsymbol{G}$ is denoted by the velocity vector $\boldsymbol{v}_G$. The time derivative of vector $\boldsymbol{E}_i$ is obtained using the relative differentiation formulation given in 3.51, as follows:

$$\left( \frac{d\boldsymbol{E}_i}{d} \right)_{fix} = \left( \frac{\partial \boldsymbol{E}_i}{\partial t} \right)_{mov} + \boldsymbol{\Omega} \times \boldsymbol{E}_i,$$

(5.17)

Since the time derivative of vector $\boldsymbol{E}_i$ with respect to the moving frame is zero, and considering that the angular velocity of the moving frame is $\boldsymbol{\Omega} = \dot{\phi}\hat{\boldsymbol{K}}$, therefore

$$\dot{\boldsymbol{E}}_i = \dot{\phi}(\hat{\boldsymbol{K}} \times \boldsymbol{E}_i).$$

(5.18)

Furthermore, the time derivative of the last term $L_i\hat{\boldsymbol{S}}_i$ is given by the following equation:

$$\frac{d}{dt}(L_i\hat{\boldsymbol{S}}_i) = \dot{L}_i\hat{\boldsymbol{S}}_i + L_i\dot{\hat{\boldsymbol{S}}}_i,$$

(5.19)

In this equation, the time derivative of the second term $L_i\dot{\hat{\boldsymbol{S}}}_i$ is determined using the relative differentiation rule similar to that for $\dot{\boldsymbol{E}}_i$, taking into account that the angular velocity of this vector is $\dot{\alpha}\boldsymbol{K}$. Hence,

$$\frac{d}{dt}(L_i\hat{\boldsymbol{S}}_i) = \dot{L}_i\hat{\boldsymbol{S}}_i + \dot{\alpha}_i L_i(\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i).$$

(5.20)

Thus, for each limb $i = 1, 2, 3, 4$, the velocity loop closure yields

$$\boldsymbol{v_G} + \dot{\phi}(\hat{\boldsymbol{K}} \times \boldsymbol{E_i}) = \dot{L}_i \hat{\boldsymbol{S}}_i + \dot{\alpha}_i L_i (\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i),$$

(5.21)

In this equation, $\boldsymbol{v_G} = [\dot{x}_G, \dot{y}_G]^T$ represents the velocity of the moving platform at point $G$, and $\hat{\boldsymbol{K}}$ is the unit vector in the $z$ direction of the fixed coordinate frame $\{\mathscr{A}\}$.

To eliminate the passive joint variable $\dot{\alpha}_i$, dot multiply both sides of Equation 5.21 by $\hat{\boldsymbol{S}}_i$.

$$\hat{\boldsymbol{S}}_i \cdot \boldsymbol{v}_G + \hat{\boldsymbol{S}}_i \cdot (\hat{\boldsymbol{K}} \times \boldsymbol{E}_i)\, \dot{\phi} = \dot{L}_i \hat{\boldsymbol{S}}_i \cdot \hat{\boldsymbol{S}}_i + \dot{\alpha}_i\, L_i\, (\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i) \cdot \hat{\boldsymbol{S}}_i.$$

(5.22)

Note that $\hat{\boldsymbol{S}}_i$ is a unit vector, hence $\dot{L}_i \hat{\boldsymbol{S}}_i \cdot \hat{\boldsymbol{S}}_i = \dot{L}_i$. Furthermore, the vector $\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i$ is orthogonal to $\hat{\boldsymbol{S}}_i$, thus $(\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i) \cdot \hat{\boldsymbol{S}}_i = 0$. Using the rules of dot and cross multiplication interchange,

$$\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{b} \cdot (\boldsymbol{c} \times \boldsymbol{a}).$$

(5.23)

Equation 5.22 simplifies to

$$\hat{\boldsymbol{S}}_i \cdot \boldsymbol{v_G} + \hat{\boldsymbol{K}} \cdot (\boldsymbol{E_i} \times \hat{\boldsymbol{S}}_i)\, \dot{\phi} = \dot{L}_i.$$

(5.24)

Rewriting Equation 5.24 in matrix form yields

$$\dot{L}_i = \begin{bmatrix} S_{i_x} \mid S_{i_y} \mid E_{i_x} S_{i_y} - E_{i_y} S_{i_x} \end{bmatrix} \cdot \begin{matrix} v_{G_x} \\ v_{G_y} \\ \dot{\phi} \end{matrix} \quad . \tag{5.25}$$

Using Equation 5.25 for $i = 1, 2, 3, 4,$ the manipulator Jacobian matrix $\boldsymbol{J}$ is obtained.

$$\boldsymbol{J} = \left[\begin{array}{c|c|c} S_{1_x} & S_{1_y} & E_{1_x} S_{1_y} - E_{1_y} S_{1_x} \\ S_{2_x} & S_{2_y} & E_{2_x} S_{2_y} - E_{2_y} S_{2_x} \\ S_{3_x} & S_{3_y} & E_{3_x} S_{3_y} - E_{3_y} S_{3_x} \\ S_{4_x} & S_{4_y} & E_{4_x} S_{4_y} - E_{4_y} S_{4_x} \end{array}\right] . \tag{5.26}$$

Note that the Jacobian matrix $\boldsymbol{J}$ is a nonsquare $4 \times 3$ matrix because the manipulator is redundantly actuated.

Furthermore, to solve for the passive limb velocities $\dot{\alpha}_i$, cross-multiply both sides of Equation 5.21 by $\hat{\boldsymbol{S}}_i$:

$$\hat{\boldsymbol{S}}_i \times \boldsymbol{v}_G + \dot{\phi} \left( \boldsymbol{E_i} \cdot \hat{\boldsymbol{S}}_i \right) \hat{\boldsymbol{K}} = \dot{\alpha}_i \, L_i \, \hat{\boldsymbol{K}}. \tag{5.27}$$

Rewriting Equation 5.27 in matrix form results in

$$\dot{\alpha}_i = \frac{1}{L_i} \cdot \left[ -S_{i_y} \mid S_{i_x} \mid E_{i_x} S_{i_x} + E_{i_y} S_{i_y} \right] \cdot \begin{matrix} v_{Gx} \\ v_{Gy} \\ \dot{\phi} \end{matrix} .$$

(5.28)

Denote $\boldsymbol{J}_\alpha$, as the matrix relating the vector of the moving platform velocities, $\dot{\boldsymbol{\chi}} = [\dot{x}_G, \dot{y}_G, \dot{\phi}]$, to the vector of angular velocities of the limbs $\dot{\boldsymbol{\alpha}} = [\dot{\alpha}_1, \dot{\alpha}_2, \dot{\alpha}_3, \dot{\alpha}_4]$, expressed as

$$\dot{\boldsymbol{\alpha}} = \boldsymbol{J}_\alpha \cdot \dot{\boldsymbol{\chi}}.$$

(5.29)

This matrix could be represented by

$$\boldsymbol{J}_\alpha = \frac{1}{L_i} \cdot \left| \begin{matrix} -S_{1_y} & S_{1_x} & E_{1_x} S_{1_x} + E_{1_y} S_{1_y} \\ -S_{2_y} & S_{2_x} & E_{2_x} S_{2_x} + E_{2_y} S_{2_y} \\ -S_{3_y} & S_{3_x} & E_{3_x} S_{3_x} + E_{3_y} S_{3_y} \\ -S_{4_y} & S_{4_x} & E_{4_x} S_{4_x} + E_{4_y} S_{4_y} \end{matrix} \right| .$$

(5.30)

### 5.5.2 Singularity Analysis

This section elaborates on the singularity analysis of the planar 4R$\underline{\text{P}}$R parallel manipulator. Jacobian analysis in parallel manipulators is typically more intricate compared

to serial ones. A notable drawback of parallel manipulators is the existence of singular configurations within their workspace, where the manipulator can gain extra degrees-of-freedom, thereby compromising its stiffness. As outlined in [132], the study of singularities in closed-loop mechanisms involves the examination of two distinct Jacobian matrices.

To analyze singularities in the specific manipulator under consideration, we refer to the Jacobian matrix defined in Equation 5.26. Comparing this with the general Jacobian formulation in Equation 5.6 and Equation 5.5, it becomes evident that for the planar 4RPR manipulator, $\boldsymbol{J}_q = \boldsymbol{I}$ (identity matrix) and $\boldsymbol{J}_x = \boldsymbol{J}$. Therefore, the manipulator does not exhibit inverse kinematic singularities, but it may encounter forward kinematic singularities when $\boldsymbol{J}$ loses rank. Notably, due to its redundant actuation and the dimensions of $\boldsymbol{J}$ being $4 \times 3$, the manipulator enters a singular configuration only when

$$\det\left(\boldsymbol{J}^T \boldsymbol{J}\right) = 0. \tag{5.31}$$

Singular configurations in the planar 4RPR parallel manipulator can be identified either by evaluating Equation 5.31 across grid points spanning the entire workspace or by geometric analysis of the Jacobian matrix properties. In the latter approach, considering $\boldsymbol{J}$ composed of columns $\hat{S}_x$, $\hat{S}_y$, and $\boldsymbol{E} \times \hat{\boldsymbol{S}}$, singularity occurs if any column becomes

zero or if two columns become linearly dependent. From the geometric perspective of vectors $\boldsymbol{E}$ and $\hat{\boldsymbol{S}}$ illustrated in Figure 5.2, $\boldsymbol{E} \times \hat{\boldsymbol{S}} = \boldsymbol{0}$ only when the vectors align, typically happening at the central position $(x_G = y_G = 0)$ and when the moving platform orientation angle $\phi$ is $\pm 90^\circ$.

As the moving platform deviates from the central position, the geometric interpretation becomes more complex. Therefore, comprehensive singular configurations for the 4RPR manipulator are determined by evaluating Equation 5.31 at numerous grid points within the manipulator's workspace. This evaluation was conducted across 1089 grid points spanning $-50 \leq x_G \leq 50$, $-50 \leq y_G \leq 50$, and $-\pi \leq \phi \leq \pi$, resulting in the identification of singular configurations displayed in the three-dimensional plot shown in Figure 5.3. This figure confirms and extends the geometric understanding of singular configurations to include points in two planes where

$$x_G = \pm y_G \quad \text{and} \quad \phi = \pm 90^\circ. \tag{5.32}$$

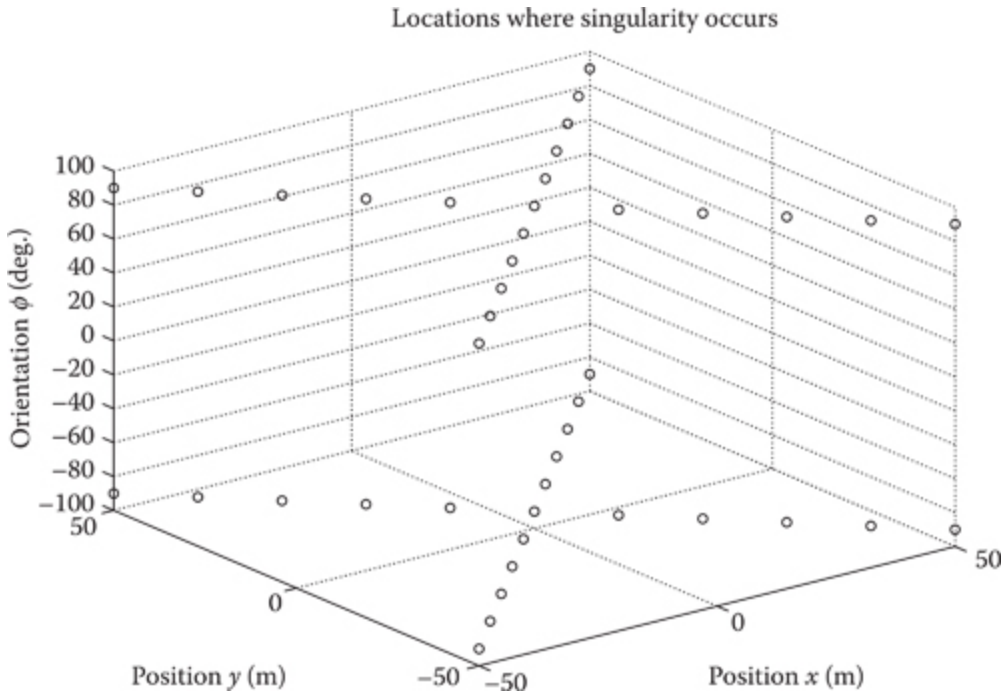



**FIGURE 5.3**
Singular configurations of the planar 4RPR manipulator. ⏎

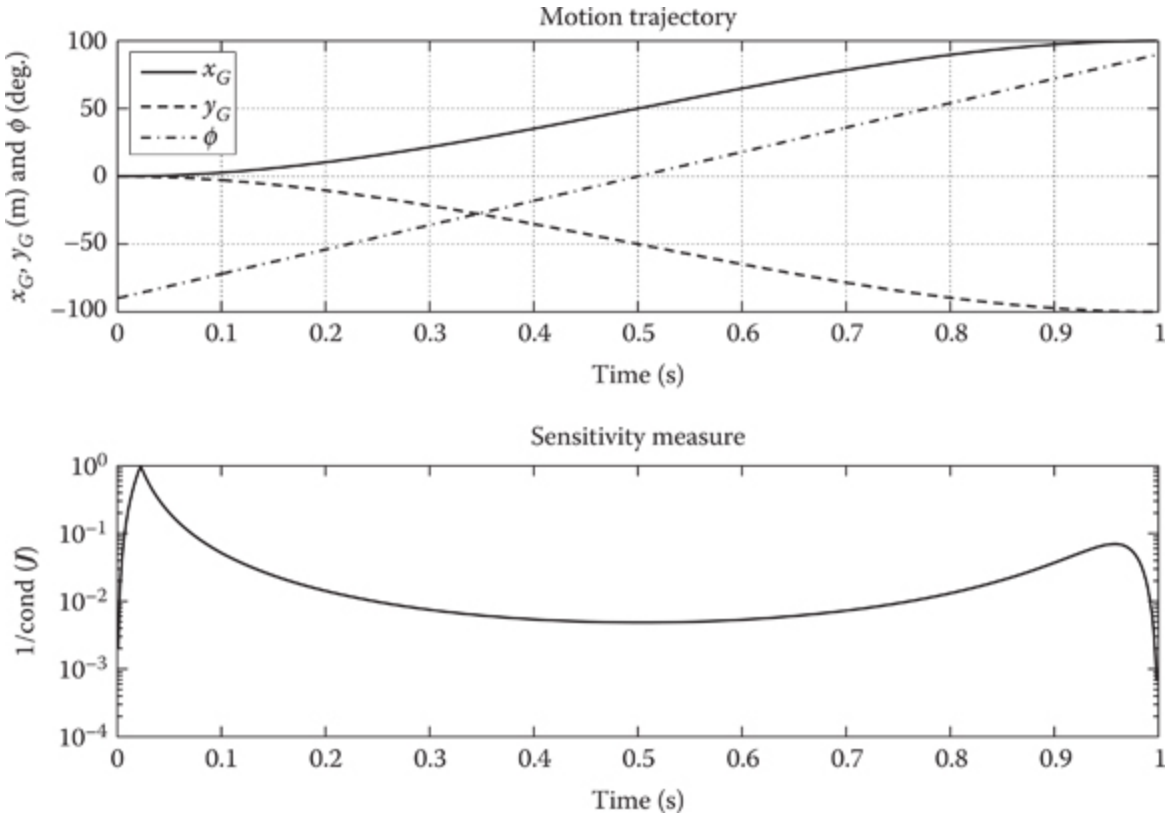
In the design of the manipulator, a cable-crossed configuration is deliberately chosen. This design decision aims to prevent a singular configuration from occurring at the central position and orientation where $x_G = y_G = \phi = 0$. With this configuration, the manipulator remains nonsingular across its entire workspace, as long as the required orientation workspace is limited to within $\pm 90^\circ$.

### 5.5.3 Sensitivity Analysis

As explored in Section 5.5.2, singular points in the manipulator primarily occur when $\phi = \pm 90^\circ$. This section examines the Jacobian matrix's conditioning near these singularities. A widely adopted metric for evaluating

Jacobian matrix conditioning is the “manipulability measure,” defined as the inverse of the Jacobian matrix's condition number [1, Section 7.3.4]. For Jacobians of redundant manipulators, which are nonsquare matrices, this measure indicates the ratio of the smallest singular value to the largest. Near singular configurations, the smallest singular value approaches zero, indicating low sensitivity. Conversely, sensitivity values close to one suggest configurations where singular values are similar, indicating near-isotropic manipulator behavior.

Figure 5.4 depicts the manipulability measure of the robot Jacobian matrix over a typical trajectory. The top figure shows the trajectory of the moving platform, with horizontal and vertical motions of approximately 100 meters and an orientation change of about 180°. Initially, the orientation is near the $-90^\circ$ singular configuration, transitioning to the next singular configuration near $+90^\circ$ at the end. The bottom figure presents a logarithmic scale plot of the manipulability measure for the manipulator Jacobian $\boldsymbol{J}$ along this trajectory. It illustrates that this sensitivity measure is low at configurations close to singular points, indicated by the initial and final positions. However, as the trajectory moves away from these singular configurations, the Jacobian matrix becomes increasingly well-conditioned. The logarithmic scale highlights that for configurations not in close proximity to singular points, the manipulator's Jacobian matrix is well-conditioned and suitable for numerical computations.




▶ Long Description for Figure 5.4

**FIGURE 5.4**

Sensitivity of the planar $4RPR$ manipulator Jacobian for a typical trajectory. ⏎

## 5.6 Case Study II: Shoulder Manipulator

In this section, we conduct a detailed Jacobian analysis of the three degrees of freedom spatial orientation shoulder manipulator. This analysis involves two primary steps. First, we determine the manipulator's Jacobian matrices, which describe the relationship between the active joint

velocities and the moving platform velocities. Next, we perform a thorough examination of the manipulator's singularities. This includes identifying configurations where the manipulator may lose its stiffness or gain unexpected degrees of freedom, which can critically affect its performance and control. By understanding these aspects, we can better predict and mitigate potential issues in the manipulator's operation.

### 5.6.1 Velocity Loop Closure

As illustrated in Figure 4.11, the shoulder manipulator's moving platform is designed to exhibit three degrees of freedom in purely orientation motion. To achieve this, the system utilizes four high-performance hydraulic piston actuators, which provide the necessary degrees of freedom while incorporating one degree of actuator redundancy for enhanced performance and reliability. The manipulator employs simple revolute joints to replace the required spherical and universal joints, maintaining a uniform kinematic structure across all limbs.

Each of the four active limbs follows a UPU kinematic configuration, where each pair of limbs is connected to the moving platform through a single linking element. Figure 4.12 presents a geometric model of this mechanism, offering a visual representation of its design. A comprehensive geometrical description of the manipulator can be found in Section 4.5.2, detailing the specific parameters and configurations that define its operation.

In this section, we derive the Jacobian matrices for the shoulder manipulator, which serve as fundamental components for the subsequent singularity and stiffness analyses. The Jacobian matrix of the shoulder manipulator establishes the relationship between the angular velocity of the moving platform, denoted as $\boldsymbol{\omega}$, and the vector of the actuated joint rates, represented as $\dot{\boldsymbol{q}} = [\dot{l}_1, \dot{l}_2, \dot{l}_3, \dot{l}_4]^T$, by the following equation:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}\boldsymbol{\omega}.$$

(5.33)

From the above definition, it is readily observed that the Jacobian of this manipulator is a $4 \times 3$ matrix, reflecting the redundant nature of the mechanism. By mapping between joint and task space variables, it is evident that the Jacobian matrix depends on both the actuated limbs and the passive supporting limb. Therefore, we first derive a $4 \times 6$ Jacobian, $\boldsymbol{J}_v$, which relates the six-dimensional (6D) twist rate of the moving platform, $\dot{\boldsymbol{\chi}}$, to the vector of actuated limbs' rates, $\dot{\boldsymbol{q}}$. Next, we derive the $6 \times 3$ Jacobian of the passive supporting limb, $\boldsymbol{J}_p$. The Jacobian of the shoulder manipulator will then be derived as follows:

$$\boldsymbol{J} = \boldsymbol{J}_v\boldsymbol{J}_p.$$

(5.34)

## Jacobian of the Actuated Limbs

The Jacobian of the actuated limbs, $\boldsymbol{J}_v$, relates the 6D twist rate of the moving platform, $\dot{\boldsymbol{\chi}}$, to the vector of actuated limb rates, $\dot{\boldsymbol{q}}$, such that

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_v \dot{\boldsymbol{\chi}},$$

(5.35)

in which the moving platform twist rate can be written as

$$\dot{\boldsymbol{\chi}} = \begin{bmatrix} {}^A\boldsymbol{v}_p \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} v_{p_x} & v_{p_y} & v_{p_z} & \omega_x & \omega_y & \omega_z \end{bmatrix}^T.$$

(5.36)

In Equation 5.36, ${}^A\boldsymbol{v}_p$ represents the velocity of the moving platform's center point, and $\boldsymbol{\omega}$ denotes the angular velocity of the moving platform. For further analysis, let us rewrite the kinematic loop closure for each actuated limb, given by Equation 4.26.

$$\boldsymbol{q}_i = l_i \, \hat{\boldsymbol{s}}_i = {}^A\boldsymbol{P} + {}^A\boldsymbol{R}_B \, {}^B\boldsymbol{P}_i - \boldsymbol{a}_i,$$

(5.37)

where $l_i$ is the length of the $i$'th actuated limb, $\hat{\boldsymbol{s}}_i$ is the unit vector pointing along the direction of the $i$'th actuated limb,

${}^{A}\boldsymbol{R}_{B}$ is the rotation matrix of the moving platform, and ${}^{A}\boldsymbol{P}$ is its center position vector.

To derive the velocity loop closure, we need to differentiate the kinematic vector loop closure 5.37 with respect to time.

$$\dot{l}_i \hat{\boldsymbol{s}}_i + (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)\, l_i = {}^{A}\boldsymbol{v}_p + \boldsymbol{\omega} \times {}^{A}\boldsymbol{R}_{B}\, {}^{B}\boldsymbol{P}_i,$$

(5.38)

where $\boldsymbol{\omega}_i$ represents the angular velocity of the $i$'th limb expressed in the base frame.

To eliminate the passive limb velocity variables, multiply Equation 5.38 by $\hat{\boldsymbol{s}}_i$.

$$\dot{l}_i = \hat{\boldsymbol{s}}_i^{T}\, {}^{A}\boldsymbol{v}_p + \left[ ({}^{A}\boldsymbol{R}_{B}\, {}^{B}\boldsymbol{P}_i) \times \hat{\boldsymbol{s}}_i \right]^{T} \boldsymbol{\omega}.$$

(5.39)

The repetition of the aforementioned equation for each actuated limb in vector form, as depicted in Equation 5.35, yields the Jacobian matrix $\boldsymbol{J}_v$ for the actuated limbs.

$$\boldsymbol{J}_v = \begin{matrix} {}^{A}\hat{\boldsymbol{s}}_1{}^{T} & ({}^{A}\boldsymbol{P}_1 \times {}^{A}\hat{\boldsymbol{s}}_1)^{T} \\ {}^{A}\hat{\boldsymbol{s}}_2^{T} & ({}^{A}\boldsymbol{P}_2 \times {}^{A}\hat{\boldsymbol{s}}_2)^{T} \\ {}^{A}\hat{\boldsymbol{s}}_3^{T} & ({}^{A}\boldsymbol{P}_3 \times {}^{A}\hat{\boldsymbol{s}}_3)^{T} \\ {}^{A}\hat{\boldsymbol{s}}_4^{T} & ({}^{A}\boldsymbol{P}_4 \times {}^{A}\hat{\boldsymbol{s}}_4)^{T} \end{matrix}_{4\times 6}.$$

(5.40)

Note that $\boldsymbol{P}_1 = \boldsymbol{P}_2$ represents the connecting point of limbs 1 and 2 to the moving platform, while $\boldsymbol{P}_3 = \boldsymbol{P}_4$ denotes the connecting point of limbs 3 and 4.

## Jacobian of the Passive Limb

To determine the manipulator Jacobian, we must establish a connection between the 6D twist rate of the moving platform, $\dot{\boldsymbol{\chi}}$, and the angular velocity of the moving platform, $\boldsymbol{\omega}$. This relationship is specified by the Jacobian of the passive limb $\boldsymbol{J}_p$ as

$$\dot{\boldsymbol{\chi}} = \begin{bmatrix} {}^{A}\boldsymbol{v}_p \\ \boldsymbol{\omega} \end{bmatrix} = \boldsymbol{J}_p \boldsymbol{\omega}.$$

(5.41)

Let us initially restate the relation of the position vector ${}^{A}\boldsymbol{P}$ with respect to *u*–*v*–*w* Euler angles of Equation 4.27 to derive ${}^{A}\boldsymbol{v}_p$.

$${}^{A}\boldsymbol{P} = l_p \begin{bmatrix} s_2 & -c_2 s_1 & c_2 c_1 \end{bmatrix}^T.$$

(5.42)

Differentiate this equation with respect to time:

$$
{}^{A}\boldsymbol{v}_p = \begin{matrix} 0 & c_2 & 0 \\ -c_2 c_1 & s_2 s_1 & 0 \\ -c_2 s_1 & -s_2 c_1 & 0 \end{matrix} l_p \dot{\boldsymbol{\theta}},
\tag{5.43}
$$

where $\dot{\boldsymbol{\theta}} = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$ represents the vector of Euler angle rates. The angular velocity of the moving platform can also be described in terms of Euler angle rates using Equation 3.39, where the rotation matrix ${}^{A}\boldsymbol{R}_B$ is defined by Equation 4.25.

$$
\boldsymbol{\omega} = \begin{matrix} 1 & 0 & s_2 \\ 0 & c_1 & -c_2 s_1 \\ 0 & s_1 & c_2 c_1 \end{matrix} \dot{\boldsymbol{\theta}}
\tag{5.44}
$$

Solve Equation 5.44 for $\dot{\boldsymbol{\theta}}$:

$$
\dot{\boldsymbol{\theta}} = \frac{1}{c_2} \begin{matrix} c_2 & s_2 s_1 & -s_2 c_1 \\ 0 & c_2 c_1 & c_2 s_1 \\ 0 & -s_1 & c_1 \end{matrix} \boldsymbol{\omega},
\tag{5.45}
$$

By substituting Equation 5.45 into Equation 5.43 and subsequently performing some simplification, we reach to:

$$
{}^{A}\boldsymbol{v}_p = \begin{matrix} 0 & c_2c_1 & c_2s_1 \\ -c_2c_1 & 0 & s_2 \\ -c_2s_1 & -s_2 & 0 \end{matrix} l_p\boldsymbol{\omega}.
$$

(5.46)

Finally, the Jacobian matrix of the passive limb is derived by augmenting the identity map, $\boldsymbol{\omega} = \boldsymbol{I}_{3\times3}\boldsymbol{\omega}$, with Equation 5.46, as referenced in Equation 5.41.

$$
\boldsymbol{J}_p = \begin{matrix} 0 & l_pc_2c_1 & l_pc_2s_1 \\ -l_pc_2c_1 & 0 & l_ps_2 \\ -l_pc_2s_1 & -l_ps_2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}
$$

(5.47)

Having $\boldsymbol{J}_v$ and $\boldsymbol{J}_p$ available, the shoulder manipulator Jacobian $\boldsymbol{J}_{4\times3}$ can be easily determined using Equation 5.34.

$$
\boldsymbol{J}_{4\times3} = \boldsymbol{J}_v\boldsymbol{J}_p = \begin{matrix} \left({}^{A}P_{i_z} - l_pc_2c_1\right){}^{A}\boldsymbol{s}_{i_y} - \left({}^{A}P_{i_y} + l_pc_2s_1\right){}^{A}\boldsymbol{s}_{i_z} \\ \left({}^{A}P_{i_x} - l_ps_2\right){}^{A}\boldsymbol{s}_{i_z} - \left({}^{A}P_{i_z} - l_pc_2c_1\right){}^{A}\boldsymbol{s}_{i_x} \\ \left({}^{A}P_{i_y} + l_pc_2s_1\right){}^{A}\boldsymbol{s}_{i_x} - \left({}^{A}P_{i_x} - l_ps_2\right){}^{A}\boldsymbol{s}_{i_y} \end{matrix}^{T} \quad i=
$$

(5.48)

### 5.6.2 Singularity Analysis

In this section, we elaborate on the singularity analysis of the shoulder manipulator. To examine potential singular configurations, we start by examining its Jacobian matrix $\boldsymbol{J}$, as defined in Equation 5.48. Comparing this with the general Jacobian definition in Equation 5.6 and Equation 5.33, it becomes evident that for the shoulder manipulator, $\boldsymbol{J}_q = \boldsymbol{I}$ and $\boldsymbol{J}_x = \boldsymbol{J}$. This indicates that the manipulator does not exhibit inverse kinematic singularities, although it may encounter forward kinematic singularities when $\boldsymbol{J}$ loses rank. It's worth noting that due to redundant actuation and the $4 \times 3$ dimensionality of $\boldsymbol{J}$, singular configurations occur when

$$\det\left(\boldsymbol{J}^T \boldsymbol{J}\right) = 0.$$

(5.49)

To investigate singular configurations of the shoulder manipulator, two approaches are employed. Firstly, Equation 5.49 is directly evaluated, and alternatively, determinants of all $3 \times 3$ minors of $\boldsymbol{J}$ are examined. A thorough analysis of Equation 5.49 across 9261 grid points within the ranges $-\pi/3 \leq \theta_1, \theta_2, \theta_3 \leq \pi/3$, indicates that no singular configurations exist throughout the entire workspace of the shoulder manipulator. This discovery

aligns with the original assertions made by the manipulator's designer [88].

Furthermore, to better understand how redundant actuators contribute to a singularity-free workspace, the analysis extends to evaluating determinants of all $3 \times 3$ minors of $\boldsymbol{J}$. It is widely recognized that for a redundant parallel manipulator, singularities typically occur when the Jacobian's rank is less than $n$, the number of degrees of freedom of the moving platform, or equivalently

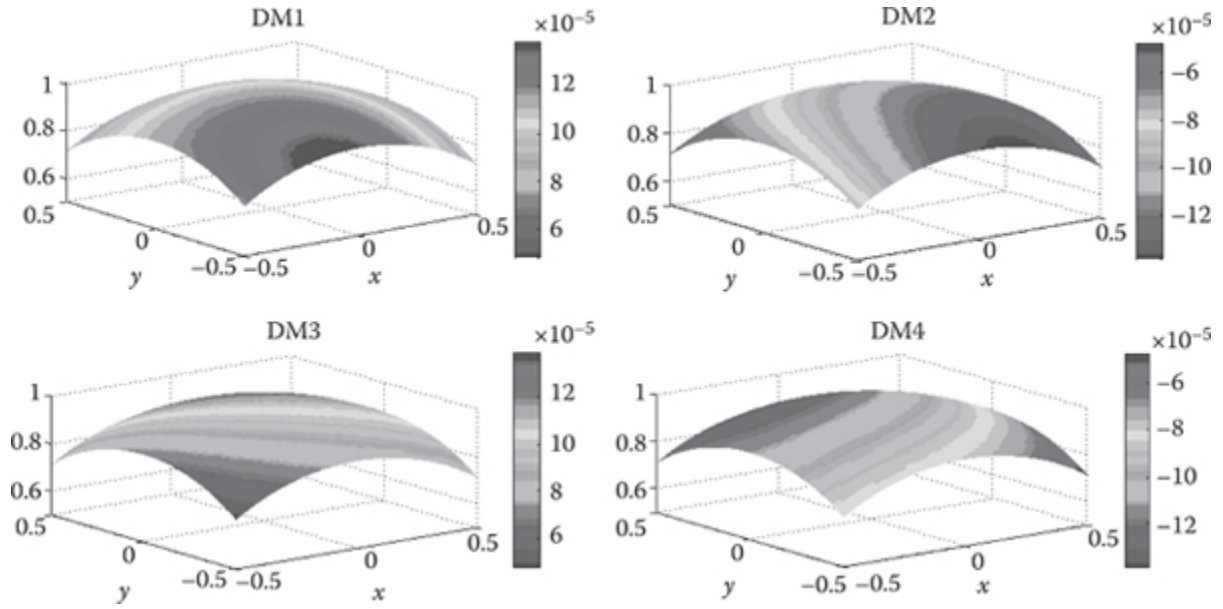
$$\det(\boldsymbol{M}) = 0, \quad \forall \boldsymbol{M} \in \{n \times n \text{ submatrices of } \boldsymbol{J}\}. \tag{5.50}$$

It's important to note that only $m - n + 1$ equations from Equation 5.50 are independent, where $m$ denotes the number of actuated joints. Therefore, singularities are located at the intersection of $m - n + 1$ hypersurfaces, forming a lower-dimensional manifold with a dimension of $n - (m - n + 1) = n - 1 - n_r$ in the task space, where $n_r$ represents the number of redundant actuators. This underscores how actuator redundancy can effectively mitigate or eliminate singularities within the workspace.

For the shoulder manipulator, as detailed in the preceding section, the linear velocities of the actuators $\dot{\boldsymbol{q}}$ are linked to the angular velocity of the moving platform $\boldsymbol{\omega}$ through Equation 5.33, where $\boldsymbol{J}$ is defined in Equation 5.48. Singularities manifest as rank deficiencies in $\boldsymbol{J}$, which occur if the determinants of all $3 \times 3$ minors of $\boldsymbol{J}$ are

uniformly zero. These minors correspond to the Jacobian matrices of the shoulder manipulator with one of the actuating limbs removed. Consequently, the redundant manipulator enters a singular configuration only if all configurations resulting from removing one actuating limb exhibit singularity—a scenario absent in the shoulder manipulator's workspace—as previously demonstrated. Indeed, a notable benefit of adding the fourth actuator is the elimination of singularities across the entire workspace.

Figure 5.5 illustrates the determinants of four minor Jacobian matrices computed across the manipulator's workspace, labeled as DM1 through DM4. Notably, DM1, DM2, DM3, and DM4 are derived by removing limbs 4, 3, 2, and 1, respectively, to evaluate the corresponding nonredundant configurations. As depicted in Figure 5.5, the values of these minor Jacobian matrices vary across the spherical surface of the manipulator's workspace. However, they do not collectively reach their minimum simultaneously at any configuration. For clarity, observe in Figure 5.5 that while DM2 reaches a minimum at certain configurations, DM1 and DM3 show intermediate values, and DM4 remains relatively high. Similar patterns are observed when other Jacobian minors attain their minima. Consequently, no configuration exists within the entire manipulator workspace where all minor Jacobian matrices simultaneously achieve their minimum values. The risk of entering a singular configuration significantly increases when one of the redundant actuators is disabled.




▶ Long Description for Figure 5.5

**FIGURE 5.5**

Minor determinants for the nonredundant structures for the shoulder manipulator; in computing DM1 through DM4, limbs 4, 3, 2, and 1 were removed, respectively. ⏎

It's important to note that while this analysis pertains to a specific parallel manipulator, minor Jacobian matrices can similarly be employed to analyze singularity in other parallel manipulators. In cases lacking actuation redundancy, a comparable analysis can identify singular configurations. Moreover, in scenarios where a malfunctioning actuator impacts manipulator singularity, a detailed examination of its effect can be undertaken.

## 5.7 Case Study III: Stewart–Gough Platform

This section focuses on a detailed Jacobian analysis of the Stewart-Gough platform. The analysis begins with the determination of manipulator Jacobian matrices, followed by a comprehensive singularity analysis across the entire workspace of the manipulator.

### 5.7.1 Velocity Loop Closure

As shown in Figure 4.16, the spatial motion of the Stewart-Gough moving platform is facilitated by six identical piston-cylinder actuators. Each actuator comprises two parts connected via a prismatic joint, linking the fixed base to the moving platform through spherical joints located at points $A_i$ and $B_i$, where $i = 1, 2, \ldots, 6$. Notably, in Figure 4.16, the attachment points $A_i$ lie in the base plane, while $B_i$ lie in the moving platform plane. However, in a general Stewart-Gough platform, these attachment points may not necessarily lie in a single plane.

To dive into the kinematics of the manipulator, as depicted in Figure 4.16, a fixed frame $\{\mathscr{A}\}$ is affixed to the fixed base at point $O_A$, with attachment points on the fixed base denoted by vectors $\boldsymbol{a}_i$. Similarly, a moving frame $\{\mathscr{B}\}$ is attached to the moving platform at point $O_B$, with attachment points on the moving platform designated by vectors $\boldsymbol{b}_i$. Each limb's geometry is defined by its length $\ell_i$ and direction by a unit vector $\hat{\boldsymbol{s}}_i$. The position of point $O_B$ on the moving platform is described by the position vector

${}^A\boldsymbol{P} = [p_x \quad p_y \quad p_z]^T$, while the orientation of the moving platform is represented by the rotation matrix ${}^A\boldsymbol{R}_B$.

The kinematic structure of each limb follows an $SPS$ arrangement. As detailed in Example 1.3 in Chapter 1, this manipulator boasts six degrees of freedom, making it a versatile parallel manipulator capable of generating complete spatial motion. Notably, the number of actuators equals the number of degrees of freedom, indicating no actuator redundancy in this setup.

From the geometric configuration illustrated in Figure 4.16, the input joint rates are represented by $\dot{\boldsymbol{q}} = [\dot{\ell}_1, \dot{\ell}_2, \dot{\ell}_3, \dot{\ell}_4, \dot{\ell}_5, \dot{\ell}_6]^T$, while the output twist rate encompasses the velocity of point $P$ and the angular velocity of the moving platform, denoted as $\dot{\boldsymbol{\chi}} = [{}^A\boldsymbol{v}_p, {}^A\boldsymbol{\omega}]^T$. The Jacobian matrix is formulated by establishing a velocity loop closure equation for each limb. These loop closure equations, derived for each limb $i = 1, 2, \ldots, 6$, are initially presented in Equation 4.51 and are reiterated here for further analysis.

$$ {}^A\boldsymbol{P} + {}^A\boldsymbol{R}_B\, {}^B\boldsymbol{b}_i = \ell_i\, {}^A\hat{\boldsymbol{s}}_i + {}^A\boldsymbol{a}_i. \tag{5.51}$$

Differentiate Equation 5.51 with respect to time

$$ {}^A\boldsymbol{v}_p + \dot{.}{}^A\boldsymbol{R}_B\, {}^B\boldsymbol{b}_i + {}^A\boldsymbol{R}_B\, {}^B\dot{\boldsymbol{b}}_i = \dot{\ell}_i\, {}^A\hat{\boldsymbol{s}}_i + \ell_i\, {}^A\dot{\hat{\boldsymbol{s}}}_i + {}^A\dot{\boldsymbol{a}}_i. $$

(5.52)

Note that the time derivative of vector $\boldsymbol{b}_i$ with respect to the moving frame $\{\mathscr{B}\}$ is zero, ${}^{B}\dot{\boldsymbol{b}}_i = 0$, and similarly, ${}^{A}\dot{\boldsymbol{a}}_i = 0$. Additionally,

$$\dot{}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{b}_i = {}^{A}\boldsymbol{\omega} \times ({}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{b}_i) = {}^{A}\boldsymbol{\omega} \times {}^{A}\boldsymbol{b}_i,$$

where ${}^{A}\boldsymbol{\omega}$ denotes the angular velocity of the moving platform expressed in the fixed frame $\{\mathscr{A}\}$. Similarly, the time derivative of the unit vector $\hat{\boldsymbol{s}}_i$ is determined using the relative differentiation rule (Equation 3.51) as

$$\ell_i\,{}^{A}\dot{\hat{\boldsymbol{s}}}_i = \ell_i\,({}^{A}\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i),$$

where ${}^{A}\boldsymbol{\omega}_i$ is the angular velocity of limb $i$ expressed in fixed frame $\{\mathscr{A}\}$. Hence, the velocity loop closure (Equation 5.52) simplifies to

$${}^{A}\boldsymbol{v}_p + {}^{A}\boldsymbol{\omega} \times {}^{A}\boldsymbol{b}_i = \dot{\ell}_i\,{}^{A}\hat{\boldsymbol{s}}_i + \ell_i\,({}^{A}\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i),$$

(5.53)

where ${}^{A}\boldsymbol{v}_p = [\dot{x}_p, \dot{y}_p, \dot{z}_p]^T$ represents the velocity of the moving platform at point $P$, ${}^{A}\boldsymbol{\omega}$ denotes the angular velocity of the moving platform expressed in the fixed frame $\{\mathscr{A}\}$, and $\hat{\boldsymbol{s}}_i$ is the unit vector in the direction of limb $i$. To eliminate the passive joint rate ${}^{A}\boldsymbol{\omega}_i$, dot multiply both

sides of Equation 5.53 by $\hat{\boldsymbol{s}}_i$ and utilize dot and cross multiplication interchange rules (Equation 5.23) to derive

$$\hat{\boldsymbol{s}}_i \cdot \boldsymbol{v}_p + (\boldsymbol{b}_i \times \hat{\boldsymbol{s}}_i)\, \boldsymbol{\omega} = \dot{\ell}_i,$$

(5.54)

where the superscript $A$ is omitted for brevity in all vectors. Repeat Equation 5.54 six times for $i = 1, 2, \ldots, 6$ and organize it into matrix form as

$$\dot{\boldsymbol{q}} = \boldsymbol{J} \dot{\boldsymbol{\chi}},$$

(5.55)

where $\dot{\boldsymbol{q}} = [\dot{\ell}_1, \dot{\ell}_2, \dot{\ell}_3, \dot{\ell}_4, \dot{\ell}_5, \dot{\ell}_6]^T$ denotes the active input joint rates, $\dot{\boldsymbol{\chi}} = [\boldsymbol{v}_p, \boldsymbol{\omega}]^T$ represents the output twist rate, and the Jacobian matrix $\boldsymbol{J}$ is a $6 \times 6$ square matrix defined as

$$\boldsymbol{J} = \begin{matrix} \hat{\boldsymbol{s}}_1^T & (\boldsymbol{b}_1 \times \hat{\boldsymbol{s}}_1)^T \\ \hat{\boldsymbol{s}}_2^T & (\boldsymbol{b}_2 \times \hat{\boldsymbol{s}}_2)^T \\ \vdots & \vdots \\ \hat{\boldsymbol{s}}_6^T & (\boldsymbol{b}_6 \times \hat{\boldsymbol{s}}_6)^T \end{matrix}.$$

(5.56)

### 5.7.2 Singularity Analysis

This section focuses on analyzing the singularity aspects of the Stewart-Gough platform. A critical drawback of parallel manipulators is the possibility of singular configurations being present within their workspace, where the manipulator can lose one or more degrees of freedom, rendering it uncontrollable. Moreover, such configurations can lead to excessive actuator forces, potentially causing mechanical failure. Thus, it is crucial to avoid singularities within the designated workspace.

To investigate the singularity configurations of the Stewart-Gough platform, we consider the manipulator's Jacobian matrix as defined in Equation 5.56. Referring to the general Jacobian definition in Equation 5.6 and comparing it with Equation 5.5, it becomes evident that for the Stewart-Gough platform, $\boldsymbol{J}_q = \boldsymbol{I}$ and $\boldsymbol{J}_x = \boldsymbol{J}$. Therefore, the manipulator does not exhibit inverse kinematic singularities within its workspace but may encounter forward kinematic singularities when $\boldsymbol{J}$ becomes rank deficient. Inverse kinematic singularities can occur at the workspace boundary, where one or more limbs reach fully extended or retracted positions. Forward kinematic singularities, on the other hand, manifest within the manipulator's workspace precisely when $\boldsymbol{J}$ satisfies the condition of rank deficiency. Given that the manipulator's Jacobian is a square $6 \times 6$ matrix, this occurs when

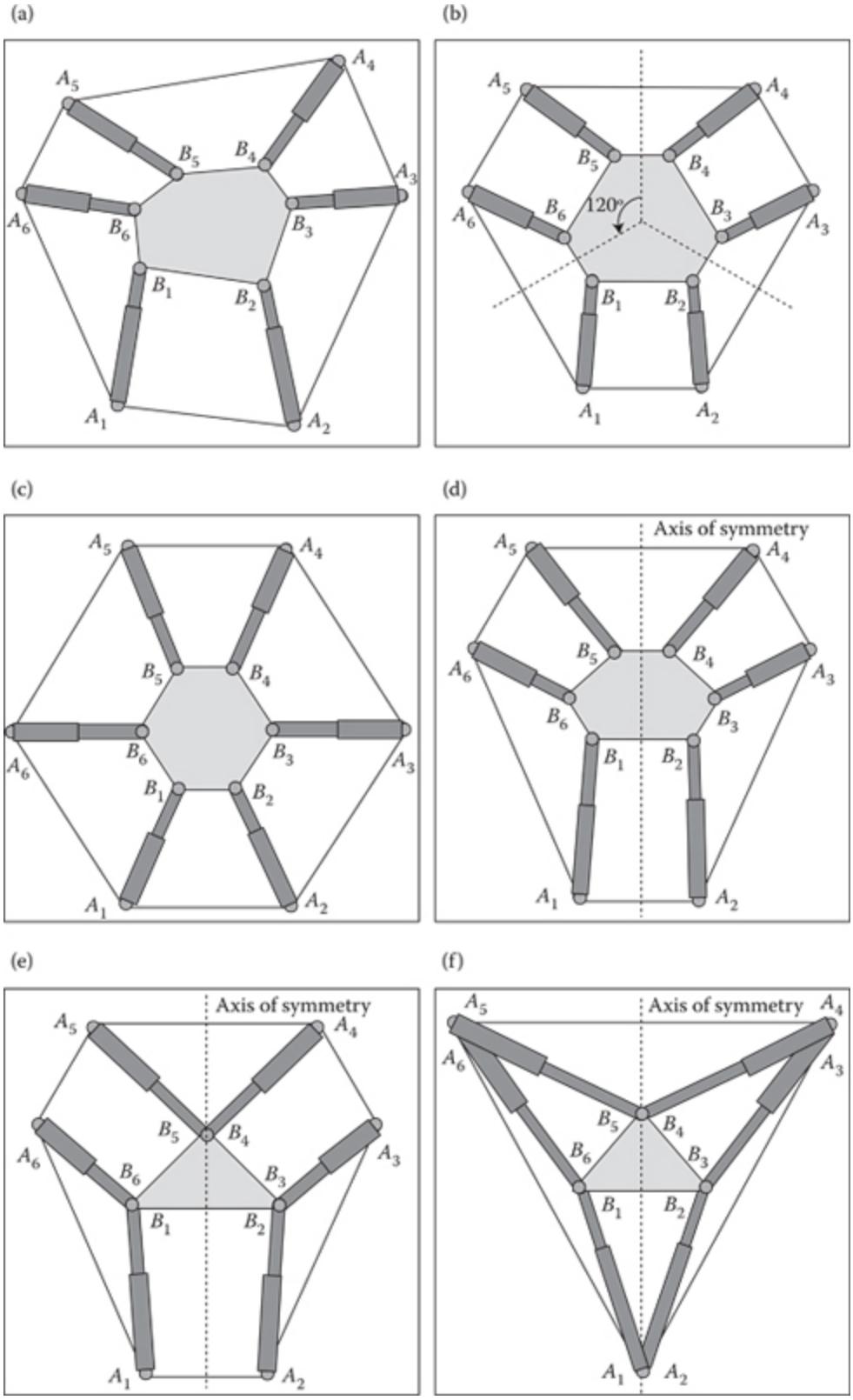
$$\det\left(\boldsymbol{J}\right) = 0. \tag{5.57}$$

This condition signifies a critical point where the manipulator loses its capability to offer a unique solution for the velocities of its moving platform, indicating a potential singularity within its operational workspace.

## Background Literature

Identifying the precise locations of such singularities within the manipulator's workspace is a complex challenge that has been extensively studied. Hunt [141] found that a singularity can occur when all lines associated with the prismatic actuators intersect at a single line. Fichter [142] examined the specific configuration of the Gough-Stewart platform, known as the Triangle Simplified Symmetric Manipulator (TSSM), and identified another type of singularity that arises when the platform is rotated about an axis perpendicular to the base plane by $\pm 90^{\circ}$.

Huang [143] investigated the loci and distribution characteristics of singularities in the TSSM architecture, classifying all singularities into three distinct linear-complex categories, as depicted in Figure 5.6e. In [144], the 3-6 fully parallel manipulator (Figure 5.6e, TSSM architecture) was analyzed, and a singularity locus equation was derived using Rodrigues parameters. Subsequently, Di Gregorio [145] presented the singularity locus expression for a 6-6 fully parallel manipulator (Figure 5.6d, Simplified Symmetric Manipulator (SSM) architecture) using mixed products of vectors.




▶ Long Description for Figure 5.6

**FIGURE 5.6**

Top view of some Stewart–Gough architectures. (a) Platform with irregular hexagons, (b) platform with semiregular hexagons, (c) platform with regular hexagons, (d) SSM architecture, (e) TSSM architecture, and (f) MSSM architecture. ⏎

Wolf and Shoham [146] utilized line geometry and screw theory to determine singular configurations of parallel mechanisms and their behavior at these points. They analyzed examples from Hunt's and Fichter's studies, as well as selected instances from Merlet's work [136], using this method to deepen the understanding of singularities in parallel manipulators.

Merlet [136] employed Grassmann geometry to analyze simplified versions of the Gough-Stewart platforms, including the Simplified Symmetric Manipulator (SSM) (Figure 5.6d), the Triangle Simplified Symmetric Manipulator (TSSM) (Figure 5.6e), and the Minimal Simplified Symmetric Manipulator (MSSM) (Figure 5.6f). Each of these configurations exhibits different geometric conditions leading to singular configurations. While the method is applicable to various parallel mechanisms, expressing these conditions mathematically can be challenging. Notably, in 1998, Kim and Chung [147] derived an analytical expression for the singularity locus equation of the general Stewart-Gough platform with constant orientation using a local structurization method

with additional sensors, which resulted in a third-degree polynomial in three position variables.

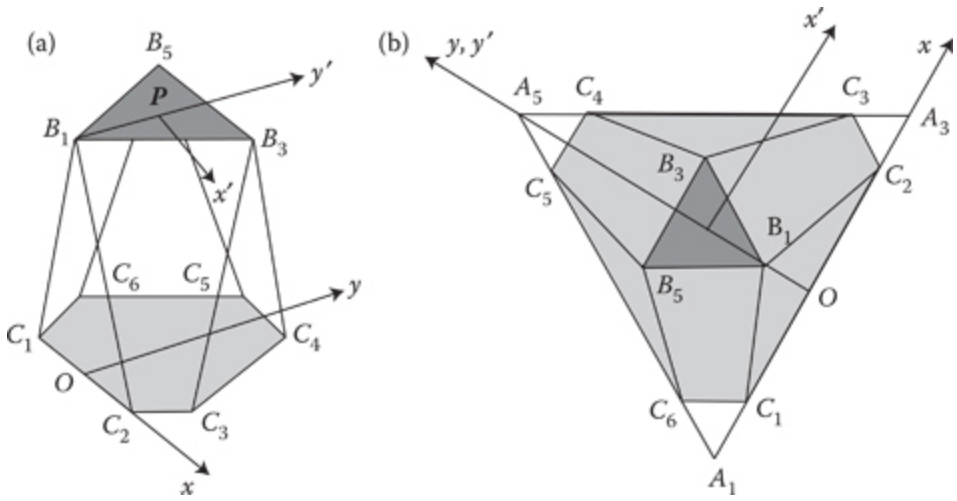
Subsequently, Mayer St-Onge and Gosselin [148] independently arrived at the same result by expanding the Jacobian matrix of the mechanism with constant orientation using linear decomposition and cofactor expansion techniques. From a design perspective, having an analytical expression of the singularity locus is advantageous as it allows designers to visualize and understand where singularities occur within the workspace based on given design parameters. This understanding facilitates the avoidance or mitigation of singularities during the design phase. In a later work [27], the same authors extended their analysis to derive an analytical expression for the singularity locus in 6D Cartesian space using the velocity equation of the Stewart-Gough platform. This expression, typically a polynomial in six variables (three position and three orientation variables), despite its complexity, can be managed both numerically and analytically. Given the complexity of visualizing a singularity locus in 6D space, [27] introduced a numerical procedure to represent the singularity locus graphically within a 3D workspace, assuming constant orientation or position, or fixing any three out of the six variables. It is important to note that since the Stewart-Gough platform lacks redundant actuation, there can be multiple possible singularity loci within its workspace. To illustrate this in detail, the

singular configuration of a 3-6 Stewart-Gough platform is examined in the following subsection.

## A 3–6 Stewart–Gough Platform

The detailed study of forward kinematic singularities in a 3-6 Stewart-Gough platform with general orientation is extensively covered in [28]. Due to its relative simplicity and relevance, this analysis has been adopted in this book. Leveraging the principles of singularity kinematics, [28] proposes an equivalent planar mechanism for singularity analysis. This approach transforms the complex task of analyzing singularities in the parallel mechanism into a more straightforward positional analysis of the planar mechanism.

A typical 3-6 Stewart-Gough platform is depicted schematically in Figure 5.7. It features an equilateral triangle moving platform $B_1B_3B_5$ and a semiregular hexagon base $C_1, C_2, \ldots, C_6$. A fixed frame $O$: $xyz$ is attached to the base platform, while a moving frame $P$: $x'y'z'$ is attached to the moving platform. Point $P$, represented by the position vector $\boldsymbol{P} = [x, y, z]^T$ in the fixed frame, serves as the geometric center of the moving platform. The orientation of the moving platform is described by the $w - v - w$ Euler angles $(\alpha, \beta, \gamma)$.




▶ Long Description for Figure 5.7

**FIGURE 5.7**

(a) Schematic of a 3–6 Stewart–Gough platform and (b) its top view. ⏎

In [28], the structure of the singularity surface of the 3-6 Stewart-Gough platform for all different orientations $(\alpha, \beta, \gamma)$ is classified as follows:

When $\beta = 0$, the moving and base plates are parallel to each other. This special configuration of the mechanism at $(90^\circ, 0, 0)$ renders the mechanism singular for all positions of the moving platform [142].

When $\beta = 0$ and $Z = 0$, the moving platform and the base coincide. The mechanism is singular for all other two Euler angles.

When $\beta \neq 0$, $\alpha = (\pm 30^\circ, \pm 90^\circ, \pm 150^\circ)$, and $\gamma \neq (\pm 30^\circ, \pm 90^\circ, \pm 150^\circ)$, the singularity loci for this orientation include a plane (Hunt plane) and a

hyperbolic paraboloid [143]. The intersecting line between the moving platform and the base is parallel to one side of the triangle $A_1A_3A_5$.

When $\beta \neq 0$, $\alpha = (\pm 30^\circ, \pm 90^\circ, \pm 150^\circ)$, and $\gamma$ is one of the angles $\pm 30^\circ$, $\pm 90^\circ$, or $\pm 150^\circ$, the singularity loci for this orientation are three intersecting planes [143].

When $\beta \neq 0$, $\alpha \neq (\pm 30^\circ, \pm 90^\circ, \pm 150^\circ)$, and $\gamma = (\pm 30^\circ, \pm 90^\circ, \pm 150^\circ)$, the singularity loci for this orientation also include a plane and a hyperbolic paraboloid.

When $\beta \neq 0$ and neither $\alpha$ nor $\gamma$ is equal to any one of the angles $\pm 30^\circ$, $\pm 90^\circ$, $\pm 150^\circ$, it is the most general case. The singularity equation for this orientation is a special irresolvable polynomial expression of degree three, and the structure of the singularity loci in infinite parallel principal sections includes a parabola, four pairs of intersecting straight lines, and an infinity of hyperbolas.

To visualize the locations of singularities within the workspace of a 3–6 Stewart–Gough platform, singular configurations are determined using Equation 5.13 at numerous grid points throughout the manipulator's workspace. Given that the workspace is six-dimensional, it is impractical to visualize it entirely in a 3D plot. Therefore, the analysis focuses on varying three coordinates while keeping the other three fixed. Specifically, calculations are performed for 4410 grid points within the ranges

$-1 \leq x \leq 1, \quad -1 \leq y \leq 1, \quad$ and $\quad 0 \leq \theta \leq \pi/2, \quad$ with the manipulator height fixed at $z = 1.0$.

The analysis investigates three scenarios where rotations are applied along the $x$, $y$, and $z$ axes. Singular configurations are only observed when $\theta_z = \pi/2$, as depicted in Figure 5.8. Throughout the constant orientation workspace, the manipulator remains devoid of singularities. This analysis underscores the robustness of the Stewart–Gough platform's design across its entire operational range, underscoring its suitability for demanding applications such as flight simulators.

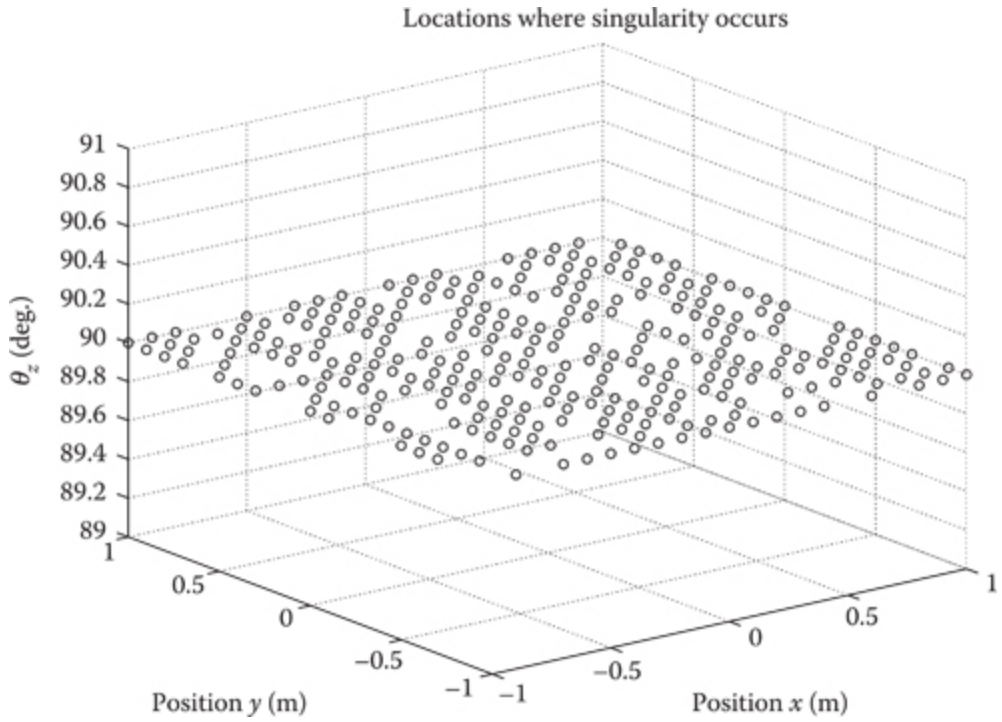



**FIGURE 5.8**

Singular configurations of a 3–6 Stewart–Gough platform for fixed $z = 1.0$, and $\theta_z$ rotation about $z$-axis. ⏎

## Problems

5.1 Consider the five-bar mechanism depicted in Figure 4.27 and discussed in Problem 4.1. Determine the end-effector velocity vector $\boldsymbol{v}_p = [v_x, v_y]^T$ in terms of the two input angular velocities $\dot{\boldsymbol{q}} = [\dot{\theta}_1, \dot{\theta}_2]^T$. Derive the Jacobian matrix of the manipulator and identify configurations where the manipulator becomes singular.

5.2 Consider the five-bar mechanism depicted in Figure 4.28 and detailed in Problem 4.2. Calculate the end-effector velocity vector $\boldsymbol{v}_p = [v_x, v_y]^T$ in terms of the two input velocity variables, $\dot{\boldsymbol{q}} = [\dot{\ell}_1, \dot{\theta}_2]^T$. Derive the Jacobian matrix for the manipulator and investigate its singular configurations.

5.3 Examine the planar pantograph mechanism illustrated in Figure 4.29 and detailed in Problem 4.3. Determine the end-effector velocity vector $\boldsymbol{v}_p = [v_x, v_y]^T$ in relation to the linear velocities of its two input links, $\dot{\boldsymbol{q}} = [\dot{x}_1, \dot{y}_2]^T$. Determine the Jacobian matrix for this manipulator and analyze its singular configurations.

5.4 Consider the planar three degrees of freedom 3RRR parallel manipulator shown in Figure 4.30 and described in Problem 4.4.

a. Formulate the vector loop closure equations for this manipulator and establish relationships between the

twist rate of the moving platform $\dot{\boldsymbol{\chi}} = [v_x, v_y, \dot{\phi}]^T$, and the angular velocity of the actuated limbs $\dot{\alpha}_i$ 's and passive limbs $\dot{\beta}_i$ 's.

b. Eliminate the angular velocities of the passive limbs $\dot{\beta}_i$ obtained in part (a), and derive expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. Identify configurations of inverse and forward singularities of the manipulator based on the structure of the Jacobian matrices.

d. Implement a MATLAB® or Python program using the following numerical values for the manipulator parameters. Generate a grid covering the entire workspace of the manipulator, compute the numerical values of Jacobian matrices at each grid point, and detect singular configurations. Visualize these singular configurations of the manipulator in a 3D plot and compare them with the results obtained in part (c).

$$a_i = b_i = 1\,(m), \quad L = 5\,(m), \quad \ell = 1.5\,(m).$$

Analyze the manipulator's workspace within the specified limits:

$$-2(m) \leq x \leq 2(m), \quad -2(m) \leq y \leq 2(m), \quad -90^\circ \leq$$

e. Determine if there are any configurations where the manipulator exhibits combined singularities, if applicable.

5.5 Consider the planar four-degrees-of-freedom 4RRR parallel manipulator depicted in Figure 4.31 and described in Problem 4.5.

a. For this manipulator, establish the vector loops closure and derive a relationship between the moving platform twist rate $\dot{\boldsymbol{\chi}} = [v_x, v_y, \dot{\phi}]^T$ and the angular velocity of the actuated limbs $\dot{\alpha}_i$ 's and passive limbs $\dot{\beta}_i$ 's.

b. Remove the passive limb angular velocities $\dot{\beta}_i$ 's from the equation derived in part (a) and determine the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. By inspecting the structure of the Jacobian matrices, identify the inverse and forward singularity configurations of the manipulator.

d. Create a MATLAB or Python program with the following numerical values for the manipulator parameters. Generate a grid space for the entire workspace of the manipulator, calculate the numerical values of the Jacobian matrices at each grid point, and identify singular configurations. Illustrate the singular configurations of the

manipulator in a 3D plot, and compare the result to the one obtained in part (c).

$$R_A = 3\,(m), R_B = 1\,(m), a_i = b_i = 1.5\,(m),$$
$$\theta_{A_i} = \theta_{B_i} = [-135^\circ, -45^\circ, 45^\circ, 135^\circ]^T.$$

Consider the manipulator workspace within the following limits:

$$-2\,(m) \leq x \leq 2\,(m), \quad -2\,(m) \leq y \leq 2\,(m), \quad -90^\circ \leq$$

e. Identify any configuration in which the manipulator has combined singularity, if any.

5.6 Consider the spatial parallel manipulator shown in Figure 4.32 and detailed in Problem 4.6.

a. For this manipulator, establish the vector loops closure and derive a relationship between the moving platform angular velocity $\boldsymbol{\omega}$ and the linear velocity vector of the actuated limbs $\dot{\ell}_i$ 's and their angular velocity $\omega_i$'s.

b. Remove $\omega_i$'s from the equations derived in part (a) and determine the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. By inspecting the structure of the Jacobian matrices, identify the inverse and forward singularity

configurations of the manipulator.

d. Assign some parametric values to the geometry of the manipulator, and develop a MATLAB or Python program to generate a grid space for the entire workspace of the manipulator. Calculate the numerical values of the Jacobian matrices at each grid point, and identify singular configurations. Illustrate the singular configurations of the manipulator in a 3D plot, and compare the result to the one obtained in part (c). Consider the manipulator workspace within the following limits:

$$-30^\circ \leq \alpha, \beta, \gamma \leq 30^\circ.$$

5.7 Consider the 3UPU parallel manipulator shown in Figure 4.33 and described in Problem 4.7.

a. For this manipulator, establish the vector loops closure and determine the relationship between the moving platform velocity $\boldsymbol{v}_P$ and the linear velocity vector of the actuated limbs $\dot{q}_i$ 's and the angular velocities of the unactuated joints $\omega_i$'s.

b. Eliminate $\omega_i$'s from the equations obtained in part (a) and determine the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. Identify the inverse and forward singularity configurations of the manipulator by examining the structure of the Jacobian matrices.

d. Using the parametric values given in Problem 4.7, develop a MATLAB or Python program to generate a grid space for the entire workspace of the manipulator. Calculate the numerical values of the Jacobian matrices at each grid point and identify singular configurations. Illustrate the singular configurations of the manipulator in a 3D plot, and compare the results to those obtained in part (c). Consider the manipulator workspace within the following limits:

$$-1\,(m) \leq x, y, z \leq 1\,(m).$$

5.8 Consider a 3UPS parallel manipulator introduced in [51], shown in Figure 4.34 and described in Problem 4.8.

a. For this manipulator, establish the velocity loop closure and derive the relationship between the 6D twist rate of the moving platform, $\dot{\boldsymbol{\chi}}$, and the vector of actuated limb velocities, $\dot{\boldsymbol{q}}$, given by $\dot{\boldsymbol{q}} = \boldsymbol{J}_v \dot{\boldsymbol{\chi}}$. Then, determine the Jacobian of the passive supporting limb, $\boldsymbol{J}_p$, which relates the 6D twist rate of the moving platform, $\dot{\boldsymbol{\chi}}$, to the vector of linear and angular velocity of the passive limb, $\boldsymbol{v}_p = [\dot{h}, \omega_{p_x}, \omega_{p_y}]^T$, by $\dot{\boldsymbol{\chi}} = \boldsymbol{J}_p \boldsymbol{v}_p$.

b. Determine the Jacobian matrix of the manipulator relating the actuator velocities $\dot{\boldsymbol{q}}$ to the moving

platform generalized velocity $\boldsymbol{v}_p$, given by $\dot{\boldsymbol{q}} = \boldsymbol{J} \cdot \boldsymbol{v}_p$, using the active and passive limb Jacobian matrices $\boldsymbol{J} = \boldsymbol{J}_v \boldsymbol{J}_p$.

c. Using the parametric values given in Problem 4.8, develop a MATLAB or Python program to generate a grid space for the entire workspace of the manipulator. Calculate the numerical values of the Jacobian matrix at each grid point and identify singular configurations. Illustrate the singular configurations of the manipulator in a 3D plot. Consider the manipulator workspace within the following limits:

$$0 \leq h \leq 0.4\,(m), \quad -30^\circ \leq \psi, \phi \leq 30^\circ.$$

5.9 Consider the three degrees of freedom 3RPS manipulator depicted in Figure 4.35 and described in Problem 4.9.

a. For this manipulator, establish the velocity loop closure and determine the relationship between the moving platform linear velocity $\boldsymbol{v}_E$ and the linear velocity vector of the actuated limbs, $\dot{\ell}_i$ 's, as well as their angular velocities $\omega_i$'s.

b. Remove $\omega_i$'s from the equations obtained in part (a) and derive the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. By examining the structure of the Jacobian matrices, identify the inverse and forward singularity configurations of the manipulator.

d. Assign specific parametric values to the geometry of the manipulator, and develop a MATLAB or Python program to generate a grid space for the entire workspace of the manipulator. Calculate the numerical values of the Jacobian matrices at each grid point and identify singular configurations. Illustrate the singular configurations of the manipulator in a 3D plot and compare the results to those obtained in part (c).

5.10 Consider the three degrees of freedom 3PRC translational parallel manipulator represented in Figure 4.35 and described in Problem 4.10.

a. For this manipulator, establish the velocity loop closure and derive the relationship between the moving platform linear velocity $\boldsymbol{v}_P$ and the linear velocity vector of the actuated limbs $\dot{d}_i$ 's, as well as the limbs' angular velocities $\omega_i$'s.

b. Remove $\omega_i$'s from the equations obtained in part (a) and find the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. By examining the structure of the Jacobian matrices, identify the inverse and forward singularity configurations of the manipulator.

d. Assign specific parametric values to the geometry of the manipulator, and develop a MATLAB or Python program to generate a grid space for the entire workspace of the manipulator. Calculate the numerical values of the Jacobian matrices at each grid point and identify singular configurations. Illustrate the singular configurations of the manipulator in a 3D plot and compare the results to those obtained in part (c).

5.11 Consider the Delta robot represented in Figure 4.38 and described in Problem 4.11.

a. For this manipulator, establish the velocity loop closure and derive the relationship between the moving platform linear velocity $\boldsymbol{v}_P$ and the angular velocity vector of the actuated limbs $\dot{\theta}_i$ 's, as well as the angular velocities of the passive limbs $\dot{\beta}_i$ 's.

b. Remove $\dot{\beta}_i$ 's from the equations obtained in part (a) and determine the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. By examining the structure of the Jacobian matrices, identify the inverse and forward singularity configurations of the manipulator.

d. Assign specific parametric values to the geometry of the manipulator, and develop a MATLAB or Python program to generate a grid space for the entire workspace of the manipulator. Calculate the

numerical values of the Jacobian matrices at each grid point and identify singular configurations. Illustrate the singular configurations of the manipulator in a 3D plot and compare the results to those obtained in part (c).

5.12 Consider the Quadrupteron robot represented in Figure 4.39 and described in Problem 4.12 [55].

a. For this manipulator, establish the velocity loop closure and determine the relationship between the moving platform linear velocity $\boldsymbol{v}_P$ and its angular velocity $\dot{\phi}$ to the linear velocity vector of the actuated limbs $\dot{\rho}_i$ 's, as well as the angular velocities of the passive limbs.

b. Eliminate the angular velocities of the passive limbs from the equations obtained in part (a) and derive the expressions for the Jacobian matrices $\boldsymbol{J}_x$ and $\boldsymbol{J}_q$ of the manipulator.

c. By examining the structure of the Jacobian matrices, identify the inverse and forward singularity configurations of the manipulator.

d. Assign specific parametric values to the geometry of the manipulator, and develop a MATLAB or Python program to generate a grid space for the entire constant orientation workspace of the manipulator. Calculate the numerical values of the Jacobian matrices at each grid point and identify singular

configurations. Illustrate the singular configurations of the manipulator in a 3D plot and compare the results to those obtained in part (c).

# 6

# *Jacobian Matrix: Mappings & Characteristics*

DOI: 10.1201/9781003590408-6

This chapter explores the mappings and intrinsic characteristics of the Jacobian matrix, highlighting its role in understanding the behavior and performance of parallel manipulators. It begins with an examination of static force analysis using both the free-body diagram and virtual work approaches, offering complementary perspectives for analyzing force distribution. Three case studies—planar cable robots, shoulder manipulators, and the Stewart–Gough platform—demonstrate the practical application of these methodologies in static force computation.

The discussion transitions to the intrinsic characteristics of the Jacobian matrix, with a focus on geometric and performance-related metrics. Key concepts such as the dexterity ellipsoid, singularity, isotropy, and measures of manipulability and dexterity are introduced, providing insights into the workspace capabilities of robotic systems.

Global performance measures are also discussed to evaluate overall system efficiency and reliability.

Stiffness analysis is explored next, emphasizing the importance of stiffness matrices and ellipsoids in understanding system rigidity and compliance. This section is supported by detailed case studies for each robotic platform, which illustrate the interplay between design parameters and stiffness characteristics. The exploration of inverse problems in differential kinematics and force mapping further deepens the understanding of Jacobian-related analyses, showcasing methods for resolving complex challenges in robotic motion and force control.

The chapter concludes by addressing redundancy resolution, presenting methods for optimizing redundant systems. Techniques such as Lagrange and Karush–Kuhn–Tucker multipliers, iterative solutions, numerical approaches, and hybrid methods are discussed in detail. Practical problems encourage readers to apply these concepts to challenging scenarios, fostering a deeper understanding of Jacobian mappings and their implications for robotic design and performance optimization.

## 6.1 Introduction

Researchers frequently draw inspiration from the biological skeletal structures of creatures in nature, particularly humans, when seeking optimal design solutions for

robotics. By emulating these natural structures, robotic designs have been developed to mirror the extraordinary characteristics found in these remarkable biological systems.

To deepen our mathematical understanding of attributes such as agility and dexterity in robots, let's consider some examples of the musculoskeletal composition of certain animals. Achieving full movement in three-dimensional space requires six degrees of freedom. This is why many industrial robots, whether serial or parallel with full actuation, offer these degrees of freedom. This raises an intriguing question: does nature follow a similar pattern? Do the musculoskeletal structures of animals also utilize six degrees of freedom?

To address these questions, consider the skeletal structures of animals and humans. One notable example is the variation in the number of vertebrae in the necks of mammals and birds. The neck of all mammals typically comprises seven vertebrae, whereas the neck of all birds consists of fourteen vertebrae. This variability in articulation serves to minimize or entirely eliminate singular configurations in the neck movements of mammals. For birds, whose survival, food acquisition, hunting, and predator evasion depend on rapid and agile neck movements, the number of neck vertebrae doubles, resulting in eight degrees of redundancy in neck articulation. It is astonishing that a giraffe, with its long

neck, has only seven vertebrae, while a tiny hummingbird has fourteen vertebrae in its neck.

This phenomenon is equally fascinating when considering the number of actuators present in the human shoulder, arm, and wrist structure. The human shoulder, like that of many animals, facilitates complete rotational motion with three degrees of freedom for the arm. The shoulder joint is ingeniously designed as a ball-and-socket joint, providing these three degrees of freedom simultaneously. However, this movement is induced by four muscle groups, introducing an additional degree of actuation redundancy. One crucial reason for incorporating actuation redundancy in this structure is to prevent singular configurations in the shoulder joint, as mathematically verified in the mechanical counterpart case study elaborated in Section 5.6.

Examining the musculoskeletal structure of the human arm reveals three degrees of freedom in the shoulder, two degrees of movement in the elbow, and two degrees of movement in the wrist. Thus, the human arm utilizes seven actuators to achieve six degrees of freedom in motion, with an extra degree of actuation redundancy. The intricately intertwined musculoskeletal structure of the elbow and wrist involves a total of seven muscle groups contributing to the creation of four degrees of freedom in motion. The design of the human wrist is also a masterpiece, utilizing eight bones and four muscle groups to generate three rotational movements.

In these instances, as well as many others observed in nature, redundancy in actuation and appropriate configuration are utilized to attain maximum accessibility, achieve optimal speed, or distribute optimal wrench to the manipulating objects. Translating these effective structures into inspiration, it becomes imperative to establish connections between these physical concepts and computable parameters for optimization in the design and implementation of parallel robots.

Unfortunately, formulating criteria that comprehensively encompass all desired features in the optimal design of a robot is not straightforward and necessitates extensive investigation. Consequently, researchers, drawing inspiration from nature and interpreting certain aspects, have introduced diverse criteria grounded in the singularity characteristics of the Jacobian matrix, each addressing a facet of reality. In the subsequent section, we will explore the mathematical definition of some of these criteria and metrics.

In the previous chapter, we elaborated on the fundamental role of the Jacobian matrix, showcasing its critical function in mapping the twist rate of the moving platform to the active joint velocities in robotic systems. This matrix, a cornerstone of robotic kinematics, enabled us to identify and define scenarios where the system becomes singular. Singularities, which occur when the determinant of the Jacobian matrix is zero, represent configurations where in the context of parallel robots, the

manipulator loses its control on certain degrees of freedom, resulting in an inability to apply forces in specific directions. We explored various types of singularities pertinent to parallel robots and thoroughly examined the underlying physical concepts that influence the inverse velocity solutions in these cases, emphasizing their practical implications and theoretical foundations.

In this chapter, we focus deeper on the multifaceted characteristics of the Jacobian matrix, expanding our analysis beyond velocity mapping to include force mapping. This shift in focus allows us to explore new dimensions of robotic performance and control. We will first demonstrate that the Jacobian matrix provides the crucial transformations needed to determine the actuator forces from the wrench components acting on the moving platform, as well. This dual functionality of the Jacobian matrix reveals its indispensable role in both kinematic and static force analyses.

By bridging the concepts of singularity and dexterity in the context of force mapping, we open new avenues for understanding and optimizing robotic performance. We will investigate how the Jacobian matrix facilitates a deeper comprehension of the force transmission capabilities of parallel manipulators, highlighting its impact on the design and control strategies aimed at enhancing robot dexterity and efficiency. Through this exploration, we aim to provide a comprehensive framework that links the theoretical

insights of the Jacobian matrix to practical applications in advanced robotic systems.

## 6.2 Static Forces in Parallel Manipulators

In tasks such as grinding and assembly operations, where robots closely interact with objects by applying forces or torques, understanding the force interactions within the robot's closed kinematic chain is essential. The actuator forces or torques in the robot's active limbs directly impact the forces and torques exerted by the robot's moving platform on the objects it handles. This section examines the static relationship between these internal forces and torques in joint space, translating into the corresponding wrench in task space, under static conditions.

In this analysis, the robot is assumed to be in a state of static equilibrium, without considering dynamic motion. To establish a direct mapping between the moving platform wrench and the actuator forces/torques, the robot's weight is not considered. As previously defined, to mathematically represent the amount of force and torque applied to the environment, we use a six-dimensional array called a wrench, where the first three components represent force and the last three components represent torque.

Two main methods have been developed for static force analysis in robots. The first method uses free-body diagrams of the forces and Newton-Euler laws in a static

state. In this approach, all robot links are separated, and free-body diagrams for the forces are applied. The connections between the links are introduced by the active and support forces and torques in the joints. Then, the Newton-Euler laws in the static state are applied to each robot link to determine the actuator forces/torques by eliminating the support forces and torques. This method requires considering all support forces and torques in the robot joints and then determining or eliminating them from the free-body diagram equilibrium equations, resulting in significant computational costs.

Another efficient method discussed in this book is the use of the principle of virtual work in static force analysis. In this method, support forces are not included in the calculations, leading to a substantial reduction in computational efforts. Furthermore, we will show that the relationship between the actuator forces/torques in joint space and the wrench applied to the environment in task space is determined using the transpose of the Jacobian matrix. This representation facilitates a conceptual description of this mapping using the Jacobian matrix, and concepts related to the intrinsic characteristics of the Jacobian matrix applied to the static force analysis domain.

### 6.2.1 Free-Body Diagram Approach

In this section, we investigate the free-body diagram approach for static force analysis. A parallel manipulator consists of a movable platform connected to a stationary

base via multiple limbs. The number of limbs is at least equal to the degrees of freedom of the moving platform, ensuring each limb is driven by at most one actuator. Due to its closed-loop structure, the joints of all limbs are not independently controlled; some are actuated while others remain passive.

The static analysis of such manipulators is significantly simplified if we assume that the limbs consist of binary links capable of bearing only tension or compression forces. This assumption holds generally if no external forces, other than actuator forces, are applied to the links, and all external forces and disturbances are applied to the moving platform. Therefore, in a planar manipulator, if no external force acts on a binary link, the static force can be assumed to be along the link axis, subjecting the link to pure tension or compression. Similarly, in a spatial manipulator, if no external force acts on a spherical-spherical binary link chain, the link will be subjected to tension or compression along a line passing through the two spherical joints.

In the free-body diagram approach, a detailed analysis of all forces and moments exerted on each link of the manipulator and the moving platform is conducted. Each component, including the manipulator's links and moving platform, is examined individually by drawing separate free-body diagrams. These diagrams illustrate all the forces and moments acting on the respective parts, providing a clear visualization of the static forces at play. The external forces and torques acting on the moving platform are

assumed to be known, while the actuator forces and torques need to be determined. To achieve this, we write the static equilibrium equations for forces and moments for each free-body diagram. These equations account for the balance of forces and moments, ensuring that each link and the moving platform are in static equilibrium.

Next, by systematically eliminating the internal forces that act between the base and the links, between the links themselves, and between the links and the moving platform, we can identify the relationship between the external forces/torques and the actuator forces. This step involves algebraic manipulation to remove the internal forces from the equations, simplifying the system to focus solely on the relationship of interest.

It can be demonstrated that for a general parallel manipulator, the relationship between the external wrench applied to the moving platform and the actuator forces in the limbs can be expressed in matrix form, where the Jacobian transpose matrix defines this mapping. This matrix formulation allows for a more efficient and compact representation of the relationships, facilitating easier computation and analysis. By utilizing the Jacobian transpose, the complexities of internal force interactions are reduced, providing a clear and direct way to relate external wrenches to the necessary actuator forces, thus enhancing the overall understanding and control of the manipulator's static force behavior.

This method is further illustrated and applied in various case studies presented in the following subsections. These case studies provide practical examples and detailed walkthroughs of the free-body diagram approach, highlighting its effectiveness in deriving the necessary relationships for static force analysis in parallel manipulators.

### 6.2.2 Virtual Work Approach

In this section, we will study the static force analysis in serial robots using the *Principle of Virtual Work*. To introduce this principle, it is necessary first to define virtual displacement. Virtual displacement refers to an infinitesimally small change in the position and orientation of the robot's moving platform caused by an infinitesimally small movement in the robot's active joints. The term “virtual” is used to distinguish it from actual motion. During actual motion, the geometric structure of the robot, the magnitudes of forces and torques applied to the environment, and the torques exerted by the actuators change. However, in virtual displacement, it is assumed that the robot's geometry and the distribution of forces remain unchanged despite this infinitesimal movement.

To distinguish virtual variables from real ones during this infinitesimally small movement, the notation $\delta$ is used. This contrasts with real motions where the differential symbol $d$ is used. Thus, the virtual displacement of the robot's active

joints is represented by $\delta \boldsymbol{q}$, where $\boldsymbol{q}$ represents the active joint variables.

$$\delta \boldsymbol{q} = \begin{bmatrix} \delta q_1 & \delta q_2 & \dots \delta q_n \end{bmatrix}^T \tag{6.1}$$

This notation helps clarify the difference between virtual displacements and actual displacements occurring during the robot's motion.

Additionally, the virtual displacement of the robot's moving platform is represented by the notation $\delta \boldsymbol{\chi}$. As before, this is interpreted through a six-dimensional array using screw coordinates. This array encapsulates both the infinitesimal positional and orientation changes of the moving platform, providing a comprehensive representation of the virtual displacement in the robot's task space.

$$\delta \boldsymbol{\chi} = \begin{bmatrix} \delta x & \delta y & \delta z & \delta \theta_x & \delta \theta_y & \delta \theta_z \end{bmatrix}^T \tag{6.2}$$

Note that in this representation, the orientation variables, represented by screw coordinates, are given by:

$$\delta \boldsymbol{\theta} = \begin{bmatrix} \delta \theta_x & \delta \theta_y & \delta \theta_z \end{bmatrix}^T \tag{6.3}$$

Similarly, as illustrated in Figure 6.1, the array representing the forces and torques exerted by the limb's actuators is denoted as

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \dots & \tau_n \end{bmatrix}^T$$

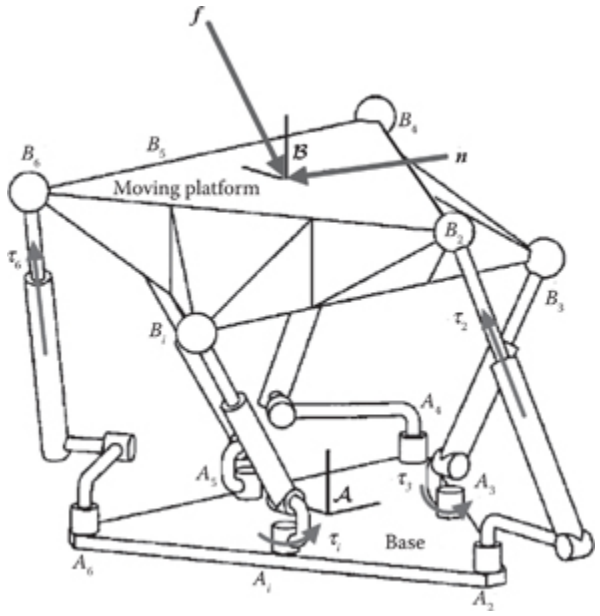



**FIGURE 6.1**
Schematic of a general parallel manipulator on which the applied external wrench and the corresponding actuator forces/torques are indicated. ⏎

where $\tau_i$ denotes the torque at revolute joints or the force at prismatic joints generated by the corresponding rotary or linear actuator $i$. It's important to note that the symbol $\tau$ is uniformly used to represent both torques and forces in this notation.

Furthermore, the forces and torques applied to the robot's moving platform due to interaction with the environment are typically represented by a six-dimensional array known as a “wrench,” defined as:

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{n} \end{bmatrix} \tag{6.4}$$

In this context, $\boldsymbol{f} = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$ represents the force vector, and $\boldsymbol{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ denotes the torque vector exerted on the robot's moving platform by the environment.

According to the virtual work principle, by neglecting gravitational and friction forces, the total virtual work done by both actuators and external interactive forces sums to zero, considering virtual displacements in both the moving platform and the limb's active joints. If virtual work is denoted by the scalar variable $\delta W$, this principle can be expressed as:

$$\delta W = \boldsymbol{\tau}^T \delta \boldsymbol{q} - \boldsymbol{\mathcal{F}}^T \delta \boldsymbol{\mathcal{X}} = 0. \tag{6.5}$$

Here, $\boldsymbol{\tau}^T \delta \boldsymbol{q}$ represents the virtual work done by the actuators, while $-\boldsymbol{\mathcal{F}}^T \delta \boldsymbol{\mathcal{X}}$, denotes the virtual work done on the interactive wrench. The negative sign accounts for the fact that $\boldsymbol{\mathcal{F}}$ describes the wrench applied to the robot's

moving platform by the environment. Thus, the virtual work on the environment is calculated with a negative sign to reflect this wrench applied to the moving platform in the computations.

Given that the Jacobian matrix defines the relationship between the derivatives of joint and task space variables through the equation $\dot{\boldsymbol{q}} = \boldsymbol{J}\dot{\boldsymbol{\chi}}$, we can derive the relationship of virtual displacements between joint and task space variables using this mapping as well:

$$\delta \boldsymbol{q} = \boldsymbol{J}\, \delta \boldsymbol{\chi},$$

(6.6)

where for a general parallel manipulator, $\boldsymbol{J} = \boldsymbol{J}_q^{-1}\boldsymbol{J}_x$ as defined in Equation 5.7. Substituting this mapping into Equation 6.5, we obtain:

$$\left(\boldsymbol{\tau}^T \boldsymbol{J} - \boldsymbol{\mathcal{F}}^T\right)\delta \boldsymbol{\chi} = 0.$$

(6.7)

Since this equation holds for any arbitrary virtual displacement $\delta\boldsymbol{\chi}$, we derive:

$$\boldsymbol{\tau}^T \boldsymbol{J} - \boldsymbol{\mathcal{F}}^T = 0.$$

(6.8)

This equation establishes the relationship between the force/torque array in joint space, and the resulting moving platform wrench in task space, through the transpose of the robot's Jacobian matrix:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}. \tag{6.9}$$

Equation 6.9 demonstrates that the Jacobian matrix not only maps between the velocities of joint and task variables but also, its transpose describes the relationship between the actuator forces/torques in joint space and the resulting end-effector wrench in task space. This mapping is depicted in Figure 6.2.

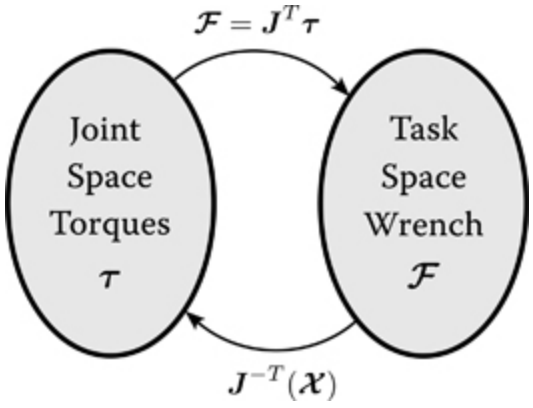



**FIGURE 6.2**
The transposed Jacobian matrix maps the actuator forces/torques in joint space to the wrench applied to the robot's moving platform in task space. ⏎

### 6.2.3 Case Study I: Planar Cable Robot

In this section, we conduct a detailed static force analysis specific to the planar 4RPR cable robot. We begin by employing the free-body diagram approach to derive the relationships between the active limb forces and the resulting wrench applied by the environment to the moving platform, confirming that this relationship is captured by the transpose of the robot's Jacobian matrix.

Let's revisit the architecture of the planar 4RPR cable robot, which is extensively described in Section 4.4.1 and depicted in Figure 4.4. As previously discussed, this manipulator configuration features a moving platform supported by four limbs, each adhering to an identical kinematic structure. Each limb connects the fixed base to the moving platform through a sequence of joints: a revolute joint (R), an actuated prismatic joint (P), and another revolute joint (R). The prismatic joint can represent either a piston-cylinder actuator or a cable-driven mechanism, depending on the design specifics.

Figure 1.6 illustrates key points of interest: $A_i$ denotes the fixed base points of the limbs, $B_i$ indicates the points where the limbs connect to the moving platform, $L_i$ represents the lengths of the limbs, and $\alpha_i$ signifies the angles of the limbs. The position of the center point $G$ of the moving platform is denoted by $G = [x_G, y_G]$, and the orientation of the manipulator's moving platform is represented by $\phi$ relative to the fixed coordinate frame. Consequently, the manipulator possesses three degrees of

freedom $\boldsymbol{\chi} = [x_G, y_G, \phi]$, with an additional degree of actuation redundancy.

In this section, we undertake a static force analysis specifically tailored for the planar 4RPR manipulator. We begin by employing the free-body diagram approach, where a fixed frame $O : xy$ is established for the system, and separate diagrams are drawn for the moving platform and each limb. Actuator forces are denoted by $\boldsymbol{\tau} = [f_1, f_2, f_3, f_4]^T$, while the force $\boldsymbol{f} = [f_x, f_y, 0]^T$ and moment $\boldsymbol{n} = [0, 0, n_z]^T$ applied to the environment at the center point $G$ of the moving platform are considered. Since the manipulator operates in a plane, only three components of the wrench $\boldsymbol{\mathcal{F}}$ are relevant, hence consider $\boldsymbol{\mathcal{F}} = [f_x, f_y, n_z]^T$ for analysis purposes. For simplicity and due to the similarity in applied forces by the limbs to the moving platform, a single limb is exemplified in Figure 6.3.

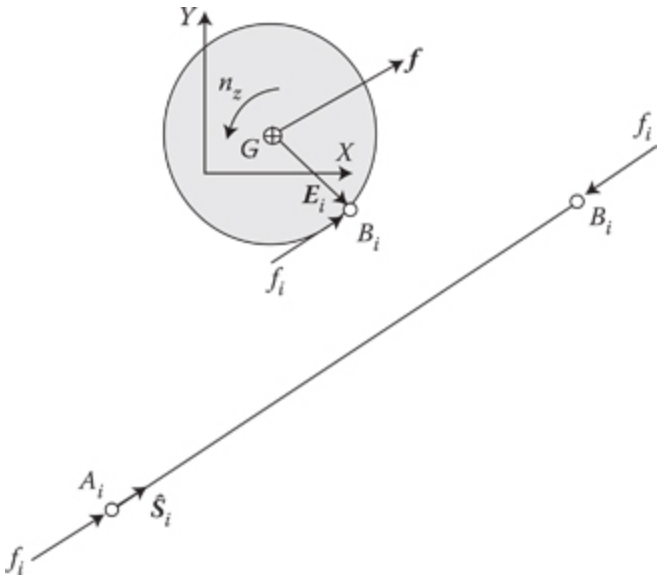



**FIGURE 6.3**

Free-body diagram of forces and moments acting on the moving platform and each limb of planar $4\underline{R}\underline{P}R$ manipulator. ⏎

Initially, we consider the external forces applied to each limb, assuming no external forces directly act on the limbs apart from the actuator forces. Therefore, the static force along the limb axis $\hat{\boldsymbol{S}}_i$ is assumed to be $-f_i$. Moving on to the free-body diagram of forces and moments applied to the moving platform, the sum of all acting forces on the platform must equate to zero. Thus, we have:

$$-\boldsymbol{f} + \sum_{i=1}^{4} f_i \hat{\boldsymbol{S}}_i = \boldsymbol{0}, \tag{6.10}$$

where $-\boldsymbol{f}$ represents the external force applied to the moving platform by the environment.

Additionally, the summation of moments about $G$ due to all forces acting on the moving platform results in:

$$-n_z \hat{\boldsymbol{Z}} + \sum_{i=1}^{4} \boldsymbol{E}_i \times f_i \hat{\boldsymbol{S}}_i = \boldsymbol{0}, \tag{6.11}$$

where $-n_z \hat{\boldsymbol{Z}}$ denotes the external moment applied to the moving platform by the environment, with $\hat{\boldsymbol{Z}}$ as the unit

vector along the $Z$-axis.

By simplifying Equations 6.10 and 6.11 and eliminating redundant equations, we can express the remaining three equations in matrix form as:

$$\begin{bmatrix} S_{i_x} \\ S_{i_y} \\ E_{i_x} S_{i_y} - E_{i_y} S_{i_x} \end{bmatrix}_{3\times 4} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ n \end{bmatrix}, \tag{6.12}$$

where $S_{i_x}$ and $S_{i_y}$ represent the $x$ and $y$ components of the vectors $\hat{\boldsymbol{S}}_i$, and $E_{i_x}$ and $E_{i_y}$ represent the $x$ and $y$ components of the vector $\boldsymbol{E}_i$, respectively. The matrix on the left-hand side of Equation 6.12 is a $3 \times 4$ matrix, with each column defined accordingly.

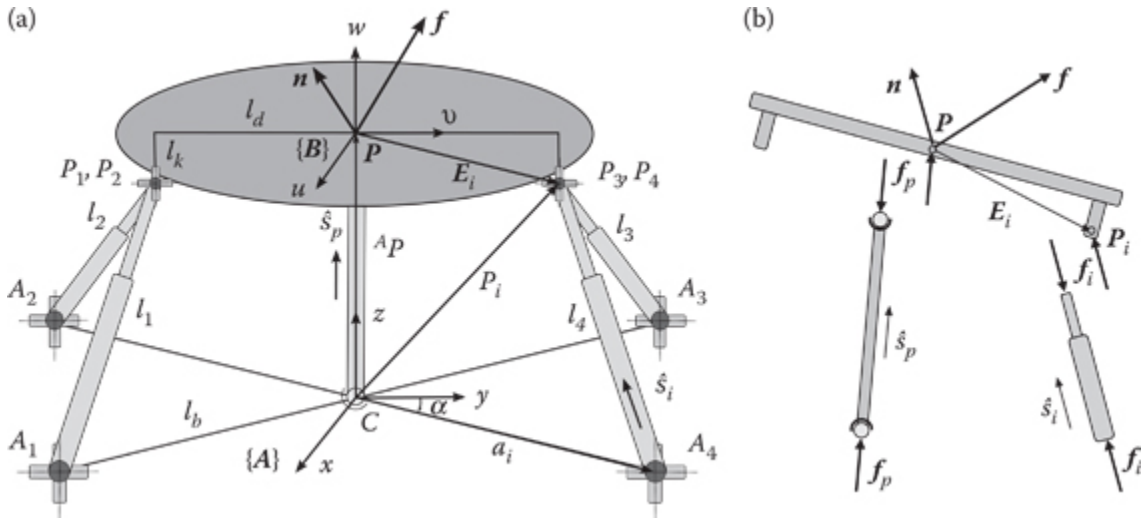
Comparing this matrix with the Jacobian matrix of the 4RPR manipulator given in Equation 5.26, it is evident that this matrix corresponds to the transpose of the manipulator's Jacobian matrix. Therefore, Equation 6.12 can be succinctly rewritten to verify the result using the virtual work approach:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}, \tag{6.13}$$

in this relation, $\boldsymbol{\tau} = [f_1, f_2, f_3, f_4]^T$ represents the vector of actuator forces, and $\boldsymbol{\mathcal{F}} = [f_x, f_y, n_z]^T$ denotes the wrench generated by the planar manipulator.

### 6.2.4 Case Study II: Shoulder Manipulator

In this Section, we employ a comprehensive approach combining free-body diagram analysis with verification through the virtual work method to analyze static forces in the shoulder manipulator detailed in Section 4.5.1. The architecture of the shoulder manipulator, illustrated in Figure 4.11, features a spatial orientation design supported by four identical limbs. Figure 6.4 provides a geometric model of the manipulator, highlighting key components such as the moving platform center point $P$, actuator lengths $l_i$, attachment points $P_i$ for the actuators, and fixed attachment points $A_i$.




▶ Long Description for Figure 6.4

**FIGURE 6.4**

(a) The geometry of shoulder manipulator, and (b) the free-body diagram of forces and moments acting on the moving platform and on each limb. ⏎

For analysis purposes, we establish two coordinate frames: $\{\mathscr{A}\}$ (attached to the fixed base at point $C$) and $\{\mathscr{B}\}$ (attached to the center of the moving platform $P$). Actuator forces are denoted by $\boldsymbol{\tau} = [f_1,\, f_2,\, f_3,\, f_4]^T$, while $\boldsymbol{f}_p$ represents the force acting on the passive limb. External forces and moments applied at point $P$ by the environment are represented by $\boldsymbol{f}$ and $\boldsymbol{n}$, respectively. Given the manipulator's spatial orientation with three rotational degrees of freedom, the resulting external wrench $\boldsymbol{\mathcal{F}}$ applied to the moving platform is a six-dimensional array encompassing both force and moment components, $\boldsymbol{\mathcal{F}} = [\boldsymbol{f},\, \boldsymbol{n}]$.

The analysis assumes that external forces are solely applied through the actuators, and no additional external forces act directly on the limbs. Consequently, the free-body diagrams for each limb indicate pure tension/compression forces along the limb axes $\hat{\boldsymbol{s}}_i$ for the active limbs and $\hat{\boldsymbol{s}}_p$ for the passive limb, respectively. This setup ensures that the static equilibrium conditions are rigorously analyzed to determine the distribution of forces within the manipulator system.

Next, we examine the free-body diagram depicting forces and moments applied to the moving platform, illustrated in

Figure 6.4. At static equilibrium, the sum of all forces acting on the moving platform must be zero, leading to the equation:

$$\boldsymbol{f} + \sum_{i=1}^{4} f_i \hat{\boldsymbol{s}}_i + f_p \hat{\boldsymbol{s}}_p = \mathbf{0}, \tag{6.14}$$

where $\boldsymbol{f}$ represents the external force applied to the moving platform by the environment. This equation allows us to ascertain how much of the external force is supported by the passive force component $\boldsymbol{f}_p = f_p \hat{\boldsymbol{s}}_p$, providing valuable insights for design considerations.

To determine the actuator forces, we focus on the moments about point $P$:

$$\boldsymbol{n} + \sum_{i=1}^{4} \boldsymbol{E}_i \times f_i \hat{\boldsymbol{s}}_i = \mathbf{0}, \tag{6.15}$$

where $\boldsymbol{n}$ denotes the external moment applied to the moving platform by the environment. In Figure 6.4, $\boldsymbol{E}_i$ represents the vector from point $P$ to the point of application $P_i$ of the $i$th actuator force. Notably, the passive force $\boldsymbol{f}_p = f_p \hat{\boldsymbol{s}}_p$ does not contribute to the moments about point $P$, hence it is absent in Equation 6.15. This equation

directly relates the actuator forces necessary to generate an external moment $\boldsymbol{n}$.

The vector $\boldsymbol{E}_i$ can be defined using previously established vectors ${}^{A}\boldsymbol{P}$ and ${}^{A}\boldsymbol{P}_i$ as:

$$ {}^{A}\boldsymbol{E}_i = {}^{A}\boldsymbol{P}_i - {}^{A}\boldsymbol{P}. \tag{6.16}$$

where ${}^{A}\boldsymbol{E}_i$ denotes the vector from point $P$ to $P_i$ in the fixed coordinate frame $\{\mathscr{A}\}$, crucial for calculating moments in Equation 6.15. This geometric relationship aids in accurately analyzing the static forces and moments within the shoulder manipulator system.

Therefore,

$$ \boldsymbol{\mathcal{f}} = \boldsymbol{n} = -\sum_{i=1}^{4} f_i \left[ \left( {}^{A}\boldsymbol{P}_i - {}^{A}\boldsymbol{P} \right) \times {}^{A}\hat{\boldsymbol{s}}_i \right]. \tag{6.17}$$

Substituting ${}^{A}\boldsymbol{P} = l_p \begin{bmatrix} s_2 & -c_2 s_1 & c_2 c_1 \end{bmatrix}^T$ from Equation 4.27 and utilizing the components of the vector ${}^{A}\boldsymbol{P}_i = {}^{A}\boldsymbol{R}_B {}^{B}\boldsymbol{P}_i = \begin{bmatrix} {}^{A}P_{i_x} & {}^{A}P_{i_y} & {}^{A}P_{i_z} \end{bmatrix}^T$ into Equation 6.16, we express this equation component-wise as follows:

$$
{}^{A}\boldsymbol{P}_i - {}^{A}\boldsymbol{P} = \begin{bmatrix} {}^{A}P_{i_x} - l_p s_2 \\ {}^{A}P_{i_y} + l_p c_2 s_1 \\ {}^{A}P_{i_z} - l_p c_2 c_1 \end{bmatrix}.
$$

(6.18)

Thus, Equation 6.17 can be reformulated in matrix form as:

$$
\begin{bmatrix} ({}^{A}P_{i_z} - l_p c_2 c_1)\,{}^{A}\boldsymbol{s}_{i_y} - ({}^{A}P_{i_y} + l_p c_2 s_1)\,{}^{A}\boldsymbol{s}_{i_z} \\ ({}^{A}P_{i_x} - l_p s_2)\,{}^{A}\boldsymbol{s}_{i_z} - ({}^{A}P_{i_z} - l_p c_2 c_1)\,{}^{A}\boldsymbol{s}_{i_x} \\ ({}^{A}P_{i_y} + l_p c_2 s_1)\,{}^{A}\boldsymbol{s}_{i_x} - ({}^{A}P_{i_x} - l_p s_2)\,{}^{A}\boldsymbol{s}_{i_y} \end{bmatrix}_{3\times 4} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \boldsymbol{\mathcal{F}}
$$

(6.19)

where the left matrix represents the Jacobian transpose of the shoulder manipulator given in Equation 5.48. This matrix is a $3 \times 4$ matrix, and for brevity, only its $i$th column is explicitly detailed in Equations 6.19 and 5.48. Hence, Equation 6.19 is consistent with the formulation given in the virtual work approach:

$$
\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau},
$$

(6.20)

Here, $\boldsymbol{\tau} = [f_1,\, f_2,\, f_3,\, f_4]^T$ is the vector of actuator forces, and $\boldsymbol{\mathcal{F}} = \boldsymbol{n}$ represents the wrench applied to the spatial orientation manipulator. This equation establishes the

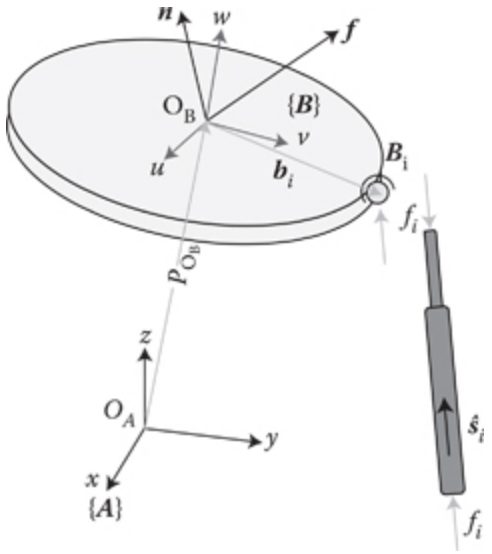
relationship between the forces applied by the actuators in joint space and the resulting wrench exerted on the moving platform in task space, through the transpose of the robot Jacobian matrix.

### 6.2.5 Case Study III: Stewart–Gough Platform

Consider the Stewart-Gough platform introduced in Section 4.6.1 and illustrated in Figure 4.16. This type of manipulator achieves spatial motion of its moving platform using six identical piston-cylinder actuators. Each actuator connects the fixed base to the moving platform through spherical joints located at points $A_i$ and $B_i$ for $i = 1, 2, \ldots, 6$ .

In Figure 4.16, a fixed frame $\{\mathscr{A}\}$ is fixed to the base at point $O_A$, with fixed attachment points represented by vectors $\boldsymbol{a}_i$. Furthermore, a mobile frame $\{\mathscr{B}\}$ is attached to the moving platform at point $O_B$, and the attachment points are denoted by vectors $\boldsymbol{b}_i$. The geometry of each limb is defined by its length $\ell_i$ and direction described by a unit vector $\hat{\boldsymbol{s}}_i$.

In Figure 6.5, the motion of the moving platform is described by a six dimensional twist rate $\dot{\boldsymbol{\chi}} = \left[{}^A\boldsymbol{v}_p, {}^A\boldsymbol{\omega}\right]^T$, where ${}^A\boldsymbol{v}_p$ represents the velocity of point $O_B$, and ${}^A\boldsymbol{\omega}$ denotes the angular velocity of the moving platform.




▶ Long Description for Figure 6.5

**FIGURE 6.5**
Free-body diagram of forces and moments acting on the moving platform and each limb of the Stewart–Gough platform. ⏎

Additionally, consider the external wrench exerted by the manipulator and applied to the environment at point $O_B$, denoted as $\boldsymbol{\mathcal{F}} = [\boldsymbol{f}, \boldsymbol{n}]^T$. Here, $\boldsymbol{f}$ indicates the force generated by the moving platform and applied to the environment at point $O_B$, while $\boldsymbol{n}$ represents the moment applied to the environment. For clarity and due to the uniformity of the limbs, only one representative limb is depicted in Figure 6.5.

Firstly, consider the external forces applied to each limb of the Stewart-Gough platform. It is assumed that no external forces other than the actuator forces are applied to the limbs. Therefore, the static force along each limb

axis $\hat{\boldsymbol{s}}_i$ can be represented by a tension/compression force $f_i$. Moving on to the free-body diagram of forces and moments applied to the moving platform, at static equilibrium, the sum of all forces acting on the platform must be zero. Thus, we have:

$$-\boldsymbol{f} + \sum_{i=1}^{6} f_i \hat{\boldsymbol{s}}_i = \mathbf{0},$$

(6.21)

where $-\boldsymbol{f}$ represents the external force applied to the moving platform by the environment. Furthermore, considering the moments about point $O_B$:

$$-\boldsymbol{n} + \sum_{i=1}^{6} \boldsymbol{E}_i \times f_i \hat{\boldsymbol{s}}_i = \mathbf{0},$$

(6.22)

where $-\boldsymbol{n}$ denotes the external moment applied to the moving platform by the environment, and $\boldsymbol{b}_i$ is the position vector from point $O_B$ to attachment point $\boldsymbol{B}_i$ on the moving platform.

Combining Equations 6.21 and 6.22 into a matrix form yields:

$$\begin{bmatrix} \hat{\boldsymbol{s}}_1 & \hat{\boldsymbol{s}}_2 & \cdots & \hat{\boldsymbol{s}}_6 \\ \boldsymbol{b}_1 \times \hat{\boldsymbol{s}}_1 & \boldsymbol{b}_2 \times \hat{\boldsymbol{s}}_2 & \cdots & \boldsymbol{b}_6 \times \hat{\boldsymbol{s}}_6 \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_6 \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{n} \end{bmatrix}.$$

(6.23)

Comparing the matrix on the left-hand side of Equation 6.23 with the Jacobian matrix of the Stewart-Gough platform given in Equation 5.56, it is evident that this matrix is equivalent to the transpose of the Jacobian matrix of the manipulator. Therefore, Equation 6.23 can be expressed by the virtual work formulation as:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau},$$

(6.24)

where $\boldsymbol{\tau} = [f_1, f_2, \ldots, f_6]^T$ represents the vector of actuator forces, and $\boldsymbol{\mathcal{F}} = [\boldsymbol{f}, \boldsymbol{n}]^T$ denotes the six-dimensional wrench exerted by the manipulator on the environment.

## 6.3 Jacobian Matrix: Intrinsic Characteristics

Thus far, we have thoroughly examined the profound relationship between the Jacobian matrix and its mapping

properties within both velocity and force fields. The Jacobian matrix serves as a fundamental tool in understanding how motion and forces are transferred within a robotic system. Its intrinsic properties are essential for the optimal design of a robot's kinematic structure, making it a focal point in numerous research studies. This section focuses deeper into these intrinsic characteristics, aiming to uncover the intricacies that contribute to the overall performance and efficiency of robotic systems.

As previously mentioned, the optimal design of robotic manipulators frequently draws inspiration from the biological musculoskeletal systems found in nature. These natural designs often incorporate redundancy in actuation to achieve several goals: maximizing accessibility to different positions and orientations, optimizing the speed of movements, and improving the distribution of forces applied to the environment. By studying these biological systems, engineers can gain valuable insights into creating more adaptable and efficient robotic structures.

To translate the structural characteristics observed in nature into practical applications, it is essential to establish clear connections between physical concepts and quantifiable parameters. This task is challenging, as it involves formulating criteria that comprehensively encompass all desired features for the optimal design of a robot. Such criteria must consider a variety of factors,

including kinematic performance, force distribution behavior, and the ability to handle various tasks and loads.

Consequently, researchers, inspired by the ingenuity of natural systems and interpreting specific aspects of their design, have developed a range of criteria grounded in the singularity characteristics of the Jacobian matrix. Each of these criteria addresses a particular aspect of robotic performance, such as dexterity, manipulability or force distribution. By focusing on singularities, researchers can identify and avoid configurations that compromise the robot's effectiveness.

In the upcoming sections, we will explore the mathematical definitions of some of these criteria and metrics in greater detail. This exploration will include a discussion of how these criteria can be applied to evaluate and enhance the design of robotic systems. By understanding and leveraging the intrinsic properties of the Jacobian matrix, we aim to develop robots that are not only more capable and efficient but also more resilient and adaptable to a wide range of tasks and environments.

### 6.3.1 Dexterity Ellipsoid

Given the robot's structure and its specific configuration $\boldsymbol{\chi}$, we can construct the Jacobian matrix to linearly map the wrench applied to the moving platform $\boldsymbol{\mathcal{F}}$, from the active joint force/torque vector by transpose the Jacobian matrix, $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$. By treating this mapping as a linear system with the active joint force/torque vector as the input and the

moving platform wrench as the output, one can analyze its characteristics, such as gain, to describe its behavior.

This linear mapping is crucial for understanding how forces and torques are transmitted through the robotic system, providing insights into the efficiency and effectiveness of force distribution. By studying the gain of the system, one can formulate strategies to either increase or decrease the magnitude of the applied wrench on the moving platform relative to the actuator forces/torques. This is particularly important for tasks requiring precise control of force and torque, such as delicate manipulation or high-force applications.

It is important to acknowledge that this mapping involves multiple variables, including the geometric configuration of the robot, the properties of the actuators, and the external loads acting on the system. To define the system's gain, one can rely on the Euclidean norms of the input and output arrays. This approach provides a scalar measure of the magnitude of the forces and torques, allowing for a straightforward comparison and analysis.

Moreover, the singular values of the Jacobian matrix can provide additional insights into the system's performance. The singular values indicate the amplification or attenuation of the input forces and torques as they are mapped to the output wrench. By examining these values, one can identify configurations where the system is particularly efficient or prone to instability.

To better illustrate the idea, let us consider a scenario where all the robot actuators apply uniform force/torque with a magnitude less than or equal to one. In this case, the Euclidean norm of the *n*-dimensional vector $\boldsymbol{\tau}$ is constrained to be less than or equal to one.

$$\| \boldsymbol{\tau} \|_2^2 = \tau_1^2 + \tau_2^2 + \cdots + \tau_n^2 \leq 1$$

(6.25)

If the Jacobian matrix of the robot is square and the robot is not in a singular configuration, using the inverse of the Jacobian matrix transpose relates the actuator force/torques to the moving platform wrench.

$$\boldsymbol{\tau} = \boldsymbol{J}^{T^{-1}} \boldsymbol{\mathcal{F}}$$

(6.26)

In the case where the robot exhibits redundancy in its actuation, solving the inverse force problem can be accomplished using the *pseudo-inverse* of the Jacobian matrix transpose.

$$\boldsymbol{\tau} = \boldsymbol{J}^{T^{\dagger}} \boldsymbol{\mathcal{F}}$$

(6.27)

In general, the relationship between the Euclidean norms of actuator force/torque vector and the moving platform

wrench can be calculated as follows.

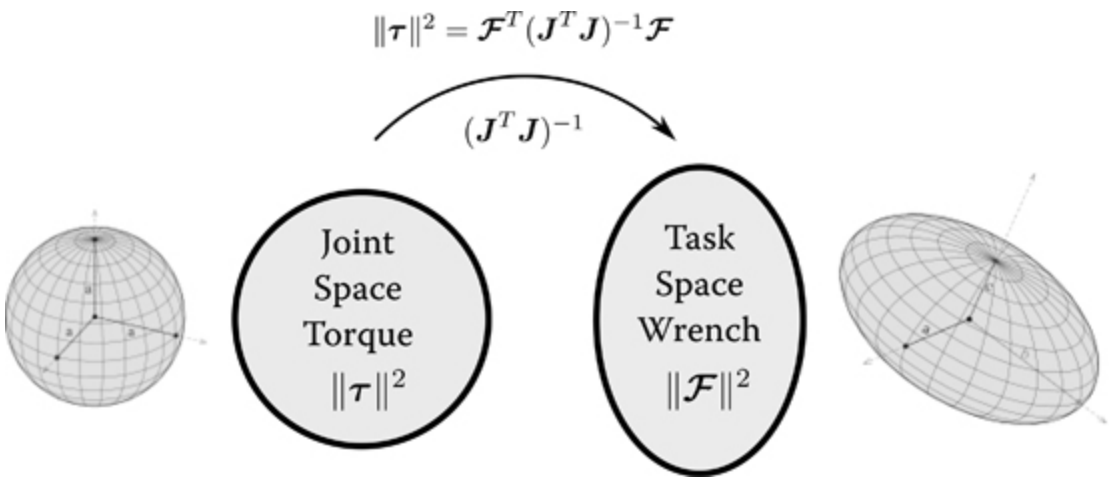
$$\begin{aligned} \| \boldsymbol{\tau} \|^2 = \boldsymbol{\tau}^T \boldsymbol{\tau} &= [\boldsymbol{J}^{T^\dagger} \boldsymbol{\mathcal{F}}]^T [\boldsymbol{J}^{T^\dagger} \boldsymbol{\mathcal{F}}] \\ &= \boldsymbol{\mathcal{F}}^T \boldsymbol{J}^\dagger \boldsymbol{J}^{T^\dagger} \boldsymbol{\mathcal{F}} \\ &= \boldsymbol{\mathcal{F}}^T (\boldsymbol{J}^T \boldsymbol{J})^\dagger \boldsymbol{\mathcal{F}} \end{aligned}$$

Considering that the matrix $\boldsymbol{J}^T \boldsymbol{J}$ is always square for any type of robot, regardless of being underactuated or redundant, this relationship simplifies to the following form.

$$\begin{aligned} \| \boldsymbol{\tau} \|^2 &= \boldsymbol{\mathcal{F}}^T (\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{\mathcal{F}} \\ &\leq \frac{1}{\| \boldsymbol{J}^T \boldsymbol{J} \|} \cdot \| \boldsymbol{\mathcal{F}} \|^2 \end{aligned} \tag{6.28}$$

This equation quantitatively represents the relationship between the moving platform wrench and the active joint force/torque vector of the robot. If the matrix $\boldsymbol{J}^T \boldsymbol{J}$, which is called the “Dexterity Matrix,” is full rank, and its inverse exists and is bounded, the relationship between the Euclidean norms of active joint force/torque vector and the moving platform wrench is given by a weighted norm along the ellipsoidal direction, as defined by this equation. This representation is referred to as the “Dexterity Ellipsoid” of the robot.

According to Figure 6.6, in the case where the robot actuator force/torque are uniform and lie within an *n*-dimensional unit sphere ($\| \boldsymbol{\tau} \|^2 = 1$), the six-dimensional array representing the moving platform wrench will be ellipsoidal. Along the ellipsoid longitudinal axis, this mapping has the maximum magnitude, while along its transverse axis, it has the minimum magnitude, all while maintaining a unit volume.




▶ Long Description for Figure 6.6

**FIGURE 6.6**
The manipulability ellipsoid of a parallel robot. ⏎

In this way, the ellipsoidal dexterity not only determines the relationship between the maximum and minimum sizes of the input and output arrays of this mapping but also specifies the directions in which the moving platform wrench components reach their maximum or minimum values.

### 6.3.2 From Singularity to Isotropy

The dexterity ellipsoid in parallel robots delineates crucial features within the robot's Jacobian matrix. In contrast to the Jacobian matrix, which is limited to being square only for fully actuated robots, the dexterity matrix $\boldsymbol{J}^T\boldsymbol{J}$ for diverse robots—encompassing underactuated, fully actuated, and even redundantly actuated cases—is invariably square. Therefore, the attributes of the dexterity ellipsoid can be extended to various types of parallel robots.

Now, let us explore the relationship between the determinant of the dexterity matrix and its eigenvalues. We know that the determinant of any square matrix is equal to the product of its eigenvalues. Therefore, in the dexterity matrix,

$$\det\left(\boldsymbol{J}^T\boldsymbol{J}\right) = \lambda_1 \cdot \lambda_2 \cdots \lambda_n \tag{6.29}$$

The size of the dexterity matrix is fixed regardless of the type of manipulator. For planar robots, $n = 3$, while for robots with spatial motion in three-dimensional space, $n = 6$.

One of the key features of the dexterity ellipsoid is captured by the determinant of this matrix. The determinant provides a measure of the volume of the ellipsoid, which in turn reflects the manipulator's ability to

manipulate the object in various directions. A larger determinant indicates a higher degree of dexterity, suggesting that the robot can apply a wide range of wrenches with ease.

As observed, this characteristic is directly related to the product of the eigenvalues of the matrix. Each eigenvalue represents a principal axis of the ellipsoid, and their product gives the volume. Thus, the determinant effectively encapsulates the overall dexterity of the manipulator, linking geometric properties of the ellipsoid with the robot's performance capabilities. Understanding this relationship is crucial for optimizing the design and control of robotic systems, ensuring they can achieve the desired movements and tasks efficiently.

For a deeper understanding, consider two limiting cases to define the spectrum influenced by this feature. In the first limit, assume that the determinant of the dexterity matrix $(\boldsymbol{J}^T\boldsymbol{J})$ becomes zero in a specific configuration $\boldsymbol{\chi}$. In this case, the robot is in a singular configuration, and according to Equation 6.29, at least one of the eigenvalues of the dexterity matrix is zero. In this limiting case, the ellipsoidal dexterity is stretched along the eigenvector corresponding to the zero eigenvalue, turning into a cylinder, as depicted in Figure 6.7. The physical interpretation of this limit in the dexterity ellipsoid aligns directly with the characteristics of a singular configuration.

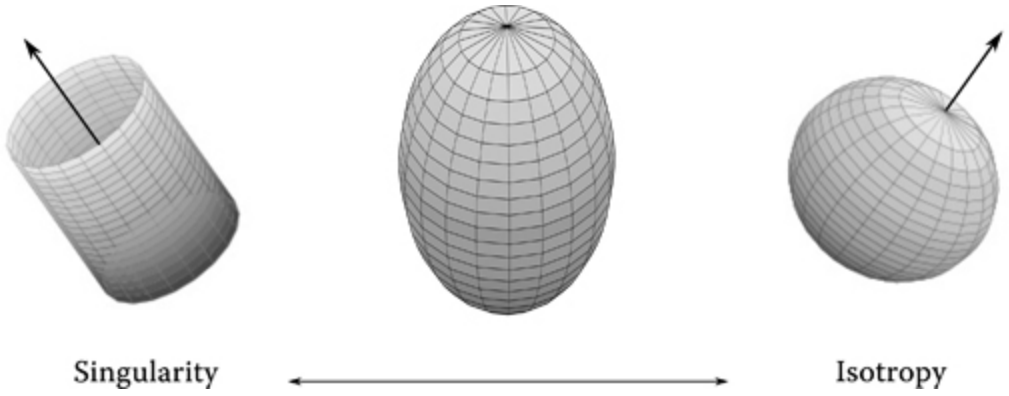



**FIGURE 6.7**
Dexterity ellipsoid of the parallel robot: From singularity to isotropy. ⏎

Considering that in Equation 6.28 the inverse of the dexterity matrix $\boldsymbol{J}^T\boldsymbol{J}$, is multiplied by the norm of the moving platform wrench, when the determinant of this matrix becomes zero, its inverse becomes infinite along the eigenvector corresponding to the zero eigenvalue. This means that to produce an infinitesimal wrench along this eigenvector requires infinitely large actuator force/torques. Therefore applying such wrench is not possible, and the robot looses its controllability in this direction.

Now consider the second limiting case where all the eigenvalues of the dexterity matrix in a specific configuration $\boldsymbol{\chi}$ are equal to one, and therefore, according to Equation 6.29, the determinant of the dexterity matrix $(\boldsymbol{J}^T\boldsymbol{J})$ will be equal to one. In this case, the dexterity ellipsoid transforms into an *n*-dimensional hypersphere with unit dimensions in all directions. This limiting case is

also depicted in Figure 6.7 and is referred to as an *isotropic* configuration.

The physical interpretation of this limiting case in dexterity ellipsoid is in contrast to the singularity configuration of the robot. In this limit, applying equal force/torques to all actuators results in uniform wrench in the moving platform. In other words, in an isotropic state, the ability to apply wrench in all direction of the robot's moving platform is uniformly possible by the actuators. This uniformity feature is highly advantageous as it maximizes the robot's dexterity in applying force.

When a robot is in an isotropic configuration, it can efficiently and effectively exert force and torque in any direction without bias or restriction. This optimal state enhances the robot's performance, making it more versatile and capable in executing a wide range of tasks. By ensuring that the ability to apply force is uniformly distributed, the robot can operate at peak efficiency, with the dexterity ellipsoid providing a clear geometric representation of this ideal condition.

Note that the dexterity matrix is a function of the robot and its moving platform pose $\boldsymbol{\chi}$. Therefore, features like singularity or isotropy may not occur universally but rather may only be observed in specific, limited situations within the robot's workspace. Despite this, a spectrum between singularity and isotropy will exist at different points in the workspace, reflecting varying degrees of dexterity and performance.

In the optimal design of robots inspired by nature, there is a strong inclination toward achieving isotropy. Designers aim for a uniform distribution among the eigenvalues of the dexterity matrix, ensuring that the moving platform wrench components are evenly distributed in different directions. This approach helps the robot avoid singularities within its workspace, enabling it to perform dexterous and agile maneuvers while maintaining the capability to apply the desired wrench to the environment.

By striving for isotropy, the robot's design ensures that it can operate efficiently and effectively across a wide range of tasks. This uniformity in dexterity enhances the robot's adaptability, allowing it to handle various operational demands without encountering performance bottlenecks. Ultimately, this leads to a more robust and versatile robotic system, capable of executing complex and precise movements with ease.

### 6.3.3 Dexterity Measure

As mentioned in the previous section, the characteristics of the dexterity matrix of a parallel robot, $\boldsymbol{J}^T\boldsymbol{J}$, can vary across a wide spectrum of states, from isotropic conditions to singularities, in different configurations. Various metrics have been defined in the field of robotics to quantitatively express different aspects of the manipulability matrix. Among them, the “Dexterity Measure” has gained more popularity in robotics literature [149, 150]. In the following, we elaborate on an examination of this metric.

The determinant of the dexterity matrix, according to Equation 6.29, is obtained from the product of the eigenvalues of this matrix. However, the eigenvalues of the dexterity matrix $\boldsymbol{J}^T\boldsymbol{J}$ are directly related to the singular values of the Jacobian matrix itself, as expressed in the following equation.

$$\sigma_i(\boldsymbol{J}) = \sqrt{\lambda_i(\boldsymbol{J}^T\boldsymbol{J})}$$

(6.30)

This relationship holds even for non-square Jacobian matrices. Thus, without the need to compute the product of the Jacobian matrix and its transpose $\boldsymbol{J}^T\boldsymbol{J}$, the dexterity measure can be determined by calculating the singular values of the Jacobian matrix itself.

With this introduction, the dexterity measure $\mu$ is defined, based on the singular values of the Jacobian matrix $\boldsymbol{J}$, as follows.

$$\begin{aligned}\mu &= \sqrt{\det\left(\boldsymbol{J}^T\boldsymbol{J}\right)} = \sqrt{\lambda_1 \cdot \lambda_2 \cdots \lambda_n} \\ &= \sigma_1 \cdot \sigma_2 \cdots \sigma_n\end{aligned}$$

(6.31)

where $\lambda_i$ are the eigenvalues of the dexterity matrix $\boldsymbol{J}^T\boldsymbol{J}$, and $\sigma_i$ are the singular values of the Jacobian matrix $\boldsymbol{J}$. Since the Jacobian matrix depends on the robot's structure

and configuration, the defined dexterity measure is a function of the robot's structure and configuration, as well. If the dexterity measure $\mu$ tends toward one, the robot moves toward its isotropic configuration.

On the other hand, when the dexterity measure $\mu$ tends toward zero, at least one of the singular values of the Jacobian matrix approaches zero. This occurs when the robot is close to its singular configuration. In this case, the rank of the Jacobian matrix becomes smaller than the number of degrees of freedom, i.e., $rank(\boldsymbol{J})) < n$. Thus, avoiding singularity is manifested by keeping this measure away from zero. In redundantly actuated robots, due to the added column to the Jacobian matrix, the tendency towards singularity is reduced, and essentially, robots have a better dexterity measure. This has been qualitatively demonstrated in examples of musculoskeletal structures studied earlier in this chapter.

In addition to the eigenvalues of the dexterity matrix, the eigenvectors of this matrix also play an important role in the analysis of the dexterity ellipsoid of a robot. To analyze this feature in fully actuated robots, where the Jacobian matrix is square, it is not necessary to directly calculate the eigenvectors of the manipulability matrix $\boldsymbol{J}^T\boldsymbol{J}$. This is because the eigenvectors of the dexterity matrix can be obtained by decomposing the singular value matrix of the Jacobian.

Let's reconsider the relationship between the norms of the active joint force/torque and the applied wrench by the

moving platform as expressed in Equation 6.28. The dexterity measure provides an upper bound to the dexterity matrix norm, hence:

$$\begin{aligned} \| \boldsymbol{\tau} \|^2 &= \boldsymbol{\mathcal{F}}^T (\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{\mathcal{F}} \\ &\leq \frac{1}{\| \boldsymbol{J}^T \boldsymbol{J} \|} \cdot \| \boldsymbol{\mathcal{F}} \|^2 \\ &\leq \frac{1}{\mu^2} \cdot \| \boldsymbol{\mathcal{F}} \|^2 \end{aligned}$$

(6.32)

This inequality establishes bounds on the magnitude of the moving platform wrench based on the manipulability measure $\mu$:

$$\| \boldsymbol{\mathcal{F}} \| \geq \mu \cdot \| \boldsymbol{\tau} \|$$

(6.33)

The physical interpretation of these relationships in singular and isotropic configurations is insightful. In a singular configuration, where $\mu = 0$, certain directions within the robot's workspace result in a zero-moving platform wrench. This indicates that the robot cannot apply wrench in those directions, thus limiting its controllability and operational capability.

Conversely, in an isotropic configuration where $\mu$ approaches unity, applying uniform force/torque across the

actuators leads to a uniform moving platform wrench in all directions. This scenario signifies that the robot possesses optimal dexterity, enabling it to manipulate objects effectively in every direction within its workspace.

These concepts illustrate how the manipulability measure $\mu$ serves as a critical indicator of the robot's operational flexibility and capability. By understanding and optimizing $\mu$, engineers can design robots that maximize their ability to perform a wide range of tasks efficiently and precisely, while ensuring robustness and reliability in diverse operational scenarios.

In addition to the magnitude of forces, their directions, interpreted by the eigenvectors of the dexterity matrix, play a crucial role in the analysis of the dexterity ellipsoid. For this purpose, consider the singular value decomposition of the Jacobian matrix $\boldsymbol{J}$ with dimensions $n \times m$ as follows:

$$\boldsymbol{J} = \boldsymbol{U}\, \boldsymbol{\Sigma}\, \boldsymbol{V}^T \tag{6.34}$$

where matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are given by

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots \boldsymbol{u}_n \end{bmatrix}_{n \times n} \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots \boldsymbol{v}_m \end{bmatrix}_{m \times m} \tag{6.35}$$

are orthogonal matrices, and the matrix $\boldsymbol{\Sigma}$ can be expressed in terms of the Jacobian matrix singular values as:

$$\boldsymbol{\Sigma} = \left[\begin{array}{cccc} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{array}\right]_{m \times n}$$

(6.36)

Efficient numerical methods have been developed for the decomposition of singular value matrices and are employed in engineering software tools[1].

Suppose the maximum singular value of the Jacobian matrix in a specific configuration is denoted as $\sigma_{\max}$ along the direction $\boldsymbol{u}_{\max}$, and similarly, the minimum singular value is denoted as $\sigma_{\min}$ along the direction $\boldsymbol{u}_{\min}$. It can be demonstrated that these directions are aligned with the eigenvectors corresponding to the maximum and minimum singular values of the dexterity matrix.

$$\boldsymbol{u}_{\max} = \boldsymbol{e}_{\max}(\boldsymbol{J}\boldsymbol{J}^T) \quad \boldsymbol{u}_{\min} = \boldsymbol{e}_{\min}(\boldsymbol{J}\boldsymbol{J}^T)$$

(6.37)

In this context, $\boldsymbol{e}_{\max}$ represents the eigenvector of the dexterity matrix $\boldsymbol{J}\boldsymbol{J}^T$ corresponding to its largest eigenvalue, while $\boldsymbol{e}_{\min}$ corresponds to the eigenvector associated with the smallest eigenvalue. Consequently, the maximum and minimum wrench values exerted on the

robot's moving platform can be determined along the directions defined by these eigenvectors.

$$\boldsymbol{\mathcal{F}}_{\max} = |\sigma_{\max}| \ \| \boldsymbol{\tau} \| \rightarrow \ \text{along} \ \boldsymbol{u}_{\max} \tag{6.38}$$

$$\boldsymbol{\mathcal{F}}_{\min} = |\sigma_{\min}| \ \| \boldsymbol{\tau} \| \rightarrow \ \text{along} \ \boldsymbol{u}_{\min} \tag{6.39}$$

In this way, not only the bounds of maximum and minimum wrench of the robot's moving platform can be determined, but also the directions of maximum and minimum dexterity ellipsoid axes can be obtained using these relationships.

The importance of utilizing the dexterity measure in the quantitative and qualitative analysis of the features of the dexterity matrix has been emphasized in these equations. Note that for determining the maximum and minimum singular values and their corresponding directions, it is sufficient to perform the singular value decomposition of the Jacobian matrix, and there is no need to carry out these calculations on the more complex dexterity matrix $\boldsymbol{J}^T \boldsymbol{J}$.

### 6.3.4 Manipulability Measure

The dexterity measure encompasses all the singular values of the Jacobian matrix, which are associated with the dexterity ellipsoid in the robot. However, as this measure is defined as the product of all the singular values of the

Jacobian matrix, it is typically not dimensionless. Its value can exceed one, and at times, it could become significantly large. Additionally, assessing isotropy with this measure faces the challenge that, while individual singular values may not be unitary, their distribution might result in their product equaling one. In such a scenario, even though the dexterity measure is equal to one, as anticipated in an isotropic configuration, a uniform distribution in the robot's moving platform wrench components may not be present.

To address these issues, a manipulability measure has been proposed in the robotic literature to assess the manipulability status of the robot. This measure is based on the condition number of the Jacobian matrix and can be utilized for underactuated, fully actuated, and redundant robots. The condition number of a matrix is defined as the ratio of its maximum singular value to its minimum singular value.

$$\text{Condition Number of } \boldsymbol{J} = \frac{\sigma_{\max}(\boldsymbol{J})}{\sigma_{\min}(\boldsymbol{J})} \tag{6.40}$$

[1]Such as the svd command in MATLAB ⏎

In general, the condition number of a matrix can take values from 1 to infinity. If the condition number is close to 1, the singular values of the matrix are close to each other, indicating a near-isotropic condition. As the condition

number increases, the ratio of the maximum singular value to the minimum singular value grows, and in the limit where the matrix becomes singular, the condition number approaches infinity. The use of the matrix condition number in linear algebra reduces numerical computation errors on matrices. When a matrix has a large condition number, appropriate numerical methods must be employed in calculations involving the matrix inverse and its counterparts to maintain accuracy [151].

With this explanation, to define a suitable dimensionless measure for assessing robot dexterity that ranges between zero and one, the reciprocal of the matrix condition number of the Jacobian matrix is utilized. The manipulability measure in a robot is defined as the inverse of the Jacobian matrix condition number in each configuration.

$$\kappa = \frac{\sigma_{\min}(\boldsymbol{J})}{\sigma_{\max}(\boldsymbol{J})} \tag{6.41}$$

In this definition, $\sigma_{\min}(\boldsymbol{J})$ and $\sigma_{\max}(\boldsymbol{J})$ represent the minimum and maximum singular values of the Jacobian matrix of the robot, respectively. With this definition, the manipulability measure $\kappa$ is a dimensionless quantity ranging between zero and one. A value close to zero indicates a singular configuration, while a value close to one signifies an isotropic condition in the robot.

The manipulability measure $\kappa$ varies across the workspace due to its dependency on the robot's structural design and configuration $\boldsymbol{\chi}$. This measure, like other dexterity metrics, provides crucial insights into the robot's operational capabilities. By understanding the kinematic structure of the robot, engineers can compute the Jacobian matrix and subsequently determine the manipulability measure at any given point in the workspace.

A well-designed robot exhibits regions within its workspace where the manipulability measure $\kappa$ approaches unity, indicating optimal dexterity. These regions are characterized by the robot's ability to efficiently maneuver and apply forces in multiple directions. Conversely, near singular configurations, if they exist, the manipulability measure tends towards zero. This signifies that in these critical points, the robot's ability to effectively perform tasks may be severely restricted, requiring careful consideration during design and operational planning.

Understanding the distribution of manipulability across the workspace allows for strategic placement of tasks and operations. Engineers can optimize the robot's performance by focusing on areas where manipulability is high, ensuring robustness and versatility in task execution. This approach not only enhances the robot's operational efficiency but also minimizes the risk of encountering performance limitations near singularities.

In addition to characterizing isotropy and singularity, the manipulability measure also reveals insights into the

directions of maximum and minimum dexterity through the eigenvectors associated with the maximum and minimum singular values, denoted as $\boldsymbol{u}_{\max}$ and $\boldsymbol{u}_{\min}$. While the definition of the manipulability measure does not explicitly account for all ellipsoidal axes unlike some other dexterity metrics, these additional directions can still be discerned using the singular value decomposition of the Jacobian matrix.

The vectors $\boldsymbol{u}_{\max}$ and $\boldsymbol{u}_{\min}$ represent the principal directions in which the robot can achieve maximum and minimum manipulability, respectively. By analyzing these directions alongside the manipulability measure $\kappa$, engineers gain a comprehensive understanding of how the robot's performance varies across its workspace. This knowledge aids in optimizing the robot's design and operational strategies to ensure efficient and effective task execution.

Utilizing the singular value decomposition allows for a detailed examination of the Jacobian matrix's structure, revealing critical information about the robot's capabilities beyond simple isotropy and singularity considerations. This approach enhances the ability to predict and enhance the robot's performance across diverse operational scenarios, ensuring robustness and adaptability in real-world applications.

### 6.3.5 Global Measures

As we have seen, dexterity measures in robots are functions of the robot's structure and its configuration $\boldsymbol{\chi}$, changing from one point to another in the workspace. If we intend to use these measures to optimize the kinematic structure of a robot and determine optimal kinematic parameters, we need to examine the condition of these measures across the entire workspace, instead of focusing on specific points. To address this, global performance measures have been introduced in the robotics literature [152], defined by averaging the desired measure over the entire robot's workspace.

In calculating global performance measures, the integral of the relevant measure is obtained over the entire workspace, while divided by the volume of the workspace. For example, if we consider the manipulability measure, the global manipulability measure $G\kappa$ is defined as follows.

$$G\kappa = \frac{A}{B} \tag{6.42}$$

where,

$$A = \int_W \kappa \, dW, \quad B = \int_W dW \tag{6.43}$$

In this definition, $A$ is computed by integrating the desired metric over the entire robot workspace $W$, and $B$ determines the volume of the robot's workspace. The average manipulability index is a measure of the robot's point-to-point manipulability throughout the entire workspace. Given that the manipulability index itself is dimensionless and lies in the range between zero and one, the global manipulability measure inherits these characteristics.

A similar definition can be applied to other global measures; it is sufficient to substitute the desired metric value for $\kappa$ in Equation 6.43. If the metric of interest is dimensionless, its global measure will also be dimensionless.

Furthermore, in some applications, it is not necessary to compute the metric over the entire workspace; instead, if the goal is for the robot to exhibit dexterity in a specific region of the workspace, compute the metric of interest only in that particular region. In practical applications, when performing numerical computations to evaluate global measures, it is common to discretize the desired workspace into a uniform grid of points. The summation of the metric at each grid point in the area of interest can then be employed as an alternative to integration.

By defining the global measure, it can be utilized in optimizing the desired robot structure and determining optimal parameters. In simpler structures, finding the optimal point can be achieved by setting the derivative of

the global measure with respect to the design parameters $\boldsymbol{h}$ to zero.

$$\frac{\partial G\kappa}{\partial \boldsymbol{h}} = 0 \tag{6.44}$$

In complex structures where the number of design parameters increases, numerical optimization methods can be employed.

## 6.4 Stiffness Analysis

In numerous practical applications, the moving platform of a parallel manipulator often interacts with a rigid environment, exerting wrenches upon it. As demonstrated in the preceding section, the Jacobian transpose serves as a projection map linking the wrench applied to the environment to the actuator forces responsible for generating this wrench. This relationship is pivotal for understanding how external forces influence the manipulator's performance.

In this section, our focus shifts to the deflections experienced by the manipulator's moving platform in response to these applied wrenches. These deflections are influenced by both the magnitude of the applied wrench and the structural stiffness of the manipulator itself. Thus,

the stiffness characteristics of a manipulator play a critical role in determining its overall positioning accuracy, particularly when the moving platform is in contact with an object within its workspace.

When analyzing robot stiffness, we assess how the applied wrench at the robot's moving platform correlates with the resulting displacement. When the robot interacts statically with its environment, and flexibility inherent in the robot's components can lead to slight deformations in the moving platform. This displacement, which varies with the robot's structure and configuration, directly affects its accuracy and precision. Therefore, a thorough examination of these deflections is essential for optimizing the robot's performance in various applications. However, it's important to recognize that compliance, or flexibility, in robots is not always undesirable; in fact, it can be advantageous in certain scenarios, especially those involving human-robot interactions where safety and adaptability are paramount.

The factors contributing to compliance in robots are diverse and multifaceted. In the following sections, we will examine some of the primary factors that contribute to compliance in robot systems, highlighting the implications for design and operational considerations.

**Robot Structure:** The kinematic structure of a robot significantly influences its stiffness and compliance characteristics. Opting for open kinematic chains extends the robot's operational range, allowing for

greater workspace flexibility. However, this design choice often introduces challenges such as increased compliance and higher tracking errors during motion. In contrast, adopting closed kinematic chains enhances the robot's stiffness, enabling more robust force application on the moving platform. The assessment of how the robot's structure impacts stiffness can be comprehensively explored by analyzing its Jacobian matrix, providing insights into how different structural configurations affect performance metrics like dexterity and operational manipulability.

**Links:** Industrial robots commonly employ rigid links to achieve precise motion control, prioritizing accuracy and stability in manufacturing processes. However, this approach often results in increased overall robot weight, which can be disadvantageous in weight-sensitive applications such as space and cable-driven robotics. In these scenarios, engineers often opt for lightweight and long links to mitigate weight and workspace constraints while inherently introducing higher compliance and enhancing the robot's structural flexibility. This strategic choice balances the need for precision with the operational demands of the required application, ensuring optimal performance in challenging environments where weight efficiency is paramount.

**Joints:** Robot joints are usually engineered to be sufficiently rigid to reduce unwanted oscillations during

the robot's movement, ensuring precise operation in industrial settings. However, in applications involving human-robot interaction, a different approach is often adopted. In these contexts, introducing compliance into robot joints is advantageous as it enhances the robot's ability to interact safely and smoothly with humans. Consequently, robots designed for collaborative tasks or service applications commonly incorporate compliant joints. This design choice not only improves flexibility but also supports safer and more natural interactions between robots and their human counterparts, promoting efficiency and user acceptance in various operational environments.

**Sensors and Actuators:** Actuators and sensors play a crucial role in contributing to compliance within the robot's structure. Hydraulic actuators and direct-drive systems are inherently compliant due to their mechanical design and operational characteristics. Similarly, many force sensors incorporate compliant elements as part of their sensing mechanisms, further influencing the overall compliance of the robot. These components are essential in enabling precise force control and feedback in robotic applications but can introduce flexibility and compliance into the system, affecting its overall stiffness and responsiveness to external forces.

**Power Transmission:** The adoption of harmonic drive power transmission systems is a significant contributor to compliance in robot joints. Harmonic drives are renowned for their high efficiency and compact design, making them popular choices in industrial robotics. These gearboxes utilize a specialized gear known as the “Flexspline,” which inherently exhibits compliant characteristics. The exceptional transmission ratio achieved by harmonic drives is largely attributed to this flexible element within the system.

As a consequence of employing harmonic drives, robot joints can exhibit compliance due to the flexibility of the Flexspline. This compliance directly influences the overall stiffness and performance of the robot, affecting its ability to accurately handle forces and motions in diverse operational environments. Understanding the implications of harmonic drive systems on robot compliance is crucial for optimizing robot design and performance across various industrial applications.

**Control System:** Apart from mechanical components, sensors, and actuators, the robot's control system significantly influences its stiffness and compliance characteristics. Traditional control architectures in robots employ high control gains to achieve precise tracking and effective disturbance rejection in system outputs. These high gains contribute to increased stiffness in the robot's response, enhancing its ability to

maintain positional accuracy and control under various operational conditions.

However, in scenarios involving human-robot interaction, excessive stiffness can be counterproductive, potentially compromising safety and efficiency. In such cases, reducing control gains becomes imperative to adjust the robot's compliance. By lowering control gains, the robot's responsiveness can be moderated, allowing for smoother interactions and safer operation in dynamic environments where human presence or collaboration is crucial. This adaptive control strategy ensures that the robot can dynamically balance between stiffness and compliance based on the specific operational requirements and safety considerations at hand.

As described, the stiffness of a manipulator depends on several factors [153]. One of the most important factors influencing the robot's stiffness is the structure of the manipulator. Utilizing closed-kinematic chains in the robot's design significantly enhances stiffness and improves positioning accuracy, which is a key advantage of parallel robots over their serial counterparts.

Conversely, the actuated joints and links of parallel robots are usually the major sources of manipulator compliance, especially in space robots and cable-driven manipulators that employ long and lightweight links. This is because the actuators, transmissions, and possibly flexible elements used for force and torque transducers contribute

significantly to the compliance of the limbs. For the purposes of this discussion, we assume that the primary sources of manipulator compliance are the actuators and their transmission systems.

In the upcoming sections, we will assess the stiffness of the parallel robots and its relation to robot compliance. Additionally, we will investigate the extent of deflections occurring in the robot's moving platform when such compliance is present in the robot structure.

### 6.4.1 Stiffness Matrix

The actuators of a typical parallel manipulator are composed of either hydraulic piston-cylinders or electrically driven linear/rotary servo systems with geared transmissions and potentially integrated force/torque transducers. Each component within these actuators plays a crucial role in determining the compliance characteristics of the limb. As forces are applied through these actuators to generate the required wrench on the manipulator's moving platform, each actuator undergoes deflection along the axis of the applied force/torque. This deflection is influenced not only by the actuator's design and materials but also by the overall mechanical configuration of the manipulator.

The relationship between the applied actuator force/torque $\tau_i$ and the resulting small deflection $\Delta q_i$ along the corresponding axis can be approximated by a linear function. This linear relationship captures the stiffness

constant $k_i$, which represents the collective compliance of all components within the actuator.

$$\tau_i = k_i \cdot \Delta q_i,$$

(6.45)

where $k_i$ represents the *stiffness constant* of the actuator caused by all its compliant components. Rewriting Equation 6.45 for all limbs in a matrix form results in:

$$\boldsymbol{\tau} = \mathscr{K} \cdot \Delta \boldsymbol{q},$$

(6.46)

where $\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \cdots & \tau_m \end{bmatrix}^T$ is the vector of actuator force/torques, the vector of the corresponding actuator deflections is denoted by $\Delta \boldsymbol{q} = \begin{bmatrix} \Delta q_1 & \Delta q_2 & \cdots & \Delta q_m \end{bmatrix}^T$, and $\mathscr{K} = \operatorname{diag} \begin{bmatrix} k_1 & k_2 & \cdots & k_m \end{bmatrix}$ is an $m \times m$ diagonal matrix composed of the actuator stiffness constants.

In parallel manipulators, the Jacobian matrix not only reveals the relationship between the infinitesimal deflection of the actuators $\Delta \boldsymbol{q}$ and that of the moving platform $\Delta \boldsymbol{\mathcal{X}}$, but it also establishes the transformation map between the actuator forces/torques $\boldsymbol{\tau}$ and the resulting wrench exerted on the moving platform $\boldsymbol{\mathcal{F}}$. This matrix not only describes the mechanical interaction between the limbs of the manipulator but also serves as a fundamental tool for understanding and controlling its operational behavior.

Expressing the Jacobian relation given in Equation 5.6 for infinitesimal deflections provides a detailed insight into how small changes in the actuator positions translate into corresponding movements on the manipulator's moving platform:

$$\Delta \boldsymbol{q} = \boldsymbol{J} \Delta \boldsymbol{\mathcal{X}},$$

(6.47)

where

$$\Delta \boldsymbol{\mathcal{X}} = [\Delta x \quad \Delta y \quad \Delta z \quad \Delta \theta_x \quad \Delta \theta_y \quad \Delta \theta_z],$$

represents the infinitesimal linear and angular deflection of the moving platform. Additionally, rewriting the Jacobian relation for parallel manipulators as the projection of actuator force/torques to the moving platform wrench 6.9 yields:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}.$$

(6.48)

Thus, by substituting Equations 6.47 and 6.48 into Equation 6.46, this equation can be expressed in task space:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \mathscr{K} \boldsymbol{J} \cdot \Delta \boldsymbol{\mathcal{X}}.$$

(6.49)

The general stiffness relation of a manipulator relates the infinitesimal deflection of the moving platform to the applied wrench as:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K} \cdot \Delta \boldsymbol{\mathcal{X}},$$

(6.50)

where $\boldsymbol{K}$ is the manipulator “Stiffness Matrix” computed from the following equation:

$$\boldsymbol{K} = \boldsymbol{J}^T \mathscr{K} \boldsymbol{J}.$$

(6.51)

Equation 6.51 implies that the moving platform's output wrench is related to its deflection by the stiffness matrix $\boldsymbol{K}$ .

The stiffness matrix has desirable characteristics for analysis: it is a symmetric positive definite matrix, although it is configuration dependent. Moreover, if the manipulator actuators are all of the same type and have the same stiffness constants, that is, $k_1 = k_2 = \cdots = k_m = k$, the stiffness matrix reduces to the simple form of $\boldsymbol{K} = k\boldsymbol{J}^T\boldsymbol{J}$, which is directly related to the dexterity matrix.

Note that the stiffness matrix is invertible if:

$$\det\left(\boldsymbol{J}^T\boldsymbol{J}\right) \neq 0.$$

(6.52)

Referring to the singularity conditions elaborated in Section 5.4.2, this condition means that the parallel manipulator has no forward kinematics singularity. From a physical perspective, at a forward kinematic singularity configuration, the manipulator gains one or more uncontrollable degrees-of-freedom and thus loses its stiffness in those directions.

If condition 6.52 holds, the “Compliance Matrix” of the manipulator, denoted by $\boldsymbol{C}$, is defined as the inverse of the stiffness matrix:

$$\boldsymbol{C} = \boldsymbol{K}^{-1} = (\boldsymbol{J}^T \mathscr{K} \boldsymbol{J})^{-1}.$$

(6.53)

The compliance matrix of a manipulator reveals the mapping of the moving platform wrench to its deflection by:

$$\Delta \boldsymbol{\mathcal{X}} = \boldsymbol{C} \cdot \boldsymbol{\mathcal{F}}.$$

(6.54)

If the parallel manipulator actuators are all of the same types and have the same stiffness constants, the stiffness matrix reduces to the simple form of $\boldsymbol{K} = k\boldsymbol{J}^T\boldsymbol{J}$, and the compliance matrix may be derived from the inverse of the dexterity matrix as follows:

$$\boldsymbol{C} = \frac{1}{k}(\boldsymbol{J}^T\boldsymbol{J})^{-1}.$$

(6.55)

Note that the compliance matrix of a parallel manipulator, like the stiffness matrix, is a function of not only the compliance of each limb but also the Jacobian matrix. This means that the overall compliance characteristics of the manipulator are influenced by the configuration and arrangement of the limbs as well as the intrinsic properties of the Jacobian matrix. As the Jacobian matrix encapsulates the geometric and kinematic relationships of the manipulator, any changes in the manipulator's configuration directly affect both the compliance and stiffness matrices. Therefore, the performance and precision of the manipulator in terms of deflection and resistance to applied forces are inherently linked to the Jacobian, making it a critical component in the design and analysis of parallel manipulators. Understanding and accurately modeling these matrices is essential for predicting the behavior of the manipulator under various loading conditions and for ensuring optimal performance in practical applications.

### 6.4.2 Stiffness Ellipsoid

As elucidated before, if similar actuators with the same stiffness structure are used in all the limbs, the joint

stiffness can be considered uniform,

$$k_1 = k_2 = \cdots = k_m = k$$

then,

$$\boldsymbol{K} = k\,\boldsymbol{J}^T\boldsymbol{J}, \quad \boldsymbol{C} = \frac{1}{k}\,(\boldsymbol{J}^T\boldsymbol{J})^{-1}$$

(6.56)

It's interesting to note that in this case, the calculation of the stiffness matrix in the robot leads to the weighted dexterity matrix $\boldsymbol{J}^T\boldsymbol{J}$, where the weight of this matrix is equal to the stiffness level in the actuated joints. Naturally, by weighting a matrix, its eigenvectors will not change, and only its eigenvalues will be weighted. Thus, the robot's stiffness matrix will inherit all ellipsoidal dexterity features in a weighted manner.

Now, if the moving platform wrench is considered to be uniform and with a unit norm, $\| \boldsymbol{\mathcal{F}} \|_2 = 1$, the Euclidean norm of the moving platform deflection due to the application of this wrench can be obtained from the stiffness ellipsoid.

$$\boldsymbol{\mathcal{F}}^T\boldsymbol{\mathcal{F}} = \Delta\boldsymbol{\mathcal{X}}^T(\boldsymbol{K}^T\boldsymbol{K})\Delta\boldsymbol{\mathcal{X}}$$

(6.57)

Consider the singular value decomposition of the robot's stiffness matrix,

$$\mathrm{svd}(\mathbf{K}) = \mathbf{U}_\mathrm{k}\,\mathbf{\Sigma}_\mathrm{k}\,\mathbf{V}_\mathrm{k}^\mathrm{T} \tag{6.58}$$

and represent the maximum and minimum singular values of the stiffness matrix as $\sigma_{max}$ and $\sigma_{min}$, respectively. Let the corresponding singular vectors in the matrix $\boldsymbol{U}_k$ be denoted by $\boldsymbol{u}_{max}$ and $\boldsymbol{u}_{min}$. Then, the maximum and minimum deflections in the robot's moving platform can be obtained along these vectors.

$$\Delta\boldsymbol{\chi}_{max} = \frac{1}{|\sigma_{min}|} \parallel \boldsymbol{\mathcal{F}} \parallel \rightarrow \text{along } \boldsymbol{u}_{min} \tag{6.59}$$

$$\Delta\boldsymbol{\chi}_{min} = \frac{1}{|\sigma_{max}|} \parallel \boldsymbol{\mathcal{F}} \parallel \rightarrow \text{along } \boldsymbol{u}_{max} \tag{6.60}$$

where, the direction of the minimum and maximum eigenvalues of the stiffness matrix, corresponds directly to that of the dexterity matrix $\boldsymbol{J}^T\boldsymbol{J}$.

In the upcoming section, we will elaborate on the assessment of the stiffness characteristics of parallel robots and their correlation with robot compliance using several

case studies. This exploration is crucial for gaining a deeper understanding of how stiffness influences the behavior of parallel robots in practical applications. By examining these properties, we aim to develop effective methods for representing and evaluating the stiffness of parallel robots, enhancing their performance and reliability across different operational scenarios.

Furthermore, we will focus on the significance of employing redundant structures in parallel robots to maximize their stiffness and precision measures. This detailed examination underscores the benefits of redundancy in enhancing the manipulator's ability to withstand external forces and maintain precision over its movements. This analysis aims to highlight how strategic design choices, such as incorporating redundancy, can significantly improve the overall performance and operational capabilities of parallel robots in various practical applications.

### 6.4.3 Case Study I: Planar Cable Robot

The architecture of a planar 4RPR parallel cable robot discussed here is detailed in Section 4.4.1 and illustrated in Figure 4.4. To conduct the stiffness analysis of the manipulator, we define the vector of actuated joint forces as $\boldsymbol{\tau} = [f_1 \quad f_2 \quad f_3 \quad f_4]$, and the corresponding vector of infinitesimal displacements of the actuated limbs as $\Delta \boldsymbol{L} = [\Delta l_1 \quad \Delta l_2 \quad \Delta l_3 \quad \Delta l_4]$.

The relationship between the joint displacements $\Delta \boldsymbol{L}$ and the actuator forces $\boldsymbol{\tau}$ is governed by a diagonal $4 \times 4$ matrix of joint stiffness constants $\mathscr{K} = \text{diag}\begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix}$, expressed as:

$$\boldsymbol{\tau} = \mathscr{K} \cdot \Delta \boldsymbol{L}.$$

(6.61)

This matrix formulation succinctly characterizes how the applied forces at each joint translate into corresponding deflections, providing a foundational framework for analyzing the manipulator's stiffness properties in practical applications. From the definition of the Jacobian in Equation 5.6, we derive the relationship:

$$\Delta \boldsymbol{L} = \boldsymbol{J} \cdot \Delta \boldsymbol{\chi},$$

(6.62)

where $\Delta \boldsymbol{\chi} = \begin{bmatrix} \Delta x_{_G} & \Delta y_{_G} & \Delta \phi \end{bmatrix}$ represents the vector of infinitesimal linear and angular motions of the moving platform. The output wrench vector $\boldsymbol{\mathcal{F}} = \begin{bmatrix} f_x & f_y & n_z \end{bmatrix}$ exerted on the moving platform is related to the vector of actuated joint forces $\boldsymbol{\tau}$ through the principle of virtual work:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \cdot \boldsymbol{\tau}.$$

(6.63)

By substituting Equations 6.61 and 6.62 into Equation 6.63, we obtain:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K} \cdot \Delta \boldsymbol{\chi},$$

(6.64)

where the stiffness matrix $\boldsymbol{K}$ is defined as:

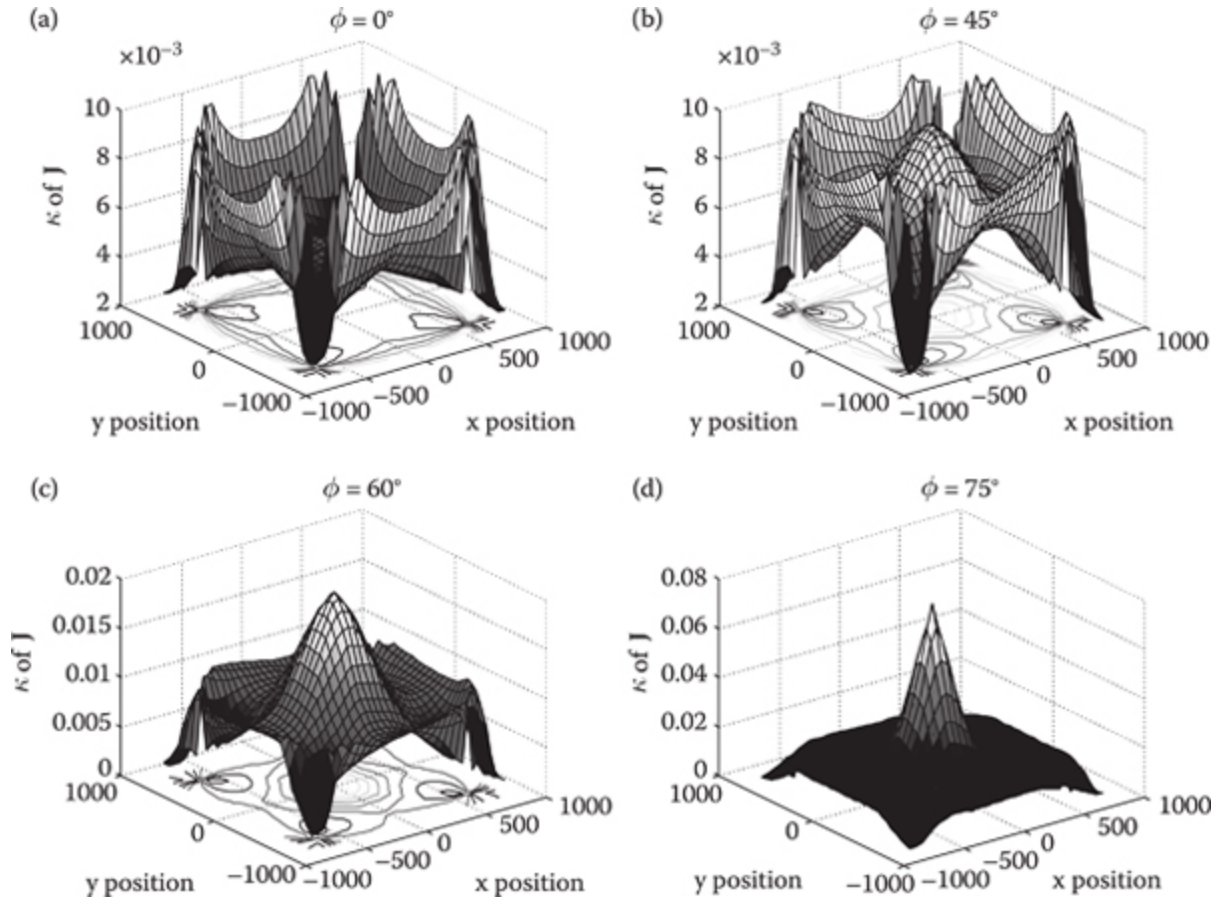
$$\boldsymbol{K} = \boldsymbol{J}^T \mathscr{K} \boldsymbol{J}.$$

(6.65)

The stiffness matrix $\boldsymbol{K}$ characterizes the stiffness properties of the planar 4RPR manipulator and is inherently dependent on its configuration. This matrix encapsulates how external forces applied to the manipulator's moving platform relate to its resulting deflections, providing critical insights into the manipulator's structural rigidity under varying operational conditions.

For a given configuration of the moving platform, the eigenvalue of the stiffness matrix represents the stiffness of the manipulator in the corresponding eigenvector direction. These directions are, in fact, represented by twist rate, which describes the motion characteristics of the platform. Additionally, the manipulability measure, or the reciprocal of the condition number of the stiffness matrix can be used to represent the dexterity of the manipulator, which is a crucial aspect of its performance.

This measure of dexterity is illustrated in the insightful and detailed plot of Figure 6.8, which assumes equal unitary stiffness constants for all actuators, ensuring a consistent basis for comparison. Since the manipulability measure is dependent on the configuration, Figure 6.8 comprehensively depicts the manipulability measure of the planar cable robot throughout its entire translational workspace, while maintaining a constant orientation of the moving platform. In this detailed figure, the manipulability measure $\kappa$, is shown for four distinct orientations, specifically $\phi = 0^\circ, 45^\circ, 60^\circ$, and $75^\circ$.




▶ Long Description for Figure 6.8

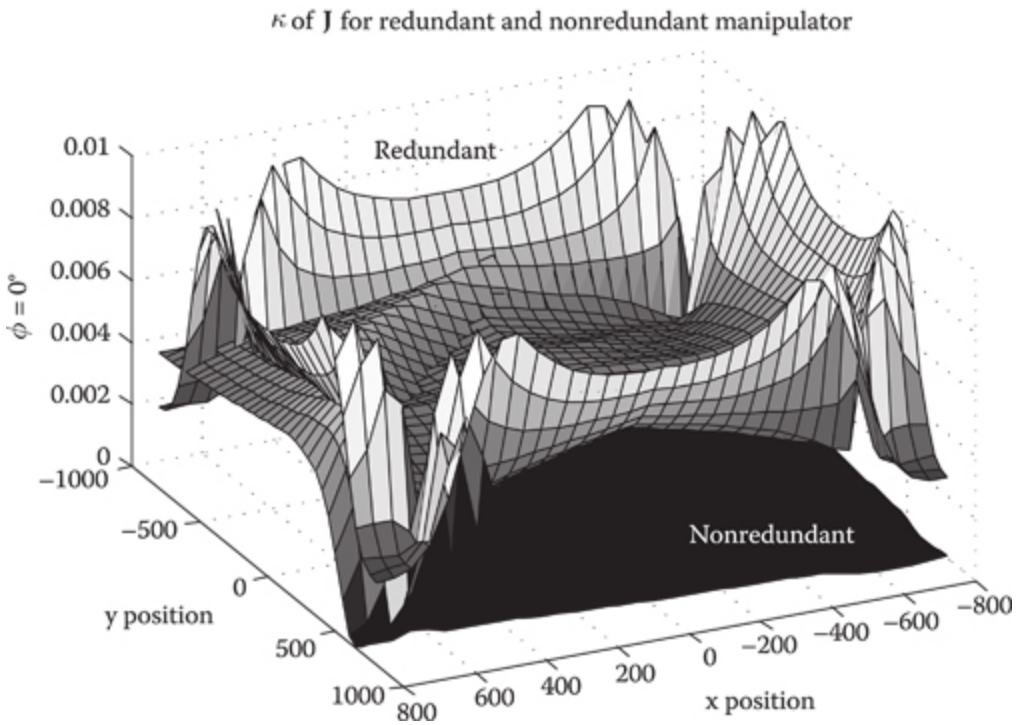
**FIGURE 6.8**
Manipulability measure (reciprocal of the condition number) of planar manipulator stiffness matrix with equal unitary stiffness constants for all actuators, over the constant orientation workspace: (a) $\phi = 0^\circ$, (b) $\phi = 45^\circ$, (c) $\phi = 60^\circ$, and (d) $\phi = 75^\circ$. ⏎

As the manipulability measure approaches one, the twist rate and wrench transformations are better conditioned, indicating more efficient and reliable manipulator performance. Conversely, as the dexterity measure decreases, the transformation characteristics deteriorate, leading to less effective manipulator behavior. This relationship is crucial for understanding and optimizing the manipulator's function.

A close examination of Figure 6.8 reveals that, in general, the manipulability measure $\kappa$, is not close to one, indicating that the manipulator is not isotropic throughout its entire workspace. This characteristic means that the manipulator's performance varies with its position within the workspace. However, the figure shows that the manipulator's manipulability measure $\kappa$, is better conditioned near its central position, suggesting optimal performance in this region. As the moving platform approaches the edges of the workspace, the manipulability measure $\kappa$, becomes smaller and ill-conditioned, indicating a decline in performance.

When comparing different constant orientation workspaces, it is evident that at the central position, the dexterity measure improves with an increase in the orientation of the moving platform. However, at the corners of the workspace, a larger orientation of the moving platform results in a worse manipulability measure, highlighting the trade-offs involved in different operational configurations. This nuanced understanding of the manipulator's behavior is essential for designing and operating it effectively in various applications.

Furthermore, it can be analyzed how actuator redundancy can enhance the overall stiffness of the planar manipulator. This enhancement is clearly demonstrated in Figure 6.9, where the manipulability measure $\kappa$, is plotted for both redundant and nonredundant manipulators. In this plot, one of the redundant limbs is suppressed, providing a comparison over the constant orientation workspace of the manipulator with $\phi = 0^{\circ}$.




▶ Long Description for Figure 6.9

**FIGURE 6.9**

Comparison of stiffness values for the redundant and nonredundant structures of planar manipulator within its constant orientation workspace with $\phi = 0^\circ$. ⏎

In Figure 6.9, it is evident that losing one actuator significantly reduces the dexterity of the manipulator. This reduction is particularly pronounced at the positions where the suppressed actuator is located, highlighting the critical role of actuator redundancy in maintaining the manipulator's performance. This figure vividly illustrates the importance of each actuator in ensuring optimal dexterity and the overall efficiency of the manipulator,

emphasizing the benefits of having redundant actuators to sustain high levels of dexterity across the workspace.

### 6.4.4 Case Study II: Shoulder Manipulator

As explained before, the stiffness of a parallel manipulator at a given point in its workspace can be characterized by its stiffness matrix. This matrix relates the wrench exerted by the moving platform in the Cartesian space to the corresponding linear and angular Cartesian deflections.

In this section, a detailed stiffness analysis is performed for the shoulder manipulator. This analysis serves as a critical initiating point for further research into this manipulator in fields such as precise position control schemes or the problem of controlling the manipulator in environments with kinematic and force constraints. In these scenarios, a desired and appropriate stiffness must be synthesized through various stiffness control schemes. Additionally, the stiffness matrix provides information on important kinematic properties of the manipulator, such as dexterity or manipulability [154].

To conduct the stiffness analysis for the shoulder manipulator, assume that the vector of actuated joint forces is denoted by $\boldsymbol{\tau} = [f_1 \quad f_2 \quad f_3 \quad f_4]$, and the corresponding vector of infinitesimal deflections of the actuated limbs is denoted by $\Delta \boldsymbol{L} = [\Delta l_1 \quad \Delta l_2 \quad \Delta l_3 \quad \Delta l_4]$. The relationship between joint displacement $\Delta \boldsymbol{L}$ and actuator force $\boldsymbol{\tau}$ is described by a diagonal $4 \times 4$ matrix of joint stiffness constants $\mathscr{K} = \text{diag}\,[k_1 \quad k_2 \quad k_3 \quad k_4]$ as follows:

$$\boldsymbol{\tau} = \mathscr{K} \cdot \Delta \boldsymbol{L}.$$

(6.66)

This fundamental relationship underpins the stiffness analysis, allowing for a thorough understanding of the manipulator's behavior under various force and displacement conditions. From the definition of the Jacobian in Equation 5.33, we have the relationship:

$$\Delta \boldsymbol{L} = \boldsymbol{J} \cdot \Delta \theta$$

(6.67)

where $\Delta\theta = [\Delta\theta_x \quad \Delta\theta_y \quad \Delta\theta_z]$ represents the vector of infinitesimal angular displacement of the moving platform. Additionally, the 3D vector of the moving platform output wrench, denoted as $\boldsymbol{\mathcal{F}} = [n_x \quad n_y \quad n_z]$, is related to the vector of actuated joint forces $\boldsymbol{\tau}$ by:

$$\boldsymbol{\tau} = \boldsymbol{J}^T \cdot \boldsymbol{\mathcal{F}}.$$

(6.68)

By substituting Equations 6.67 and 6.68 into Equation 6.66, we obtain:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K} \cdot \Delta\theta,$$

(6.69)

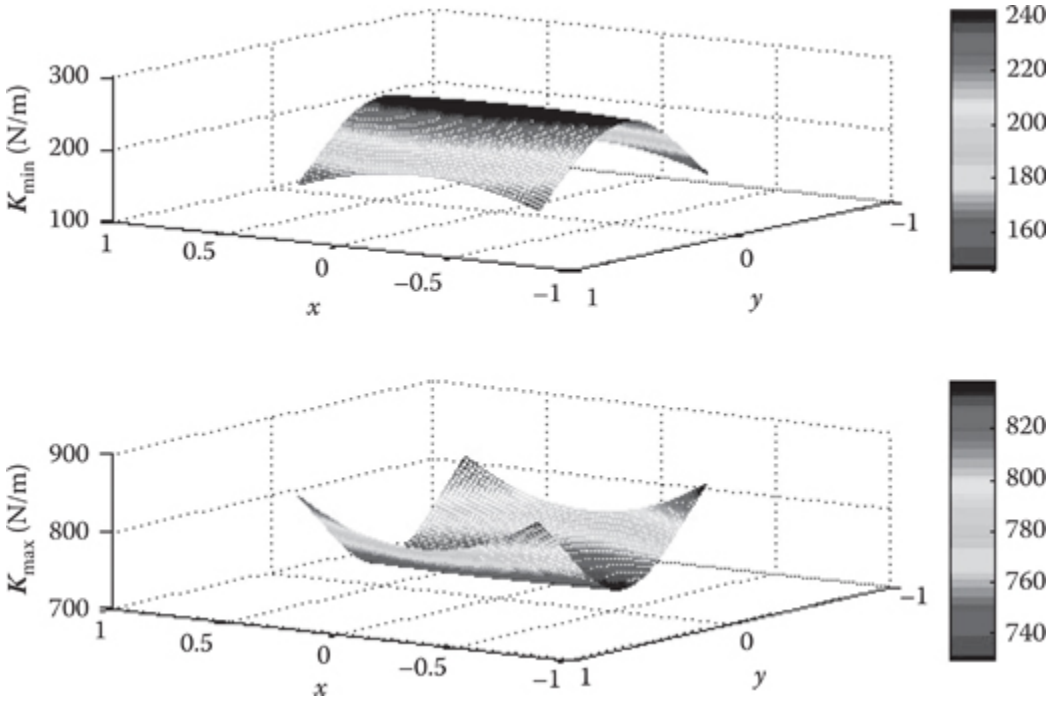
where

$$\boldsymbol{K} = \boldsymbol{J}^T \mathscr{K} \boldsymbol{J}.$$

(6.70)

Here, $\boldsymbol{K}$ is referred to as the stiffness matrix of the shoulder manipulator, which is evidently configuration-dependent. This matrix encapsulates the relationship between the wrench exerted on the moving platform and its corresponding angular deflections, providing crucial insights into the manipulator's stiffness characteristics in various configurations.

For a given configuration, the eigenvalue of the stiffness matrix represents the stiffness of the shoulder manipulator in the corresponding eigenvector direction. These directions are actually represented by twist rate vectors, which describe the manipulator's motion characteristics. Similarly, minimum and maximum stiffness mappings can be obtained using the corresponding eigenvalues and eigenvectors. These mappings are depicted in Figure 6.10, which assumes equal stiffness constants of $10^5\ N/m$ for all the actuators.




▶ Long Description for Figure 6.10

**FIGURE 6.10**

Minimum and maximum stiffness mappings of the shoulder manipulator throughout its entire workspace. ⏎

As shown in this figure, the minimum stiffness of the manipulator reaches its peak value when the manipulator is in its home orientation, with the moving platform at zero inclination. As the moving platform tilts toward its boundaries, the minimum stiffness decreases. In contrast, the maximum value of the stiffness matrix is lowest at the home position and highest at the boundaries. Additionally, it is observed that the maximum stiffness of the manipulator is approximately three times greater than its minimum value at any given orientation.

Moreover, the manipulability measure computed by reciprocal of the manipulator Jacobian condition number can be used as a measure of the manipulator's dexterity [93]. This measure allows for the analysis of how actuator redundancy has improved the total stiffness of the shoulder manipulator. This improvement is shown in Figure 6.11, where the Euclidean norms of the stiffness matrix are computed for both the redundant and nonredundant manipulators by suppressing one of the redundant legs.

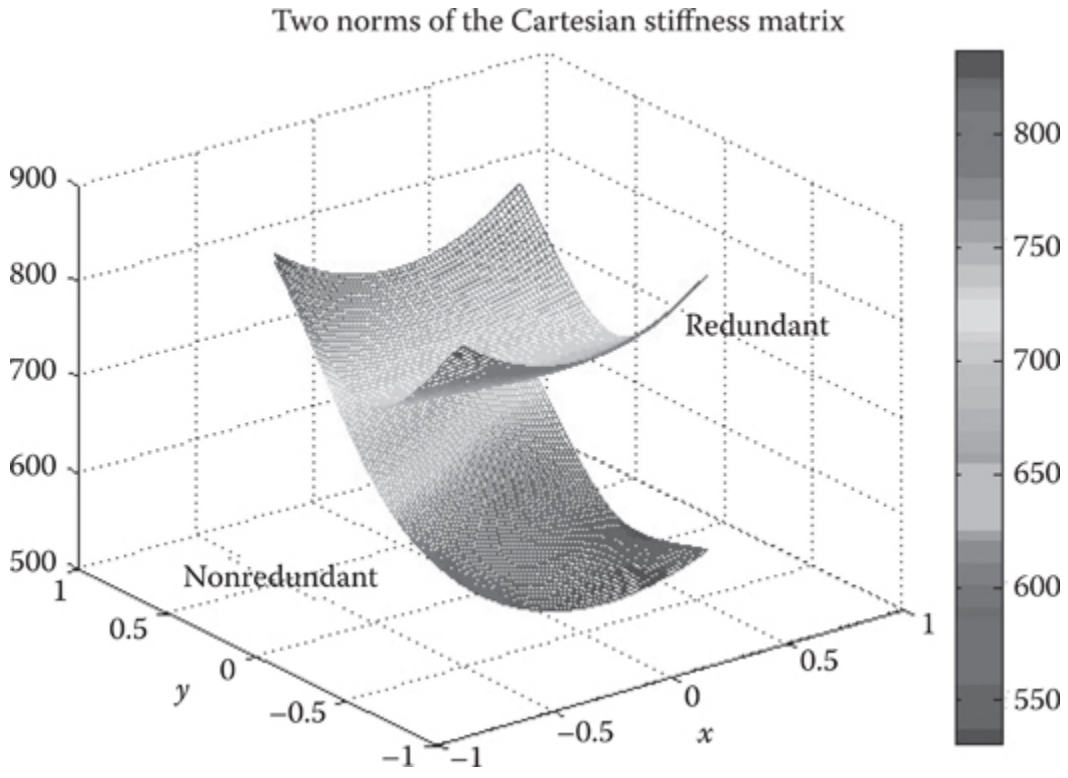


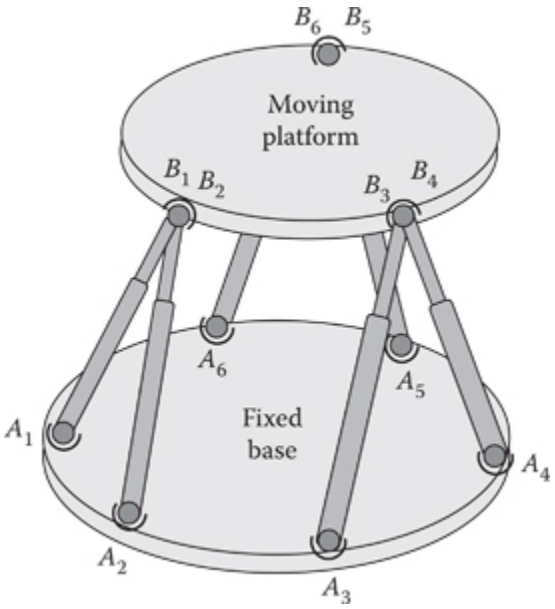

**FIGURE 6.11**

Comparison of the stiffness values for the redundant and nonredundant structures of the shoulder manipulator throughout its entire workspace. ⏎

It is clearly observed in Figure 6.11 that losing one actuator significantly reduces the stiffness of the manipulator, especially at positions where the suppressed actuator is located. This figure vividly illustrates the importance of actuator redundancy in maintaining high levels of stiffness and, consequently, the overall performance and stability of the manipulator.

### 6.4.5 Case Study III: Stewart–Gough Platform

The architecture of the Stewart–Gough platform is described in detail in Section 4.6.1 and illustrated in Figure 4.16. To perform the stiffness analysis for this manipulator, different structures of the manipulator may be examined. In this section, we restrict our analysis to a 3–6 structure of this manipulator. As shown in Figure 6.12, there exist six distinct attachment points $A_i$'s on the fixed base. The moving platform is connected through six actuators to only three moving attachment points $B_i$'s.




▶ Long Description for Figure 6.12

**FIGURE 6.12**
Schematic of a 3–6 Stewart–Gough platform. ⏎

Let the vector of actuated joint forces be denoted by $\boldsymbol{\tau} = [f_1 \quad f_2 \quad f_3 \quad f_4 \quad f_5 \quad f_6]$, and the corresponding vector of infinitesimal deflections of actuated limbs be denoted by $\Delta \boldsymbol{L} = [\Delta l_1 \quad \Delta l_2 \quad \Delta l_3 \quad \Delta l_4 \quad \Delta l_5 \quad \Delta l_6]$. The relationship between the joint deflections $\Delta \boldsymbol{L}$ and actuator force $\boldsymbol{\tau}$ is described by a diagonal $6 \times 6$ matrix of joint stiffness constants $\mathscr{K} = \text{diag}\,[k_1 \quad k_2 \quad k_3 \quad k_4 \quad k_5 \quad k_6]$ as follows:

$$\boldsymbol{\tau} = \mathscr{K} \cdot \Delta \boldsymbol{L}. \tag{6.71}$$

From the definition of the Jacobian in Equation 5.6, we have:

$$\Delta \boldsymbol{L} = \boldsymbol{J} \cdot \Delta \boldsymbol{\mathcal{X}},$$

(6.72)

where $\Delta \boldsymbol{\mathcal{X}} = [\Delta x \quad \Delta y \quad \Delta z \quad \Delta \theta_x \quad \Delta \theta_y \quad \Delta \theta_z]$ represents the vector of infinitesimal linear and angular motions of the moving platform. Additionally, a six-dimentional array of the moving platform output wrench denoted by $\boldsymbol{\mathcal{F}} = [f_x \quad f_y \quad f_z \quad n_x \quad n_y \quad n_z]$ is related to the vector of actuated joint forces $\boldsymbol{\tau}$ by:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \cdot \boldsymbol{\tau}.$$

(6.73)

By substituting Equations 6.71 and 6.72 into Equation 6.73, we obtain:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K} \cdot \Delta \boldsymbol{\mathcal{X}},$$

(6.74)

where

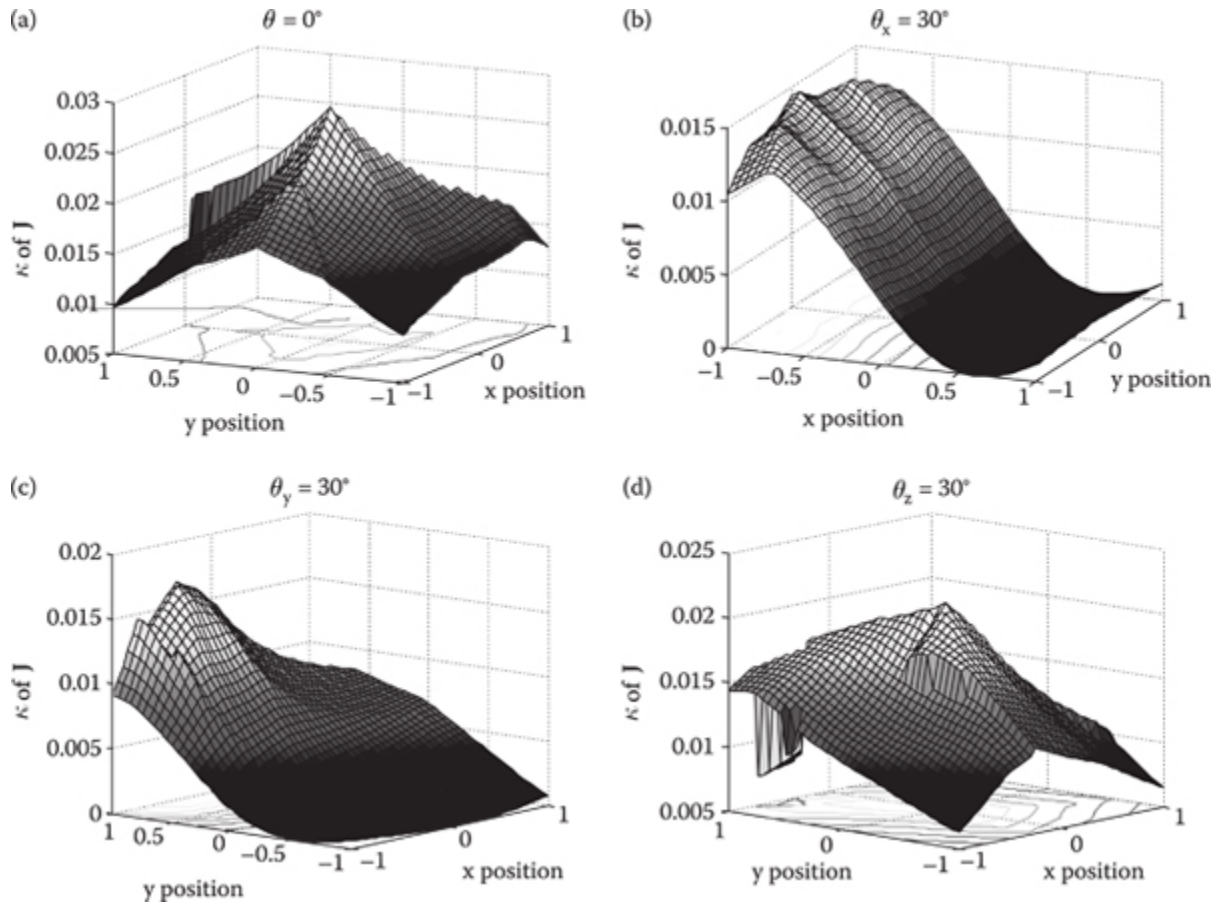
$$\boldsymbol{K} = \boldsymbol{J}^T \mathscr{K} \boldsymbol{J}.$$

(6.75)

Here, $\boldsymbol{K}$ is referred to as the stiffness matrix of the Stewart–Gough manipulator, which characterizes how the output wrench $\boldsymbol{\mathcal{F}}$ relates to the infinitesimal motions $\Delta \boldsymbol{\mathcal{X}}$ of

the moving platform. This matrix provides essential information on the manipulator's stiffness properties, crucial for understanding and optimizing its performance in various applications.

For a given configuration of the moving platform, the eigenvalue of the stiffness matrix represents the stiffness of the manipulator in the corresponding eigenvector direction. These directions are effectively represented by twist rate vectors. Additionally, the manipulability measure calculated by the reciprocal of the stiffness matrix condition number can be used to measure the dexterity of the manipulator. This measure is visualized in Figure 6.13, assuming equal unitary stiffness constants for all actuators.




▶ Long Description for Figure 6.13

**FIGURE 6.13**

Reciprocal of the condition number of a 3–6 Stewart–Gough platform stiffness matrix over the constant orientation workspace with varying $x$ and $y$and fixed $z = 0$ and (a) $\boldsymbol{\theta}_x = 0$, (b) $\theta_x = 30^\circ$, (c)$\theta_y = 30^\circ$, and (d) $\theta_z = 30^\circ$. ⏎

Given the configuration-dependent nature of dexterity, Figure 6.13 illustrates the dexterity measure of the 3–6 Stewart–Gough manipulator across its entire translational workspace, maintaining a constant heave $z = 1$ and a fixed orientation of the moving platform. The figure depicts the

manipulability measure for four distinct orientations: (a) $\theta_x = 0, \theta_y = 0, \theta_z = 0$, (b) $\theta_x = 30^\circ, \theta_y = 0, \theta_z = 0$, (c) $\theta_x = 0, \theta_y = 30^\circ, \theta_z = 0$, and (d) $\theta_x = 0, \theta_y = 0, \theta_z = 30^\circ$.

As the manipulability measure approaches one, the manipulator's twist rate and wrench transformations are better conditioned. Conversely, a lower manipulability measure indicates less favorable transformation characteristics. Figure 6.13 generally shows that the manipulability measure does not reach one across the entire workspace, indicating non-isotropic behavior of the manipulator. However, configurations (a) and (d), where rotations about the *x*- and *y*-axes are zero, exhibit better dexterity near the central position. As the moving platform moves towards the workspace boundaries, the manipulator's dexterity decreases. Conversely, configurations (b) and (c), involving rotations about either the *x*- or *y*-axes, demonstrate significantly reduced manipulability towards the workspace edges.

Moreover, exploring the maximum stiffness of the manipulator involves analyzing the maximum singular value of its Jacobian matrix. As discussed previously, this value determines the longest axis of the stiffness ellipsoid at each configuration. Figure 6.14 illustrates this maximum singular value over a constant orientation workspace with varying *x* and *y* coordinates, and fixed $z = 0$ and $\boldsymbol{\theta} = 0$. It is evident that as the moving platform approaches the workspace boundaries, the largest singular value of the Jacobian matrix increases. Consequently, the dexterity

ellipsoid from a perfect hyper sphere shape. This observation supports the findings from the dexterity measure analysis, underscoring reduced dexterity towards the edges of the constant orientation workspace. Similar insights can be derived from such analyses at different orientations of the manipulator, providing a comprehensive understanding of its operational capabilities across varied configurations and workspace boundaries.

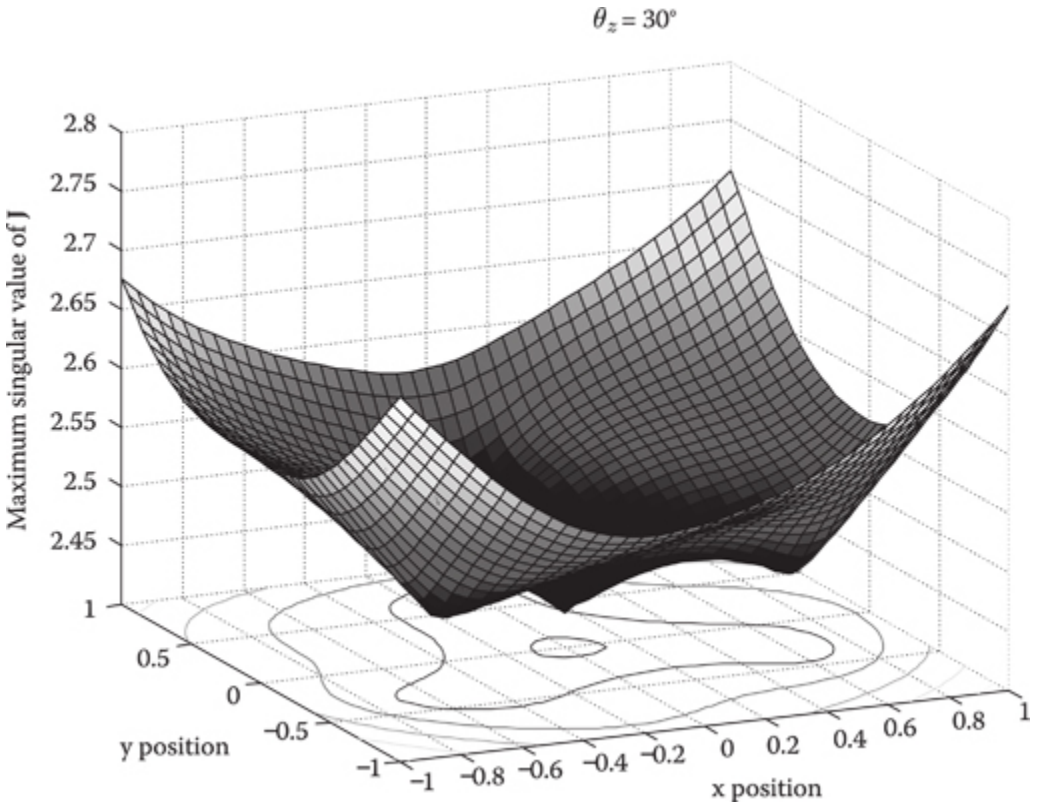



**FIGURE 6.14**

Maximum singular value of the Jacobian matrix over the constant orientation workspace with fixed $z = 0$ and $\theta_x = 0, \theta_y = 0$, and $\theta_z = 30^o$. ⏎

## 6.5 Inverse Problems

As elaborated before, the Jacobian matrix provides a direct mapping from the task space twist rate $\dot{\boldsymbol{\chi}}$ to the active joint velocities $\dot{\boldsymbol{q}}$. Specifically, the relationship is given by:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}\dot{\boldsymbol{\chi}} \tag{6.76}$$

Additionally, the transpose of the Jacobian matrix provides a direct mapping between the actuator forces/torques $\boldsymbol{\tau}$ and the moving platform wrench $\boldsymbol{\mathcal{F}}$, expressed as:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau} \tag{6.77}$$

The inverse problem in robotics involves using the Jacobian matrix to calculate the inverse mapping in differential kinematics and force domains. This problem is crucial for determining the necessary task space twist rate of the moving platform, and for computing the required actuator torques to generate a specific moving platform wrench. In the sections that follow, we will elaborate on this problem separately for both the velocity and force domains.

### 6.5.1 Inverse Velocity Problem

The Jacobian matrix plays a crucial role in robotics as it describes the relationship between the velocities of the moving platform and the active joint velocities. Specifically, if we know the twist rate of the moving platform, we can use the Jacobian matrix to directly determine the corresponding active joint velocities.

Now, consider a situation where we have the active joint velocities of the robot, denoted as $\dot{\boldsymbol{q}}$, and we need to find the twist rate of the moving platform, denoted as $\dot{\boldsymbol{\chi}}$. This scenario requires solving what is known as the inverse problem in differential kinematics. Fortunately, solving this inverse problem is not as challenging as in kinematics, since the Jacobian matrix explicitly governs this relationship.

In the case of robots with full actuation, the Jacobian matrix is a square matrix. This allows us to directly use the inverse of the Jacobian matrix to solve the inverse problem. The relationship is mathematically expressed as follows:

$$\dot{\boldsymbol{\chi}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{q}} \tag{6.78}$$

It is important to note that this solution is unique as long as the robot is not in a singular configuration, which would make the Jacobian matrix non-invertible.

To further explore the inverse problem, we can extend our analysis from velocities to accelerations. To do this, we

take the derivative of both sides of Equation 6.76 with respect to time:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}\ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{\chi}}$$

(6.79)

In this equation, $\ddot{\boldsymbol{q}}$ represents the active joint accelerations and $\ddot{\boldsymbol{\chi}}$ represents the acceleration of the moving platform. The term $\dot{\boldsymbol{J}}$ accounts for the time variation of the Jacobian matrix itself.

To solve for the acceleration array $\ddot{\boldsymbol{\chi}}$, we need to isolate it in the equation. By rearranging the terms and substituting the expression for $\dot{\boldsymbol{\chi}}$ from Equation 6.78, we obtain:

$$\ddot{\boldsymbol{\chi}} = \boldsymbol{J}^{-1}\left[\ddot{\boldsymbol{q}} - \dot{\boldsymbol{J}}\boldsymbol{J}^{-1}\dot{\boldsymbol{q}}\right]$$

(6.80)

This equation shows that the inverse acceleration problem is significantly more complex than the inverse velocity problem. It requires not only the inverse of the Jacobian matrix but also involves the time derivative of the Jacobian matrix. Analytically solving the inverse acceleration problem using Equation 6.80, is only feasible for fully actuated robots, where the Jacobian matrix is square and invertible.

It is important to note that if the robot is not fully actuated or enters a singular configuration, the Jacobian matrix may not be invertible. As a result, the straightforward relationship in Equation 6.78 for inverse velocity problem and Equation 6.80, cannot be used for computations.

### 6.5.2 Inverse Force Problem

Similar to the inverse velocity problem for parallel robots, the inverse problem in force mapping can also be explored using the Jacobian matrix. It's important to understand that the transpose of the Jacobian matrix, known as the robot's structural matrix $\boldsymbol{S}$, describes the mapping between the actuator forces/torques in the joint space and the wrenches applied to the robot's moving platform.

When the robot is fully actuated, one can utilize the inverse of the structural matrix to determine the required actuator forces and torques in the active joints to apply a desired wrench on the moving platform. This relationship can be mathematically expressed as follows:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{S}\,\boldsymbol{\tau} \quad \rightarrow \quad \boldsymbol{\tau} = \boldsymbol{S}^{-1}\boldsymbol{\mathcal{F}} \tag{6.81}$$

In this equation, $\boldsymbol{\mathcal{F}}$ represents the wrench applied to the moving platform, $\boldsymbol{\tau}$ denotes the vector of forces/torques applied in the active joints, and $\boldsymbol{S}$ is the structural matrix.

By inverting the structural matrix $\boldsymbol{S}$, one can effectively map the desired platform wrench $\boldsymbol{\mathcal{F}}$ back to the required joint forces and torques $\boldsymbol{\tau}$.

Nevertheless, if the robot is not fully actuated or encounters a singular configuration, the Jacobian matrix becomes uninvertible, rendering the direct relationship for inverse computation inapplicable. It's crucial to understand that the structural matrix of the robot, derived from the transpose of the Jacobian matrix, typically exhibits dimensions different from those of the Jacobian matrix. This discrepancy means that solving the inverse problem for joint velocities may not parallel its counterpart in force space.

Let's start by considering an underactuated robot, where the number of actuated joints, denoted as $m$, is less than the spatial dimension $n$. In such scenarios, the structural matrix has more rows than columns. The inverse force problem is solvable only when the applied wrench $\boldsymbol{\mathcal{F}}$ on the moving platform lies within the range space of the structural matrix. This condition can be checked using the following relationship:

$$\operatorname{rank}(\boldsymbol{S}) = \operatorname{rank}\left[\, \boldsymbol{S} \mid \boldsymbol{\mathcal{F}} \,\right] \tag{6.82}$$

This condition implies that not every wrench vector $\boldsymbol{\mathcal{F}}$ can be mapped back to arbitrary joint torques $\boldsymbol{\tau}$ through the structural matrix $\boldsymbol{S}$. Instead, the inverse problem has

solutions only for specific wrench vectors that satisfy this rank condition.

In contrast, for redundantly actuated robots, the situation is different. Here, the structural matrix has more columns than rows, leading to an inverse problem with infinitely many solutions. Similar to addressing the inverse velocity problem, one can select a solution that not only resolves the inverse force problem but also optimizes various performance criteria. These may include minimizing energy consumption in joint torques, avoiding singular configurations, enhancing operational flexibility, or achieving other specific objectives. The pursuit of optimal force distribution in such scenarios mirrors the strategies used in optimizing velocity distribution for redundant robots. This crucial aspect will be extensively discussed in the upcoming Section 6.6.

## 6.6 Redundancy Resolution

Redundancy is a fundamental characteristic of robot manipulators designed for tasks demanding high levels of dexterity. Typically, manipulators are engineered with an exact number of actuators corresponding to the degrees of freedom required for their specific tasks. While this non-redundant structure is cost-effective, it inherently limits the manipulator's dexterity and manipulability. In contrast, as exemplified before, many biological musculoskeletal

mechanisms, benefit from redundancy in actuation. This redundancy allows for greater manipulability in movement and control.

Redundant manipulators, incorporate more actuators than strictly necessary to perform their tasks. These additional actuators are strategically employed to enhance specific kinematic characteristics, such as avoiding singular configurations [155], optimizing posture control [156], ensuring collision avoidance [157], and navigating within joint limits [158]. This surplus actuation capability enables manipulators to operate with improved agility and manipulability, essential for applications requiring precise and dexterous movements. In cable-driven manipulators, redundancy is not merely advantageous but often a prerequisite for functionality. Cables can only exert pulling forces and cannot push, necessitating redundant actuators to maneuver the moving platform effectively while maintaining tension in all cables.

Techniques developed to manage these additional degrees of freedom optimally are referred to as “redundancy resolution” schemes. Due to its practical applications, redundancy resolution has received significant attention from researchers in the past three decades. Many of these methods are based on the pseudo-inverse of the Jacobian matrix and least squares error minimization.

This approach was first introduced by Whitney in 1969 [159] and further developed by Liégeois in 1977 [160]. In

this method, the Euclidean norm of the optimization variable is minimized, which from a physical standpoint, is equivalent to minimizing the required energy. This method has proven to be highly applicable and has been utilized in various contexts, such as torque minimization in robot actuators [161], optimization based on the weighted pseudo-inverse of the Jacobian matrix [162], optimization of elliptical dexterity features [163, 164], and determining the optimal configuration of the robot in joint and task spaces [165, 158], among many other applications.

The complexity of redundancy resolution intensifies in parallel manipulators, particularly in cable-driven systems where dynamic constraints and cable tension requirements play critical roles. Limited research has addressed redundancy in cable-driven parallel robots, focusing primarily on ensuring the feasibility of wrench solutions and optimizing system performance under stringent constraints [166, 167]. Recent advances by Barrette and Gosselin [168] have explored redundancy resolution in cable robots through advanced optimization techniques involving equality and inequality constraints, highlighting the intricate balance required to maintain cable tensions while achieving desired motion.

Further advancements, such as those investigated by Hassan and Khajepour [169], focus on numerical projection methods to distribute actuator forces optimally in cable robots. These approaches seek to minimize the overall force exertion on cables and redundant limbs, essential for

ensuring operational efficiency and longevity in cable-driven systems. Recent advancements have expanded the scope of redundancy resolution to include planar cable robots, addressing challenges at both kinematic and dynamic levels [170]. These studies treat redundancy resolution as an optimization problem aimed at minimizing the Euclidean norm of actuator forces and mobile platform velocities while ensuring cables remain under positive tension throughout operation. Such approaches underscore ongoing efforts to refine the operational efficiency and reliability of cable-driven robotic platforms, paving the way for enhanced performance in diverse robotic applications.

### 6.6.1 Problem Formulation

As discussed in Section 6.5.2, the inverse force solution is frequently encountered in parallel robotics analysis. For instance, in many motion control applications, a wrench command in task space is computed based on the tracking error in motion control. This command needs to be transformed into active joint forces/torques applied to the robots. This mapping process is commonly referred to as the "Inverse Force Solution," or "Force Distribution Method."

In the inverse force problem, the objective is to map the desired wrench $\mathcal{F}$ in task space to its corresponding actuator forces/torques $\boldsymbol{\tau}$ in joint space. For fully parallel manipulators ($m = n$), the inverse mapping from $\mathcal{F}$ to $\boldsymbol{\tau}$ at

nonsingular configurations is straightforwardly obtained using the inverse of $\boldsymbol{S}$:

$$\boldsymbol{\tau} = \boldsymbol{S}^{-1} \cdot \boldsymbol{\mathcal{F}}, \quad \text{if } m = n. \tag{6.83}$$

In cases where the manipulator is redundantly actuated ( $m > n$), solving for $\boldsymbol{\tau}$ given $\boldsymbol{\mathcal{F}}$ becomes more challenging due to the underdetermined nature of the equations represented by $\boldsymbol{\mathcal{F}} = \boldsymbol{S}\boldsymbol{\tau}$. Here, $\boldsymbol{S}$ being a rectangular matrix (more columns than rows) implies infinitely many solutions for $\boldsymbol{\tau}$. The pseudo-inverse $\boldsymbol{S}^{\dagger}$ provides a suitable inverse solution:

$$\boldsymbol{\tau}_o = \boldsymbol{S}^{\dagger} \cdot \boldsymbol{\mathcal{F}}, \quad \text{if } m > n. \tag{6.84}$$

The pseudo-inverse minimizes the Euclidean norm of $\boldsymbol{\tau}$, a characteristic particularly useful in optimizing performance metrics of redundant manipulators. This solution for a rectangular matrix $\boldsymbol{S}$, where the number of columns $m$ exceeds the number of rows $n$, is defined as:

$$\boldsymbol{S}^{\dagger} \doteq \boldsymbol{S}^{T}\left(\boldsymbol{S}\boldsymbol{S}^{T}\right)^{-1}, \quad \text{if } m > n. \tag{6.85}$$

This solution can be rigorously obtained through a constrained optimization problem, where the objective is to minimize the Euclidean norm of $\boldsymbol{\tau}$. Additionally, the optimization is subject to the constraint that the kinematic relation between the joint space variable $\boldsymbol{\tau}$ and its corresponding task space projection $\boldsymbol{\mathcal{F}}$ adheres to Equation 6.77. This optimization problem represents the fundamental formulation of redundancy resolution for a redundant manipulator, where the extra degrees of freedom in choosing $\boldsymbol{\tau}$ among the infinite solutions are utilized to minimize the energy required by the actuator efforts $\| \boldsymbol{\tau} \|_2$.

Thus, in its basic form, the redundancy resolution problem for a redundant manipulator can be framed as the following constrained optimization problem:

---

**Basic Optimization Problem:**
Find $\boldsymbol{\tau}$ such that

$$\min \| \boldsymbol{\tau} \|_2,$$

$$\text{subject to } \boldsymbol{\mathcal{F}} = \boldsymbol{S} \cdot \boldsymbol{\tau}.$$

(6.86)

---

While this approach represents the most straightforward form of redundancy resolution, it offers an effective

solution for actuators by minimizing the energy required for task execution through optimized force distribution. In practice, if no additional user-defined objectives or constraints need to be met, this optimization formulation is highly suitable. The Moore-Penrose pseudo-inverse solution $\boldsymbol{\tau}_o$ defined in Equation 6.84 can be regarded as the base solution for the redundancy resolution scheme. Moreover, all other solutions to the mapping described in Equation 6.77 can be derived from this base solution $\boldsymbol{\tau}_o$.

To identify the complete set of solutions $\boldsymbol{\tau}$ that satisfy Equation 6.77, let's introduce practical constraints to the problem. As previously noted, maintaining tension in cable-driven parallel manipulators is crucial. Let's formulate the redundancy resolution scheme for such scenarios as an optimization problem, seeking solutions from the base solution $\boldsymbol{\tau}_o$. The requirement for positive cable tensions can be incorporated as an additional inequality constraint in the optimization problem:

---

**Positive Tension Optimization:**
Find $\boldsymbol{\tau}$ such that

$$\min \| \boldsymbol{\tau} \|_2,$$

$$\text{subject to } \begin{cases} \boldsymbol{\mathcal{F}} & = \boldsymbol{S} \cdot \boldsymbol{\tau} \\ \boldsymbol{\tau} & \geq \boldsymbol{\tau}_{\min} \end{cases},$$

(6.87)

---

In this formulation, $\boldsymbol{\tau}_{\min}$ represents a user-defined lower bound for the actuator force. For a positive tension solution, it is sufficient to set $\boldsymbol{\tau}_{\min} > \mathbf{0}$ for cable-driven manipulators. It's important to note that the base solution $\boldsymbol{\tau}_o$, as defined earlier, may not inherently satisfy the requirement for positive tension in cables, which is expressed as the final inequality constraint in Equation 6.87. Therefore, $\boldsymbol{\tau}_o$ might not be suitable as the solution for this specific application.

The complete set of solutions that satisfy Equation 6.77 can be expressed by the following equation (For further details and proofs, refer to Appendix A.):

$$\boldsymbol{\tau} = \boldsymbol{\tau}_o + \left(\boldsymbol{I} - \boldsymbol{S}^{\dagger}\boldsymbol{S}\right)\boldsymbol{y}, \qquad \boldsymbol{y} \in \mathbb{R}^m,$$

(6.88)

In this equation, $\boldsymbol{\tau}_o$ represents the base pseudo-inverse solution as given in Equation 6.84. The matrix $\boldsymbol{I}$ is an $m \times m$ identity matrix, and $\boldsymbol{y}$ is any arbitrary vector in $\mathbb{R}^m$. This general solution is composed of two parts: the base solution $\boldsymbol{\tau}_o$ and a component that spans the null space of the matrix $\boldsymbol{S}$.

To understand why this general solution works, let's prove that the solution $\boldsymbol{\tau}$ given in Equation 6.88 satisfies the projection requirement described in Equation 6.77. By

multiplying both sides of Equation 6.88 by $\boldsymbol{S}$ and simplifying, we get:

$$\begin{aligned}\boldsymbol{S}\cdot\boldsymbol{\tau} &= \boldsymbol{S}\cdot\boldsymbol{\tau}_o + \boldsymbol{S}(\boldsymbol{I}-\boldsymbol{S}^\dagger\boldsymbol{S})\boldsymbol{y}\\ &= \boldsymbol{\mathcal{F}} + (\boldsymbol{S}-\boldsymbol{S}\boldsymbol{S}^\dagger\boldsymbol{S})\boldsymbol{y}\\ &= \boldsymbol{\mathcal{F}} + (\boldsymbol{S}-\boldsymbol{S}\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{S}^T)^{-1}\boldsymbol{S})\boldsymbol{y}\\ &= \boldsymbol{\mathcal{F}} + (\boldsymbol{S}-\boldsymbol{S})\boldsymbol{y}\\ &= \boldsymbol{\mathcal{F}}.\end{aligned}$$

This proof shows that the second part of the solution, $(\boldsymbol{I}-\boldsymbol{S}^\dagger\boldsymbol{S})\boldsymbol{y}$, contributes nothing to the task space wrench $\boldsymbol{\mathcal{F}}$. Hence, the term $\boldsymbol{M}\boldsymbol{y}=(\boldsymbol{I}-\boldsymbol{S}^\dagger\boldsymbol{S})\boldsymbol{y}$ spans the null space of the matrix $\boldsymbol{S}$ for any arbitrary vector $\boldsymbol{y}$. This confirms that the general solution $\boldsymbol{\tau}$ includes the base pseudo-inverse solution plus a component that resides in the null space of $\boldsymbol{S}$, allowing for flexibility in choosing $\boldsymbol{\tau}$ while still generating the required task space wrench.

Among the numerous possible solutions generated by the arbitrary selection of $\boldsymbol{y}$, it is essential to find the one that satisfies the inequality constraint presented in Equation 6.87, while still minimizing the actuator energy required. To address this optimization problem, which includes both equality and inequality constraints, a variety of effective search algorithms have been developed. These algorithms can be based on gradient-based iterative methods such as active set and interior point methods [171], or on soft computing techniques like genetic algorithms [172].

Recently, an iterative-analytic scheme has also been introduced to solve this optimization problem [173]. The specific details of these techniques are discussed in the next section.

The range and effectiveness of these numerical techniques are such that even more complex optimization problems, involving multiple cost functions and various equality and inequality constraints, can be resolved. In the following sections, we will present two additional redundancy resolution schemes framed as optimization problems to illustrate the versatility of these techniques for parallel manipulators.

Firstly, consider the scenario where the actuator forces are constrained by certain saturation limits, denoted by $\boldsymbol{\tau}_{max}$, in addition to the previously mentioned constraints. This added requirement can be incorporated into the optimization problem with an additional inequality constraint as follows:

---

**Constrained Optimization Problem:**

Find $\boldsymbol{\tau}$ such that

$$\min \| \boldsymbol{\tau} \|_2,$$

$$\text{subject to } \begin{cases} \boldsymbol{\mathcal{F}} = \boldsymbol{S} \cdot \boldsymbol{\tau} \\ \boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max} \end{cases}$$

(6.89)

In this redundancy resolution scheme, additional constraints are imposed on the actuator forces. To meet these requirements, a higher degree of redundancy is typically necessary. It is important to note that if the degree of redundancy is insufficient, it may be impossible to find a feasible solution to the optimization problem that satisfies all the constraints.

As a final case, consider a scenario where a user-defined cost function is provided to address the redundancy. Let $f(\boldsymbol{\tau})$ represent this user-defined cost function. This situation can also be formulated as an optimization problem, making it suitable for numerical solutions as follows:

**Full Optimization Problem:**

Find $\boldsymbol{\tau}$ such that

$$\min f(\boldsymbol{\tau}),$$

$$\text{subject to } \begin{cases} \boldsymbol{\mathcal{F}} = \boldsymbol{S} \cdot \boldsymbol{\tau} \\ \boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max} \\ \vdots \end{cases}$$

(6.90)

### 6.6.2 Lagrange and Karush–Kuhn–Tucker Multipliers

As explored in the preceding section, redundancy resolution strategies address complex optimization challenges that involve both equality and inequality constraints. These strategies aim to find an optimal solution that strictly satisfies these constraints. A common approach to achieve this involves integrating Lagrange multipliers, which effectively incorporate the constraints into the primary objective function. This augmented formulation allows the use of analytical or numerical methods to find the local minimum of the modified cost function.

Furthermore, ensuring the feasibility of the solution within this augmented framework necessitates establishing critical conditions on the introduced multipliers. This process ensures that the solution not only optimizes the primary objective but also respects all imposed constraints effectively.

To illustrate this methodology, let's initially consider the simplest optimization problem introduced in Equation 6.86, where the goal is to minimize a specific cost function while satisfying a set of equality constraints. We can generalize this problem into a broader formulation as follows:

---

**Equality Constrained Optimization Problem:**

Find $\boldsymbol{x}$ such that

$$\min f(\boldsymbol{x}),$$

$$\text{subject to } \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0} \tag{6.91}$$

---

In this context, the cost function $f : \mathbb{R}^m \to \mathbb{R}$ represents any arbitrary functional of the optimization variable $\boldsymbol{x} \in \mathbb{R}^m$, while the equality constraints are encapsulated by a vector-valued function $\boldsymbol{g}(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}^n$. In the context of redundancy resolution schemes, the functional $f$ often denotes the Euclidean norm of the optimization vector $\boldsymbol{x} = \boldsymbol{\tau}$, where $\boldsymbol{\tau}$ represents actuator forces or similar variables. The equality constraints typically manifest as a linear function of $\boldsymbol{\tau}$, expressed through a structure matrix $\boldsymbol{S}$. However, the formulation presented in Equation 6.91 offers a generalized representation of such optimization problems under equality constraints.

To focus deeper into the optimization solution, it is essential to define the concepts of a *regular point* and a *stationary value*.

**Definition 6.1** *A point $\boldsymbol{x}_o$ that satisfies the equality constraints given in Equation 6.91 is termed a regular point if the vectors*

$$\left( \frac{\partial g_i}{\partial \boldsymbol{x}} \right)^T, \quad i = 1, 2, \ldots, n. \tag{6.92}$$

*are linearly independent at* $\boldsymbol{x} = \boldsymbol{x}_o$.

To determine whether $\boldsymbol{x}_o$ satisfies Equation 6.92, one can evaluate the rank of the Jacobian matrix of $\boldsymbol{g}$ with respect to $\boldsymbol{x}$ at $\boldsymbol{x} = \boldsymbol{x}_o$. If the rank of the Jacobian matrix $\partial \boldsymbol{g} / \partial \boldsymbol{x} \big|_{\boldsymbol{x}=\boldsymbol{x}_o}$ equals $n$, then $\boldsymbol{x}_o$ qualifies as a regular point.

**Definition 6.2** *In the context of a constraint optimization problem defined in Equation 6.91, a functional* $f$ *is said to have a stationary value at* $\boldsymbol{x} = \boldsymbol{x}_o$ *if its partial derivative with respect to* $\boldsymbol{x}$ *evaluates to zero at* $\boldsymbol{x} = \boldsymbol{x}_o$*, that is,*

$$\left. \frac{\partial f}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\boldsymbol{x}_o} = \mathbf{0}. \tag{6.93}$$

A stationary point of a functional signifies a local extremum, whether it is a minimum, maximum, or saddle point. Equation 6.93 represents a necessary condition for identifying such points in the optimization process.

**Theorem 6.1 (Lagrange Multiplier)** *Assume* $\boldsymbol{x}_o$ *is an extremum of the optimization problem subject to equality*

*constraints given by Equation 6.91. If $\boldsymbol{x}_o$ is a regular point of the constraints, then there exist scalars $\lambda_i$, for $i = 1, 2, \ldots, n$, known as "Lagrange Multipliers," such that the augmented functional $f_a$ attains a stationary value at $\boldsymbol{x}_o$ :*

$$f_a(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}), \quad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} \in \mathbb{R}^n. \tag{6.94}$$

*Here, $f_a$ combines the original objective function $f(\boldsymbol{x})$ with the linear combination of the constraint functions $\boldsymbol{g}(\boldsymbol{x})$ weighted by the Lagrange multipliers $\boldsymbol{\lambda}$.*

The Lagrange multipliers $\lambda_i$ play a crucial role in determining how the constraints influence the optimization process. They ensure that the constraints are effectively incorporated into the optimization objective, allowing for the identification of critical points where the objective function is optimized while adhering to the given constraints. Equation 6.94 illustrates how the Lagrange multipliers enable the formulation of the augmented functional $f_a$, essential for solving constrained optimization problems efficiently.

The significance of the Lagrange multiplier Theorem lies in its treatment of equality constraints within optimization problems. By introducing Lagrange multipliers as additional variables, the theorem enables the identification of stationary values for multivariable functionals. These stationary values correspond to the local extrema of the functional, where the partial derivatives with respect to the optimization variable $\boldsymbol{x}$ are set to zero. This fundamental principle states that at a local extremum, the slope of the function with respect to all variables must be zero.

To integrate the Lagrange multipliers into the optimization framework, we define an augmented variable $\boldsymbol{x}_a$ as:

$$\boldsymbol{x}_a = \begin{bmatrix} \boldsymbol{x}, & \boldsymbol{\lambda} \end{bmatrix}^T$$

where $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_n \end{bmatrix}^T$ are the Lagrange multipliers associated with the constraints. The augmented functional $f_a$ is then expressed as:

$$f_a(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$$

where $\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} g_1(\boldsymbol{x}) & g_2(\boldsymbol{x}) & \cdots & g_n(\boldsymbol{x}) \end{bmatrix}^T$ represents the vector of constraint functions.

Seeking an extremum of $f_a$ is akin to the traditional optimization problem. At the extremum point $\boldsymbol{x}_o$, the conditions are:

$$\left.\frac{\partial f_a}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_o} = \mathbf{0}$$

which ensures zero slope of $f_a$ with respect to each component of $\boldsymbol{x}$, and

$$\left.\frac{\partial f_a}{\partial \boldsymbol{\lambda}}\right|_{\boldsymbol{x}=\boldsymbol{x}_o} = \boldsymbol{g}(\boldsymbol{x}_o) = \mathbf{0}.$$

ensuring that the constraints $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$ are satisfied at $\boldsymbol{x}_o$. The first condition provides $m$ equations corresponding to the derivatives of $f_a$ with respect to $\boldsymbol{x}$, incorporating contributions from both the objective function $f$ and the constraint function $\boldsymbol{g}$. The latter condition explicitly verifies that the constraints are met precisely at the extremum point $\boldsymbol{x}_o$.

The Lagrange multiplier theorem is pivotal in optimization theory as it simplifies the problem of minimizing a cost function subject to equality constraints into a problem of finding stationary values without constraints. This foundational result has been extended to handle optimization problems that include inequality constraints. Originally detailed by Harold W. Kuhn and Albert W. Tucker in their work referenced as [174], these conditions were actually predated by William Karush, who laid out the necessary conditions in his master's thesis [175].

Consider the most general optimization problem introduced in Equation 6.90, which seeks to minimize a cost function subject to both equality and inequality constraints. This problem can be formulated as follows:

---

**General Constrained Optimization Problem:**

Find $\boldsymbol{x}$ such that

$$\min f(\boldsymbol{x}),$$

$$\text{subject to } \begin{cases} \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}, & \boldsymbol{g} : \mathbb{R}^m \to \mathbb{R}^n \\ \\ \boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}, & \boldsymbol{h} : \mathbb{R}^m \to \mathbb{R}^\ell \end{cases},$$

(6.95)

---

where $f : \mathbb{R}^m \to \mathbb{R}$ is an arbitrary functional representing the cost function to be minimized, $\boldsymbol{g}(\boldsymbol{x})$ represents the vector-valued equality constraints mapping $\mathbb{R}^m$ to $\mathbb{R}^n$, and $\boldsymbol{h}(\boldsymbol{x})$ represents the vector-valued inequality constraints mapping $\mathbb{R}^m$ to $\mathbb{R}^\ell$.

To analyze the solution to this optimization problem, we extend the notion of a *regular point*:

**Definition 6.3** *A point $\boldsymbol{x}_o$ satisfying the constraints defined in Equation 6.95 is termed a regular point if the*

*gradients of the equality constraints* $(\partial g_i/\partial \boldsymbol{x})^T$ *evaluated at* $\boldsymbol{x} = \boldsymbol{x}_o$*, are linearly independent.*

$$\left(\frac{\partial g_i}{\partial \boldsymbol{x}}\right)^T, \quad i = 1, 2, \ldots, n; \qquad \left(\frac{\partial h_j}{\partial \boldsymbol{x}}\right)^T, \quad j \in J. \tag{6.96}$$

The condition that the gradients $(\partial g_i/\partial \boldsymbol{x})^T$ are linearly independent at $\boldsymbol{x} = \boldsymbol{x}_o$ ensures that the equality constraints are sufficiently differentiable and non-redundant at $\boldsymbol{x}_o$. This condition is essential because it allows for the application of the Lagrange multiplier theorem and the subsequent derivation of Kuhn-Tucker conditions, which are crucial for optimizing problems involving both equality and inequality constraints.

**Theorem 6.2 (Karush–Kuhn–Tucker (KKT) Theorem)** *Assume that a point* $\boldsymbol{x}_o$ *is an extremum of the optimization problem subject to equality and inequality constraints represented by Equation 6.95. If* $\boldsymbol{x}_o$ *is a regular point of the constraints, then there exist scalars* $\lambda_i, i = 1, 2, \ldots, n$ *and nonnegative scalars* $\mu_j \geq 0, j = 1, 2, \ldots, \ell$ *that provide the stationary value for the following augmented functional* $f_a$ *at* $\boldsymbol{x}_o$:

$$f_a(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{x}), \tag{6.97}$$

*where*

$$\boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{x}_o) = 0,$$

(6.98)

*and*

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} \in \mathbb{R}^n, \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_\ell \end{bmatrix} \in \mathbb{R}^\ell.$$

(6.99)

*The scalars $\lambda_i$ and $\mu_i$ are called the "KKT Multipliers."*

Note that since $\mu_i \geq 0$ and $h_i \leq 0$, Equation 6.98 implies that $\mu_i = 0$ for $h_i(\boldsymbol{x}) < 0$ and $\mu_i \geq 0$ for $h_i(\boldsymbol{x}) = 0$. The KKT theorem plays a pivotal role in optimization as it transforms a problem with equality and inequality constraints into a form where the search for an extremum is unrestricted by the constraints themselves. Moreover, it establishes the necessary conditions for a solution to be feasible. The significance of the KKT theorem lies in its role in establishing the necessary conditions for the existence of feasible solutions in optimization problems with mixed constraints. In the next section, we will explore iterative methods used to numerically solve such complex

optimization problems, leveraging the insights provided by the KKT conditions to guide the search for optimal solutions.

### 6.6.3 Iterative Solutions

Various iterative techniques have been developed to tackle constrained optimization problems. The overarching goal of these methods is to simplify the original problem by transforming it into more manageable subproblems. This approach facilitates iterative procedures that aim to locate the optimal solution efficiently. One prevalent strategy involves converting the constrained optimization problem into an unconstrained one using cost functions. These cost functions penalize violations of constraints, pushing solutions away from constraint boundaries or enforcing them directly toward the solution. This method allows for the application of unconstrained optimization techniques to handle the problem iteratively.

State-of-the-art optimization methods often revolve around satisfying the Karush–Kuhn–Tucker (KKT) conditions. These conditions provide the necessary criteria for identifying potential solutions within the constraints. By incorporating KKT multipliers associated with both equality and inequality constraints, the KKT theorem effectively guides iterative methods towards feasible solutions. In the subsequent section, we will describe specific iterative techniques employed to solve constrained optimization problems. These methods leverage the principles outlined

by the KKT conditions to refine and converge towards optimal solutions efficiently.

### 6.6.4 Numerical Methods

As highlighted in the preceding section, the KKT equations play a crucial role as necessary conditions for optimizing a constrained problem. When dealing with a convex programming case where the cost function $f(\boldsymbol{x})$ and the constraint functions $g_i(\boldsymbol{x}), i = 1, 2, \ldots, n$ and $h_i(\boldsymbol{x}), i = 1, 2, \ldots, \ell$ are convex, these conditions not only guarantee optimality but also provide sufficient conditions for achieving a globally optimal solution $\boldsymbol{x}^\star$.

Referring to the KKT theorem, for the optimization problem described by Equation 6.95, the KKT equations for the optimal solution $\boldsymbol{x}^\star$, alongside the original constraints in Equation 6.95, are as follows:

$$\nabla f(\boldsymbol{x}^\star) + \sum_{i=1}^{n} \lambda_i \nabla g_i(\boldsymbol{x}^\star) + \sum_{j=1}^{\ell} \mu_j \nabla h_j(\boldsymbol{x}^\star) = 0 \tag{6.100}$$

and

$$\lambda_i g_i(\boldsymbol{x}^\star) = 0, \quad i = 1, 2, \ldots, n \tag{6.101}$$

$$\mu_j \geq 0, \quad j = 1, 2, \ldots, \ell.$$

(6.102)

To achieve optimality with $\boldsymbol{x}^\star$, Equation 6.100 imposes critical conditions ensuring that gradients of the objective function and active constraints cancel each other at the solution point. This cancellation is achieved by appropriately adjusting the KKT multipliers $\lambda_i$ and $\mu_j$ to balance the deviations in magnitude between the objective function and the constrained gradients. Importantly, inactive constraints, as implied by the last two equations, are disregarded in this balancing act, necessitating their respective KKT multipliers to be zero.

The solution of KKT equations serves as the cornerstone for numerous nonlinear programming algorithms, wherein these algorithms aim to compute KKT multipliers directly. Among these methods, the active-set approach stands out as a notable representative.

The active set optimization approach is an iterative method used to solve constrained optimization problems, particularly those involving linear and nonlinear constraints. In this method, a subset of constraints, known as the active set, is identified and treated as equality constraints at each iteration. The algorithm iteratively adjusts this active set by solving a sequence of subproblems, where it optimizes the objective function while satisfying the active constraints. If a constraint in the

active set is no longer binding, it is removed, and if a non-active constraint becomes binding, it is added to the active set. This approach efficiently navigates the feasible region defined by the constraints, making it effective for handling complex optimization problems with multiple constraints.

Another straightforward yet effective concept widely used in optimization methods is the trust-region approach. The trust-region optimization method is also an iterative approach for finding the minimum of a nonlinear function. It approximates the objective function with a simpler model, typically quadratic, within a specified region around the current point, called the trust region. The size of this region is dynamically adjusted based on how well the model predicts the actual function behavior. If the model performs well, the region is expanded; otherwise, it is contracted. This method is known for its robustness and stability, making it effective for complex optimization problems where traditional methods may struggle. It is widely used in fields requiring reliable solutions for large-scale nonlinear optimization challenges.

One of the most effective methods for solving the constrained optimization problem described in Equation 6.95 is the interior-point approach. Unlike methods that work along the boundaries of the feasible region, the interior-point method starts from a point within the feasible region and iteratively moves towards the optimal solution by traversing the interior of the feasible region. This method tackles the constrained optimization problem by

iteratively solving a series of approximate minimization problems. Specifically, the original optimization problem 6.95 is approximated through the following formulation:

---

**Interior Point Optimization Formulation:**

Find $\boldsymbol{x}, \boldsymbol{s}$ such that

$$\min f_\nu(\boldsymbol{x}) = \min f(\boldsymbol{x}) - \nu \sum_{i=1}^{\ell} \ln s_i, \tag{6.103}$$

$$\text{subject to } \begin{cases} \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}, \\ \\ \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{s} = \mathbf{0}. \end{cases} \tag{6.104}$$

---

In this formulation, $f_\nu(\boldsymbol{x})$ represents the modified objective function where the original objective function $f(\boldsymbol{x})$ is adjusted by penalizing deviations from the constraints $\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{s} = \mathbf{0}$ using a logarithmic barrier function. The parameter $\nu > 0$ controls the strength of this penalty. Furthermore, the vector $\boldsymbol{s}$, which has the same dimension as the inequality constraints, $\boldsymbol{s} \in \mathbb{R}^\ell$, is referred to as the slack variable. Introducing slack variables into the cost

function ensures that $s_i$'s remain positive, thereby bounding $\ln s_i$. Hence, this logarithmic term serves as a barrier function.

As the parameter $\nu$ decreases towards zero, the minimum of $f_\nu$ approaches the minimum of the original objective function $f$. The approximate problem formulated in (6.103) consists of a sequence of equality-constrained problems, which are generally easier to handle computationally than the original inequality-constrained problem presented in Equation 6.95.

To solve the approximate problem, the interior-point algorithm employs one of two main types of steps at each iteration: either a Newton step in $(\boldsymbol{x}, \boldsymbol{s})$ or a conjugate gradient step within a trust region. The Newton step aims to solve the KKT equations (6.100) and (6.101) using a linear approximation of the problem. On the other hand, the conjugate gradient approach adjusts both $\boldsymbol{x}$ and $\boldsymbol{s}$, ensuring $\boldsymbol{s}$ remains positive, and minimizes a quadratic approximation of the problem under linearized constraints. By default, the algorithm initially attempts a Newton step and resorts to a conjugate gradient step if the former fails to converge. For a comprehensive understanding of these numerical techniques, interested readers are encouraged to refer to detailed discussions in standard references such as [172, 176, 171].

These numerical techniques are integrated into various commercial software packages, including MATLAB Optimization Toolbox. Users can leverage functions like

`fmincon`, which implement different numerical methods such as trust-region, active-set, and interior-point approaches to solve inequality-constrained optimization problems represented by Equation 6.95. For detailed implementation specifics, consult the user manual [177].

### 6.6.5 An Iterative-Analytical Method

It is crucial to emphasize that in the application of all numerical methods, iterative computations are indispensable for achieving the optimal solution. This principle is illustrated in Section 11.2 and exemplified in Figure 11.2, where redundancy resolution schemes are employed to distribute forces in a closed loop. To utilize such techniques effectively within a closed loop, real-time problem-solving capability is essential. Therefore, the optimization routine must converge to a solution within a fixed and brief timeframe. This requirement contrasts with conventional numerical algorithms, which proceed with variable time steps and terminate only upon reaching a specified level of accuracy.

To address this challenge, a recent development introduces an iterative-analytical method [173]. The primary advantage of employing an iterative-analytical approach for redundancy resolution is its ability to ensure that the overall solution process remains feasible within a small and acceptable timeframe suitable for real-time implementation. This method leverages the Karush-Kuhn-Tucker theorem to analyze the optimization problem and

generate a series of potential analytical solutions. Subsequently, a practical iterative algorithm is employed to efficiently identify a suitable solution.

To illustrate how this method operates in practice, let's reformulate the constrained optimization problem given in Equation 6.87 as follows:

Find $\boldsymbol{\tau}$ such that

$$\min \| \boldsymbol{\tau} \|_2,$$

$$\text{subject to } \begin{cases} \boldsymbol{\mathcal{F}} & = \boldsymbol{S}\boldsymbol{\tau} \\ \boldsymbol{\tau} & \geq \boldsymbol{\tau}_{\min}, \end{cases}$$

(6.105)

In this problem formulation, the set of all solutions satisfying the equality constraint can be described by Equation 6.88, which is reiterated here for ease of use:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_o + \left(\boldsymbol{I} - \boldsymbol{S}^{\dagger}\boldsymbol{S}\right)\boldsymbol{y}, \qquad \boldsymbol{y} \in \mathbb{R}^m.$$

(6.106)

As previously explained, in this equation, $\boldsymbol{S}$ denotes the transpose of the manipulator Jacobian matrix $\boldsymbol{J}^T$, $\boldsymbol{\tau}_o$ represents the base pseudo-inverse solution given in Equation 6.84, $\boldsymbol{I}$ denotes an $m \times m$ identity matrix, and $\boldsymbol{y}$ is an arbitrary vector in $\mathbb{R}^m$.

The solution consists of two parts: $\boldsymbol{\tau}_o$, the base solution, and the subspace generated by $\boldsymbol{My} = (\boldsymbol{I} - \boldsymbol{S}^\dagger \boldsymbol{S})\boldsymbol{y}$, which spans the null space of matrix $\boldsymbol{S}$ for any arbitrary vector $\boldsymbol{y}$. Among all possible solutions generated by varying $\boldsymbol{y}$, we seek the solution that satisfies the inequality constraint introduced in Equation 6.105 while minimizing actuator energy.

Let matrix $\boldsymbol{B}$ be defined by collecting and orthonormalizing the linearly independent column vectors of $\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{S}^\dagger \boldsymbol{S})$. Thus, the set of all solutions that satisfy the equality constraint can be expressed as:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_o + \boldsymbol{By}, \qquad \boldsymbol{y} \in \mathbb{R}^m. \tag{6.107}$$

Using this formulation, the optimization problem with both equality and inequality constraints can be simplified to a problem with only the inequality constraint:

Find $\boldsymbol{y}$ such that

$$\begin{aligned} \min f(\boldsymbol{y}) \quad & = \| \boldsymbol{\tau} \|_2^2 = \boldsymbol{\tau}^T \boldsymbol{\tau} \\ & = (\boldsymbol{\tau}_o + \boldsymbol{By})^T (\boldsymbol{\tau}_o + \boldsymbol{By}) \\ \text{subject to} \quad & \boldsymbol{h}(y) = \boldsymbol{\tau}_{\min} - (\boldsymbol{\tau}_o + \boldsymbol{By}) \leq \boldsymbol{0}. \end{aligned} \tag{6.108}$$

To find the solution $\boldsymbol{y}^\star$, we apply the Karush-Kuhn-Tucker (KKT) theorem by defining the augmented functional $f_a$ and

the KKT multipliers $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_m \end{bmatrix}^T$ as follows:

$$\begin{aligned} f_a(\boldsymbol{y}) &= f(\boldsymbol{y}) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{y}) \\ &= \boldsymbol{\tau}_o^T \boldsymbol{\tau} + \boldsymbol{\tau}_o^T \boldsymbol{B}\boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{B}^T \boldsymbol{\tau}_o + \boldsymbol{y}^T \boldsymbol{B}^T \boldsymbol{B}\boldsymbol{y} + \boldsymbol{\mu}^T (\boldsymbol{\tau}_{min} - \end{aligned} \tag{6.109}$$

The optimal point $\boldsymbol{y} = \boldsymbol{y}^\star$ must satisfy the following conditions:

$$\left. \frac{\partial f_a}{\partial \boldsymbol{y}} \right|_{\boldsymbol{y}=\boldsymbol{y}^\star} = \boldsymbol{0} \tag{6.110}$$

Additionally, the following second condition must be met for the KKT multipliers $\boldsymbol{\mu}$ to be positive:

$$\boldsymbol{\mu}^T \, \boldsymbol{h}(\boldsymbol{y}) \big|_{\boldsymbol{y}=\boldsymbol{y}^\star} = 0. \tag{6.111}$$

These two conditions can be simplified to:

$$2\boldsymbol{\tau}_o^T \boldsymbol{B} + 2\boldsymbol{y}^{\star T} \boldsymbol{B}^T \boldsymbol{B} - \boldsymbol{\mu}^T \boldsymbol{B} = \boldsymbol{0} \tag{6.112}$$

$$\boldsymbol{\mu}^T (\boldsymbol{\tau}_{min} - \boldsymbol{\tau}_o - \boldsymbol{B}\boldsymbol{y}^\star) = 0.$$

(6.113)

Note that matrix $\boldsymbol{B}$ is generated by orthonormal column vectors of matrix $\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{S}^{\dagger}\boldsymbol{S})$; therefore, $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}$. Utilizing this fact, we can write the transpose of Equation 6.112 as follows:

$$2\boldsymbol{B}^T\boldsymbol{\tau}_o + 2\boldsymbol{y}^{\star} - \boldsymbol{B}^T\boldsymbol{\mu} = \boldsymbol{0},$$

(6.114)

$$\boldsymbol{\mu}^T(\boldsymbol{\tau}_{min} - \boldsymbol{\tau}_o - \boldsymbol{B}\boldsymbol{y}^{\star}) = 0.$$

(6.115)

Equation 6.115 can be expressed component-wise in the following form:

$$\sum_{i=1}^{m} \mu_i h_i(\boldsymbol{y}^{\star}) = 0.$$

(6.116)

where $\boldsymbol{y}^{\star}$ is a feasible solution if all $\mu_i \geq 0$. Since all $\mu_i$ must be positive for the regular point $\boldsymbol{y}^{\star}$ to be a feasible solution for the optimization problem, it can be concluded that:

$$\mu_i = 0 \qquad \text{for } h_i(\boldsymbol{y}^{\star}) < 0,$$

(6.117)

$$\mu_i \geq 0 \qquad \text{for } h_i(\boldsymbol{y}^\star) = 0.$$

(6.118)

These relations imply that $\mu_i > 0$ may only occur when the corresponding inequality constraint reaches its boundary $h_i(\boldsymbol{y}^\star) = 0$, at which point the corresponding actuator force reaches its lower limit $\tau_i = \tau_{\min_i}$. Furthermore, for instances where the inequality constraint holds $\tau_{\min_i} - \tau_i < 0$, the corresponding KKT multiplier must be equal to zero $\mu_i = 0$ to satisfy Equation 6.116. Considering these relationships, the solution of the optimization problem can be derived from the following three cases:

- Case 1: Suppose all forces are within the solution set defined by the inequality constraint, and $\mu_i = 0$ for every $i = 1, 2, \ldots, m$. In this scenario, we have $h_i(\boldsymbol{y}^\star) > 0$ for all $i = 1, 2, \ldots, m$. Consequently, the solution to the optimization problem can be simplified by using Equation 6.114 as follows:

$$\boldsymbol{y}^\star = -\boldsymbol{B}^T \boldsymbol{\tau}_o.$$

(6.119)

Here, $\boldsymbol{\mu} = \boldsymbol{0}$, indicating that the actuator forces can be easily determined by

$$\boldsymbol{\tau} = \boldsymbol{\tau}_o + \boldsymbol{B}\boldsymbol{y}^\star,$$

(6.120)

where $\boldsymbol{y}^{\star}$ is defined in Equation 6.119.

- Case 2: Consider the situation where, for the optimal solution, $\mu_i > 0$ for some indices $i$ within $i = 1, 2, \ldots, m$, and $\mu_j = 0$ for the remaining indices, specifically when $j \neq i$ within $j = 1, 2, \ldots, m$. In this case, for the components where $\mu_i > 0$, the corresponding actuator forces lie at the boundary $\boldsymbol{\tau}_i = \boldsymbol{\tau}_{\min_i}$. For the remaining $\mu_j$ values, the corresponding actuator forces can be calculated from a linear equation derived from Equation 6.114 by eliminating the rows and columns of matrix $\boldsymbol{B}$ corresponding to the zero $\mu_i$ values. Therefore, $\boldsymbol{y}^{\star}$ and $\mu_j$ values can be determined by solving the following linear equation:

$$\begin{bmatrix} -2\boldsymbol{I}_{m\times m} & \begin{bmatrix} \vec{\boldsymbol{b}}_1^T & \cdots \vec{\boldsymbol{b}}_j^T \end{bmatrix} \\ \begin{bmatrix} \vec{\boldsymbol{b}}_1 \\ \vdots \\ \vec{\boldsymbol{b}}_j \end{bmatrix} & \boldsymbol{0}_{j\times j} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{y}^{\star} \\ \mu_1 \\ \vdots \\ \mu_j \end{bmatrix} = \begin{bmatrix} 2\boldsymbol{B}^T\tau_o \\ \tau_{min_1} - \tau_{o_1} \\ \vdots \\ \tau_{min_j} - \tau_{o_j} \end{bmatrix}.$$

(6.121)

In this equation, matrix $\boldsymbol{B}$ is decomposed into its rows as $\boldsymbol{B} = \begin{bmatrix} \vec{\boldsymbol{b}}_1 \\ \vdots \\ \vec{\boldsymbol{b}}_m \end{bmatrix}$, where each $\vec{\boldsymbol{b}}_j$ is a row vector. Only the row vectors corresponding to nonzero $\mu_j$ values are included in this equation. As demonstrated in [173], the left-hand side matrix in Equation 6.121 is always invertible, ensuring a solution can be found. However, this solution is feasible only if all the KKT multipliers are strictly positive, meaning $\mu_j > 0$ for all $j$.

- Case 3: In this scenario, all forces lie on the boundary of the inequality constraints, which implies that all the KKT multipliers are non-negative, $\mu_i \geq 0$ for every $i = 1, 2, \ldots, m$. Consequently, $h_i(\boldsymbol{y}^\star) = 0$ for all $i = 1, 2, \ldots, m$. The optimal actuator forces for all joints can thus be straightforwardly calculated using:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\min}. \tag{6.122}$$

In this case, the solution can be directly obtained from Equation 6.122, eliminating the need to solve Equation 6.114 for $\boldsymbol{y}^\star$.

To seek the optimal solution $\boldsymbol{y}^\star$, a search algorithm is proposed in Reference [173],. This algorithm begins with an initial assumption that all forces lie on the boundary of

the solution set defined by the inequality constraints, meaning $\mu_i > 0$ for all $i = 1, 2, \ldots, m$, as categorized in Case 3. In this scenario, the solution is calculated using Equation 6.122. This initial solution is considered feasible if it satisfies the equality constraint $\boldsymbol{\mathcal{F}} = \boldsymbol{S}\boldsymbol{\tau}$.

If the equality constraint is not satisfied with the initial assumption, the algorithm proceeds to search through combinations of actuator forces that might lie on the boundaries of the inequality constraints. There are $\binom{\boldsymbol{m}}{S}$ possible combinations to consider, where $\boldsymbol{m} = \begin{bmatrix} 1 & 2 & \cdots & m \end{bmatrix}^T$ and $S$ represents the subset size, ranging from 1 to $m$. These combinations are systematically explored by iterating through all possible values of $S$ in a loop, ensuring a thorough examination of potential solutions.

A solution is deemed feasible for the optimization problem only if all KKT multipliers are strictly positive, i.e., $\mu_j > 0$ for every $j$. If no feasible solution is found after exhaustively searching through all these combinations, the algorithm then checks Case 1. In Case 1, all the actuator forces lie within the boundary, and this can be verified if the actuator forces derived from Equation 6.120 are all positive.

This search algorithm is efficiently implemented in Reference [178]. When compared to three other numerical optimization methods—the active set, trust region, and interior point methods—simulations demonstrate that the

iterative-analytic scheme significantly reduces the elapsed time for implementation. This time efficiency is particularly evident in a special case study involving a planar cable-driven robot [173]. The study highlights the algorithm's practical advantages in optimizing the performance of robotic systems, showcasing its robustness and speed in finding optimal solutions.

## Problems

6.1 Reconsider the five-bar mechanism depicted in Figure 4.27 and detailed in Problems 4.1 and 5.1.

a. Utilize the free-body diagram approach to compute the actuator torques $\boldsymbol{\tau} = [\tau_a, \tau_2]^T$ necessary to produce an output force $\boldsymbol{F} = [f_x, f_y]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed in the form $\boldsymbol{F} = \boldsymbol{J}^T\boldsymbol{\tau}$, and compare the Jacobian matrix $\boldsymbol{J}$ obtained here with the one derived in Problem 5.1.

c. Investigate the conditions under which the input torques become unbounded.

6.2 Revisit the five-bar mechanism depicted in Figure 4.28, which is elaborated upon in Problems 4.2 and 5.2.

a. Using the free-body diagram approach, compute the actuator force/torques $\boldsymbol{\tau} = [f_1, \tau_2]^T$ necessary to generate an output force $\boldsymbol{F} = [f_x, f_y]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{F} = \boldsymbol{J}^T\boldsymbol{\tau}$, and compare the obtained Jacobian matrix $\boldsymbol{J}$ with the one derived in Problem 5.2.

c. Explore the conditions under which the input torques may become unbounded.

6.3 Consider the planar pantograph mechanism depicted in Figure 4.29, which is explored in detail in Problems 4.3 and 5.3.

a. Using the free-body diagram approach, determine the actuator torques $\boldsymbol{\tau} = [f_1, f_2]^T$ needed to generate an output force $\boldsymbol{F} = [f_x, f_y]^T$ at point $P$.

b. Show that the relationship established in part (a) can be expressed in the form $\boldsymbol{F} = \boldsymbol{J}^T\boldsymbol{\tau}$, and validate the resulting Jacobian matrix $\boldsymbol{J}$ against the one derived in Problem 5.3.

c. Investigate the conditions under which the input torques may become unbounded.

6.4 Explore the planar three-degrees-of-freedom 3RRR parallel manipulator depicted in Figure 4.30, detailed in Problems 4.4 and 5.4.

a. Utilize the free-body diagram method to compute the actuator torques $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T$ required to generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, f_y, n_z]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$, and validate the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.4.

c. Assume unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Implement a MATLAB or Python program using the numerical values provided in Problem 5.4. Compute the manipulability measure $\kappa$ of the dexterity matrix across its entire constant orientation workspace for four different orientations: $\phi = 0^\circ, 30^\circ, 60^\circ$, and $75^\circ$. Present the manipulability measures in four separate 3D plots. Analyze the dexterity plots and provide insights into the manipulator's design.

e. Extend the program developed in part (d) to calculate and plot the maximum singular value and the two norms of the stiffness matrix over the entire constant orientation workspace with $\phi = 0^\circ$. Analyze the plots to evaluate the stiffness characteristics of the manipulator.

6.5 Revisit the planar redundant 4RRR parallel manipulator shown in Figure 4.31, discussed in detail in Problems 4.5 and 5.5.

a. Apply the free-body diagram approach to compute the actuator torques $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3, \tau_4]^T$ required to generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, f_y, n_z]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$, and validate the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.5.

c. Assume unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Implement a MATLAB or Python program using the numerical values provided in Problem 5.5. Calculate the manipulability measure $\kappa$ of the dexterity matrix across its entire constant orientation workspace for four different orientations: $\phi = 0^\circ, 30^\circ, 60^\circ$, and $75^\circ$. Present the manipulability measures in four separate 3D plots. Analyze the manipulability plots and provide insights into the design of the manipulator.

e. Extend the program developed in part (d) to calculate and plot the maximum singular value and the two norms of the stiffness matrix over the entire constant orientation workspace with $\phi = 0^\circ$. Analyze

the plots to evaluate the stiffness characteristics of the manipulator.

f. Deactivate one of the actuators of the redundant manipulator, and repeat parts (d) and (e). Provide comparison plots between the redundant and non-redundant configurations, and discuss the impact of redundancy on the stiffness characteristics and manipulability of the manipulator.

6.6 Consider the spatial parallel manipulator depicted in Figure 4.32, discussed thoroughly in Problems 4.6 and 5.6.

a. Utilize the free-body diagram approach to compute the actuator forces $\boldsymbol{\tau} = [f_1, f_2, f_3]^T$ required to produce an output wrench $\boldsymbol{\mathcal{F}} = [n_x, n_y, n_z]^T$ at the moving platform.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$, and verify the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.6.

c. Assume unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Develop a MATLAB or Python program using the numerical values provided in Problem 5.6. Calculate the manipulability measure $\kappa$ of the dexterity matrix across its entire constant orientation workspace for

four different orientations: $\gamma = 0^\circ, 30^\circ, 60^\circ$, and $75^\circ$. Present the manipulability measures in four separate 3D plots. Analyze the manipulability plots and provide insights into the design of the manipulator.

e. Extend the program developed in part (d) to compute and plot the maximum singular value and the two norm of the stiffness matrix over the entire constant orientation workspace with $\gamma = 0^\circ$. Analyze the plots to evaluate the stiffness characteristics of the manipulator.

6.7 Revisit the 3UPU parallel manipulator shown in Figure 4.33, discussed in detail in Problems 4.7 and 5.7.

a. Apply the free-body diagram approach to calculate the actuator forces $\boldsymbol{\tau} = [f_1, f_2, f_3]^T$ required to generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, f_y, f_z]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$, and verify the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.7.

c. Assume unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Develop a MATLAB or Python program using the numerical values provided in Problem 5.7. Calculate

the manipulability measure $\kappa$ of the dexterity matrix over its entire workspace for four different heave values: $z = 0, 0.2, 0.4, 0.6$. Present the manipulability measures in four separate 3D plots. Analyze the manipulability plots and provide insights into the design of the manipulator.

e. Extend the program developed in part (d) to compute and plot the maximum singular value and the two norm of the stiffness matrix over the entire workspace with $z = 0$. Analyze the plots to evaluate the stiffness characteristics of the manipulator.

6.8 Revisit the 3UPS parallel manipulator introduced in [51], depicted in Figure 4.34, and discussed in Problems 4.8 and 5.8.

a. Apply the free-body diagram approach to calculate the actuator forces $\boldsymbol{\tau} = [f_1, f_2, f_3]^T$ required to generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, n_x, n_y]^T$ at point $M$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$, and verify the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.8.

c. Assume unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Develop a MATLAB or Python program using the numerical values provided in Problem 5.8. Calculate the manipulability measure $\kappa$ of the dexterity matrix over its entire workspace for four different heave values: $h = 0, 0.1, 0.2,$ and $0.4$. Present the manipulability measures in four separate 3D plots. Analyze the manipulability plots and provide insights into the design of the manipulator.

e. Extend the program developed in part (d) to compute and plot the maximum singular value and the two norms of the stiffness matrix over the entire workspace with $h = 0.4$. Analyze the plots to evaluate the stiffness characteristics of the manipulator.

6.9 Reconsider the three-degrees-of-freedom 3RPS manipulator represented in Figure 4.35 and described in Problems 4.9 and 5.9.

a. For this manipulator, apply the free-body diagram approach and calculate the actuator forces $\boldsymbol{\tau} = [f_1, f_2, f_3]^T$ required to generate an output wrench of $\boldsymbol{\mathcal{F}} = [f_x, f_y, f_z]^T$ at point $P$.

b. Show that the relation obtained in part (a) can be written in the form of $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$, and verify the obtained Jacobian matrix with the one derived in Problem 5.9.

c. Consider unit stiffness values for all manipulator actuators, and derive an expression for the stiffness matrix of the manipulator.

d. Develop a program in MATLAB or Python using the numerical values considered in Problem 5.9, and calculate the manipulability measure $\kappa$ of the manipulator's dexterity matrix over the entire workspace. Use four different heave values and plot the manipulability measures in four separate 3D plots. Analyze the dexterity plots and comment on the design of the manipulator.

e. Extend the program developed for part (d), and calculate and plot the maximum singular value and the two norms of the stiffness matrix over the entire workspace with $z = 0$. Analyze the plots and comment on the stiffness characteristics of the manipulator.

6.10 Revisit the three-degrees-of-freedom 3RPS manipulator shown in Figure 4.35, discussed in Problems 4.9 and 5.9.

a. Apply the free-body diagram approach to calculate the actuator forces $\boldsymbol{\tau} = [f_1, f_2, f_3]^T$ required to generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, f_y, f_z]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$, and verify the

resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.9.

c. Consider unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Implement a program in MATLAB or Python using the numerical values provided in Problem 5.9. Compute the manipulability measure $\kappa$ of the dexterity matrix across its entire workspace. Generate four 3D plots depicting manipulability measures for varying heave values. Analyze these plots to gain insights into the manipulator's dexterity and design considerations.

e. Extend the program developed in part (d) to compute and plot the maximum singular value, and two norms of the stiffness matrix over the entire workspace with $z = 0$. Evaluate and discuss the stiffness characteristics of the manipulator based on these plots.

6.11 Revisit the Delta robot depicted in Figure 4.38, detailed in Problems 4.11 and 5.11.

a. Apply the free-body diagram approach to compute the actuator torques $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T$ required to generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, f_y, f_z]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$, and verify the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.11.

c. Consider unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Implement a program in MATLAB or Python using the numerical values provided in Problem 5.11. Compute the manipulability measure $\kappa$ of the dexterity matrix across its entire workspace. Generate four 3D plots illustrating manipulability measures for different heave values. Analyze these plots to gain insights into the manipulator's dexterity and design.

e. Extend the program developed in part (d) to calculate and plot the maximum singular value, and two norms of the stiffness matrix over the entire workspace with $z = 0$. Evaluate and discuss the stiffness characteristics of the manipulator based on these plots.

6.12 Consider the Quadrupteron robot represented in Figure 4.39 and described in Problem 4.12 [55].

a. Apply the free-body diagram approach to compute the actuator torques $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3 \tau_4]^T$ required to

generate an output wrench $\boldsymbol{\mathcal{F}} = [f_x, f_y, f_z, n_z]^T$ at point $P$.

b. Demonstrate that the relationship derived in part (a) can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$, and verify the resulting Jacobian matrix $\boldsymbol{J}$ against the one obtained in Problem 5.12.

c. Consider unit stiffness values for all manipulator actuators and derive an expression for the stiffness matrix of the manipulator.

d. Implement a program in MATLAB or Python using the numerical values provided in Problem 5.12. Calculate the manipulability measure $\kappa$ of the dexterity matrix over its entire workspace for four different heave values: $z = 0, 0.2, 0.4, 0.6$, and $\theta_z = 0$. Present the manipulability measures in four separate 3D plots. Analyze the manipulability plots and provide insights into the design of the manipulator.

e. Extend the program developed in part (d) to compute and plot the maximum singular value and the two norms of the stiffness matrix over the entire workspace with $z = 0$ and $\theta_z = 0$. Analyze the plots to evaluate the stiffness characteristics of the manipulator.

# Part III

# Dynamics

# 7

# *Dynamics Analysis: Newton-Euler Approach*

DOI: 10.1201/9781003590408-7

This chapter elaborates on the analysis of the dynamic formulation of parallel robots. Unlike the previous chapters that primarily focused on kinematic analysis, which examined motion without considering the forces causing it, this chapter explores the intricate relationship between the forces and torques applied to the robot and the resulting dynamic maneuvers. Dynamic analysis is crucial in various aspects of robotics, such as designing robust robot components, selecting appropriate sensors and actuators, conducting precise motion simulations, and analyzing complex motion maneuvers, particularly within the context of robot control.

The chapter begins by providing a comprehensive review of rigid body dynamics, ensuring that readers who may not be familiar with dynamic analysis concepts are well-prepared. This foundational knowledge is essential for

understanding the subsequent detailed discussions. Special emphasis is placed on the Newton-Euler method, a fundamental technique for deriving the dynamic formulation of parallel robots. Initially, the chapter elucidates the basic principles and equations governing this method, providing clear explanations and illustrative examples to enhance comprehension.

Following this, the chapter presents two case studies that serve to reinforce understanding and demonstrate the practical application of the recursive method. These case studies are carefully selected to cover a range of scenarios, showcasing the versatility and effectiveness of the Newton-Euler method in real-world situations. Through these case studies, readers gain valuable insights into the process of dynamic analysis and its implications for the design and control of parallel robots.

Finally, a series of problems is provided at the end of the chapter, offering readers an opportunity to apply the fundamental theories and methods covered throughout the chapter. By working through these exercises, readers will be better equipped to tackle the complexities of dynamic formulation in parallel robots and enhance their overall expertise in this critical area of robotics.

## 7.1 Introduction

Various methods have been studied for the dynamic analysis of systems, and among these approaches, the use of Newton-Euler equations as a fundamental method in this field has gained considerable popularity.

Newtonian mechanics is established based on three laws, first formulated by Sir Isaac Newton (1642–1727) in *The Mathematical Principles of Natural Sciences,* published in 1687. Newton effectively built upon the findings of his predecessors, including renowned scientists like Galileo and Kepler. Although the first two laws were recognized by Galileo, Newton was the first to clearly articulate all three laws. These laws are formulated for a particle, requiring the existence of a reference coordinate system for their validity, known as an inertial frame or the Galilean reference system. The laws of motion introduced by Newton are as follows:

**First Law:** The Law of Inertia.

> An object will stay at rest or keep moving at a constant velocity if there is no net force acting on it.

**Second Law:** The Law of Force and Acceleration.

> Force is equal to the mass of an object times its acceleration ($F = ma$). The acceleration of an object is directly proportional to the force and inversely proportional to the mass.

**Third Law:** The Action and Reaction Law.

> For every action, there is an equal and opposite reaction. This law describes how forces always come in pairs.

In Newtonian mechanics, the dynamic formulation is directly derived from the application of Newton's second law, which describes dynamic systems in terms of forces and torques. This approach involves the use of free-body diagrams and Newton-Euler equations in a dynamic state. In this method, it is necessary to separate all the links of the robot from each other and study them individually using free-body diagrams.

In the free-body diagram, the relationships between the links are analyzed through support forces/torques and actuator force/torque at the robot joints. By examining the acceleration of the center of mass of the links, the Newton-Euler equations in dynamic states are applied to each link of the robot to determine the relationship between these forces and torques in producing the motion of each link. However, it should be noted that ultimately, the support forces are eliminated from the equations to establish a direct relationship between actuator force/torque and the motion of the robot links.

In contrast to open-chain serial manipulators, the dynamic analysis of parallel manipulators presents an inherent complexity due to their closed-loop structure. Nevertheless, dynamic modeling is crucial for control, particularly for parallel manipulators used in applications where precise positioning and desirable dynamic

performance are prime requirements. Several approaches have been proposed for the dynamic analysis of parallel manipulators.

The fundamental Newton-Euler formulation is used for the dynamic analysis of general parallel manipulators [179], and also for the Stewart-Gough platform [180]. In this formulation, the equations of motion for each limb and the moving platform must be derived, which inevitably leads to a large number of equations and less computational efficiency. On the other hand, this method allows for the computation of all reaction forces, which is very useful in the design of a parallel manipulator.

Despite the complexity and computational demands, the Newton-Euler formulation offers a comprehensive approach to understanding the dynamics of parallel manipulators. By breaking down the system into its constituent parts and analyzing the forces and torques acting on each component, this method provides detailed insights into the dynamic behavior of the entire system. This information is essential for optimizing the design and control strategies, ensuring that the manipulator performs with high precision and efficiency.

Additionally, other methods have been developed to address the computational challenges associated with the Newton-Euler formulation. These methods aim to simplify the dynamic equations and improve computational efficiency without compromising accuracy. By leveraging advanced mathematical techniques and computational

tools, researchers continue to enhance the dynamic modeling of parallel manipulators, making them more accessible and practical for a wider range of applications.

Another efficient approach is to use the principle of virtual work, in which the computation of the constraint forces is bypassed [181]. In this method, inertial forces and moments are computed using the linear and angular accelerations of the bodies. Then, the whole manipulator is considered to be in static equilibrium by using d'Alembert's principle, and the principle of virtual work is applied to derive the input forces or torques [181]. Since constraint forces and moments do not need to be computed for the purpose of dynamics simulations and control, this approach leads to faster computational algorithms for a parallel manipulator [182].

As an alternative, the Euler-Lagrange formulation eliminates all the unwanted reaction forces at the outset, and therefore, it is quite efficient [183]. The Euler-Lagrange formulation is a powerful reformulation of classical mechanics that combines conservation of momentum with conservation of energy. This approach is particularly effective for the dynamic analysis of multibody systems, where numerous reaction forces and moments exist between the components [184]. The intricacies of this method are elaborated in Chapter 8

By focusing on the essential forces and torques required for motion, the principle of virtual work offers another streamlined and efficient method for dynamic analysis. This

method significantly reduces the computational burden compared to traditional approaches, making it particularly suitable for real-time control applications. Researchers and engineers can thus leverage this approach to achieve faster formulations, enhancing the performance and reliability of parallel manipulators in various practical applications. This method is elucidated in Chapter 9

In this chapter, we begin with a review of fundamental definitions and algorithms essential for the dynamic analysis of rigid bodies. Following this, the Newton-Euler method for dynamic analysis is introduced and explored through several case studies to illustrate its practical application. In the subsequent two chapters, we will explore dynamic analysis methods based on virtual work and the Euler-Lagrange formulation, respectively. These methods will be examined in detail, using the same case studies presented in this chapter to facilitate a comparative understanding of their applications and effectiveness.

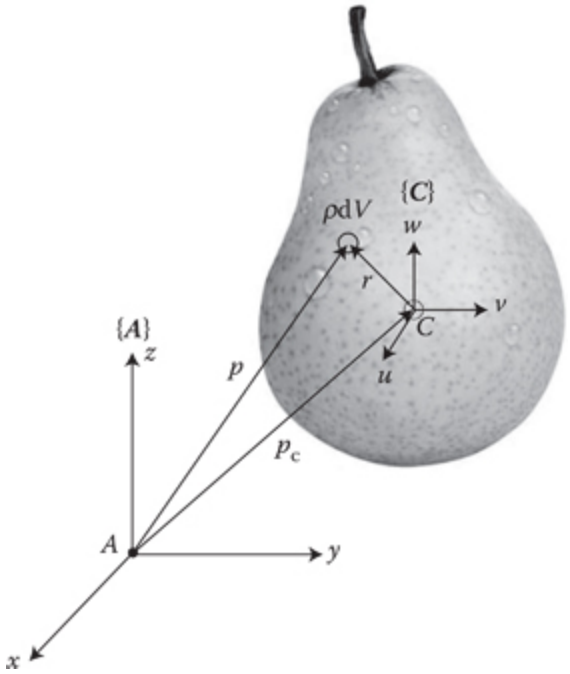
## 7.2 Dynamics of Rigid Bodies: A Review

In this section, fundamental definitions and relationships for the dynamic analysis of rigid bodies will be examined. Considering that in Chapter 3 and in Section 3.6 methods for determining the linear acceleration of a point and the angular acceleration of a body have been discussed, the subsequent focus will be on the mass properties of the rigid

body, linear and angular momentum, and the kinetic energy of the rigid body.

### 7.2.1 Center of Mass

Engineering students are introduced to the concept of mass in basic physics courses, where it is defined as a property of a body that determines its weight in a gravitational field. As shown in Figure 7.1, the distribution of mass within a rigid body affects its size, shape, and weight.




▶ Long Description for Figure 7.1

**FIGURE 7.1**
Mass properties of a rigid body. ⏎

In a given coordinate system $\{\mathscr{A}\}$, where the mass properties are measured, we assume that the body has a

uniform volumetric density $\rho$. For such a mass distribution, the mass of a small differential element of the body can be represented as $\rho dV$, where $dV$ is the infinitesimal volume of the element. If we denote the position vector of this element within the coordinate system $\{\mathscr{A}\}$ by $\boldsymbol{p}$, the total mass of the body is obtained by integrating the mass of all differential elements over the entire volume of the body. This integration process considers the uniform density and spatial distribution to calculate the total mass accurately.

$$\boldsymbol{p}_c = \frac{1}{m} \int_V \boldsymbol{p}\, \rho dV,$$

(7.1)

In that case, the position vector of the center of mass $\boldsymbol{C}$, denoted by $\boldsymbol{p}_c$ as depicted in the figure, can be determined based on the mass distribution within the body.

$$m = \int_V \rho dV.$$

(7.2)

The center of mass is a hypothetical point within an object where the entire mass of the object can be considered to be concentrated, with the mass distributed evenly around this point. When an object is supported at its center of mass, it will balance perfectly on a sharp edge, ensuring that the

weight is equally distributed on both sides and achieving equilibrium.

In the dynamic analysis of rigid bodies, the center of mass is particularly important. Rather than analyzing the distribution of mass throughout the entire object, the dynamics can be simplified by modeling the mass as if it is concentrated at the center of mass. This approach reduces the complexity of the calculations and makes it easier to analyze the object's behavior.

The mass properties of a rigid body are defined by four key parameters: the body's total mass, denoted as $m$, and the three components of the position vector of its center of mass, which are $\boldsymbol{p}_c = \begin{bmatrix} p_{c_x} & p_{c_y} & p_{c_z} \end{bmatrix}^T$. These parameters provide a clear description of the mass and its spatial distribution, facilitating a more manageable and effective analysis of the body's dynamics.

### 7.2.2 Moments of Inertia

Unlike mass, which opposes linear accelerations, the moment of inertia describes a body's resistance to angular accelerations. It characterizes how mass is distributed relative to the axis of rotation, affecting the body's response to rotational forces.

For a rigid body in rotational motion, as depicted in Figure 7.1, the moment of inertia with respect to a reference coordinate system $\{\mathscr{A}\}$ is represented by the inertia matrix ${}^{A}I$. This matrix captures the distribution of mass around the axis of rotation and is crucial for

understanding the body's rotational dynamics and resistance to changes in angular velocity.

$$
{}^{A}I = \begin{matrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{matrix} ,
$$

(7.3)

where

$$
\begin{aligned}
I_{xx} &= \int_V (y^2 + z^2)\rho dV, \qquad & I_{xy} = I_{yx} &= -\int_V xy\ \rho dV, \\
I_{yy} &= \int_V (x^2 + z^2)\rho dV, \qquad & I_{yz} = I_{zy} &= -\int_V yz\ \rho dV, \\
I_{zz} &= \int_V (x^2 + y^2)\rho dV, \qquad & I_{xz} = I_{zx} &= -\int_V xz\ \rho dV.
\end{aligned}
$$

In the equations for the inertia matrix, the diagonal components are directly proportional to the square of the radial distance from the mass particle to the axis of rotation. This contrasts with the off-diagonal components, which are proportional to the product of the distances between the mass particle and the other axes.

Consequently, the inertia matrix is symmetric and requires only six parameters for its complete description. Importantly, the inertia of the body is dependent on the coordinate system in which the calculations are performed.

Typically, these calculations are carried out with respect to a coordinate system fixed at the body's center of mass, denoted as $\{\mathscr{C}\}$.

Direct calculation of the inertia matrix using its definition is rarely performed. For bodies with simple geometries, the inertia matrix is often obtained from standard reference tables, such as those provided in [185, Appendix 5]. For more complex bodies, computer-aided design (CAD) software is commonly used to automate these computations. The primary task then becomes transforming the inertia matrix from one coordinate system to another, which will be detailed in the following sections.

## Principal Axes

As previously discussed, the inertia matrix of a body varies depending on the coordinate system used for measurement. It is possible, however, to find a coordinate system where the inertia matrix is diagonal. This specific coordinate system is known as the principal axes of inertia.

In this principal coordinate system, the inertia matrix is expressed as:

$$
{}^{A}I = \begin{matrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{matrix}
\tag{7.4}
$$

In this diagonal form, $I_{xx}$, $I_{yy}$, and $I_{zz}$ represent the principal moments of inertia. These values are fundamental mass properties of the body and can be obtained by applying a transformation matrix composed of the eigenvectors of the inertia matrix. Physically, these principal moments align with the body's symmetry axes. This approach allows us to view the inertia matrix as a mapping where the principal moments define the matrix's bounds. Since the inertia matrix governs the mass distribution and resistance to rotational acceleration, the principal moments of inertia are always positive.

## Inertia Matrix Transformations

As previously mentioned, direct computation of a body's inertia matrices according to their definition is rarely performed. Typically, these values are derived from reference tables or generated using CAD software. However, these inertia matrix values are often provided with respect to a coordinate system attached to the body at its center of mass or at specific points along the body. This contrasts with dynamic analysis, where it is essential to determine inertia matrix values with respect to different coordinate systems. Therefore, understanding how to transform the inertia matrix from one coordinate system to another is crucial.

As illustrated in Figure 7.1, if the change in coordinates for the inertia matrix is limited to translational motion,

these calculations can be performed using the parallel-axis theorem [186] and the following equation:

$$^{A}I = {}^{C}I + m\left(\boldsymbol{p}_c^T\boldsymbol{p}_c I_{3\times 3} - \boldsymbol{p}_c\boldsymbol{p}_c^T\right), \tag{7.5}$$

In this relation, $^{A}\boldsymbol{I}$ represents the inertia matrix relative to the coordinate system $\{\mathscr{A}\}$, and $^{C}\boldsymbol{I}$ is the inertia matrix relative to the coordinate system attached to the center of mass $\{\mathscr{C}\}$. Additionally, $m$ denotes the mass of the body, the vector $\boldsymbol{p}_c = [x_c, y_c, z_c]^T$, as shown in Figure 7.1, represents the translation vector from the coordinate system $\{\mathscr{C}\}$ to the coordinate system $\{\mathscr{A}\}$, and $\boldsymbol{I}_{3\times 3}$ is the identity matrix of size $3 \times 3$. This relation can be rewritten in terms of the components of the inertia matrix as follows.

$$\begin{aligned}
^{A}I_{xx} &= {}^{C}I_{xx} + m(y_c^2 + z_c^2), & ^{A}I_{xy} &= {}^{C}I_{xy} + m\,x_c y_c,\\
^{A}I_{yy} &= {}^{C}I_{yy} + m(x_c^2 + z_c^2), & ^{A}I_{yz} &= {}^{C}I_{yz} + m\,y_c z_c,\\
^{A}I_{zz} &= {}^{C}I_{zz} + m(x_c^2 + y_c^2), & ^{A}I_{xz} &= {}^{C}I_{xz} + m\,x_c z_c.
\end{aligned} \tag{7.6}$$

As observed in this relation, the diagonal components of the inertia matrix are increased proportionally to the square of the radial distance between the two coordinate systems, and the off-diagonal components are increased proportionally to the product of the distances of the corresponding axes of the two coordinate systems.

If the coordinate transformation in determining the body's inertia matrix is limited to pure rotational motion, this transformation can be performed using the following equation.

$$^{A}I = {}^{A}\boldsymbol{R}_{C}\,{}^{C}I\,{}^{A}\boldsymbol{R}_{C}^{T}. \tag{7.7}$$

In this equation, ${}^{A}\boldsymbol{R}_{C}$ is the rotation matrix from the coordinate system $\{\mathscr{C}\}$ to $\{\mathscr{A}\}$. In the dynamic analysis of parallel robots, the inertia matrix of the links is often obtained relative to the coordinate system located at their own center of mass. Equation 7.7 is then often used to transform it into other coordinate systems that have pure rotational motion relative to it.

### 7.2.3 Linear and Angular Momentum

The Newton-Euler and Euler-Lagrange methods are based on principles related to linear and angular momentum, as well as kinetic energy. A thorough exploration of these subjects in various coordinate systems demands a more extensive investigation, which falls outside the scope of this book. Enthusiastic readers are encouraged to read the references provided in this area [185, 187]. Here, we will concisely outline the key outcomes of these studies that are directly relevant to the dynamic analysis of parallel robots.

The linear momentum of a rigid body can be obtained by multiplying its mass by the velocity of its center of mass.

$$^{A}\boldsymbol{G} = m\,{}^{A}\boldsymbol{v}_{c}$$

(7.8)

In this context, it's important to emphasize that we consider the mass distribution around the center of mass as if it were concentrated solely at the center of mass itself. Consequently, we avoid computing individual particle momenta. This relationship is only valid using the velocity of the body's center of mass $\boldsymbol{v}_c$, and it cannot be determined using the velocities of other points on the body. Hence, the significance of precisely defining the center of mass and determining its velocity becomes evident in the dynamic analysis of rigid bodies.

As indicated by Equation 7.8, the linear momentum of a body, akin to its center of mass velocity, is a vector that can be interpreted across various coordinate systems. In this equation, the linear momentum of the body with respect to the coordinate system $\{\mathscr{A}\}$ is represented, and for this purpose, the velocity of the center of mass calculated in the coordinate system $\{\mathscr{A}\}$ is used. Given the vector properties of linear momentum, the homogeneous transformation matrix can be employed to transform it into another coordinate system as follows.

$$^{A}\boldsymbol{G} = \ {}^{A}\boldsymbol{T}_{C}\,{}^{C}\boldsymbol{G} = m\,{}^{A}\boldsymbol{T}_{C}\,{}^{C}\boldsymbol{v}_{c}$$

(7.9)

The angular momentum of a rigid body is also a vector, calculated with respect to the coordinate system attached to the body at its center of mass $\{\mathscr{C}\}$ as follows:

$$^{C}\boldsymbol{H} = {}^{C}\boldsymbol{I}\,\boldsymbol{\Omega}$$

(7.10)

With this definition, angular momentum extends the concept of linear momentum to rotational motions. Since the angular velocity $\boldsymbol{\Omega}$ is the same for all points on the body, calculating angular momentum only requires determining the inertia matrix of the body in the $\{\mathscr{C}\}$ coordinate system. This approach simplifies the analysis by focusing on a single, consistent rotational behavior throughout the entire body, eliminating the need to compute angular velocities at different points.

If we intend to calculate angular momentum with respect to another coordinate system, such as $\{\mathscr{A}\}$, it is no longer straightforward to accomplish this using a homogeneous transformation matrix. Instead, we need to use the following equation:

$$^{A}\boldsymbol{H} = \boldsymbol{p}_c \times \boldsymbol{G}_c + {}^{C}\boldsymbol{H}$$

(7.11)

As observed in this equation, to determine the angular momentum of the body with respect to the coordinate system $\{\mathscr{A}\}$, denoted by ${}^{A}\boldsymbol{H}$, it is necessary to sum the angular momentum of the body with respect to the center of mass coordinate system $\{\mathscr{C}\}$ with the cross product of the vector representing the distance from the center of mass to the origin of the coordinate system $\boldsymbol{p}_c$ and the linear momentum $\boldsymbol{G}_c$.

This approach underscores that the mass distribution around the center of mass is also treated as a concentrated mass at the center of mass. Therefore, when calculating the angular momentum with respect to the coordinate system $\{\mathscr{A}\}$, it suffices to combine the angular momentum with respect to the center of mass coordinate system $\{\mathscr{C}\}$ and the effect of the linear momentum of the body about the $\{\mathscr{A}\}$ coordinate system.

### 7.2.4 Kinetic Energy

In the analysis of dynamics using the Euler-Lagrange method, the kinetic energy of a rigid body plays a crucial role. Kinetic energy of a rigid body is defined as

$$K = \frac{1}{2} \int_V \boldsymbol{v} \cdot \boldsymbol{v} \rho dV. \tag{7.12}$$

Similar to the approach discussed in the previous section, the velocity of a differential mass $\rho, dV$ can be represented by the linear velocity of the center of mass and the angular velocity of the rigid body as follows:

$$\boldsymbol{v} = \boldsymbol{v}_p + \boldsymbol{\Omega} \times \boldsymbol{r}.$$

(7.13)

Substituting Equation 7.13 into Equation 7.12, and using the property of the center of mass, the kinetic energy of the rigid body can be derived by performing some manipulations:

$$K = \frac{1}{2}\boldsymbol{v}_c \cdot \boldsymbol{G}_c + \frac{1}{2}\boldsymbol{\Omega} \cdot {}^{C}\boldsymbol{H},$$

(7.14)

In this equation, the kinetic energy of the body is a scalar quantity. Its linear motion component is calculated by the dot product of the linear velocity of the center of mass and its linear momentum, while the angular motion component is determined by the dot product of the angular velocity of the body and its angular momentum about the center of mass. If the motion of the body is confined to the $x - y$ plane, the following simplified relationship can be utilized.

$$K = \frac{1}{2}m\boldsymbol{v}_c^2 + \frac{1}{2}{}^{C}I_{zz}\Omega_z^2$$

(7.15)

In this equation, $\boldsymbol{v}_c = [v_{c_x}, v_{c_y}]^T$ represents the velocity of the center of mass of the body in the $x - y$ plane, and $\Omega_z$ is the angular velocity of the body about the $z$ axis. It is for this reason that, in determining the angular momentum, only the inertia component of the body with respect to the coordinate system centered at the center of mass and about the $z$ axis, denoted by ${}^C I_{zz}$, has been used. This equation will be employed in dynamic formulation for planar mechanisms.

### 7.2.5 Newton–Euler Laws

In this section, we review the Newton and Euler laws as they apply to dynamic analysis. These fundamental principles of mechanics can be formulated for different scenarios: when angular motion is about a fixed point in space, about the center of mass, or about an arbitrary moving point in space. For clarity and brevity, we focus on the case where all rotations are represented by the center of mass. This approach simplifies the analysis by assuming that the rotational effects are considered relative to the center of mass, which is a common practice in rigid body dynamics.

However, it is important to note that these laws can be adapted to the other two cases. While the center of mass representation provides a solid foundation for dynamic analysis, other representations might offer simplifications

in specific contexts. For instance, analyzing rotations about a fixed point or an arbitrary moving point could potentially streamline the formulation and computational process, depending on the particular requirements of the problem at hand. By concentrating on the center of mass, we can leverage well-established techniques and insights from previous sections, making it easier to apply the Newton and Euler laws in practice.

Consider a rigid body in general motion, which involves both translation and rotation. Newton's law provides a relationship between the change in linear momentum of the rigid body and the external forces acting upon it. This relationship is expressed as:

$$\sum \boldsymbol{f}_{\text{ext}} = \frac{d\boldsymbol{G}_c}{dt}.$$

(7.16)

In the case where the rigid body has constant mass, this law simplifies to:

$$\sum \boldsymbol{f}_{\text{ext}} = m\frac{d\boldsymbol{v}_c}{dt} = m\mathbf{a}_c,$$

(7.17)

where $\boldsymbol{a}_c$ represents the linear acceleration of the center of mass.

These equations are vector equations and can be expressed componentwise for detailed analysis. The time derivatives in these equations are taken with respect to a fixed coordinate frame. For simplification, one can use formulations that relate relative accelerations to absolute accelerations, such as Equations 3.77 and 3.78.

It is important to note that $\Sigma \boldsymbol{f}_{\text{ext}}$ encompasses all external forces applied to the body. Thus, constructing a free-body diagram is a useful approach for identifying these forces and understanding their effects on the body's motion.

The Euler law provides a fundamental relationship between the change in angular momentum of a rigid body and the external moments acting on it. Specifically, it relates the time derivative of the angular momentum vector to the sum of all external torques applied to the body about its center of mass:

$$\sum {}^{c}\boldsymbol{n}_{\text{ext}} = \frac{d}{dt}({}^{c}\boldsymbol{H}).$$

(7.18)

In the context of a rigid body with constant mass, this relationship simplifies to:

$$\sum {}^{c}\boldsymbol{n}_{\text{ext}} = \frac{d}{dt}({}^{c}\boldsymbol{I}\,\boldsymbol{\Omega}) = {}^{c}\boldsymbol{I}\,\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times ({}^{c}\boldsymbol{I}\,\boldsymbol{\Omega}),$$

(7.19)

In this equation, $\Sigma^{c}\boldsymbol{n}_{\text{ext}}$ denotes the sum of all external moments (or torques) applied to the rigid body about its center of mass, ${}^{c}\boldsymbol{I}$ represents the moment of inertia matrix with respect to the center of mass, capturing how the mass is distributed relative to the rotation axes, and $\boldsymbol{\Omega}$ is the angular velocity vector, and $\dot{\boldsymbol{\Omega}}$ is the angular acceleration vector of the rigid body.

The moment of inertia matrix ${}^{c}\boldsymbol{I}$ essentially encodes the rotational inertia of the body, influencing how external torques change its angular velocity. The term $\boldsymbol{\Omega} \times ({}^{c}\boldsymbol{I}, \boldsymbol{\Omega})$ represents the torque due to the rotation of the body itself, known as the gyroscopic effect.

For a more specific case where the coordinate frame $\{\mathscr{C}\}$ is aligned with the principal axes of the rigid body—where the moment of inertia matrix is diagonal—Equation 7.19 can be broken down into its component forms:

$$
\begin{aligned}
\sum {}^{c}n_x &= I_{xx}\dot{\Omega}_x - \Omega_y\Omega_z(I_{yy} - I_{zz}), \\
\sum {}^{c}n_y &= I_{yy}\dot{\Omega}_y - \Omega_z\Omega_x(I_{zz} - I_{xx}), \\
\sum {}^{c}n_z &= I_{zz}\dot{\Omega}_z - \Omega_x\Omega_y(I_{xx} - I_{yy}),
\end{aligned}
\tag{7.20}
$$

where $\sum {}^{c}n_x$, $\sum {}^{c}n_y$, and $\sum {}^{c}n_z$ are the components of the external torques about the principal axes, $I_{xx}$, $I_{yy}$, and $I_{zz}$ are the principal moments of inertia around the respective principal axes.

This component-wise representation simplifies the analysis of rotational dynamics by exploiting the alignment of the coordinate system with the body's principal axes. By doing so, the inertia matrix becomes diagonal, making it easier to work with and interpret the dynamics of the body.

## 7.3 Newton–Euler Formulation

The Newton–Euler formulation stands out as one of the most prevalent methods for deriving the dynamic equations of motion in parallel manipulators. These manipulators, exemplified by the system shown in Figure 1.11, typically consist of a moving platform connected to a fixed base through multiple limbs. Due to the manipulator's closed-loop architecture, not all joints are controlled independently. Instead, some joints are driven by actuators, while others remain passive, creating a more complex dynamic system.

To use the Newton–Euler formulation, one must first create free-body diagrams for each limb and the moving platform. These diagrams help visualize and isolate each component of the system, making it easier to apply the Newton–Euler laws. The formulation requires an understanding of the linear and angular accelerations for each link and the moving platform. Specifically, the linear acceleration of the links, their centers of mass, and the angular acceleration of the links must be accurately

determined. This necessitates a thorough acceleration analysis of all the links and the moving platform.

Identifying all external forces and moments that act on the links and the moving platform is crucial. These external forces include gravitational forces that affect the centers of mass, frictional forces and moments at the joints, and any additional disturbance forces or moments. Among these external forces, the actuator forces/torques, denoted by $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_m]^T$, are particularly important as they directly influence the manipulator's motion.

In addition to external forces, the internal forces that act between the base and the links, among the links themselves, and between the links and the moving platform must also be considered. The Newton–Euler laws are applied to each link and the moving platform to derive these forces and moments. By systematically eliminating the support forces and torques from the equations, one can derive the dynamic equations that relate the actuator forces and moments $\boldsymbol{\tau}$ to the motion variables of the moving platform, including position $\boldsymbol{\chi}$, velocity $\dot{\boldsymbol{\chi}}$, and acceleration $\ddot{\boldsymbol{\chi}}$.

This comprehensive approach ensures that the dynamic behavior of the parallel manipulator is accurately modeled, taking into account both internal and external influences. The detailed process and application of these principles are further demonstrated through examples in the subsequent subsections, providing practical insights into how these

dynamic equations are formulated and utilized in real-world scenarios.

## 7.4 Case Study I: Planar Cable Robot

In this section, we explore the dynamic formulation of a planar 4RPR manipulator, examining its behavior and characteristics in detail. The architecture of this manipulator, thoroughly described in Section 4.4.1 and illustrated in Figure 4.4, involves a moving platform supported by four limbs, each featuring an RPR kinematic structure. To ensure optimal performance and to avoid kinematic singularities at the central position of the manipulator, it is crucial that the limbs are configured to cross each other. This crossing arrangement helps maintain singularity free configuration throughout the manipulator's range of motion.

The dynamic analysis of the planar 4RPR manipulator requires a comprehensive understanding of its kinematic and dynamic properties. A thorough Jacobian analysis, which assesses the relationship between joint velocities and the moving platform's velocity, is essential for this purpose. The complete Jacobian and singularity analyses for this manipulator are extensively discussed in Section 5.5, where the mathematical tools and methodologies used to analyze the manipulator's dynamic behavior are presented.

### 7.4.1 Acceleration Analysis

Acceleration analysis of the limbs and the moving platform is a crucial step in the Newton–Euler formulation for any parallel manipulator. In this section, we specifically focus on the acceleration analysis for the planar 4R<u>P</u>R manipulator. The objective of the acceleration analysis is to derive expressions for both the linear and angular accelerations of the limbs. These are denoted by $\ddot{L}_i$ and $\ddot{\alpha}_i$, respectively, and are functions of the moving platform's acceleration vector $\ddot{\boldsymbol{\chi}} = [\ddot{x}_{_G}, \ddot{y}_{_G}, \ddot{\phi}]^T$. This vector encapsulates the accelerations in the *x* and *y* directions as well as the angular acceleration $\ddot{\phi}$ of the moving platform.

To establish these relationships, we start by rewriting the velocity loop closure, which is fundamental to our analysis. This equation, as given by:

$$\boldsymbol{v}_{_G} + \dot{\phi}\,(\hat{\boldsymbol{K}} \times \boldsymbol{E_i}) = \dot{L}_i \hat{\boldsymbol{S}}_i + \dot{\alpha}_i\, L_i\, (\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i). \tag{7.21}$$

represents the velocity closure constraint for the system. Here, $\boldsymbol{v}_{_G}$ is the velocity of the moving platform, $\dot{\phi}$ is the rate of change of the platform's orientation, and $\boldsymbol{E_i}$ and $\hat{\boldsymbol{S}}_i$ are vectors related to the kinematic structure of the limbs. The term $(\hat{\boldsymbol{K}} \times \boldsymbol{E_i})$ accounts for the contribution of the angular velocity of the platform to the relative velocities of the limbs.

Differentiate Equation 7.21 with respect to time to obtain the acceleration loop closure equations for each limb of the planar 4RPR manipulator. To facilitate this, recall the definitions of the vectors $\hat{\boldsymbol{S}}_i$ and $\boldsymbol{E_i}$, which are illustrated in Figure 5.2. Specifically, note that $\dot{\hat{\boldsymbol{S}}}_i = \dot{\alpha}_i(\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i)$ and $\dot{\boldsymbol{E}}_{\boldsymbol{i}} = \dot{\phi}(\hat{\boldsymbol{K}} \times \boldsymbol{E_i})$. By differentiating and manipulating the loop closure equation, the acceleration loop closure equations for each limb, indexed by $i = 1, 2, 3, 4$, can be derived as:

$$\boldsymbol{a}_{_G} + \ddot{\phi}\,(\hat{\boldsymbol{K}} \times \boldsymbol{E_i}) - \dot{\phi}^2 \boldsymbol{E_i} = \ddot{L}_i\,\hat{\boldsymbol{S}}_i + 2\,\dot{L}_i\,\dot{\alpha}_i\,(\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i) + \ddot{\alpha}_i\,L_i \tag{7.22}$$

Here, $\boldsymbol{a}_{_G}$ represents the linear acceleration of the moving platform, while $\ddot{\phi}$ denotes the angular acceleration. The terms $\dot{\phi}^2 \boldsymbol{E_i}$ and $\dot{\alpha}_i^2 L_i \hat{\boldsymbol{S}}_i$ account for the effects of angular velocities on the acceleration of the limbs.

To eliminate $\ddot{\alpha}_i$ and obtain an expression for $\ddot{L}_i$, take the dot product of both sides of Equation 7.22 with $\hat{\boldsymbol{S}}_i$ and rearrange to get:

$$\ddot{L}_i = \boldsymbol{a}_{_G} \cdot \hat{\boldsymbol{S}}_i + \ddot{\phi}\,\hat{\boldsymbol{K}}(\boldsymbol{E_i} \times \hat{\boldsymbol{S}}_i) - \dot{\phi}^2(\boldsymbol{E_i} \cdot \hat{\boldsymbol{S}}_i) + \dot{\alpha}_i^2\,L_i. \tag{7.23}$$

This equation provides the linear acceleration of each limb, taking into account the contributions from the platform's

acceleration and angular effects.

To derive an expression for $\ddot{\alpha}_i$, cross-multiply both sides of Equation 7.22 by $\hat{\boldsymbol{S}}_i$:

$$\hat{\boldsymbol{S}}_i \times \boldsymbol{a}_{_G} + \ddot{\phi}\,(\boldsymbol{E_i} \cdot \hat{\boldsymbol{S}}_i)\,\hat{\boldsymbol{K}} - \dot{\phi}^2(\hat{\boldsymbol{S}}_i \times \boldsymbol{E_i}) = (2\,\dot{L}_i\,\dot{\alpha}_i + \ddot{\alpha}_i\,L_i)\,\hat{\boldsymbol{K}} \tag{7.24}$$

Simplify this to obtain:

$$\ddot{\alpha}_i = \frac{1}{L_i}[-S_{i_y}|S_{i_x}|E_{i_x}S_{i_x} + E_{i_y}S_{i_y}] \quad \begin{matrix} a_{Gx} \\ a_{Gy} \\ \ddot{\phi} \end{matrix} \quad - \frac{1}{L_i}\Big((E_{i_y}S_{i_x} - E \tag{7.25}$$

This final expression provides the angular acceleration $\ddot{\alpha}_i$ for each limb. Note that when written for all four limbs, the first term in this equation corresponds to $\boldsymbol{J}_\alpha$, as defined in Equation 5.30.

To complete the analysis of manipulator acceleration, it is crucial to derive expressions for the linear accelerations of the center of mass of each limb. Given that the manipulator is cable-driven, the center of mass of each limb is located at the midpoint of the limb. Let the velocity and acceleration of the center of mass of each limb be denoted by $\boldsymbol{v}_{c_i}$ and $\boldsymbol{a}_{c_i}$, respectively.

The velocity of the center of mass, $\boldsymbol{v}_{c_i}$, can be decomposed into tangential and normal components. The tangential component of the velocity is $\dot{L}_i \hat{\boldsymbol{S}}_i$, which represents the rate at which the length of the cable is changing along the limb. The normal component, which varies along the length of the limb, is given by $\frac{1}{2} L_i \dot{\alpha}_i (\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i)$. Therefore, the total velocity of the center of mass can be expressed as:

$$\boldsymbol{v}_{c_i} = \dot{L}_i \hat{\boldsymbol{S}}_i + \frac{1}{2} L_i \, \dot{\alpha}_i \, (\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i).$$

(7.26)

To obtain the acceleration of the center of mass, differentiate Equation 7.26 with respect to time:

$$\boldsymbol{a}_{c_i} = \frac{1}{2} \Big( (2\ddot{L}_i - L_i \, \dot{\alpha}^2) \, \hat{\boldsymbol{S}}_i + (L_i \, \ddot{\alpha}_i + 3 \, \dot{L}_i \, \dot{\alpha}_i)(\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i) \Big).$$

(7.27)

This expression for $\boldsymbol{a}_{c_i}$ accounts for both the changes in the tangential length of the limb and the variations in the normal component due to angular acceleration.

The velocities and accelerations of the centers of mass of the limbs, $\boldsymbol{v}_{c_i}$ and $\boldsymbol{a}_{c_i}$, depend on the limb velocities and accelerations, specifically $\dot{L}_i$, $\dot{\alpha}_i$, $\ddot{L}_i$, and $\ddot{\alpha}_i$. The relationships between these quantities and the manipulator's overall twist rate and acceleration, $\dot{\boldsymbol{\chi}}$ and $\ddot{\boldsymbol{\chi}}$,

are described by Equations 5.25, 5.28, 7.23, and 7.25, respectively. This comprehensive approach ensures that all dynamic effects are accurately accounted for in the manipulator's motion analysis.

### 7.4.2 Dynamic Formulation of the Limbs

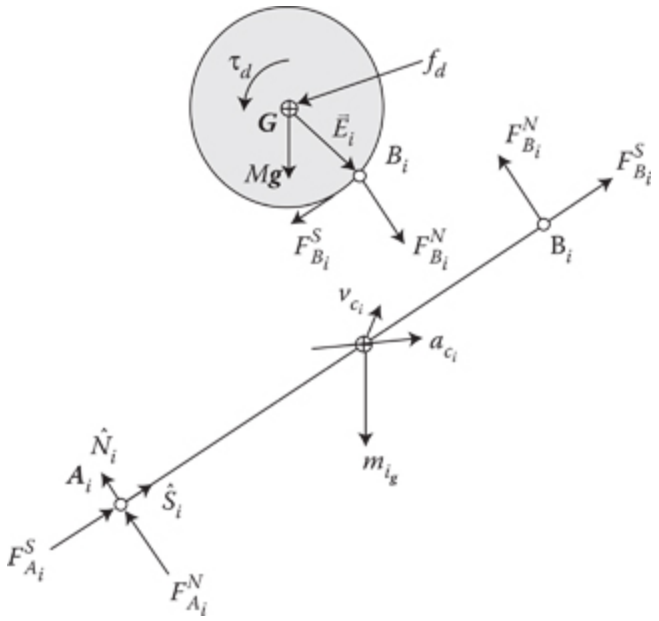
To derive the dynamic formulation of the planar $4RPR$ manipulator, certain assumptions and parameters need to be established. First, assume that the center of mass of the moving platform is located at point $G$ and that the platform has a mass denoted by $M$. The moment of inertia of the platform about its center of mass, specifically around the $z$-axis, is represented by $I_{c_i}$. Additionally, the cables in the manipulator are assumed to be homogeneous with a circular cross-section. Their density per unit length is denoted by $\rho_m$. For simplicity, the cables are treated as straight lines and modeled as rigid bodies. The mass of each cable is given by $m_i = \rho_m \, L_i$, where $L_i$ is the length of the cable.

The moment of inertia of each cable is calculated assuming that the cables are slender bars of length $L_i$. The mass and moment of inertia of the cables about their centers of mass $c_i$ in the $z$-axis are given by:

$$m_i = \rho_{_m} \, L_i; \quad I_{c_i} = \frac{m_i}{12} \, L_i^2 = \frac{\rho_{_m}}{12} \, L_i^3. \tag{7.28}$$

Here, $m_i$ represents the mass of each cable, and $I_{c_i}$ is its moment of inertia about its center of mass. The use of these parameters in the dynamic formulation ensures accurate modeling of the manipulator's components and their contributions to the overall dynamics.

To complete the dynamic formulation of the planar 4RPR manipulator, it is important to consider the orientation of the manipulator and the forces acting on it. Assume that the planar 4RPR manipulator is not confined to a horizontal plane, and thus gravity forces, represented by the vector $m_i \boldsymbol{g}$, affect the system. As shown in Figure 7.2, the gravity forces act on the manipulator's components.




▶ Long Description for Figure 7.2

**FIGURE 7.2**

Free-body diagram of the planar $4RPR$ manipulator's limb and the moving platform. ⏎

In this analysis, the free-body diagrams for the limbs and the moving platform are considered, as depicted in Figure 7.2. For each limb, the reaction forces at the fixed attachment points $A_i$ are resolved into components: $F_{A_i}^S$, which acts along the limb direction ($\hat{\boldsymbol{S}}_i$), and $F_{A_i}^N$, which acts perpendicular to the limb direction ($\hat{\boldsymbol{N}}_i$). Similarly, the internal forces at the attachment points $B_i$ are also decomposed into components $F_{B_i}^S$ and $F_{B_i}^N$.

The velocity and acceleration of each limb's center of mass, denoted by $\boldsymbol{v}_{c_i}$ and $\boldsymbol{a}_{c_i}$, respectively, are illustrated in the free-body diagram. Additionally, assume that the only external disturbance wrench acting on the moving platform is denoted by $\boldsymbol{\mathcal{F}}_d = [f_{d_x}, f_{d_y}, \tau_d]^T$, where $f_{d_x}$ and $f_{d_y}$ are the components of the disturbance force in the $x$ and $y$ directions, respectively, and $\tau_d$ represents the disturbance moment about the $z$-axis.

To derive the equations of motion for the limbs of the planar 4RPR manipulator, we use the Newton–Euler formulation. The Newton's law for the linear motion of the limbs is given by

$$\sum \boldsymbol{F}_{\text{ext}} = m_i \, \boldsymbol{a}_{c_i}, \tag{7.29}$$

where $\Sigma \boldsymbol{F}_{\text{ext}}$ is the sum of all external forces acting on each limb, and $m_i$ represents the mass of the limb. The

acceleration of the center of mass of the limb, $\boldsymbol{a}_{c_i}$, is determined from the previous analysis.

The Euler's law for rotational motion about the center of mass is expressed as

$$\sum {}^{c_i}n_{\text{ext}} = I_{c_i}\,\ddot{\alpha}_i,$$

(7.30)

where $\Sigma^{c_i}n_{\text{ext}}$ denotes the total external moments in the $z$-direction about the center of mass of the limb $c_i$, and $I_{c_i}$ is the moment of inertia of the limb about its center of mass. The angular acceleration of the limb, $\ddot{\alpha}_i$, is also derived from the previous calculations.

Incorporating the free-body diagram of the limb shown in Figure 7.2, the resulting forces and moments on the limb can be expressed in vector form as follows.

For the forces, we have

$$\left(F^S_{B_i} + F^S_{A_i}\right)\hat{\boldsymbol{S}}\boldsymbol{i} + \left(F{B_i}^N + F^N_{A_i}\right)\hat{\boldsymbol{N}}\boldsymbol{i} + m_i\boldsymbol{g} = m_i\,\boldsymbol{a}_{c_i},$$

(7.31)

where $F^S_{B_i}$ and $F^S_{A_i}$ are the force components along the limb direction $\hat{\boldsymbol{S}}_i$, and $F^N_{B_i}$ and $F^N_{A_i}$ are the perpendicular components along $\hat{\boldsymbol{N}}_i$. The term $m_i\boldsymbol{g}$ represents the gravitational force acting on the limb.

For the moments about the center of mass, the equation is

$$\left(F_{B_i}^N - F_{A_i}^N\right)\frac{L_i}{2} = I_{c_i}\,\ddot{\alpha}_i,$$

(7.32)

where $\frac{L_i}{2}$ is the distance from the center of mass to the point of application of the normal forces, and $I_{c_i}$ is the moment of inertia of the limb.

To derive the detailed equations of motion for the planar $4RPR$ manipulator, substitute the expressions for $\boldsymbol{v}_{c_i}$ and $\boldsymbol{a}_{c_i}$ from Equations 7.26 and 7.27 into the equations derived earlier. These substitutions yield componentwise equations in the directions $\hat{\boldsymbol{S}}_i$ and $\hat{\boldsymbol{N}}_i$.

Starting with the force balance in the limb direction, we have

$$F_{B_i}^S + F_{A_i}^S = \frac{\rho_m}{2}\left(2L_i\ddot{L}_i - (L_i\dot{\alpha}_i)^2 - 2L_i g^S\right),$$

(7.33)

where $\rho_m$ is the mass density per unit length of the cables, $L_i$ is the length of each limb, $\ddot{L}_i$ is the linear acceleration, $\dot{\alpha}_i$ is the angular velocity, and $g^S$ represents the component of gravitational acceleration along $\hat{\boldsymbol{S}}_i$.

For the force balance in the normal direction, we get

$$F_{B_i}^N + F_{A_i}^N = \frac{\rho_m}{2}\left(L_i^2\ddot{\alpha}_i + 3L_i\dot{L}_i\dot{\alpha}_i - 2L_i g^N\right),$$

(7.34)

where $g^N$ is the component of gravitational acceleration along $\hat{\boldsymbol{N}}_i$.

For the moment balance about the center of mass, we derive

$$F_{B_i}^N - F_{A_i}^N = \frac{\rho_m}{6} L_i^2 \ddot{\alpha}_i.$$

(7.35)

In these equations, the gravitational acceleration vector $\boldsymbol{g}$ is decomposed into its components:

$$g^S = \boldsymbol{g} \cdot \hat{\boldsymbol{S}}_i; \quad g^N = \boldsymbol{g} \cdot \hat{\boldsymbol{N}}_i; \quad g^K = \boldsymbol{g} \cdot \hat{\boldsymbol{K}},$$

(7.36)

where $g^S$ and $g^N$ represent the gravitational components along the directions of $\hat{\boldsymbol{S}}_i$ and $\hat{\boldsymbol{N}}_i$, respectively. These components can be derived from the dot product of the gravitational vector $\boldsymbol{g}$ with the respective unit vectors $\hat{\boldsymbol{S}}_i$, $\hat{\boldsymbol{N}}_i$, and $\hat{\boldsymbol{K}}$.

To derive the dynamic formulation of the planar 4R<u>P</u>R manipulator, we first determine the forces acting on each limb. The pivot reaction force, $F_{A_i}^N$, is the force exerted at the fixed points $A_i$, while $F_{B_i}^N$ represents the internal force between the limbs and the moving platform. Additionally,

the actuator forces $F_{A_i}^S$ act on the limbs and are denoted as $F_{A_i}^S = -\tau_i$, where $\tau_i$ is the actuator torque.

Substituting the expressions for $\boldsymbol{v}_{c_i}$ and $\boldsymbol{a}_{c_i}$ from Equations 7.26 and 7.27 into the force balance equations from Equations 7.33 through 7.35, we get the following component-wise relations for the forces:

$$F_{A_i}^N = \rho_m \left( \frac{1}{6} L_i^2 \ddot{\alpha}_i + \frac{3}{4} L_i \dot{L}_i \dot{\alpha}_i - \frac{1}{2} L_i g^N \right) \tag{7.37}$$

$$F_{B_i}^N = \rho_m \left( \frac{1}{3} L_i^2 \ddot{\alpha}_i + \frac{3}{4} L_i \dot{L}_i \dot{\alpha}_i - \frac{1}{2} L_i g^N \right) \tag{7.38}$$

$$F_{B_i}^S = \tau_i + \rho_m \left( L_i \ddot{L}_i - \frac{1}{2} (L_i \dot{\alpha}_i)^2 - L_i g^S \right). \tag{7.39}$$

In these equations, $\rho_m$ is the linear density of the cables, and $g^S$ and $g^N$ represent the components of gravitational acceleration in the directions of $\hat{\boldsymbol{S}}_i$ and $\hat{\boldsymbol{N}}_i$, respectively. The resulting forces and moments provide a comprehensive account of the system dynamics, considering the effects of the cables' mass, acceleration, and gravitational forces.

These relations are critical for formulating the dynamic equations of the moving platform.

### 7.4.3 Dynamic Formulation of the Moving Platform

Now, we turn our attention to the dynamic analysis of the moving platform using the free-body diagram illustrated in Figure 7.2. To derive the equations of motion for the moving platform, we apply the Newton–Euler equations as follows:

$$\sum \boldsymbol{F}_{\text{ext}} \quad = M\boldsymbol{g} + \boldsymbol{f}_d - \sum_{i=1}^{4} \left( F_{B_i}^S \hat{\boldsymbol{S}}_i + F_{B_i}^N \hat{\boldsymbol{N}}_i \right) = M\boldsymbol{a}_G \tag{7.40}$$

$$\sum {}^G\boldsymbol{n}_{\text{ext}} \quad = \tau_d \, \hat{\boldsymbol{K}} - \sum_{i=1}^{4} \boldsymbol{E}_i \times \left( F_{B_i}^S \hat{\boldsymbol{S}}_i + F_{B_i}^N \hat{\boldsymbol{N}}_i \right) = I_m \, \ddot{\phi} \, \hat{\boldsymbol{K}}. \tag{7.41}$$

The first equation, representing the force balance on the moving platform, is written component-wise to obtain:

$$M\ddot{x}_G - M(\boldsymbol{g} \cdot \hat{\boldsymbol{x}}) - f_{d_x} + \sum_{i=1}^{4} \left( F_{B_i}^S S_{i_x} - F_{B_i}^N S_{i_y} \right) = 0$$

(7.42)

$$M\ddot{y}_G - M(\boldsymbol{g} \cdot \hat{\boldsymbol{y}}) - f_{d_y} + \sum_{i=1}^{4} \left( F_{B_i}^S S_{i_y} + F_{B_i}^N S_{i_x} \right) = 0$$

(7.43)

Here, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are the unit vectors in the $x$ and $y$ directions, respectively. The gravitational force $\boldsymbol{g}$ and disturbance force $\boldsymbol{f}_d$ are considered in these equations.

Next, for the moment balance around the center of mass, the equation is given by:

$$I_m \ddot{\phi} - \tau_d - \sum_{i=1}^{4} \left( F_{B_i}^S (E_{i_x} S_{i_y} - E_{i_y} S_{i_x}) + F_{B_i}^N (\boldsymbol{E}_i \cdot \hat{\boldsymbol{S}}_i) \right) = 0$$

(7.44)

In these equations, the unit vectors are expressed component-wise as $\hat{\boldsymbol{S}}_i = [S_{i_x}, S_{i_y}]^T$ and $\hat{\boldsymbol{N}}_i = [-S_{i_y}, S_{i_x}]^T$. This ensures that the calculations for forces and moments are accurately represented in the coordinate directions.

Equations 7.42 through 7.44 represent the governing equations of motion for the manipulator. In these equations, $\boldsymbol{\mathcal{F}}_d = [f_{Dx}, f_{Dy}, \tau_d]^T$ denotes the disturbance wrench applied to the moving platform. The interaction forces between the limbs and the moving platform, $F_{B_i}^S$ and $F_{B_i}^N$, are derived from the limb dynamics as given in

Equations 7.39 and 7.38, respectively, and the vectors $\boldsymbol{E}_i$ and $\hat{\boldsymbol{S}}_i$ can be determined as follows:

$$\hat{\boldsymbol{S}}_i \quad = [\cos(\alpha_i) \quad \sin(\alpha_i)]^T, \tag{7.45}$$

$$\boldsymbol{E}_i \quad = [R_B \cos(\theta_{B_i} + \phi) \quad R_B \sin(\theta_{B_i} + \phi)]^T. \tag{7.46}$$

The Equations 7.42 through 7.44 can be expressed in an implicit vector form as:

$$\boldsymbol{f}(\ddot{\boldsymbol{\chi}}, \dot{\boldsymbol{\chi}}, \boldsymbol{\chi}, \boldsymbol{g}, \boldsymbol{\mathcal{F}}_d, \boldsymbol{\tau}) = \mathbf{0}. \tag{7.47}$$

As shown, the dynamic formulation of this parallel robot is represented in implicit form. This means the robot's dynamics are described through equations that include both the state variables and their derivatives without explicitly solving for the accelerations $\ddot{\boldsymbol{\chi}}$. This approach is typical for systems with constraints, such as parallel robots, where these constraints enforce relationships between the variables.

While the implicit dynamic formulation provides a comprehensive description of parallel robots' dynamics, incorporating constraints directly into the model, it also

introduces significant challenges. These challenges include numerical stability, computational complexity, and the necessity for specialized integration and control methods. Addressing these issues requires advanced techniques and careful implementation to ensure accurate and efficient operation of parallel robots.

To overcome these challenges, researchers have attempted to derive closed-form representations of dynamic formulations. For this means, one may choose a set of independent generalized coordinates and then express the dynamic equations in terms of these coordinates and their time derivatives. Typically, these coordinates represent the position of the moving platform. To achieve this the kinematic variables of internal components should be expressed as functions of the chosen generalized coordinates. If this process is executed accurately, the dynamic equations can be formulated in an explicit form:

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{\mathcal{F}}, \tag{7.48}$$

where $\boldsymbol{\chi}$ is the vector of generalized coordinates, $\boldsymbol{M}(\boldsymbol{\chi})$ is the system mass matrix, $\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is denoted as the Christoffel matrix, $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$ denotes the gravity vector, and $\boldsymbol{\mathcal{F}}$ stands for the generalized force.

Deriving the dynamic formulation in the form of Equation 7.48 is well-developed for serial manipulators [1, 188]. However, for parallel manipulators, the closed-loop

constraints make it challenging to derive explicit equations of motion in terms of independent generalized coordinates [184]. Although various methods have been proposed in the literature to derive dynamic equations for parallel manipulators, there is no single, standardized approach proposed for this problem.

In this domain, significant contributions have been made to the dynamic analysis of parallel manipulators using various methods. For example, Khalil and Ibrahim [189] developed a closed-form method for deriving dynamic equations using different Jacobian matrices and joint variables. Their approach offers explicit expressions for the manipulator's dynamics, facilitating both analysis and control. Additionally, Dasgupta and Choudhury [190] extended this work by establishing a comprehensive procedure using the Newton-Euler method to derive inverse dynamic formulations for parallel manipulators. Their method converts these formulations into closed-form solutions, simplifying the computational process and enhancing the accuracy of dynamic simulations.

In contrast, this book introduces a novel matrix-based manipulation technique that simplifies the derivation process by avoiding the cumbersome component-wise approach. This technique offers a more manageable way to handle dynamic equations, significantly reducing complexity while maintaining computational efficiency. The method is designed to address complex manipulator dynamics without needing symbolic software packages,

making it practical and efficient for real-time applications and advanced dynamic simulations. Details of this method are elaborated on in Section 7.5.4.

## 7.5 Case Study II: Stewart–Gough Platform

This section offers a comprehensive examination of the dynamic analysis of the Stewart–Gough platform, a widely utilized parallel manipulator. The detailed architecture and design of this manipulator are discussed in Section 4.6.1 and visually represented in Figure 4.16. In this setup, the spatial motion of the moving platform is generated by six identical piston-cylinder actuators, which provide precise control over the platform's position and orientation in three-dimensional space. For a thorough understanding of how the Stewart–Gough platform operates, a complete kinematic analysis is provided in Section 4.6. Additionally, Section 5.7 offers an in-depth Jacobian analysis, which plays a crucial role in understanding the velocity and force transmission characteristics of the manipulator.

### 7.5.1 Acceleration Analysis

The acceleration analysis of the limbs and the moving platform has been conducted to facilitate the application of the Newton–Euler formulation. This analysis aims to derive expressions for the linear and angular accelerations of the

limbs, denoted as $\ddot{\ell}_i$ and $\dot{\omega}_i$, respectively, in terms of the acceleration of the moving platform, $\ddot{\boldsymbol{\chi}} = [\dot{\boldsymbol{v}}_p, \dot{\boldsymbol{\omega}}]^T$.

To establish this relationship, we start by revisiting the velocity loop closure Equation 5.53. This equation can be rewritten as follows:

$$\boldsymbol{v}_p + \boldsymbol{\omega} \times \boldsymbol{b}_i = \dot{\ell}_i \, \hat{\boldsymbol{s}}_i + \ell_i \, (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i).$$

(7.49)

Here, $\boldsymbol{v}_p$ represents the linear velocity of the moving platform, $\boldsymbol{\omega}$ is its angular velocity, $\boldsymbol{b}_i$ denotes the position vector of the limb's attachment point, $\dot{\ell}_i$ is the rate of change of the limb's length, and $\hat{\boldsymbol{s}}_i$ is the unit vector along the limb's direction. The term $\ell_i, (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)$ accounts for the contribution of the limb's rotational motion to the velocity.

To begin, let's define an intermediate variable, $\boldsymbol{v}_{b_i}$, which represents the velocity of the point $b_i$ on the manipulator:

$$\boldsymbol{v}_{b_i} = \boldsymbol{v}_p + \boldsymbol{\omega} \times \boldsymbol{b}_i.$$

(7.50)

Here, $\boldsymbol{v}_p$ denotes the linear velocity of the moving platform, $\boldsymbol{\omega}$ is its angular velocity, and $\boldsymbol{b}_i$ is the position vector from the origin to point $b_i$. This definition helps to isolate the contribution of the platform's motion to the velocity of point $b_i$.

Next, to derive the angular velocity of the limbs, denoted as $\boldsymbol{\omega}_i$, we need to manipulate the velocity loop closure equation. Cross-multiply both sides of Equation 7.49 by $\hat{\boldsymbol{s}}_i$, which is the unit vector in the direction of the limb:

$$\hat{\boldsymbol{s}}_i \times \boldsymbol{v}_p + \hat{\boldsymbol{s}}_i \times (\boldsymbol{\omega} \times \boldsymbol{b}_i) = \ell_i \left(\hat{\boldsymbol{s}}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)\right). \tag{7.51}$$

In this equation, $\hat{\boldsymbol{s}}_i \times \boldsymbol{v}_p$ represents the cross product of the limb's direction with the platform's linear velocity, while $\hat{\boldsymbol{s}}_i \times (\boldsymbol{\omega} \times \boldsymbol{b}_i)$ accounts for the rotational contribution of the platform to the velocity at point $b_i$. The term $\ell_i \left(\hat{\boldsymbol{s}}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)\right)$ captures the contribution of the limb's angular velocity to the overall velocity at $b_i$. This manipulation is crucial for isolating and solving for the angular velocity $\boldsymbol{\omega}_i$ of the limbs.

It is important to note that, given the absence of actuation torque about the direction of $\hat{\boldsymbol{s}}_i$, the angular velocity and acceleration vectors of the limbs are orthogonal to $\hat{\boldsymbol{s}}_i$, provided the following assumptions about the platform are met:

- Both end joints of the limb are spherical. Notably, if the Stewart–Gough platform was equipped with universal joints at the base, the angular velocity and acceleration vectors of the limbs would have a component along $\hat{\boldsymbol{s}}_i$ [191].
- The limbs are symmetric relative to their axes.

- Frictional effects in the spherical joints are neglected.

Under these conditions, it can be concluded that the limbs are unable to rotate about their own axes. Consequently, the following relationships hold:

$$\boldsymbol{\omega}_i \cdot \hat{\boldsymbol{s}}_i = \dot{\boldsymbol{\omega}}_i \cdot \hat{\boldsymbol{s}}_i = 0. \tag{7.52}$$

This implies that the angular velocity vector $\boldsymbol{\omega}_i$ is perpendicular to the limb axis $\hat{\boldsymbol{s}}_i$. Therefore, it follows that:

$$\hat{\boldsymbol{s}}_i \times \left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right) = \boldsymbol{\omega}_i. \tag{7.53}$$

In other words, the cross product of $\hat{\boldsymbol{s}}_i$ with the vector resulting from the cross product of $\boldsymbol{\omega}_i$ and $\hat{\boldsymbol{s}}_i$ yields the angular velocity vector $\boldsymbol{\omega}_i$, reinforcing its perpendicularity to the limb axis $\hat{\boldsymbol{s}}_i$.

Taking into account Equations 7.50 and 7.53, the angular velocity of the limb, $\boldsymbol{\omega}_i$, is calculated using the following Equation:

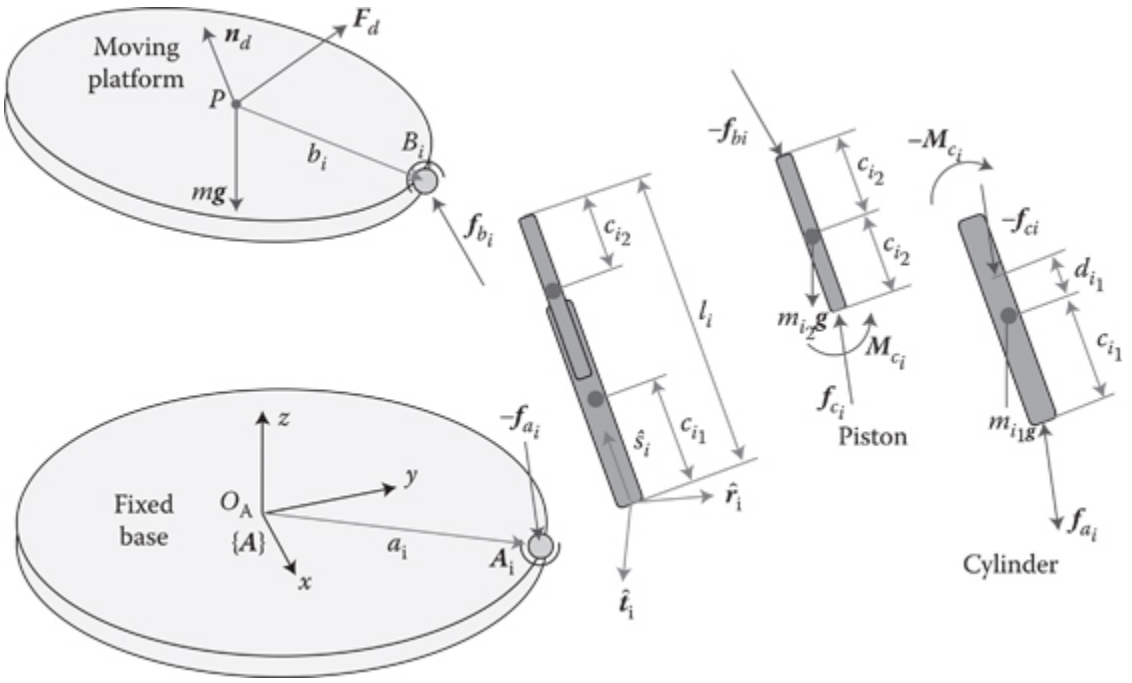
$$\begin{aligned} \boldsymbol{\omega}_i &= \frac{1}{\ell_i}\left(\hat{\boldsymbol{s}}_i \times \boldsymbol{v}_p + \hat{\boldsymbol{s}}_i \times \left(\boldsymbol{\omega} \times \boldsymbol{b}_i\right)\right) \\ &= \frac{1}{\ell_i}\left(\hat{\boldsymbol{s}}_i \times \boldsymbol{v}_{b_i}\right). \end{aligned} \tag{7.54}$$

In this equation, $\boldsymbol{v}_p$ represents the velocity of the moving platform, and $\boldsymbol{\omega} \times \boldsymbol{b}_i$ accounts for the contribution of the platform's angular velocity to the velocity at point $b_i$. By isolating the term $\hat{\boldsymbol{s}}_i \times \boldsymbol{v}_{b_i}$, we determine the angular velocity vector $\boldsymbol{\omega}_i$ of the limb relative to its length $\ell_i$.

Furthermore, as shown in Figure 7.3, the limb's piston-cylinder structure is divided into two distinct segments. Each segment has its mass, denoted by $m_{i_1}$ and $m_{i_2}$ respectively. To determine the position vectors of the centers of mass for these segments, we use:

$$\boldsymbol{p}_{i_1} = \boldsymbol{a}_i + c_{i_1} \hat{\boldsymbol{s}}_i \tag{7.55}$$

$$\boldsymbol{p}_{i_2} = \boldsymbol{a}_i + (\ell_i - c_{i_2}) \hat{\boldsymbol{s}}_i. \tag{7.56}$$




▶ Long Description for Figure 7.3

**FIGURE 7.3**

Free-body diagram of the limbs and the moving platform of a general Stewart–Gough manipulator. ⏎

Here, $\boldsymbol{a}_i$ represents the base attachment point of the limb, $c_{i_1}$ and $c_{i_2}$ are the distances from the base to the centers of mass of the first and second segments respectively, and $\hat{\boldsymbol{s}}_i$ is the direction vector of the limb.

To find the velocities of these centers of mass, differentiate the position vectors:

$$\boldsymbol{v}_{i_1} = c_{i_1}(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) \tag{7.57}$$

$$\boldsymbol{v}_{i_2} \quad = (\ell_i - c_{i_2})(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) + \dot{\ell}_i \hat{\boldsymbol{s}}_i.$$

(7.58)

In these equations, $\boldsymbol{v}_{i_1}$ represents the velocity of the center of mass of the first segment, which depends on the angular velocity of the limb and its distance $c_{i_1}$ from the base. Similarly, $\boldsymbol{v}_{i_2}$ is the velocity of the center of mass of the second segment, incorporating both the limb's angular velocity and its length change, $\dot{\ell}_i$.

Having established the linear velocities of the limb centers of mass and the angular velocities of the limbs, we now proceed to derive the acceleration terms. Start by differentiating Equation 7.49 with respect to time:

$$\begin{aligned}\boldsymbol{a}_p + \dot{\boldsymbol{\omega}} \times \boldsymbol{b}_i + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{b}_i) \quad &= \ddot{\ell}_i \hat{\boldsymbol{s}}_i + 2\dot{\ell}_i(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) + \ell_i(\dot{\boldsymbol{\omega}}_i \times \\ &\quad + \ell_i \left(\boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)\right).\end{aligned}$$

(7.59)

Similarly to how we defined the linear velocity in Equation 7.50, we can express the linear acceleration of point $\boldsymbol{b}_i$ as:

$$\boldsymbol{a}_{b_i} = \boldsymbol{a}_p + \dot{\boldsymbol{\omega}} \times \boldsymbol{b}_i + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{b}_i).$$

(7.60)

To derive an expression for the linear acceleration $\ddot{\boldsymbol{\ell}}_i$ and eliminate $\dot{\boldsymbol{\omega}}_i$, dot multiply both sides of Equation 7.59 by $\hat{\boldsymbol{s}}_i$ and rearrange:

$$\begin{aligned}\ddot{\boldsymbol{\ell}}_i \quad &= \boldsymbol{a}_p \cdot \hat{\boldsymbol{s}}_i + (\boldsymbol{b}_i \times \hat{\boldsymbol{s}}_i) \cdot \dot{\boldsymbol{\omega}} \\ &\quad + (\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \hat{\boldsymbol{b}}_i)) \cdot \hat{\boldsymbol{s}}_i - \ell_i(\boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i))) \cdot \hat{\boldsymbol{s}}_i.\end{aligned} \tag{7.61}$$

As noted in Equation 7.52, since the vector $\boldsymbol{\omega}_i$ is orthogonal to the limb axis $\hat{\boldsymbol{s}}_i$, it follows that:

$$(\boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)) \cdot \hat{\boldsymbol{s}}_i = -\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i. \tag{7.62}$$

Thus, Equation 7.61 can be simplified to:

$$\begin{aligned}\ddot{\boldsymbol{\ell}}_i \quad &= \boldsymbol{a}_{b_i} \times \hat{\boldsymbol{s}}_i + \ell_i(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i) \\ &= \boldsymbol{a}_p \cdot \hat{\boldsymbol{s}}_i + (\boldsymbol{b}_i \times \hat{\boldsymbol{s}}_i) \cdot \dot{\boldsymbol{\omega}} + (\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \hat{\boldsymbol{b}}_i)) \cdot \hat{\boldsymbol{s}}_i + \ell_i(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i\end{aligned} \tag{7.63}$$

This equation provides the expression for the linear acceleration $\ddot{\boldsymbol{\ell}}_i$ of the limb, incorporating contributions from the moving platform's acceleration, the changes in angular velocity, and the rotational effects of the limb.

To isolate $\dot{\boldsymbol{\omega}}_i$ and derive an expression for it, start by cross-multiplying both sides of Equation 7.59 with $\hat{\boldsymbol{s}}_i$:

$$\hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_p + \dot{\boldsymbol{\omega}} \times \boldsymbol{b}_i + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{b}_i)) = \ell_i \dot{\boldsymbol{\omega}}_i + 2\dot{\ell}_i \boldsymbol{\omega}_i.$$

(7.64)

Using the definition provided in Equation 7.60, this simplifies to:

$$\dot{\boldsymbol{\omega}}_i = \frac{1}{\ell_i}(\hat{\boldsymbol{s}}_i \times \boldsymbol{a}_{b_i} - 2\dot{\ell}_i \boldsymbol{\omega}_i).$$

(7.65)

This result provides the angular acceleration $\dot{\boldsymbol{\omega}}_i$ of the limb in terms of known quantities such as $\hat{\boldsymbol{s}}_i$, $\boldsymbol{a}_{b_i}$, and $\dot{\ell}_i$.

To complete the acceleration analysis for the manipulator, we need to derive expressions for the linear accelerations of the centers of mass of each limb. This requires differentiating the velocity expressions for the centers of mass of the limbs, given in Equations 7.57 and 7.58, with respect to time.

Differentiate these equations to obtain the following expressions for the accelerations of the centers of mass:

$$\boldsymbol{a}_{i_1} \quad = c_{i_1}(\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)),$$

(7.66)

$$\boldsymbol{a}_{i_2} = (\ell_i - c_{i_2})(\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i - (\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i)\hat{\boldsymbol{s}}_i) + 2\dot{\ell}_i(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) + \ddot{\ell}_i \hat{\boldsymbol{s}}_i \tag{7.67}$$

In these equations, $\boldsymbol{a}_{i_1}$ represents the linear acceleration of the center of mass of the first segment of the limb, and $\boldsymbol{a}_{i_2}$ denotes the linear acceleration of the center of mass of the second segment of the limb. The terms $c_{i_1}$ and $c_{i_2}$ are the distances from the pivot points to the centers of mass of the respective limb segments. The equations account for the effects of both linear and angular accelerations, providing a complete description of the limb's motion.

### 7.5.2 Dynamic Formulation of the Limbs

To derive the dynamic formulation of the Stewart–Gough platform, we begin by breaking down the manipulator into its fundamental components: the moving platform and the six identical limbs. This decomposition is illustrated in Figure 7.3. Each limb of the manipulator is modeled as comprising two segments: the cylinder and the piston. We assume that the velocities and accelerations of the centers of mass for these segments are known. Specifically, the centers of mass for the cylinder and piston are positioned at distances $c_{i_1}$ and $c_{i_2}$ above their respective foot points. The masses of these segments are denoted as $m_{i_1}$ and $m_{i_2}$. Furthermore, we assume that the pistons are symmetric

with respect to their axes, with their centers of mass located at the midpoint of their lengths.

The free-body diagrams for the limbs and the moving platform are detailed in Figure 7.3. In these diagrams, various forces are represented: the reaction forces at the fixed points $A_i$ are denoted by $\boldsymbol{f}_{a_i}$, while the internal forces at the moving points $B_i$ are indicated by $\boldsymbol{f}_{b_i}$. Additionally, the internal forces and moments between the cylinders and pistons are represented by $\boldsymbol{f}_{c_i}$ and $\boldsymbol{M}_{c_i}$, respectively. The figure also shows the gravitational forces acting on both the limb segments and the moving platform.

It is assumed that the only external disturbance applied to the moving platform is a wrench represented by $\boldsymbol{\mathcal{F}}_d = [\boldsymbol{F}_d, \boldsymbol{n}_d]^T$. To accurately define the internal and interaction forces on the limb segments in a component-wise manner, we introduce the unit vectors $\hat{\boldsymbol{t}}_i$ and $\hat{\boldsymbol{r}}_i$. These are defined as follows:

$$\hat{\boldsymbol{t}}_i = \frac{\hat{\boldsymbol{s}}_i \times \boldsymbol{a}_i}{\| \hat{\boldsymbol{s}}_i \times \boldsymbol{a}_i \|}, \tag{7.68}$$

$$\hat{\boldsymbol{r}}_i = \hat{\boldsymbol{s}}_i \times \hat{\boldsymbol{t}}_i. \tag{7.69}$$

Here, $\hat{\boldsymbol{t}}_i$ is the unit vector orthogonal to both the limb axis $\hat{\boldsymbol{s}}_i$ and the vector $\boldsymbol{a}_i$, normalized to ensure it has unit

length. Similarly, $\hat{\boldsymbol{r}}_i$ is defined as the cross product of $\hat{\boldsymbol{s}}_i$ and $\hat{\boldsymbol{t}}_i$, providing another orthogonal direction in the force analysis.

Based on these assumptions and definitions, the equations of motion for each limb segment can be derived as follows. Using Equations 7.16 and 7.18 as references, the Newton–Euler equations for the cylinder segments are formulated as:

$$\begin{aligned} \sum \boldsymbol{F}_{\text{ext}} &= m_{i_1} \boldsymbol{a}_{i_1} \\ &= \boldsymbol{f}_{a_i} - \boldsymbol{f}_{c_i} + m_{i_1} \boldsymbol{g} \end{aligned} \tag{7.70}$$

$$\begin{aligned} \sum {}^{c_{i_1}}\boldsymbol{n}_{\text{ext}} &= {}^{A}I_{c_{i_1}}\, \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times {}^{A}I_{c_{i_1}} \cdot \boldsymbol{\omega}_i \\ &= c_{i_1}(-\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{a_i}) + d_{i_1}(\hat{\boldsymbol{s}}_i \times -\boldsymbol{f}_{c_i}) - \boldsymbol{M}_{c_i} \end{aligned} \tag{7.71}$$

where $d_{i_1}$, is defined as:

$$d_{i_1} = \ell_i - c_{i_1} - 2c_{i_2}. \tag{7.72}$$

In this formulation, ${}^{A}I_{c_{i_1}}$ represents the inertia matrix of the cylinder, evaluated at a frame parallel to $\{\mathscr{A}\}$ and located at point $c_{i_1}$. Since the limb moves relative to this frame, the inertia matrix can be computed using the

rotation rule described in Equation 7.7 from the inertia matrix initially obtained about $\{\mathscr{A}_i\}$ attached to the cylinder at point $c_{i_1}$. The transformation is given by:

$$ {}^{A}\boldsymbol{I}_{c_{i_1}} = {}^{A}\boldsymbol{R}_{A_i}\, {}^{A_i}\boldsymbol{I}_{c_{i_1}}\, {}^{A}\boldsymbol{R}_{A_i}^{T}. $$

(7.73)

Moreover, $\Sigma \boldsymbol{F}_{\text{ext}}$ represents the total external forces applied to each cylinder. This summation accounts for all forces acting from outside the system, such as contact forces and applied loads. Similarly, $\Sigma^{c_{i_1}} \boldsymbol{n}_{\text{ext}}$ denotes the net external moments relative to the center of mass $c_{i_1}$ of each cylinder. These moments result from external torques or forces applied at points other than the center of mass. The linear acceleration at the center of mass, $\boldsymbol{a}_{i_1}$, limb angular velocity $\boldsymbol{\omega}_i$, and angular acceleration $\dot{\boldsymbol{\omega}}_i$ are given in Equations 7.66, 7.54, and 7.65, respectively. These equations describe the dynamics of the system in terms of its linear and rotational motions.

In a similar manner, the Newton–Euler equations for the piston segments are derived and expressed as follows:

$$ \begin{aligned} \sum \boldsymbol{F}_{\text{ext}} &= m_{i_2} \boldsymbol{a}_{i_2} \\ &= \boldsymbol{f}_{c_i} - \boldsymbol{f}_{b_i} + m_{i_2} \boldsymbol{g} \end{aligned} $$

(7.74)

$$\begin{aligned}\sum {}^{c_{i_2}}\boldsymbol{n}_{\text{ext}} &= {}^{A}I_{c_{i_2}}\,\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times {}^{A}I_{c_{i_2}} \cdot \boldsymbol{\omega}_i \\ &= c_{i_2}(-\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{c_i}) + c_{i_2}(\hat{\boldsymbol{s}}_i \times -\boldsymbol{f}_{b_i}) + \boldsymbol{M}_{c_i}\end{aligned} \tag{7.75}$$

where

$$ {}^{A}\boldsymbol{I}_{c_{i_2}} = {}^{A}\boldsymbol{R}_{A_i}\,{}^{A_i}\boldsymbol{I}_{c_{i_2}}\,{}^{A}\boldsymbol{R}_{A_i}^{T} \tag{7.76}$$

This equation indicates how the inertia matrix ${}^{A_i}\boldsymbol{I}_{c_{i_2}}$, which is expressed in the body frame $A_i$, transforms to the fixed frame $A$ using the rotation matrix ${}^{A}\boldsymbol{R}_{A_i}$.

The acceleration of the piston center of mass, $\boldsymbol{a}_{i_2}$, is detailed in Equation 7.67, describing the piston's linear motion. Since the moving frame $\{\mathscr{A}_i\}$ is attached to the limb $i$, and using the unit vectors $\hat{\boldsymbol{s}}_i$, $\hat{\boldsymbol{t}}_i$, and $\hat{\boldsymbol{r}}_i$ defined in Equations 7.68 and 7.69, the rotation matrix ${}^{A}\boldsymbol{R}_{A_i}$ can be represented as

$$ {}^{A}\boldsymbol{R}_{A_i} = \begin{matrix} \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{x}} & \hat{\boldsymbol{t}}_i \cdot \hat{\boldsymbol{x}} & \hat{\boldsymbol{r}}_i \cdot \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{y}} & \hat{\boldsymbol{t}}_i \cdot \hat{\boldsymbol{y}} & \hat{\boldsymbol{r}}_i \cdot \hat{\boldsymbol{y}} \\ \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{z}} & \hat{\boldsymbol{t}}_i \cdot \hat{\boldsymbol{z}} & \hat{\boldsymbol{r}}_i \cdot \hat{\boldsymbol{z}} \end{matrix}, \tag{7.77}$$

where $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ are the unit vectors of the fixed frame $A$. This matrix describes the orientation of the body frame $A_i$ with respect to the fixed frame $A$.

Additionally, if the limbs are symmetric, as previously mentioned, the inertia matrix ${}^{A_i}\boldsymbol{I}_{c_i}$ will be diagonal, simplifying the calculations:

$$
{}^{A_i}\boldsymbol{I}_{c_i} = \begin{matrix} I_{xx_i} & 0 & 0 \\ 0 & I_{xx_i} & 0 \\ 0 & 0 & I_{zz_i} \end{matrix} .
$$

(7.78)

Here, $I_{xx_i}$ and $I_{zz_i}$ are the moments of inertia about the $x$ and $z$ axes of the limb $i$, respectively, illustrating the simplified form of the inertia matrix when the system exhibits symmetry.

To streamline the analysis and eliminate internal forces from Equations 7.70 through 7.75, we begin by utilizing Equation 7.70 to express the acceleration force $\boldsymbol{f}_{a_i}$ in terms of the force $\boldsymbol{f}_{c_i}$ acting at the center of mass. The relationship is given by:

$$
\boldsymbol{f}_{\boldsymbol{a}_i} = m_{i_1}\boldsymbol{a}_{i_1} + \boldsymbol{f}_{c_i} - m_{i_1}\boldsymbol{g}.
$$

(7.79)

In this equation, $m_{i_1}\boldsymbol{a}_{i_1}$ represents the inertial force due to the linear acceleration of the center of mass, $\boldsymbol{f}_{c_i}$ is the force

at the center of mass, and $m_{i_1}\boldsymbol{g}$ denotes the gravitational force acting on the system. By substituting this expression for $\boldsymbol{f}_{a_i}$ into Equation 7.75, we can manipulate the equation to isolate the moment $\boldsymbol{M}_{c_i}$.

After performing the necessary algebraic manipulations, we derive the following expression for $\boldsymbol{M}_{c_i}$:

$$\begin{aligned}\boldsymbol{M}_{c_i} &= -c_{i_1}\hat{\boldsymbol{s}}_i \times \left(m_{i_1}\boldsymbol{a}_{i_1} + \boldsymbol{f}_{c_i} - m_{i_1}\boldsymbol{g}\right) + d_{i_1}\left(\hat{\boldsymbol{s}}_i \times -\boldsymbol{f}_{c_i}\right) \\ &\quad -{}^{A}\boldsymbol{I}_{c_{i_1}}\dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i \times {}^{A}\boldsymbol{I}_{c_{i_1}}\boldsymbol{\omega}_i.\end{aligned} \tag{7.80}$$

In this expression, the term $-c_{i_1}\hat{\boldsymbol{s}}_i \times \left(m_{i_1}\boldsymbol{a}_{i_1} + \boldsymbol{f}_{c_i} - m_{i_1}\boldsymbol{g}\right)$ accounts for the moments due to the inertial forces and the forces at the center of mass, while $d_{i_1}\left(\hat{\boldsymbol{s}}_i \times -\boldsymbol{f}c_i\right)$ represents the contribution from the actuator force. The remaining terms account for the contributions of the inertia matrix and angular velocities.

To further isolate the actuator force, $\tau_i\hat{\boldsymbol{s}}_i$, from the internal force between the cylinder and piston, we rewrite $\boldsymbol{f}_{c_i}$ as:

$$\boldsymbol{f}_{c_i} = \boldsymbol{f}_{c_i}^{n} + \boldsymbol{f}_{c_i}^{s} = \boldsymbol{f}_{c_i}^{n} + \tau_i\hat{\boldsymbol{s}}_i, \tag{7.81}$$

where $\boldsymbol{f}_{c_i}^{n}$ represents the component of the force perpendicular to $\hat{\boldsymbol{s}}_i$, and $\tau_i\hat{\boldsymbol{s}}_i$ is the actuator force along the direction of $\hat{\boldsymbol{s}}_i$. Substituting this revised expression for $\boldsymbol{f}_{c_i}$,

along with $\boldsymbol{a}_{c_{i_1}}$ and $d_{i_1}$, into Equation 7.5.2, leads to an updated relation for $\boldsymbol{M}_{c_i}$ in terms of $\boldsymbol{f}_{c_i}^n$ and the actuator force $\tau_i$. Utilizing vector triple product identities and the fact that $\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{c_i} = \boldsymbol{f}_{c_i}^n$, the final formulation for $\boldsymbol{M}_{c_i}$ is given by:

$$\begin{aligned}\boldsymbol{M}_{c_i} &= -m_{i_1} c_{i_1}^2 \dot{\boldsymbol{\omega}}_i - {}^A\boldsymbol{I}_{c_{i_1}} \dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i \times {}^A\boldsymbol{I}_{c_{i_1}} \boldsymbol{\omega}_i - (\ell_i - 2c_{i_2}) \left(\hat{\boldsymbol{s}}_i \right. \\ &\quad + m_{i_1} c_{i_1} \hat{\boldsymbol{s}}_i \times \boldsymbol{g}.\end{aligned} \tag{7.82}$$

This expression captures the effects of various forces and moments, including the contributions from the center of mass, inertia, and actuator forces.

Lastly, to facilitate further calculations, the simplified form of $\hat{\boldsymbol{s}}_i \times \boldsymbol{M}_{c_i}$ is provided:

$$\begin{aligned}\hat{\boldsymbol{s}}_i \times \boldsymbol{M}_{c_i} &= -m_{i_1} c_{i_1}^2 \left(\hat{\boldsymbol{s}}_i \times \dot{\boldsymbol{\omega}}_i\right) - \hat{\boldsymbol{s}} \times \left({}^A\boldsymbol{I}_{c_{i_1}} \dot{\boldsymbol{\omega}}_i\right) \\ &\quad - \left(\hat{\boldsymbol{s}}_i . \left({}^A\boldsymbol{I}_{c_{i_1}} \boldsymbol{\omega}_i\right)\right) \boldsymbol{\omega}_i + (\ell_i - 2c_{i_2}) \boldsymbol{f}_{c_i}^n + m_{i_1} c_{i_1} \hat{\boldsymbol{s}}_i\end{aligned} \tag{7.83}$$

To express $\boldsymbol{f}_{b_i}$ in vector form, we decompose it into two distinct components:

$$\boldsymbol{f}_{\boldsymbol{b}_i} = \boldsymbol{f}_{\boldsymbol{b}_i}^n + \boldsymbol{f}_{\boldsymbol{b}_i}^s,$$

(7.84)

where $\boldsymbol{f}_{b_i}^s$ represents the component of $\boldsymbol{f}_{b_i}$ along $\hat{\boldsymbol{s}}_i$, and $\boldsymbol{f}_{b_i}^n$ denotes the component normal to $\hat{\boldsymbol{s}}_i$. This decomposition facilitates the effective elimination of internal forces, thus simplifying the dynamic equations for each limb. It is important to note that while $\boldsymbol{f}_{b_i}^n$ and $\boldsymbol{f}_{c_i}^n$ are both normal to $\hat{\boldsymbol{s}}_i$, they are not necessarily parallel to each other.

By substituting $\boldsymbol{f}_{c_i}$, $\boldsymbol{f}_{b_i}$, and $\boldsymbol{a}_{c_{i_2}}$ into Equation 7.74 and simplifying, we derive the following equation for the piston:

$$\begin{aligned}\boldsymbol{f}_{\boldsymbol{b}_i}^n - \boldsymbol{f}_{c_i}^n + \boldsymbol{f}_{\boldsymbol{b}_i}^s \quad &= \tau_i \hat{\boldsymbol{s}}_i + m_{i_2}\boldsymbol{g} - m_{i_2}((\ell_i - c_{i_2}) \\ &\quad \times \left(\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i - (\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i)\hat{\boldsymbol{s}}_i\right) + 2\dot{l}_i \left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right) +\end{aligned}$$

(7.85)

Equation 7.85 illustrates how decomposing $\boldsymbol{f}_{b_i}$ into its normal and tangential components significantly simplifies the dynamic equations. By taking the dot product of Equation 7.85 with $\hat{\boldsymbol{s}}_i$, we can isolate $\boldsymbol{f}_{b_i}^s$ as a function of the actuator force and the kinematic parameters of the limb:

$$\boldsymbol{f}_{b_i}^s = \tau_i \hat{\boldsymbol{s}}_i + m_{i_2}\left(\ell_i - c_{i_2}\right)(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i)\hat{\boldsymbol{s}}_i - m_{i_2}\ddot{\ell}_i \hat{\boldsymbol{s}}_i + m_{i_2}\left(\boldsymbol{g} \cdot \hat{\boldsymbol{s}}_i\right)\hat{\boldsymbol{s}}$$

(7.86)

This approach simplifies the calculation of $\boldsymbol{f}_{b_i}^s$, providing a clearer view of how the actuator force and limb kinematics influence the system.

To derive additional insights, cross-multiply $\hat{\boldsymbol{s}}_i$ with Equations 7.70 and 7.74, and rearrange them to find:

$$\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{c_i} \quad = \hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{b_i} + m_{i_2} \hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_{i_2} - \boldsymbol{g}) \tag{7.87}$$

$$\begin{aligned} \hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{a_i} \quad &= \hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{c_i} + m_{i_1} \hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_{i_1} - \boldsymbol{g}) \\ &= \hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{b_i} + m_{i_2} \hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_{i_2} - \boldsymbol{g}) + m_{i_1} \hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_{i_1} - \boldsymbol{g}) \end{aligned} \tag{7.88}$$

Next, add Equations 7.71 and 7.75, and substitute $\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{c_i}$ and $\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{a_i}$:

$$\begin{aligned} &\left({}^A\boldsymbol{I}_{c_{i_1}} + {}^A\boldsymbol{I}_{c_{i_2}}\right)\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \left({}^A\boldsymbol{I}_{c_{i_1}} + {}^A\boldsymbol{I}_{c_{i_2}}\right)\boldsymbol{\omega}_i \\ &\quad = -c_{i_1}(\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{a_i}) - (\ell_i - c_{i_1} - c_{i_2})(\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{c_i}) - c_{i_2}(\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{b_i} \\ &\quad = -c_{i_1} m_{i_1} \hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_{i_1} - \boldsymbol{g}) - (\ell_i - c_{i_2}) m_{i_2} \hat{\boldsymbol{s}}_i \times (\boldsymbol{a}_{i_2} - \boldsymbol{g}) - \ell \end{aligned} \tag{7.89}$$

This equation simplifies to:

$$
\begin{aligned}
{}^{A}\boldsymbol{I}_{\mathrm{eq}}\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times {}^{A}\boldsymbol{I}_{\mathrm{eq}}\boldsymbol{\omega}_i \quad &= -c_{i_1} m_{i_1} \hat{\boldsymbol{s}}_i \times \boldsymbol{a}_{i_1} - (\ell_i - c_{i_2}) m_{i_2} \hat{\boldsymbol{s}}_i \\
&\times \boldsymbol{a}_{i_2} - \ell_i (\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{b_i}) + \ell_i m_{g_e} \hat{\boldsymbol{s}}_i \times \boldsymbol{g},
\end{aligned}
\tag{7.90}
$$

where

$$
{}^{A}\boldsymbol{I}_{\mathrm{eq}} = \left( {}^{A}\boldsymbol{I}_{c_{i_1}} + {}^{A}\boldsymbol{I}_{c_{i_2}} \right), \quad m_{g_e} = \frac{1}{\ell_i} (m_{i_1} c_{i_1} + m_{i_2} (\ell_i - c_{i_2})).
\tag{7.91}
$$

The dynamic equation for the limbs can be further simplified under the assumption of symmetry in limb segments. Through appropriate manipulations, it becomes evident that the left-hand side of Equation 7.90 can be significantly simplified. When both the cylinder and piston are symmetric, their inertia matrices relative to the frame $\{\mathscr{A}_i\}$ (attached to the limb) are diagonal. Consequently, the equivalent moment of inertia ${}^{A_i}\boldsymbol{I}_{\mathrm{eq}}$, which is the sum of the inertia matrices of these two segments, is also diagonal and can be expressed as:

$$
{}^{A_i}\boldsymbol{I}_{\mathrm{eq}} = {}^{A_i}\boldsymbol{I}_{c_{i_1}} + {}^{A_i}\boldsymbol{I}_{c_{i_2}} = \begin{matrix} I_{xx_i} & 0 & 0 \\ 0 & I_{xx_i} & 0 \\ 0 & 0 & I_{zz_i} \end{matrix}, \quad \begin{aligned} I_{xx_i} &= I_{xx_{i_1}} + I_{} \\ I_{zz_i} &= I_{zz_{i_1}} + I_{z} \end{aligned}
\tag{7.92}
$$

Furthermore, since $\boldsymbol{\omega}_i$ and $\dot{\boldsymbol{\omega}}_i$ have no components along the $\hat{\boldsymbol{z}}_i$ direction, we have:

$$\begin{aligned} {}^{A}\boldsymbol{I}_{\mathrm{eq}}\dot{\boldsymbol{\omega}}_i &= \left( {}^{A}\boldsymbol{R}_{A_i}\, {}^{A_i}\boldsymbol{I}_{\mathrm{eq}}\, {}^{A}\boldsymbol{R}_{A_i}^{T} \right) {}^{A}\dot{\boldsymbol{\omega}}_i = {}^{A}\boldsymbol{R}_{A_i}\, {}^{A_i}\boldsymbol{I}_{\mathrm{eq}}\, {}^{A_i}\dot{\boldsymbol{\omega}}_i \\ &= {}^{A}\boldsymbol{R}_{A_i} I_{xx_i}\, {}^{A_i}\dot{\boldsymbol{\omega}}_i = I_{xx_i}\, {}^{A}\dot{\boldsymbol{\omega}}_i. \end{aligned} \tag{7.93}$$

Similarly, it can be demonstrated that:

$$\boldsymbol{\omega}_i \times {}^{A}\boldsymbol{I}_{\mathrm{eq}}\dot{\boldsymbol{\omega}}_i = 0. \tag{7.94}$$

This relation allows for a significant simplification of Equation 7.90, reducing it to:

$$I_{xx_i}\dot{\boldsymbol{\omega}}_i = -\ell_i(\hat{\boldsymbol{s}}_i \times \boldsymbol{f}_{b_i}) - c_{i_1} m_{i_1} \hat{\boldsymbol{s}}_i \times \boldsymbol{a}_{i_1} - (\ell_i - c_{i_2}) m_{i_2} \hat{\boldsymbol{s}}_i \times \boldsymbol{a} \tag{7.95}$$

To isolate $\boldsymbol{f}_{b_i}^{n}$, cross-multiply Equation 7.95 by $\hat{\boldsymbol{s}}_i$, then reorder and simplify to obtain:

$$\boldsymbol{f}_{b_i} = \frac{1}{\ell_i}(I_{xx_i} + \ell_i^2 m_{c_e})\hat{\boldsymbol{s}}_i \times \dot{\boldsymbol{\omega}}_i + \frac{2}{\ell_i} m_{i_2} c_{i_2} \dot{\ell}_i \hat{\boldsymbol{s}}_i \times \boldsymbol{\omega}_i - m_{g_e} \hat{\boldsymbol{s}}_i \times \tag{7.96}$$

Here, $m_{c_e}$ is defined as

$$m_{c_e} = \frac{1}{\ell_i^2}\left(m_{i_1} c_{i_1}^2 + m_{i_2} c_{i_2}^2\right). \tag{7.97}$$

Equations 7.86 and 7.96 provide the expressions for the two components of the interaction force $\boldsymbol{f}_{b_i}$ in terms of the kinematic parameters of each limb. This interaction force $\boldsymbol{f}_{b_i}$ is crucial for the Newton–Euler formulation of the moving platform, which will be elaborated in the following section. If there is a need to evaluate other interacting forces and torques, such as $\boldsymbol{f}_{a_i}$, $\boldsymbol{f}_{c_i}$, and $\boldsymbol{M}_{c_i}$ for design purposes, similar methods can be applied as used for $\boldsymbol{f}_{b_i}$.

### 7.5.3 Dynamic Formulation of the Moving Platform

Assume that the center of mass of the moving platform is positioned at the point $P$. The platform has a total mass of $m$ and its moment of inertia is denoted by ${}^{A}\boldsymbol{I}_P$. In addition, the platform is subjected to gravitational forces and external disturbance wrenches. These disturbances are represented by $\boldsymbol{\mathcal{F}}_d = [\boldsymbol{F}_d, \boldsymbol{n}_d]^T$, where $\boldsymbol{F}_d$ is the force component and $\boldsymbol{n}_d$ is the moment component, as illustrated in the free-body diagram (Figure 7.3).

The Newton–Euler equations that govern the dynamics of the moving platform can be expressed as:

$$\sum \boldsymbol{F}_{\text{ext}} \quad = \sum_{i=1}^{6} \boldsymbol{f}_{b_i} + m\boldsymbol{g} + \boldsymbol{F}_d = m\boldsymbol{a}_p$$

(7.98)

$$\sum {}^{p}\boldsymbol{n}_{\text{ext}} \quad = \boldsymbol{n}_d + \sum_{i=1}^{6} \boldsymbol{b}_i \times \boldsymbol{f}_{b_i} = {}^{A}\boldsymbol{I}_P\, \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times {}^{A}\boldsymbol{I}_P \boldsymbol{\omega},$$

(7.99)

Here, ${}^{A}\boldsymbol{I}_P$ represents the moment of inertia of the platform relative to the fixed frame $\{\mathscr{A}\}$. This moment of inertia can be computed using the following relation:

$${}^{A}\boldsymbol{I}_P = {}^{A}\boldsymbol{R}_B {}^{B}\boldsymbol{I}_P {}^{A}\boldsymbol{R}_B{}^{T}.$$

(7.100)

To account for the dynamics more concisely, these equations can be rewritten in an implicit form as follows:

$$m(\boldsymbol{a}_p - \boldsymbol{g}) - \boldsymbol{F}_d - \sum_{i=1}^{6} \boldsymbol{f}_{b_i} = \boldsymbol{0}$$

(7.101)

$$ {}^{A}\boldsymbol{I}_{P}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times {}^{A}\boldsymbol{I}_{P}\boldsymbol{\omega} - \boldsymbol{n}_{d} - \sum_{i=1}^{6} \boldsymbol{b}_{i} \times \boldsymbol{f}_{b_{i}} = \mathbf{0}. $$

(7.102)

In these equations, $\boldsymbol{a}_{p}$ denotes the acceleration of the platform's center of mass, and $\boldsymbol{g}$ is the gravitational acceleration vector. The term $\mathbf{0}$ represents a zero vector indicating equilibrium. Equation 7.101 describes the balance of forces acting on the platform, ensuring that the sum of external forces and torques equals the product of the mass and acceleration. Equation 7.102 describes the balance of moments or torques, accounting for the contributions from the moment of inertia, angular velocity, and external disturbance moments.

These two vector equations, 7.101 and 7.102, constitute the fundamental dynamic formulation for the Stewart–Gough platform. Here, $\boldsymbol{\mathcal{F}}_{d} = [\boldsymbol{F}_{d}, \boldsymbol{n}_{d}]^{T}$ represents the external disturbance wrench acting on the moving platform, while $\boldsymbol{f}_{b_{i}}$ denotes the interaction forces between the platform and its limbs. These interaction forces are detailed in Equation 7.96. The implicit vector form of these equations can be written as:

$$ \boldsymbol{f}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}}, \boldsymbol{\mathcal{F}}_{d}, \boldsymbol{\tau}) = \mathbf{0} $$

(7.103)

where

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{\theta} \end{bmatrix}$$

(7.104)

is the vector of motion variables for the moving platform. This vector includes the linear position of point $P$, denoted by $\boldsymbol{x}_p = [x_p \;\; y_p \;\; z_p]^T$, which specifies the platform's position in space. It also includes the platform's orientation, represented by screw coordinates:

$$\boldsymbol{\theta} = \theta[s_x \quad s_y \quad s_z]^T = [\theta_x \quad \theta_y \quad \theta_z]^T.$$

(7.105)

Unlike Euler angles, screw coordinates $\boldsymbol{\theta}$ form a vector, and its derivative is also a vector that represents the angular velocity of the moving platform, $\dot{\boldsymbol{\theta}} = \boldsymbol{\omega}$. Therefore, $\dot{\boldsymbol{\chi}} = [\boldsymbol{v}_p, \boldsymbol{\omega}]^T$ describes the moving platform's twist, combining linear velocity $\boldsymbol{v}_p$ and angular velocity $\boldsymbol{\omega}$. Furthermore, $\boldsymbol{\mathcal{F}}_d = [\boldsymbol{F}_d, \boldsymbol{n}_d]^T$ represents the external disturbance wrench, while $\boldsymbol{\tau} = [\tau_1 \;\; \tau_2 \;\; \ldots \;\; \tau_6]^T$ denotes the vector of actuator forces applied to the platform. These equations can be used for simulations in both forward and inverse dynamics, similar to the approach outlined in Section 7.4.

### 7.5.4 Closed-Form Dynamics

Although the dynamic formulation represented by Equation 7.103 can be utilized to simulate the inverse dynamics of the Stewart–Gough platform, its implicit nature makes it less suitable for detailed dynamic analysis and precise control applications. The complexity and lack of clarity inherent in the implicit formulation can pose significant challenges for engineers and researchers working on the dynamic modeling and control of such platforms. Therefore, in this section, we present a more accessible and efficient method to convert the dynamic formulation into a closed form. This closed-form representation is comparable to that typically achieved for serial manipulators, providing a clearer, more straightforward approach that facilitates better understanding and implementation in practical scenarios [192].

## Closed-Form Dynamics of the Limbs

To derive a closed-form dynamic formulation for the Stewart–Gough platform as:

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{\mathcal{F}}, \tag{7.106}$$

we begin by introducing an intermediate generalized coordinate $\boldsymbol{x}_i$, which represents the position of point $b_i$. This coordinate is crucial for harmonizing the dynamic formulations of the limb and the moving platform, enabling the derivation of a closed-form structure for the entire

manipulator. By using $\boldsymbol{x}_i$, we can create a unified representation that simplifies the overall analysis and control of the system.

Next, we manipulate the dynamic equations of each limb to convert them into the desired closed form. This involves a systematic process where we introduce relations to substitute complex kinematic parameters such as $\dot{\boldsymbol{\omega}}_i$ (angular velocity), $\ddot{\boldsymbol{\omega}}_i$ (angular acceleration), and $\ddot{\ell}_i$ (linear acceleration) with the intermediate generalized coordinate $\boldsymbol{x}_i$ and its time derivatives. By doing this, we can express the dynamics in terms of more manageable and interpretable quantities.

This substitution process requires transforming vector multiplications, such as dot and cross products, into their corresponding matrix multiplications. This transformation not only simplifies the mathematical operations but also ensures consistency in the formulation. It is important to note that for any three arbitrary vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, the following equalities hold, which are fundamental to our transformation:

$$(\boldsymbol{a} \cdot \boldsymbol{b})\boldsymbol{c} = \big(\boldsymbol{a}^T\boldsymbol{b}\big)\boldsymbol{c} = \boldsymbol{c}\big(\boldsymbol{a}^T\boldsymbol{b}\big) = \big(\boldsymbol{c}\boldsymbol{a}^T\big)\boldsymbol{b} = \boldsymbol{c}\boldsymbol{a}^T\boldsymbol{b}. \tag{7.107}$$

Similarly, for the cross product of $\boldsymbol{a} \times \boldsymbol{b}$, we can employ matrix multiplication:

$$\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{a}_\times \boldsymbol{b} = -\boldsymbol{b}_\times \boldsymbol{a}$$

(7.108)

In this equation, $\boldsymbol{a}_\times$ and $\boldsymbol{b}_\times$ are skew-symmetric matrices derived from the elements of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. These vectors are represented as $\boldsymbol{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ and $\boldsymbol{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}^T$, respectively. The skew-symmetric matrix for $\boldsymbol{a}$ is given by:

$$\boldsymbol{a}^\times \text{ or } \boldsymbol{a}_\times \doteq \begin{matrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{matrix} .$$

(7.109)

As defined in Equation 3.35, this matrix representation can be used to replace vector cross products with matrix multiplication. This transformation is essential for simplifying the mathematical operations involved in dynamic analysis.

To relate the intermediate generalized coordinate $\boldsymbol{x}_i$ and its derivatives to the limb kinematic variables, we can use the following relations:

$$\begin{aligned} \dot{l}_i &= \hat{\boldsymbol{s}}_i^T \dot{\boldsymbol{x}}_i && \Rightarrow \dot{l}_i \hat{\boldsymbol{s}}_i = \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T \dot{\boldsymbol{x}}_i \\ \boldsymbol{\omega}_i &= \frac{1}{\ell_i} \hat{\boldsymbol{s}}_{i\times} \dot{\boldsymbol{x}}_i && \Rightarrow |\boldsymbol{\omega}_i|_2^2 \hat{\boldsymbol{s}}_i = \frac{-1}{{\ell_i}^2} \hat{\boldsymbol{s}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{s}}_{i\times}^2 \dot{\boldsymbol{x}}_i. \end{aligned}$$

(7.110)

These equations help in expressing the kinematic parameters in terms of $\boldsymbol{x}_i$ and its time derivatives, thereby simplifying the formulation.

Furthermore, for linear and angular accelerations, we can derive the following relationships:

$$\begin{aligned} \ddot{l}_i - \ell_i \boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i = \hat{\boldsymbol{s}}_i^T \ddot{\boldsymbol{x}}_i \quad &\Rightarrow \ddot{l}_i \hat{\boldsymbol{s}}_i - \ell_i \left(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i\right) \hat{\boldsymbol{s}}_i = \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T \ddot{\boldsymbol{x}}_i \\ l_i \dot{\boldsymbol{\omega}}_i + 2\dot{l}_i \boldsymbol{\omega}_i = \hat{\boldsymbol{s}}_{i\times} \ddot{\boldsymbol{x}}_i \quad &\Rightarrow \dot{\boldsymbol{\omega}}_i = \frac{1}{\ell_i} \Big( \hat{\boldsymbol{s}}_{i\times} \ddot{\boldsymbol{x}}_i - 2\dot{l}_i \boldsymbol{\omega}_i \Big). \end{aligned} \tag{7.111}$$

Transformations 7.110 and 7.111 are key tools for simplifying the process of deriving closed-form dynamic formulations of the limbs from their implicit formulations. By substituting kinematic parameters into the dynamic Equations 7.86 and 7.96 through these simplifications, the forces $\boldsymbol{f}_{b_i}^s$ and $\boldsymbol{f}_{b_i}^n$ can be converted into the following forms:

$$\boldsymbol{f}_{b_i}^s - \tau_i \hat{\boldsymbol{s}}_i \quad = -m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T \ddot{\boldsymbol{x}}_i + \frac{m_{i_2} c_{i_2}}{\ell_i^2} \hat{\boldsymbol{s}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{s}}_{i\times}^2 \dot{\boldsymbol{x}}_i + m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T \boldsymbol{g} \tag{7.112}$$

$$\boldsymbol{f}_{b_i}^n \quad = \frac{1}{\ell_i^2} (I_{xx_i} + \ell_i^2 m_{c_e}) \hat{\boldsymbol{s}}_{i\times}^2 \ddot{\boldsymbol{x}}_i + \frac{2}{\ell_i} m_{c_o} \dot{\ell}_i \boldsymbol{s}_{i\times}^2 \dot{\boldsymbol{x}}_i - m_{g_e} \boldsymbol{s}_{i\times}^2 \boldsymbol{g}, \tag{7.113}$$

In these equations:

$$m_{c_o} = \frac{1}{\ell_i} m_{i_2} c_{i_2} - \frac{1}{\ell_i^2}(I_{xx_i} + \ell_i^2 m_{c_e}), \quad m_{g_e} = \frac{1}{\ell_i}(m_{i_1} c_{i_1} + m_{i_2}$$

(7.114)

These expressions allow us to convert complex limb dynamics into a more straightforward, closed-form representation, facilitating easier analysis and control of the Stewart-Gough platform. The use of intermediate coordinates and matrix transformations significantly simplifies the dynamic equations, making them more practical for implementation in control algorithms and dynamic simulations.

Note that, as illustrated in the free-body diagram of the piston in Figure 7.3, $-\boldsymbol{f}_{b_i}$ is considered to act on the piston. This consideration is crucial for accurately capturing the interaction forces within the system. To derive $-\boldsymbol{f}_{b_i}$, we first need to add Equations 7.112 and 7.113, which represent different components of the force acting on the limb. Once these equations are combined, the resulting expression is then negated to reflect the direction of $-\boldsymbol{f}_{b_i}$.

Next, we need to factor out $\ddot{\boldsymbol{x}}_i$ and $\dot{\boldsymbol{x}}_i$ from the combined equation. This step is essential as it allows us to isolate the terms involving acceleration and velocity, leading to a more compact and interpretable form. By doing so, we derive a single equation that describes the dynamics of limb *i* based

on the intermediate generalized coordinate $\boldsymbol{x}_i$. This equation can be expressed in a closed form as follows:

$$\boldsymbol{M}_i\ddot{\boldsymbol{x}}_i + \boldsymbol{C}_i\dot{\boldsymbol{x}}_i + \boldsymbol{G}_i = \boldsymbol{F}_i,$$

(7.115)

In this formulation, the corresponding mass matrix $\boldsymbol{M}_i$, the Coriolis and centrifugal matrix $\boldsymbol{C}_i$, and the gravity vector $\boldsymbol{G}_i$ encapsulate the dynamic characteristics of the limb. These components can be simplified into the following forms:

$$\begin{aligned}
\boldsymbol{M}_i &= m_{i_2}\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T - \frac{1}{\ell_i^2}I_{xx_i}\hat{\boldsymbol{s}}_{i\times}^2 - m_{c_e}\hat{\boldsymbol{s}}_{i\times}^2 \\
\boldsymbol{C}_i &= -\frac{2}{\ell_i}m_{c_o}\dot{\ell}_i\hat{\boldsymbol{s}}_{i\times}^2 - \frac{1}{\ell_i^2}m_{i_2}c_{i_2}\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i\times}^2 \\
\boldsymbol{G}_i &= \left(m_{g_e}\hat{\boldsymbol{s}}_{i\times}^2 - m_{i_2}\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\right)\boldsymbol{g} \\
\boldsymbol{F}_i &= -\boldsymbol{f}_{b_i} + \tau_i\hat{\boldsymbol{s}}_i,
\end{aligned}$$

(7.116)

The parameters involved in these equations include the mass matrix $\boldsymbol{M}_i$, which accounts for the inertia of the limb, the Coriolis and centrifugal matrix $\boldsymbol{C}_i$, which captures the effects of velocity-dependent forces, and the gravity vector $\boldsymbol{G}_i$, which represents the influence of gravitational forces. These matrices are derived from the following intermediate parameters:

$$\boldsymbol{I}_{\text{eq}} = \left({}^{A}\boldsymbol{I}_{c_{i_1}} + {}^{A}\boldsymbol{I}_{c_{i_2}}\right), \quad m_{c_e} = \frac{1}{\ell_i^2}\left(m_{i_1}c_{i_1}^2 + m_{i_2}c_{i_2}^2\right),$$

$$m_{c_o} = \frac{1}{\ell_i}m_{i_2}c_{i_2} - \frac{1}{\ell_i^2}(I_{xx_i} + \ell_i^2 m_{c_e}), \quad m_{g_e} = \frac{1}{\ell_i}(m_{i_1}c_{i_1} + m_i$$

(7.117)

By structuring the dynamics of each limb in this manner, we achieve a closed-form dynamic representation that simplifies both analysis and control of the Stewart-Gough platform. This approach ensures that complex interactions within the system are captured accurately while maintaining a manageable form for practical application. Moreover, this closed-form representation is particularly valuable for real-time applications where computational efficiency is crucial. By reducing the complexity of the dynamic equations, we can achieve faster computation times, enabling more responsive and precise control of the Stewart-Gough platform.

## Closed-Form Dynamics of the Moving Platform

In this section, the equations of motion for the moving platform, originally given in Equations 7.101 and 7.102, are converted into the following closed-form representation:

$$\boldsymbol{M}_p\ddot{\boldsymbol{\chi}} + \boldsymbol{C}_p\dot{\boldsymbol{\chi}} + \boldsymbol{G}_p = \boldsymbol{\mathcal{F}}_p,$$

(7.118)

Here, $\boldsymbol{\chi}$ represents a vector of generalized coordinates that describe both the position and orientation of the moving platform. As previously detailed, the motion of the moving platform is captured by six coordinates. The first three coordinates, denoted by $\boldsymbol{x}_p$, represent the linear position of the platform, while the remaining three coordinates, $\boldsymbol{\theta}$, represent its angular orientation.

To accurately describe the angular motion, screw coordinates, as defined in Equation 7.105, are preferred. In screw coordinates, $\boldsymbol{\theta}$ is a vector, and its time derivative, $\dot{\boldsymbol{\theta}}$, directly represents the angular velocity of the moving platform, denoted as $\boldsymbol{\omega}$. Furthermore, the second time derivative of $\boldsymbol{\theta}$, $\ddot{\boldsymbol{\theta}}$, represents the angular acceleration of the platform, denoted as $\dot{\boldsymbol{\omega}}$. This formulation provides a clear and direct way to handle angular velocities and accelerations.

Therefore, the generalized coordinate vector $\boldsymbol{\chi}$ and its derivatives are defined as follows:

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{\theta} \end{bmatrix}; \quad \dot{\boldsymbol{\chi}} = \begin{bmatrix} \boldsymbol{v}_p \\ \boldsymbol{\omega} \end{bmatrix}; \quad \ddot{\boldsymbol{\chi}} = \begin{bmatrix} \boldsymbol{a}_p \\ \dot{\boldsymbol{\omega}} \end{bmatrix}.$$

(7.119)

Equations 7.101 and 7.102 can be effectively simplified into the closed-form representation provided in Equation 7.118 using the following matrix components:

$$\boldsymbol{M}_p = \begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & {}^{A}\boldsymbol{I}_p \end{bmatrix}_{6\times6} \quad ; \quad \boldsymbol{C}_p = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{\omega}_{\times}{}^{A}\boldsymbol{I}_p \end{bmatrix}_{6\times6}$$
$$\boldsymbol{G}_p = \begin{bmatrix} -m\boldsymbol{g} \\ \boldsymbol{0}_{3\times1} \end{bmatrix}_{6\times1} ; \quad \boldsymbol{\mathcal{F}}_p = \begin{bmatrix} \boldsymbol{F}_d + \sum \boldsymbol{f}_{b_i} \\ \boldsymbol{n}_d + \sum \boldsymbol{b}_{i\times} \boldsymbol{f}_{b_i} \end{bmatrix}_{6\times1},$$

(7.120)

where $\boldsymbol{I}_{3\times3}$ represents the $3 \times 3$ identity matrix, $\boldsymbol{0}_{3\times3}$ is the $3 \times 3$ zero matrix, and $\boldsymbol{0}_{3\times1}$ is the $3 \times 1$ zero vector.

## Closed-Form Dynamics of the Stewart–Gough Manipulator

To develop the closed-form dynamic formulation for the entire manipulator, it is necessary to perform a transformation that maps the intermediate generalized coordinates $\boldsymbol{x}_i$ to the principal generalized coordinates $\boldsymbol{\chi}$ [193]. This transformation allows us to consolidate the equations for both the limbs and the moving platform, thereby eliminating the internal forces $\boldsymbol{f}_{b_i}$ and yielding a closed-form dynamic model for the complete manipulator.

To achieve this, we define a Jacobian matrix $\boldsymbol{J}_i$ that relates the intermediate coordinates to the principal generalized coordinates. The relationship is expressed as follows:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{J}_i \dot{\boldsymbol{\chi}},$$

(7.121)

where $\boldsymbol{J}_i$ is given by

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{I}_{3\times3} & -\boldsymbol{b}_{i\times} \end{bmatrix}.$$

(7.122)

Taking the time derivative of Equation 7.121, we obtain:

$$\ddot{\boldsymbol{x}}_i = \boldsymbol{J}_i \ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{\chi}},$$

(7.123)

where the time derivative of $\boldsymbol{J}_i$, denoted as $\dot{\boldsymbol{J}}_i$, can be computed as follows:

$$\dot{\boldsymbol{J}}_i = \begin{bmatrix} \boldsymbol{0}_{3\times3} & -(\boldsymbol{\omega}_\times \boldsymbol{b}_i)_\times \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & -\boldsymbol{\omega}_\times \boldsymbol{b}_{i\times} + \boldsymbol{b}_{i\times} \boldsymbol{\omega}_\times \end{bmatrix}.$$

(7.124)

In these equations, $\boldsymbol{J}_i$ is the Jacobian matrix that maps the time derivative of the intermediate coordinates $\boldsymbol{x}_i$ to the time derivative of the principal coordinates $\boldsymbol{\chi}$, and $\dot{\boldsymbol{J}}_i$ is the time derivative of the Jacobian matrix, accounting for changes in angular velocity and its effect on the intermediate coordinates.

To derive the closed-form dynamic formulation for the entire manipulator, we start by substituting the expression for $\dot{\boldsymbol{x}}_i$ from Equation 7.121 into Equation 7.115. By performing this substitution, and then multiplying both sides of the resulting equation by $\boldsymbol{J}_i^T$ from the left, we obtain:

$$\begin{aligned} \boldsymbol{J}_i^T \boldsymbol{M}_i \left( \boldsymbol{J}_i \ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{\chi}} \right) + \quad & \boldsymbol{J}_i^T \boldsymbol{C}_i \boldsymbol{J}_i \dot{\boldsymbol{\chi}} + \boldsymbol{J}_i^T \boldsymbol{G}_i = \boldsymbol{J}_i^T \boldsymbol{F}_i \\ (\boldsymbol{J}_i^T \boldsymbol{M}_i \boldsymbol{J}_i) \ddot{\boldsymbol{\chi}} + (\boldsymbol{J}_i^T \boldsymbol{M}_i \dot{\boldsymbol{J}}_i \quad & + \boldsymbol{J}_i^T \boldsymbol{C}_i \boldsymbol{J}_i) \dot{\boldsymbol{\chi}} + \boldsymbol{J}_i^T \boldsymbol{G}_i = \boldsymbol{J}_i^T \boldsymbol{F}_i. \end{aligned} \tag{7.125}$$

This equation can be consolidated into a closed-form expression as:

$$\boldsymbol{M}_{li} \ddot{\boldsymbol{x}}_i + \boldsymbol{C}_{li} \dot{\boldsymbol{x}}_i + \boldsymbol{G}_{li} = \boldsymbol{\mathcal{F}}_{li}, \tag{7.126}$$

where

$$\begin{aligned} \boldsymbol{M}_{li} &= \boldsymbol{J}_i^T \boldsymbol{M}_i \boldsymbol{J}_i; \qquad & \boldsymbol{C}_{li} &= \boldsymbol{J}_i^T \boldsymbol{M}_i \dot{\boldsymbol{J}}_i + \boldsymbol{J}_i^T \boldsymbol{C}_i \boldsymbol{J}_i \\ \boldsymbol{G}_{li} &= \boldsymbol{J}_i^T G_i; & \boldsymbol{\mathcal{F}}_{li} &= \boldsymbol{J}_i^T \boldsymbol{F}_i. \end{aligned} \tag{7.127}$$

To understand why the Jacobian matrix $\boldsymbol{J}_i^T$ is used in these equations, we must examine the form of the resulting term $\boldsymbol{\mathcal{F}}_{li}$. Given the specific structure of the intermediate

Jacobian $\boldsymbol{J}_i$, as detailed in Equation 7.122, the term $\boldsymbol{\mathcal{F}}_{li}$ takes on the following form:

$$\boldsymbol{\mathcal{F}}_{li} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} \\ -\boldsymbol{b}_{i\times} \end{bmatrix} \left[ -\boldsymbol{f}_{b_i} + \tau_i \hat{\boldsymbol{s}}_i \right] = - \begin{bmatrix} \boldsymbol{f}_{b_i} \\ \boldsymbol{b}_{i\times} \boldsymbol{f}_{b_i} \end{bmatrix} + \begin{bmatrix} \tau_i \hat{\boldsymbol{s}}_i \\ \tau_i \boldsymbol{b}_{i\times} \hat{\boldsymbol{s}}_i \end{bmatrix}. \tag{7.128}$$

For simplicity, we denote the term involving the torque $\tau_i$ as $\boldsymbol{\mathcal{F}}_{\tau_i}$:

$$\boldsymbol{\mathcal{F}}_{\tau_i} = \tau_i \begin{bmatrix} \hat{\boldsymbol{s}}_i \\ \boldsymbol{b}_{i\times} \hat{\boldsymbol{s}}_i \end{bmatrix}. \tag{7.129}$$

By summing the closed-form expressions for all the limbs, as specified in Equation 7.126, with Equation 7.118 for the moving platform, we effectively eliminate the internal forces $\boldsymbol{f}_{b_i}$ from these equations. This approach provides the comprehensive closed-form dynamic formulation for the Stewart-Gough manipulator, which is expressed as follows:

$$\boldsymbol{M}(\boldsymbol{\chi}) = \boldsymbol{M}_p + \sum_{i=1}^{i=6} \boldsymbol{M}_{li}; \qquad \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{C}_p + \sum_{i=1}^{i=6} \boldsymbol{C}_{li}$$

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{G}_p + \sum_{i=1}^{i=6} \boldsymbol{G}_{li}; \qquad \boldsymbol{\mathcal{F}}(\boldsymbol{\chi}) = \boldsymbol{\mathcal{F}}_d + \sum_{i=1}^{i=6} \boldsymbol{\mathcal{F}}_{\tau_i},$$

(7.130)

where $\boldsymbol{\mathcal{F}}_d$ denotes the external disturbance wrench applied to the moving platform. This formulation encapsulates the dynamics of the Stewart-Gough manipulator by integrating the contributions from all its limbs and the moving platform, leading to a unified and comprehensive dynamic model.

## Problems

7.1 Consider the five-bar mechanism depicted in Figure 4.27 and discussed in Problems 4.1 and 5.1. For simplification, treat link $i$ as a slender bar with a mass $m_i$, with its center of mass at the midpoint of the link, and its moment of inertia about the center of mass as $I_i$. Assume that gravity acts in the $-y$ direction, and that actuator torques $\tau_1$ and $\tau_2$ are applied at joints $B$ and $A$, respectively.

a. Carry out a detailed acceleration analysis for this manipulator.

b. Develop a dynamic model using the Newton–Euler approach.

7.2 Consider the five-bar mechanism depicted in Figure 4.28 and outlined in Problems 4.2 and 5.2. Assume link $i$ is a slender bar with mass $m_i$, with its center of mass

at the midpoint of the link, and its moment of inertia about the center of mass is $I_i$. Also, consider that the gravity vector is in the $-y$ direction, with an actuator force $f_1$ applied to the slider pair and an actuator torque $\tau_2$ applied at joint $A$.

a. Perform a full acceleration analysis for this manipulator.

b. Formulate the dynamics using the Newton–Euler method.

7.3 Consider the planar pantograph mechanism shown in Figure 4.29 and detailed in Problems 4.3 and 5.3. For simplification, assume link $i$ is a slender bar with mass $m_i$, with its center of mass located at the midpoint of the link, and its moment of inertia about the center of mass is $I_i$. Also, consider the piston masses $m_{p_1}$ and $m_{p_2}$, with gravity acting in the $-y$ direction, and actuator forces $f_1$ and $f_2$ applied to sliders $A$ and $B$, respectively.

a. Conduct a thorough acceleration analysis for this manipulator.

b. Develop a dynamic model using the Newton–Euler method.

7.4 Consider the planar three-degrees-of-freedom 3RRR parallel manipulator shown in Figure 4.30 and described in Problems 4.4 and 5.4. For this manipulator, treat the links as slender bars with

masses $m_{a_i}$ and $m_{b_i}$, with their centers of mass located at the midpoint of each link, and their moments of inertia about the center of mass as $I_{a_i}$ and $I_{b_i}$. Additionally, consider the moving platform mass $m_e$, with its center of mass at the geometric center of the triangle $\triangle B_1 B_2 B_3$, and its moments of inertia about the center of mass denoted as $I_e$. The manipulator operates in the horizontal plane, while gravity acts in the $-z$ direction, and actuator torques $\tau_i$ are applied at the joints $A_i$.

a. Perform a complete acceleration analysis for this manipulator. Determine the relationship between the angular accelerations of the actuated limbs $\ddot{\alpha}_i$ and the passive limbs $\ddot{\beta}_i$ as functions of the moving platform acceleration vector $\ddot{\boldsymbol{\chi}} = [a_x, a_y, \ddot{\phi}]^T$. Additionally, derive the linear accelerations of the centers of mass of the limbs.

b. Apply the Newton–Euler laws to the limbs and moving platform, eliminate the internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.5 Consider the planar three-degrees-of-freedom 4RRR parallel manipulator shown in Figure 4.31 and discussed in Problems 4.5 and 5.5. For this manipulator, assume the links are slender bars with masses $m_{a_i}$ and $m_{b_i}$, with their centers of mass at the midpoint of each link, and their moments of inertia

about the center of mass as $I_{a_i}$ and $I_{b_i}$. Additionally, consider the moving platform with mass $m_e$, its center of mass at the center of the moving attachment point $B_i$, and its moments of inertia about the center of mass denoted as $I_e$. The manipulator operates in the horizontal plane, with gravity acting in the $-z$ direction, and actuator torques $\tau_i$ applied at joints $A_i$.

a. Conduct a comprehensive acceleration analysis for this manipulator. Determine the relationship for the angular accelerations of the actuated limbs $\ddot{\alpha}_i$ and the passive limbs $\ddot{\beta}_i$ as functions of the moving platform acceleration vector $\ddot{\boldsymbol{\chi}} = [a_x, a_y, \ddot{\phi}]^T$. Additionally, derive the linear acceleration of the centers of mass of the limbs.

b. Apply the Newton–Euler laws to the limbs and the moving platform, remove the internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.6 Consider the spatial parallel manipulator depicted in Figure 4.32 and detailed in Problems 4.6 and 5.6.

a. Perform a thorough acceleration analysis for this manipulator.

b. Apply the Newton–Euler laws to the limbs and the moving platform, remove any internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.7 Consider the 3UPU parallel manipulator shown in Figure 4.33 and discussed in Problems 4.7 and 5.7.

a. Conduct a comprehensive acceleration analysis for this manipulator.

b. Use the Newton–Euler laws for the limbs and the moving platform, eliminate any internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.8 Consider the three-degrees-of-freedom 3UPS manipulator introduced in [51], as shown in Figure 4.34 and detailed in Problems 4.8 and 5.8.

a. Perform a thorough acceleration analysis for this manipulator.

b. Apply the Newton–Euler laws to the limbs and the moving platform, remove any internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.9 Consider the three-degrees-of-freedom 3RPS manipulator shown in Figure 4.35 and discussed in Problems 4.9 and 5.9.

a. Conduct a thorough acceleration analysis for this manipulator.

b. Apply the Newton–Euler laws to the limbs and the moving platform, remove any internal forces, and

derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.10 Consider the three-degrees-of-freedom $3RPS$ manipulator shown in Figure 4.35 and discussed in Problems 4.9 and 5.9.

a. Conduct a thorough acceleration analysis for this manipulator.

b. Apply the Newton–Euler laws to the limbs and the moving platform, remove any internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.11 Consider the Delta robot depicted in Figure 4.38 and discussed in Problems 4.11 and 5.11.

a. Carry out a detailed acceleration analysis for this manipulator.

b. Apply the Newton–Euler laws to both the limbs and the moving platform, remove any internal forces, and derive the dynamic formulation of the manipulator using the Newton–Euler method.

7.12 Consider the Quadrupteron robot illustrated in Figure 4.39 and detailed in Problems 4.12 and 5.12 [55].

a. Conduct a comprehensive acceleration analysis for this manipulator.

b. Apply the Newton–Euler laws to the limbs and the moving platform, remove any internal forces, and

derive the dynamic formulation of the manipulator using the Newton–Euler method.

# 8

# *Dynamics Analysis: Euler-Lagrange Approach*

DOI: 10.1201/9781003590408-8

This chapter explores the dynamics of parallel manipulators using the Euler–Lagrange approach, emphasizing its theoretical foundation and practical application. It begins by introducing the Euler–Lagrange formulation, highlighting its reliance on generalized coordinates and forces. Both the first and second forms of the Euler–Lagrange equations are presented, providing a comprehensive framework for deriving dynamic equations of motion.

The chapter proceeds to examine key properties of dynamic formulations. The structure and characteristics of the mass matrix are discussed, followed by an exploration of the Christoffel matrix and its derivation using the Kronecker product. These elements are crucial for understanding the dynamics of complex robotic systems. Additionally, the concept of linearity in parameters is

introduced, offering insights into simplifying and optimizing dynamic models.

To demonstrate the practical application of the Euler–Lagrange approach, two detailed case studies are presented. The first focuses on a planar cable robot, covering the dynamic formulation of its limbs, moving platform, and overall system. This analysis showcases the method's capability to handle the intricacies of distributed forces and motion in parallel manipulators. The second case study examines the Stewart–Gough platform, a widely used high-precision parallel manipulator. The dynamic formulations of its limbs, moving platform, and entire structure are derived, highlighting the versatility of the Euler–Lagrange approach in addressing real-world challenges in robotic dynamics.

The chapter concludes with a series of thought-provoking problems, designed to reinforce readers’ understanding of the Euler–Lagrange method and its application to dynamic analysis. These exercises bridge theory and practice, equipping readers with the skills needed to design, analyze, and optimize advanced robotic systems.

## 8.1 Introduction

The Euler-Lagrange formulation is a powerful reformulation of classical mechanics that combines conservation of momentum with conservation of energy.

This approach is particularly effective for the dynamic analysis of multibody systems, where numerous reaction forces and moments exist between the components. By using a set of *generalized coordinates*, the Euler-Lagrange method simplifies the derivation of equations of motion, often eliminating the need to directly account for reaction forces and moments.

In the Euler-Lagrange method, generalized coordinates represent the minimal set of variables necessary to describe a system's motion. This choice reduces the complexity of the dynamic analysis. For instance, consider a small frictionless bead traveling in a groove. Using Newton-Euler mechanics would require solving for the time-varying constraint force that keeps the bead in the groove. In contrast, the Euler-Lagrange approach allows us to focus solely on the bead's path through generalized coordinates, bypassing the need to consider constraint forces directly.

By defining generalized coordinates, external forces and torques acting along these coordinates are determined as *generalized forces*. The kinetic and potential energy of the moving masses in the system are then calculated in terms of these generalized coordinates. Finally, the Euler-Lagrange equations are used to determine the equations of motion. The details of this method will be thoroughly discussed in the following section.

## 8.2 Euler-Lagrange Formulation

Unlike the dynamics of open-chain serial manipulators, deriving the dynamic equations for general parallel manipulators remains complex and lacks a universally accepted procedure. This difficulty arises from the intricate nature of closed-loop structures and their kinematic constraints. In Lagrangian mechanics, the trajectory of a multibody system is derived by solving the Lagrange equations, which are presented in two distinct forms.

The first form, known as the Euler-Lagrange equations of the first kind, treats constraints explicitly as additional equations [194]. These constraints are typically managed using Lagrange multipliers, which introduce extra variables into the system to enforce the constraints. This method allows for a clear separation between the equations of motion and the constraints, making it easier to handle complex systems with multiple interdependent constraints. For example, in a system with many interconnected components, each constraint can be addressed individually, and the Lagrange multipliers ensure that the constraints are satisfied throughout the motion.

The second form, known as the Euler-Lagrange equations of the second kind, incorporates constraints directly by choosing appropriate generalized coordinates [195]. This approach involves selecting a set of independent variables that inherently satisfy the system's constraints, thereby embedding the constraints into the equations of motion from the outset. This method reduces the number of

equations and variables, eliminating the need for separate constraint equations and Lagrange multipliers. The choice of generalized coordinates is crucial, as it simplifies the dynamic analysis by focusing on the essential degrees of freedom of the system [196]. The following sections provide a general description of both types of formulations, demonstrating how each method can be applied to derive the equations of motion for complex multibody systems.

Although various methods have been proposed in the literature to derive dynamic equations for parallel manipulators, there is no single, standardized approach. Common practices generally involve disconnecting the limbs from the moving platform, analyzing the dynamics of each limb individually, and then combining these local models to develop a comprehensive dynamic formulation for the entire manipulator.

Typically, each limb and the moving platform are described using local coordinates. For the limbs, the position vectors of the moving attachment points often serve as generalized coordinates. For the moving platform, the position of its center of mass and its orientation angle are frequently used as generalized coordinates. Local dynamic models are developed based on these coordinates, and some approaches use coordinate transformation techniques to convert these local models into a global framework that reflects the entire manipulator's dynamics.

By integrating these local dynamic models, researchers can obtain a global dynamic model for the entire

manipulator. This approach is further explored and extended in this book, where it is used to derive closed-form dynamic formulations for the case studies presented in this chapter.

Significant contributions to the dynamic analysis of parallel manipulators have been made using various methods. For instance, Abdellatif and Heimann [197] developed a closed-form method for deriving dynamic equations, highlighting the impact of selecting appropriate local generalized coordinates on the computational efficiency and simplicity of the dynamic model. They noted that while the Euler-Lagrange method avoids the need for calculating Lagrange multipliers associated with closed-chain constraints, it still requires detailed component-wise derivation for each limb. This can result in a highly complex model, especially for sophisticated manipulators like the Stewart-Gough platform, where interactions between multiple limbs and joints create intricate dynamic behaviors.

In contrast, this book introduces a novel matrix-based manipulation technique that simplifies the derivation process by avoiding the cumbersome component-wise approach. This technique offers a more manageable way to handle dynamic equations, significantly reducing complexity while maintaining computational efficiency. The method is designed to handle complex manipulator dynamics without the need for symbolic software packages,

making it a practical and efficient solution for real-time applications and advanced dynamic simulations.

### 8.2.1 Generalized Coordinates and Forces

The connection between two masses in multi-body systems is described by kinematic constraints that restrict the relative motion between them in various directions. In parallel robots, joints create these kinematic constraints, defining and limiting the system's degrees of freedom.

In a general classification, system constraints are termed “holonomic” if the relationship between position variables can be expressed by an algebraic equation between these variables and, if necessary, the time.

$$f(x_1, x_2, \ldots, t) = 0 \tag{8.1}$$

In this algebraic relationship, $x_i$ represents position or orientation variables, while their derivatives are not seen in this equation. For example, consider the angular motion of a pendulum rotating about its joint. The coordinates $x$ and $y$ of each point on the pendulum with a radial distance of $r$ from the center of rotation can be constrained by the algebraic relation $x^2 + y^2 = r^2$. Therefore, motion constrained by a revolute joint is holonomic. This property holds true for other types of joints such as prismatic, spherical, and others commonly used in robotic systems.

If kinematic constraints cannot be interpreted with an algebraic relation like 8.1, or if derivatives of position and orientation variables appear in the kinematic constraint, or if a differential relation in the position constraints appears, the system is called “non-holonomic.” In these systems, the motion variables of the body depend on the path of motion, and interpreting algebraic constraints in them will not be possible. For example, consider a four-wheeled car. If the car departs from its initial parked state, maneuvers through the surrounding streets, and precisely returns to the initial point with the same parking configuration, it is reasonable to expect that the wheel positions in the initial and final states will differ, even though the car's overall state remains identical in both instances. This discrepancy points to the existence of non-holonomic constraints within the system. By simulating the motion of the wheels on the ground, it becomes evident that a corresponding algebraic relation, akin to 8.1, cannot be identified in such a system.

The configuration of a robotic manipulator is fully specified when the positions and orientations of all its components are known relative to a reference frame. In the context of a typical parallel manipulator, as depicted in Figure 1.11, each limb consists of several interconnected bodies, forming closed kinematic chains with the moving platform. This setup creates a complex network of intermediate coordinates and constraints to accurately represent the motion of each segment within the

manipulator, with these coordinates often being interdependent.

To address these dependencies in the case of holonomic constraints—where the constraints fully define the system's motion—one can eliminate dependent coordinates through constraint equations or by introducing $m$ new independent variables, denoted as $q_1, q_2, \ldots, q_m$. These variables allow the system to express all intermediate coordinates in terms of these $m$ new independent variables. The resulting set of independent variables is referred to as the *generalized coordinates*, and their number corresponds to the degrees of freedom of the parallel robot.

It is important to note that the definition of generalized coordinates for a parallel robot is not unique and can be established in various ways. One common approach is to select a sufficient number of position and orientation variables for the moving platform that corresponds to the system's degrees of freedom. Typically, a generalized coordinate $q_j$ does not need to represent a physical length but can be defined as an orientation angle. Although Euler angles can be used for this purpose, they often involve complex transformations and are less straightforward for certain analyses.

In contrast, screw coordinates are frequently used as an alternative. Screw coordinates offer a more convenient representation because they directly link their derivatives to the angular velocities of the moving platform. This direct relationship simplifies the process of deriving dynamic

equations and analyzing the system's behavior, making screw coordinates particularly useful for advanced dynamic simulations and control applications. This choice enhances the overall efficiency and clarity of the dynamic modeling process for parallel robots.

In parallel with the introduction of a generalized coordinate $q_j$, we also define a corresponding generalized force $Q_j$, which has a conjugate role to the generalized coordinate. The process of defining generalized forces involves applying the principle of virtual work to the external forces acting on the parallel robot.

To illustrate this, consider a scenario where $n$ external forces $\boldsymbol{F}_i$ are applied at various points on the system. Typically, the forces of interest are non-conservative, as these are the ones that contribute to the generalized forces of the system. The virtual work $\delta W_a$ done by these forces is computed by summing the contributions from each force, where each contribution is the product of the force and the virtual displacement at the point of application $\delta \boldsymbol{x}_i$. This is expressed mathematically as:

$$\delta W_a = \sum_{i=1}^{n} \boldsymbol{F}_i \cdot \delta \boldsymbol{x}_i. \tag{8.2}$$

The next step is to relate this virtual work to the generalized coordinates. The projection of this work onto the generalized coordinates allows us to express it in terms

of the work done by each generalized force $Q_j$ along the corresponding generalized coordinate $q_j$. Thus, the total virtual work can be rewritten in the form:

$$\delta W_a = \sum_{i=1}^{m} Q_j \delta q_j,$$

(8.3)

where the generalized force $Q_j$ is defined as:

$$Q_j = \sum_{i=1}^{n} \boldsymbol{F}_i \cdot \frac{\partial \boldsymbol{x}_i}{\partial q_j}.$$

(8.4)

Here, the generalized force $Q_j$ represents the effective force component along the direction of the generalized coordinate $q_j$. It is calculated by summing the contributions of all external forces $\boldsymbol{F}_i$ projected onto the partial derivatives of the position vector $\boldsymbol{x}_i$ with respect to the generalized coordinate $q_j$. This formulation provides a way to translate the impact of external forces into a form that directly relates to the system's generalized coordinates, facilitating a more straightforward analysis and application in dynamic modeling and control.

In deriving the generalized forces, it is important to recognize that the equivalence of Equation 8.2 and Equation 8.3 is applied for each non-conservative external

force acting on the system. Equation 8.4 represents the mathematical definition used in the subsequent derivation of the Lagrangian formulation. Therefore, for a generalized coordinate $q_j$ defined as an orientation angle, the associated generalized force will be a torque.

### 8.2.2 Euler-Lagrange Equations of the Second Kind

The dynamics formulated using Euler-Lagrangian mechanics are known as the Euler-Lagrange equations of motion. This section briefly covers the derivation of the Euler-Lagrange equations of the second kind, which directly incorporates kinematic constraints by selecting appropriate generalized coordinates [195]. This method involves choosing independent variables that inherently satisfy the system's constraints, thus embedding them into the equations of motion from the start. This approach reduces the number of equations and variables, removing the need for separate constraint equations and Lagrange multipliers, and simplifies dynamic analysis by focusing on the system's essential degrees of freedom [196].

Let's explore the principle of virtual work applied to a system of particles, which can be extended to multibody systems with similar formulations. Consider a system with $n$ particles, each with mass $m_i$ and experiencing an acceleration $\boldsymbol{a}_i$. The external forces acting on these particles are denoted by $\boldsymbol{F}_i$. By employing d’Alembert's

principle and incorporating inertial forces, we assume that the system is in static equilibrium. Therefore, the principle of virtual work for this system can be written as:

$$\delta W = \sum_{i=1}^{n} (\boldsymbol{F}_i - m_i \boldsymbol{a}_i) \cdot \delta \boldsymbol{r}_i = 0, \tag{8.5}$$

where $\delta W$ represents the total virtual work performed by the system, and $\delta \boldsymbol{r}_i$ denotes the virtual displacement of each particle, consistent with the constraints imposed on the system. This equation indicates that the total virtual work, which includes contributions from external forces and inertial forces, must be zero for the system to be in equilibrium.

Furthermore, let's assume the system is described by $m$ independent generalized coordinates $q_i$. These coordinates completely define the displacement of each particle in the system. The position of each particle $\boldsymbol{r}_i$ can be expressed as a function of the generalized coordinates and time $t$, such that:

$$\begin{aligned} \boldsymbol{r}_1 &= \boldsymbol{r}_1(q_1, q_2, \ldots, q_m, t) \\ \boldsymbol{r}_2 &= \boldsymbol{r}_2(q_1, q_2, \ldots, q_m, t) \\ &\vdots \\ \boldsymbol{r}_n &= \boldsymbol{r}_n(q_1, q_2, \ldots, q_m, t). \end{aligned}$$

In this context, the generalized coordinates $q_1, q_2, \ldots, q_m$ effectively capture all the necessary information about the configuration of the system. Each particle's position vector $\boldsymbol{r}_i$ is thus a function of these coordinates and time, reflecting how the positions and displacements of the particles are interrelated through the generalized coordinates. This approach simplifies the analysis by reducing the system's degrees of freedom to a manageable number of variables and ensures that the virtual work principle is consistently applied across the system.

Thus, the virtual displacement or differential motion of each particle can be derived by taking the partial derivatives of $\boldsymbol{r}_i$ with respect to the generalized coordinates:

$$\delta \boldsymbol{r}_i = \sum_{j=1}^{m} \frac{\partial \boldsymbol{r}_i}{\partial q_j} \delta q_j.$$

(8.6)

By substituting this expression into the principle of virtual work Equation 8.5 and factoring out $\delta q_j$, we obtain:

$$\delta W = \sum_{j=1}^{m} \sum_{i=1}^{n} \left( \boldsymbol{F}_i \frac{\partial \boldsymbol{r}_i}{\partial q_j} - m_i \boldsymbol{a}_i \frac{\partial \boldsymbol{r}_i}{\partial q_j} \right) \cdot \delta q_j = 0.$$

(8.7)

In this context, the external forces applied to the system are split into two components: $\boldsymbol{F}_i = \boldsymbol{F}_i^c + \boldsymbol{F}_i^u$. Here, $\boldsymbol{F}_i^c$ denotes the conservative forces, which conserve mechanical energy, such as gravitational and spring forces. On the other hand, $\boldsymbol{F}_i^u$ represents non-conservative forces like friction and air drag, which dissipate energy.

For conservative forces, they can be described by a scalar potential field, $P$, such that $\boldsymbol{F}_i^c = -\nabla P$. Consequently, the work done by conservative forces can be expressed as:

$$\sum_{i=1}^{n} \boldsymbol{F}_i^c \cdot \frac{\partial \boldsymbol{r}_i}{\partial q_j} = -\sum_{i=1}^{n} \nabla P \cdot \frac{\partial \boldsymbol{r}_i}{\partial q_j} = -\frac{\partial P}{\partial q_j}. \tag{8.8}$$

This result shows that the contribution of conservative forces to the virtual work can be directly related to the gradient of the potential field $P$ with respect to the generalized coordinates. This formulation simplifies the analysis of the system by focusing on how the potential energy changes with respect to the coordinates, thereby facilitating the application of the principle of virtual work in dynamic system analysis.

Additionally, the non-conservative components of the external forces are represented by generalized forces $Q_j$, as defined in Equation 8.4:

$$Q_j = \sum_{i=1}^{n} \boldsymbol{F}_i^u \cdot \frac{\partial \boldsymbol{r}_i}{\partial q_j}.$$

(8.9)

These generalized forces account for non-conservative effects such as friction and other dissipative forces not represented by potential energy fields.

To handle the inertial forces in the system, we use the expression involving kinetic energy $K$, as specified by [198]:

$$\sum_{i=1}^{n} m_i \boldsymbol{a}_i \cdot \frac{\partial \boldsymbol{r}_i}{\partial q_j} = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_j}\right) - \frac{\partial K}{\partial q_j}.$$

(8.10)

Here, the term $\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_j}\right)$ represents the rate of change of the generalized momentum, while $\frac{\partial K}{\partial q_j}$ accounts for the dependency of kinetic energy on the generalized coordinates.

Substituting Equation 8.8 and Equation 8.10 into Equation 8.7, we simplify to:

$$\delta W = \sum_{j=1}^{m} \left(-\frac{\partial P}{\partial q_j} + Q_j - \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_j}\right) + \frac{\partial K}{\partial q_j}\right) \cdot \delta q_j = 0.$$

(8.11)

This equation must be satisfied for any arbitrary virtual displacement $\delta q_j$. Consequently, we derive the following fundamental relation for each generalized coordinate $q_j$:

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_j}\right) - \frac{\partial K}{\partial q_j} + \frac{\partial P}{\partial q_j} = Q_j.$$

(8.12)

This equation encapsulates the balance of forces and energies in the system, incorporating both conservative and non-conservative effects.

The “Lagrangian function” is defined as the difference between the system's kinetic energy ($K$) and potential energy ($P$):

$$\mathscr{L} = K - P.$$

(8.13)

The Lagrangian function $\mathscr{L}$ is a central concept because it simplifies the equations of motion by combining kinetic and potential energies into a single function. Since kinetic energy depends on both the generalized coordinates and their time derivatives, while potential energy depends only on the coordinates, the Lagrangian formulation allows us to express the equations of motion in a unified framework.

By reformulating Equation 8.12 in terms of the Lagrangian function, we get:

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathscr{L}}{\partial q_j} = Q_j, \quad j = 1, 2, \ldots, m, \tag{8.14}$$

In this equation, $q_j$ denotes the generalized coordinate, $Q_j$ represents the generalized force, and $m$ is the number of independent generalized coordinates. By applying this Euler-Lagrange Equation 8.14 for each generalized coordinate, one obtains a set of $m$ differential equations that collectively describe the dynamics of the system. This approach is particularly advantageous for complex systems with multiple interacting components, such as robotic manipulators, where it facilitates the derivation and solution of dynamic equations efficiently.

### 8.2.3 Euler-Lagrange Equations of the First Kind

In contrast to the Lagrangian formulation of the second kind, which simplifies dynamic analysis by directly incorporating constraints into the equations of motion, the Euler-Lagrange equations of the first kind operate with a set of redundant coordinates. This method requires the formulation of additional constraint equations derived from the system's kinematics. These constraints, along with their

derivatives, are integrated into the primary equations of motion through the use of Lagrange multipliers. Consequently, the number of equations in this formulation equals the number of degrees-of-freedom plus the number of redundant coordinates.

The Euler-Lagrange equations of the first kind can be expressed as:

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathscr{L}}{\partial q_j} = Q_j + \sum_{i=1}^{k} \lambda_i \frac{\partial h_i}{\partial q_j}, \quad j = 1, 2, \ldots, n, \tag{8.15}$$

In this formulation, $h_i$, denotes the constraint functions that describe the kinematic relationships and restrictions on the system, while $k$ is the number of such constraints. Furthermore, $\lambda_i$ are the Lagrange multipliers, which are additional variables introduced to enforce the constraints, $n$ represents the total number of redundant coordinates, which is the sum of the degrees-of-freedom ($m$) and the number of constraints ($k$), i.e. $n = m + k$.

To solve these equations effectively, it is helpful to partition them into two distinct sets. The first set consists of equations where the Lagrange multipliers ($\lambda_i$) are the unknowns, while the second set contains equations where the generalized forces ($Q_i$) are the unknowns.

The first set of equations, associated with the redundant coordinates, can be written as:

$$\sum_{i=1}^{k} \lambda_i \frac{\partial h_i}{\partial q_j} = \frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathscr{L}}{\partial q_j} - \hat{Q}_j, \quad j = 1, 2, \ldots, k.$$

(8.16)

In this set of equations, if any $\hat{Q}_j$ exists, represents the generalized force due to external influences, excluding gravitational and actuator forces, while $\lambda_i$ are the unknowns to be determined. By solving these equations for each redundant coordinate, you obtain a system of $k$ linear equations that allow you to calculate the Lagrange multipliers. Once these multipliers are known, the remaining dynamic equations involving only actuator-contributed generalized forces can be derived from the following set of equations:

$$Q_j = \frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathscr{L}}{\partial q_j} - \sum_{i=1}^{k} \lambda_i \frac{\partial h_i}{\partial q_j}, \quad j = k+1, \ldots, n.$$

(8.17)

Although the Euler-Lagrange formulation of the first kind may seem advantageous because it does not initially require defining a set of independent generalized coordinates equal to the number of degrees-of-freedom, it introduces complexity through the necessity of determining Lagrange multipliers. Each redundant coordinate requires

a corresponding multiplier, complicating the derivation of the final dynamic equations.

Conversely, the Lagrangian formulation of the second kind directly yields equations equal in number to the degrees-of-freedom, simplifying the dynamic model. However, this method necessitates deriving intermediate limb variables and solving a nonlinear inverse kinematics problem to eliminate these variables.

Ultimately, while both formulations involve a comparable level of complexity in deriving the equations of motion, the choice between them depends on the specific system characteristics and the practical considerations of the problem at hand. In the subsequent sections, the Lagrangian formulation of the second kind will be employed to derive the governing equations of motion for various typical parallel manipulators, showcasing its practical application and advantages in handling complex dynamic systems.

## 8.3 Properties of Dynamic Formulations

Before deriving the governing equations of motion for various typical parallel manipulators, it is beneficial to revisit and examine the dynamic formulation obtained for a robot manipulator, focusing on its structural attributes and dynamic properties. This review is crucial as it provides a foundation for understanding how these equations can be

further developed and applied in real-world scenarios. The advanced implementation of dynamic equation characteristics, which is elaborated in the subsequent chapter, will be instrumental in designing and refining different control strategies.

Consider a fully parallel manipulator with $n$ degrees of freedom. The state of such a manipulator is characterized by a set of generalized coordinates, which are represented by the vector $\boldsymbol{\chi} = [q_1, q_2, \ldots, q_n]^T$. When the dynamic equations, derived from Equation 8.14, are expressed for all the generalized coordinates in vector form, they yield the equation

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\boldsymbol{\chi}}}\right) - \frac{\partial \mathscr{L}}{\partial \boldsymbol{\chi}} = \boldsymbol{\mathcal{F}},$$

(8.18)

In this equation, $\boldsymbol{\mathcal{F}} = [Q_1, Q_2, \ldots, Q_n]^T$ represents the vector of generalized forces acting on the manipulator. The term $,\mathscr{L}$, denotes the Lagrangian of the system, which is defined as the difference between the kinetic and potential energies, $\mathscr{L} = K - P$.

For a parallel manipulator with six degrees of freedom, it is often advantageous to use the position of the center of mass of the moving platform along with its orientation variables, expressed in screw coordinates, as the generalized coordinates. Moreover, in the absence of

friction and external disturbances (wrenches) applied to the system, the generalized forces can be computed from the actuator forces using the Jacobian relationship provided in Equation 6.9. Specifically, this relationship is expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$, where $\boldsymbol{J}$ denotes the Jacobian matrix and $\boldsymbol{\tau}$ represents the vector of actuator forces.

First, let's examine the Lagrangian function. We start by defining the *mass matrix* $\boldsymbol{M}(\boldsymbol{\chi})$, which directly stems from the manipulator's kinetic energy. The kinetic energy $K(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ of the manipulator is given by:

$$K(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \frac{1}{2} \dot{\boldsymbol{\chi}}^T \boldsymbol{M}(\boldsymbol{\chi}) \dot{\boldsymbol{\chi}}. \tag{8.19}$$

Here, $\dot{\boldsymbol{\chi}}$ represents the rate of the twist vector, and $\boldsymbol{M}(\boldsymbol{\chi})$ is a symmetric matrix that depends on the generalized coordinates $\boldsymbol{\chi}$. The kinetic energy is thus expressed as a quadratic function of the velocity vector. Using this, the Lagrangian function $\mathscr{L}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ can be written as:

$$\mathscr{L}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = K(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) - P(\boldsymbol{\chi}) = \frac{1}{2} \dot{\boldsymbol{\chi}}^T \boldsymbol{M}(\boldsymbol{\chi}) \dot{\boldsymbol{\chi}} - P(\boldsymbol{\chi}). \tag{8.20}$$

In this expression, $P(\boldsymbol{\chi})$ denotes the potential energy of the system, which depends on the generalized coordinates $\boldsymbol{\chi}$ but is independent of the velocities $\dot{\boldsymbol{\chi}}$. Thus, the

Lagrangian function captures the dynamics of the system by combining kinetic and potential energy into a single formulation.

A fundamental property to recognize is that the kinetic energy is a quadratic function of the velocity vector $\dot{\boldsymbol{\chi}}$, while the potential energy is solely a function of the generalized coordinates $\boldsymbol{\chi}$.

Next, let's analyze the terms in the Lagrange equation provided in Equation 8.18:

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\boldsymbol{\chi}}}\right) &= \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\boldsymbol{\chi}}}\right), \\ &= \frac{d}{dt}\Big(\boldsymbol{M}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}\Big), \\ &= \boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}},\end{aligned} \tag{8.21}$$

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\chi}} = \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\chi}}\Big(\dot{\boldsymbol{\chi}}^T \boldsymbol{M}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}\Big) - \frac{\partial P(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}}. \tag{8.22}$$

Here, $\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\boldsymbol{\chi}}}\right)$ represents the time derivative of the partial derivative of the Lagrangian with respect to the velocity vector. This term evaluates to $\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}$, where

$\ddot{\boldsymbol{\chi}}$ denotes the acceleration vector and $\dot{\boldsymbol{M}}(\boldsymbol{\chi})$ represents the time derivative of the mass matrix.

The term $\frac{\partial \mathscr{L}}{\partial \boldsymbol{\chi}}$ represents the gradient of the Lagrangian with respect to the generalized coordinates. It can be split into two parts: the derivative of the kinetic energy term minus the derivative of the potential energy.

Consequently, the Lagrange equation can be expressed as:

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \left(\dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\chi}}\left(\dot{\boldsymbol{\chi}}^T \boldsymbol{M}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}\right)\right) + \frac{\partial P(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} =$$

(8.23)

This equation represents the balance of forces acting on the manipulator, incorporating both the inertial forces (expressed by $\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}}$), Coriolis and centrifugal effects (captured by $\dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\chi}}\left(\dot{\boldsymbol{\chi}}^T \boldsymbol{M}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}\right)$), and gravitational forces (represented by $\frac{\partial P(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}}$).

We define the *gravity vector* $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$ as:

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \frac{\partial P(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}},$$

(8.24)

which represents the generalized forces due to gravity.

Additionally, we introduce the *Coriolis and centrifugal vector* $\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ as:

$$\begin{aligned} \boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) &= \dot{M}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} - \frac{\partial K}{\partial \boldsymbol{\chi}} \\ &= \dot{M}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\chi}}\left(\dot{\boldsymbol{\chi}}^T M(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}\right). \end{aligned} \tag{8.25}$$

This vector captures the effects of Coriolis and centrifugal forces on the system.

Thus, the Lagrange equations can be succinctly written in the general form:

$$M(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{\mathcal{F}}. \tag{8.26}$$

This formulation highlights that, through the Lagrange method, the dynamic equations for parallel manipulators can be expressed in a unified and closed form. This approach is analogous to the equations typically used for serial manipulators, providing a consistent framework for analyzing and controlling complex robotic systems.

### 8.3.1 Properties of Mass Matrix

The mass matrix of the robot, denoted as $\boldsymbol{M}(\boldsymbol{\chi})$, is derived from the kinetic energy of the system using the relationship

specified in 8.19. This matrix, which is $n \times n$ in size, is a crucial component in the dynamic modeling of the robot. It depends on the robot's configuration $\boldsymbol{\chi}$, which includes the positions and orientations of its various links and joints. Given that the kinetic energy of the system is inherently non-negative, the mass matrix is guaranteed to be symmetric and positive definite.

Due to these properties, $\boldsymbol{M}(\boldsymbol{\chi})$ is invertible for any configuration $\boldsymbol{\chi}$ and is bounded by specific limits. Mathematically, this boundedness is expressed as:

$$\underline{\lambda}\,\boldsymbol{I}_{n\times n} \leq \boldsymbol{M}(\boldsymbol{\chi}) \leq \overline{\lambda}\,\boldsymbol{I}_{n\times n},$$

(8.27)

where $\boldsymbol{I}_{n\times n}$ is the identity matrix of size $n \times n$, and $\overline{\lambda}$ and $\underline{\lambda}$ are positive scalars. These scalars represent the maximum and minimum eigenvalues of the mass matrix, respectively, across the entire range of the robot's workspace. The presence of these bounds ensures that the mass matrix does not vary too drastically with changes in the configuration $\boldsymbol{\chi}$, maintaining stability in the dynamic analysis.

In addition to the mass matrix itself, its inverse, $\boldsymbol{M}^{-1}(\boldsymbol{\chi})$, is also bounded. This can be expressed as:

$$\frac{1}{\underline{\lambda}}\,\boldsymbol{I}_{n\times n} \geq \boldsymbol{M}^{-1}(\boldsymbol{\chi}) \geq \frac{1}{\overline{\lambda}}\,\boldsymbol{I}_{n\times n}.$$

(8.28)

This relationship indicates that the inverse of the mass matrix also falls within specific bounds, which is crucial for ensuring that the system's response to forces and torques remains well-behaved and predictable.

Furthermore, the boundedness of the mass matrix can also be understood in terms of its norm. The norm of a matrix provides a measure of its size and can be used to assess the matrix's behavior in various contexts. The bounded norm of the mass matrix is given by:

$$\underline{M} \leq \| \boldsymbol{M}(\boldsymbol{\chi}) \| \leq \overline{M},$$

(8.29)

where $\underline{M}$ and $\overline{M}$ are positive scalars that define the lower and upper bounds of the matrix norm. This norm can be computed using any induced matrix norm, such as the Frobenius norm. The bounded norm implies that the mass matrix maintains consistent magnitude characteristics across different configurations, ensuring stability and robustness in the dynamic analysis and control of the robot.

### 8.3.2 Christoffel Matrix

Deriving the Coriolis and centrifugal vector $\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ in the dynamics formulation of parallel robots (Equation 8.26) constitutes the most intricate part of the calculations. In

this section, by employing the Kronecker product in matrices, we replace this vector with a matrix containing Christoffel symbols, which we refer to as the “Christoffel matrix” in this book, and we will elucidate the unique features of this matrix. It can be demonstrated that the Coriolis and centrifugal vector can be substituted by the product of the Christoffel matrix $\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ and the vector of angular joint velocities as follows:

$$\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} \tag{8.30}$$

In this representation, the Christoffel matrix is not uniquely determined, and there are various representations for it. To determine the Christoffel matrix, the audience needs to be familiar with the definition and properties of the Kronecker product [199, 200]. A brief overview of the necessary concepts is provided in the following section, with more detailed information given in Appendix A.

## Kronecker Product

The Kronecker product, denoted by $\otimes$, defines a multiplication operation on two matrices of arbitrary dimensions, resulting in a block matrix. If matrix $\boldsymbol{A}$ has dimensions $m \times n$ and matrix $\boldsymbol{B}$ has dimensions $p \times q$, then the Kronecker product of these two matrices, denoted as $\boldsymbol{A} \otimes \boldsymbol{B}$, is a matrix with dimensions $mp \times nq$. It is formed

as a block matrix by taking the product of the elements of the first matrix with the whole second matrix.

With this definition, Kronecker products exhibits the following properties. The matrix $(\boldsymbol{A} \otimes \boldsymbol{B})$ is invertible, if and only if, matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible. In this case, the inverse of this matrix is calculated as follows.

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1} \tag{8.31}$$

Moreover, the transpose in this product is distributive.

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{T} = \boldsymbol{A}^{T} \otimes \boldsymbol{B}^{T} \tag{8.32}$$

Another feature of the Kronecker product is utilized in simplifying dynamic relations. Considering the definition of the vector $\boldsymbol{x} \in \mathbb{R}^n$, and since the Kronecker product is not commutative in general, we have $\boldsymbol{I}_{n\times n} \otimes \boldsymbol{x} \neq \boldsymbol{x} \otimes \boldsymbol{I}_{n\times n}$, where $\boldsymbol{I}_{n\times n}$ represents the identity matrix of dimensions $n \times n$. However, it can be shown that the following relation holds for any arbitrary vector $\boldsymbol{x} \in \mathbb{R}^n$.

$$(\boldsymbol{I}_{n\times n} \otimes \boldsymbol{x})\boldsymbol{x} = (\boldsymbol{x} \otimes \boldsymbol{I}_{n\times n})\boldsymbol{x} \tag{8.33}$$

Now, let us define the time derivative and partial derivatives of matrices using the Kronecker product and investigate the application of the Kronecker product in examining the gradients of composite matrices. Consider the matrix $\boldsymbol{A}(\boldsymbol{\chi})$, according to the convention, the partial derivative of this matrix with respect to the variable $\boldsymbol{\chi}$ can be expressed in the form of a block-column matrix as follows.

$$\frac{\partial \boldsymbol{A}(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} = \begin{bmatrix} \frac{\partial \boldsymbol{A}}{\partial q_1} \\ \vdots \\ \frac{\partial \boldsymbol{A}}{\partial q_n} \end{bmatrix} \tag{8.34}$$

With this definition, the time derivative of a matrix can be obtained using the Kronecker product, as elaborated in [200].

$$\frac{d\boldsymbol{A}}{dt} = \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\chi}}^T \left( \boldsymbol{I}_{n\times n} \otimes \dot{\boldsymbol{\chi}} \right) \tag{8.35}$$

Now, if we seek to find the relative derivative of the product of two matrices, we can utilize the following relation.

$$\frac{\partial}{\partial \boldsymbol{\chi}}(\boldsymbol{A}(\boldsymbol{\chi})\boldsymbol{B}(\boldsymbol{\chi})) = (\boldsymbol{I}_{n\times n} \otimes \boldsymbol{A})\frac{\partial \boldsymbol{B}}{\partial \boldsymbol{\chi}} + \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\chi}}\boldsymbol{B}$$

(8.36)

## Christoffel Matrix Derivation

We initiate the calculations to determine the Christoffel matrix using the Kronecker product. The computation of the Christoffel matrix is crucial for various applications in differential geometry and theoretical physics. To start, we can rewrite Equation 8.25 using the Kronecker product, which allows for a more compact and manageable representation of the terms involved. Specifically, we aim to express the second term of $\boldsymbol{\nu}$ given in Equation 8.25 in terms of the Kronecker product to facilitate the calculation of the partial derivatives. The detailed steps are as follows:

$$\begin{aligned}\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) &= \dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} - \frac{1}{2}\left[\frac{\partial}{\partial \boldsymbol{\chi}}\left(\dot{\boldsymbol{\chi}}^T \boldsymbol{M}(\boldsymbol{\chi})\right)\right]\dot{\boldsymbol{\chi}} \\ &= \left[\dot{\boldsymbol{M}}(\boldsymbol{\chi}) - \frac{1}{2}\left(\boldsymbol{I}_{n\times n} \otimes \dot{\boldsymbol{\chi}}^T\right)\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}\right]\dot{\boldsymbol{\chi}},\end{aligned}$$

(8.37)

Here, $\dot{\boldsymbol{M}}(\boldsymbol{\chi})$ represents the time derivative of the matrix $\boldsymbol{M}$ with respect to $\boldsymbol{\chi}$, and $\frac{\partial}{\partial \boldsymbol{\chi}}$ denotes the partial derivative with respect to $\boldsymbol{\chi}$. The use of the Kronecker

product $\otimes$ helps to systematically handle the multidimensional nature of these derivatives.

In this calculation, we utilize the relation given by 8.34, which helps simplify the expression by applying matrix calculus techniques. Additionally, we apply the equality $\partial \dot{\boldsymbol{\chi}}^T / \partial \boldsymbol{\chi} = \mathbf{0}$, indicating that the partial derivative of the transpose of $\dot{\boldsymbol{\chi}}$ with respect to $\boldsymbol{\chi}$ is zero. Furthermore, by using the Kronecker property described in 8.35, we arrive at the final form of the expression:

$$\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \left[ \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T \left( \boldsymbol{I}_{n \times n} \otimes \dot{\boldsymbol{\chi}} \right) - \frac{1}{2} \left( \boldsymbol{I}_{n \times n} \otimes \dot{\boldsymbol{\chi}}^T \right) \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}} \right] \dot{\boldsymbol{\chi}}$$

(8.38)

In this final expression, $\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T$ represents the transpose of the partial derivative of $\boldsymbol{M}$ with respect to $\boldsymbol{\chi}$, and the Kronecker product is used to distribute the terms effectively.

Using this relationship, the first representation of the Christoffel matrix $\boldsymbol{\mathcal{C}}_1(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ can be directly computed. The computation is expressed as:

$$\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{\mathcal{C}}_1(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \dot{\boldsymbol{\chi}}$$

(8.39)

where the components are defined as follows:

$$\boldsymbol{\mathcal{C}}_1(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{V}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) - \frac{1}{2}\boldsymbol{U}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$$

(8.40)

Here, $\boldsymbol{\mathcal{C}}_1(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is decomposed into two terms, $\boldsymbol{V}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ and $\boldsymbol{U}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$, defined by the following equations:

$$\boldsymbol{V}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \quad = \dot{\boldsymbol{M}}(\boldsymbol{\chi}) = \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T \left( \boldsymbol{I}_{n \times n} \otimes \dot{\boldsymbol{\chi}} \right)$$

(8.41)

$$\boldsymbol{U}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \quad = \left( \boldsymbol{I}_{n \times n} \otimes \dot{\boldsymbol{\chi}}^T \right) \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}$$

(8.42)

The term $\boldsymbol{V}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ represents the derivative of the matrix $\boldsymbol{M}$ with respect to $\boldsymbol{\chi}$ transposed and scaled by the Kronecker product of the identity matrix and $\dot{\boldsymbol{\chi}}$. The term $\boldsymbol{U}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$, on the other hand, involves the Kronecker product of the identity matrix and the transpose of $\dot{\boldsymbol{\chi}}$ multiplied by the partial derivative of $\boldsymbol{M}$ with respect to $\boldsymbol{\chi}$.

In this formulation, the derivation of $\partial \boldsymbol{M} / \partial \boldsymbol{\chi}$ is the cornerstone of the calculations, ensuring the accuracy and applicability of the Christoffel matrix representation in various contexts such as kinematics and dynamic systems.

As previously mentioned, the decomposition of the vector $\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ does not uniquely determine a Christoffel matrix, and different representations of this matrix are possible. An alternative representation can be calculated as follows:

$$\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{c}_2(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}},$$

(8.43)

where

$$\boldsymbol{c}_2(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{U}^T(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) - \frac{1}{2}\boldsymbol{U}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}).$$

(8.44)

This holds true because, according to definition 8.42 and the distributive property of the transpose operator on the Kronecker product, $\boldsymbol{U}^T$ can be derived as

$$\boldsymbol{U}^T = \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T \left(\boldsymbol{I}_{n\times n} \otimes \dot{\boldsymbol{\chi}}\right).$$

By noting this, Equation 8.44 is directly obtained from Equation 8.38.

The most commonly adopted representation of the Christoffel matrix is designed so that the matrix $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$ becomes skew-symmetric. This version of the Christoffel matrix is expressed as:

$$\boldsymbol{\nu}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}}$$

(8.45)

where

$$\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \frac{1}{2}\Big(\boldsymbol{V}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) + \boldsymbol{U}^T(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) - \boldsymbol{U}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\Big).$$

(8.46)

In this formulation, $\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is derived in such a way that it ensures the matrix $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$ to be skew-symmetric. This skew-symmetry property is significant in many applications, particularly in model-based control systems.

To verify this representation, we proceed with a detailed calculation. Consider the expression:

$$\begin{aligned}\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} &= \dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} - \frac{1}{2}\Big(\boldsymbol{I} \otimes \dot{\boldsymbol{\chi}}^T\Big)\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}\dot{\boldsymbol{\chi}}, \\ &= \frac{1}{2}\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T\Big(\boldsymbol{I} \otimes \dot{\boldsymbol{\chi}}\Big)\dot{\boldsymbol{\chi}} + \frac{1}{2}\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T\Big(\boldsymbol{I} \otimes \dot{\boldsymbol{\chi}}\Big)\dot{\boldsymbol{\chi}} -\end{aligned}$$

Note that in this derivation, $\dot{\boldsymbol{M}}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}}$ is divided into two equal terms, and Equation 8.41 is used to evaluate these terms. Furthermore, used the Kronecker product property given in Equation 8.33, to further simplify

$$\begin{aligned}\boldsymbol{\mathcal{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} &= \frac{1}{2}\boldsymbol{V}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \frac{1}{2}\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{\chi}}^T \Big(\dot{\boldsymbol{\chi}} \otimes \boldsymbol{I}\Big)\dot{\boldsymbol{\chi}} - \frac{1}{2}\Big(\boldsymbol{I} \\ &= \frac{1}{2}\Big[\boldsymbol{V}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}}) + \boldsymbol{U}^T(\boldsymbol{\chi},\dot{\boldsymbol{\chi}}) - \boldsymbol{U}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\Big]\dot{\boldsymbol{\chi}}.\end{aligned}$$

By combining these components, we can see that the derived form of $\boldsymbol{\mathcal{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})$ aligns with the final form of Equation 8.46.

With this definition, the computational complexity involved in calculating the Christoffel matrix does not increase substantially, as we still need to derive the matrices $\boldsymbol{V}$ and $\boldsymbol{U}$ as before. However, this specific version of the Christoffel matrix has a significant advantage over other forms because it ensures that the matrix $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$ is skew-symmetric. This is illustrated as follows:

$$\begin{aligned}\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}} &= \boldsymbol{V} - \Big[\boldsymbol{V} + \boldsymbol{U}^T - \boldsymbol{U}\Big] \\ &= \boldsymbol{U} - \boldsymbol{U}^T\end{aligned} \tag{8.47}$$

This result implies that for any vector $\boldsymbol{x}$, the scalar expression $\boldsymbol{x}^T(\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}})\boldsymbol{x}$ will be zero. In contrast, using other representations of the Christoffel matrix might not guarantee skew-symmetry for $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$. For such alternative forms, the term $\dot{\boldsymbol{\chi}}^T(\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}})\dot{\boldsymbol{\chi}}$ will only be

zero for the specific vector $\dot{\boldsymbol{\chi}}$, which does not provide the same level of generality.

Although any of the $\boldsymbol{\mathcal{C}}_i$ matrices could theoretically be used to represent the Coriolis and centrifugal forces, it is highly recommended to use the final form given in Equation 8.46. This form guarantees that $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$ retains its skew-symmetric property, a crucial aspect for maintaining consistency in the dynamic equations of the system.

With this representation, the dynamic equation for any parallel robot can be succinctly expressed in the general closed form:

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{\mathcal{F}}. \tag{8.48}$$

This formulation, known as the “closed-form dynamic formulation,” is particularly advantageous in the analysis and control of parallel manipulators. It facilitates the application of advanced control strategies, including modern adaptive and robust control methods, by providing a straightforward and consistent representation of the system's dynamics. By ensuring the skew-symmetry of $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$, this representation aids in the simplified control structure of complex robotic systems.

It is crucial to understand that the closed-form dynamic formulation is a direct extension of the Lagrangian method.

This approach contrasts with other methods discussed earlier, which often result in an implicit form of dynamic equations. Although these implicit formulations can be converted into a closed form through additional manipulations, this process involves significant algebraic transformations. For instance, the Stewart-Gough platform dynamics, derived using the Newton-Euler approach, is thoroughly detailed in Section 7.5 to illustrate such manipulations.

To maintain consistency across various formulations and to facilitate accurate comparisons between results derived from different approaches, it is beneficial to use appropriate generalized coordinates for the kinetic and potential energies of the system. Specifically, one effective choice for these coordinates is the positions of the moving attachment points where the limbs connect to the moving platform. This choice usually simplifies the representation of the system's.

Additionally, the kinetic and potential energies of the moving platform are expressed using its own generalized coordinates. By defining a clear transformation between the limb coordinates and those of the moving platform, all terms can be integrated into a unified set of generalized coordinates. This integration enables the derivation of the equations of motion for the entire manipulator in a coherent manner. It also ensures that the resulting equations are directly comparable to those obtained from

the Newton-Euler method, as demonstrated for the Stewart-Gough platform in Section 8.5.

Following this comprehensive approach, the dynamic formulation for a planar parallel manipulator is initially presented. This is followed by a detailed derivation of the dynamic equations for the Stewart-Gough manipulator using the Lagrangian formulation. This sequence of derivations not only provides a clear and consistent framework for analyzing different types of manipulators but also ensures that the dynamic equations are accurately formulated and comparable across different methodologies.

### 8.3.3 Linearity in Parameters

The dynamics formulation of parallel robots can be generally represented by the relation 8.26. In order to use these equations in dynamic simulation or model-based control design, it is essential to identify the kinematic and dynamic parameters of the robot. Various calibration methods have been developed in the robotics literature to accurately determine these parameters [201, 202, 203]. However, due to the complexity of dynamic equations, the robot calibration process is highly challenging in practice.

Fortunately, it can be demonstrated that the nonlinear and coupled dynamic equations of parallel manipulators, in terms of $\boldsymbol{\chi}$ and its derivatives, can be rewritten in the form of linear regression with respect to the kinematic and dynamic parameters of the robot. If the parameter vector of the robot can be represented in the $\ell$-dimensional space by

the symbol $\boldsymbol{\theta} \in \mathbb{R}^{\ell}$, it should be first noted that the dimension of this space is not unique. In general, in addition to kinematic and geometric parameters, the dynamic parameters of each link in the limbs consist of ten components, including the mass of the link, its three center of mass components, and six components of the link's inertia matrix. However, depending on the type of robot motion, some of these parameters may not be excited and may not appear in the robot's dynamic equations. Determining the minimum dimension of the parameter space is a challenging research topic that has been addressed in the robotics literature [204].

By determining the dimension of the robot's parameter space, the dynamic equations can be rewritten as a linear regression form with respect to these parameters.

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}})\boldsymbol{\theta}. \tag{8.49}$$

This formulation is presented in the context of linear regression, where $\boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}})$ is referred to as the regressor, and $\boldsymbol{\theta}$ is the parameter vector that needs to be identified through empirical data. In this representation, $\boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}})$ encapsulates the variables and their interactions that influence the system's behavior, while $\boldsymbol{\theta}$ represents the set of parameters that characterize the

system's dynamics and need to be estimated from experimental observations.

For a parallel manipulator, this linear regression framework facilitates the calibration process by applying least-squares techniques to estimate the parameter vector $\boldsymbol{\theta}$. Calibration experiments are conducted by gathering data from the manipulator's operations, which are then used to fit the model to the observed data. By minimizing the discrepancy between the predicted and actual values, least-squares methods provide an optimal approach to determining the manipulator's parameters, ensuring accurate modeling of its dynamic behavior.

Moreover, this linear regression representation is not only useful for calibration but also plays a crucial role in advanced adaptive control strategies. These techniques, which are extensively explored in the literature [205, 206, 207], leverage the linear model to adjust control parameters in real-time, enhancing the adaptability and performance of the control system. The principles and applications of these advanced adaptive control techniques are further detailed in Chapter 12, highlighting their significance in improving the robustness and efficiency of robotic systems. By incorporating the linear regression model into these control strategies, the system benefits from a structured approach to parameter estimation and real-time adjustment, ultimately leading to more precise and responsive control of the manipulator.

## 8.4 Case Study I: Planar Cable Robot

In this section, we represent a comprehensive dynamic analysis of a planar 4RPR manipulator using the Lagrange approach. This type of manipulator, known for its four-bar linkage structure, is meticulously described in Section 4.4.1, and an illustrative depiction can be found in Figure 4.4. As previously elucidated, the manipulator's moving platform is supported by four identical RPR kinematic limbs. Each limb comprises a revolute joint ($R$), a prismatic joint ($P$), and another revolute joint ($R$), enabling complex movements and high flexibility.

A thorough examination of the Jacobian and singularity aspects of this manipulator is presented in Section 5.5. These analyses are crucial as they help in understanding the manipulator's velocity and force transmission characteristics and identifying configurations where the manipulator may lose degrees of freedom, respectively. The dynamic analysis, leveraging the Newton–Euler method, is detailed in Section 7.4.

To derive the equations of motion for the manipulator using the second kind of Lagrange formulation, it is essential to establish the Lagrangian function for the entire manipulator system in terms of suitably generalized coordinates. The generalized coordinates for the manipulator are taken as the position and orientation of the moving platform's center of mass, represented by $\boldsymbol{\chi} = [x_G, y_G, \phi]^T$. Here, $x_G$ and $y_G$ indicate the coordinates

of the moving platform's center of mass, while $\phi$ denotes the orientation angle of the platform relative to a fixed reference frame.

Given that the manipulator operates in a planar environment, only three coordinates are required to fully describe its motion. The movements of the limbs are entirely determined by the length of each limb, $L_i$, and the directions of the limbs are defined by the vector $\hat{\boldsymbol{S}}_i$. This can be simply represented by the limb angle $\alpha_i$, as shown in Figure 4.4. Consequently, the kinetic and potential energies of the limbs, which are functions of their respective variables, can be expressed in terms of the generalized coordinates of the system, $\boldsymbol{\chi}$. This transformation is achieved through the manipulator's inverse kinematics and Jacobian relations, which are comprehensively discussed in Sections 4.4 and 5.5, respectively.

To demonstrate the importance of carefully deriving the Lagrangian function for the manipulator to avoid complications in subsequent Lagrange formulations, we will first derive the kinetic energy of the limbs in a naive approach. This will illustrate the potential issues that can arise during the necessary manipulations. The kinetic energy of the limbs can be written as follows:

$$K_i = \frac{1}{2} m_i v_{c_i}^2 + \frac{1}{2} I_{c_i} \dot{\alpha}_i^2,$$

(8.50)

where $v_{c_i}$ represents the linear velocity of the limb's center of mass, and $\dot{\alpha}_i$ is the angular velocity of the limb. By substituting the inertial parameters, $m_i = \rho_m L_i$ and $I_{c_i} = \frac{1}{12}\rho_m L_i^3$ from Equation 7.28, and the linear velocity $\boldsymbol{v}_{c_i} = \dot{L}_i \hat{\boldsymbol{S}}_i + \frac{1}{2} L_i, \dot{\alpha}_i, (\hat{\boldsymbol{K}} \times \hat{\boldsymbol{S}}_i)$ from Equation 7.26, and simplifying Equation 8.50, we derive the kinetic energy of the limb as:

$$K_i = \frac{1}{2}\rho_m L_i \left( \dot{L}_i^2 + \frac{1}{3}(L_i \dot{\alpha}_i)^2 \right).$$

(8.51)

To apply the Lagrange equation of the second kind (Equation 8.14), we need to determine the partial derivatives of the kinetic energy with respect to the generalized coordinate $\dot{\boldsymbol{\chi}}$. Therefore, it is necessary to express the limb coordinates $L_i$ and $\alpha_i$, along with their time derivatives $\dot{L}_i$ and $\dot{\alpha}_i$, as functions of the generalized coordinate $\boldsymbol{\chi}$ and its time derivative $\dot{\boldsymbol{\chi}}$. The substitution of velocities is relatively straightforward, and the Jacobian matrices $\dot{\boldsymbol{L}} = \boldsymbol{J}\dot{\boldsymbol{\chi}}$ and $\dot{\boldsymbol{\alpha}} = \boldsymbol{J}_\alpha \dot{\boldsymbol{\chi}}$, provided in Equations 5.26 and 5.30, respectively, can be used for these transformations.

However, transforming the limb coordinates $L_i$ and $\alpha_i$ into the generalized coordinate $\boldsymbol{\chi} = [x_{_G}, y_{_G}, \phi]^T$ involves utilizing the inverse kinematics relations given in Equations

4.10 and 4.11. These inverse kinematics relations are nonlinear, which significantly complicates the resulting equation for the kinetic energy of the limbs. Consequently, performing the necessary partial derivatives becomes a very challenging task. This complexity underscores the need for meticulous care when deriving the Lagrangian function, as initial oversights or simplifications can lead to prohibitively complex calculations later on.

To summarize, a detailed and precise approach is required to derive the Lagrangian function of the manipulator. By carefully examining the transformations and substitutions required, and considering the complexities introduced by nonlinear inverse kinematics, we can avoid complications in subsequent Lagrange formulations.

As a common solution to this issue, many researchers suggest a modular approach where each limb is considered separately. This involves selecting a set of generalized coordinates that fully define the motion of each limb. By deriving the dynamic model of the limb using the Lagrange formulation, and then transforming these coordinates into the generalized coordinates used for the entire manipulator through a coordinate transformation, the dynamic model for the whole manipulator can be achieved. The transformed dynamic models of the individual limbs and the moving platform are then combined.

According to Reference [197], the careful selection of local generalized coordinates for the limbs significantly

influences the computational efficiency and simplicity of the resulting dynamic model. This approach, although avoids the necessity of evaluating Lagrange multipliers for closed-chain constraints, often involves a component-wise derivation of dynamic equations for the limbs. This can result in very complex and bulky formulations for the parallel manipulator, as discussed in studies such as Reference [197].

To manage this complexity, in the following sections we introduce tools for matrix derivation of the formulation. These tools are designed to significantly reduce the complexity of the equations, making them more manageable without relying on symbolic software packages. The basis for such manipulation is found in the closed-form Newton–Euler formulation discussed earlier in Section 7.5.4.

To derive the dynamic equations for the planar manipulator, the manipulator is decomposed into the moving platform and its four limbs. Each limb's dynamics are considered separately, and the overall dynamics are then synthesized. It is also assumed that the external forces acting on the limbs are due to the actuators and gravity, with no disturbance wrenches applied to the limbs. This detailed approach ensures that the manipulator's dynamic model is both accurate and computationally efficient, which is crucial for the optimal performance of parallel manipulators in various applications.

### 8.4.1 Dynamic Formulation of the Limbs

Consider the generalized coordinate for each limb to be the position of the moving attachment point $B_i$, denoted by $\boldsymbol{x}_i$. Although only the $x$ and $y$ components of the position vector $\boldsymbol{x}_i$ are needed to fully determine the location of each limb in planar motion, for the purposes of simplifying vector manipulations into standard matrix manipulations, the dimension of $\boldsymbol{x}_i$ is considered to be $3 \times 1$. This approach ensures that we can use well-established matrix operations for further analysis and derivations. It is important to note that while a $3 \times 1$ vector is used to represent the motion of the limb, its third component is always zero and does not contribute to the system's motion.

By using this local coordinate system for each limb, the governing equation of motion for each limb can be written in a standard form:

$$\boldsymbol{M}_i \ddot{\boldsymbol{x}}_i + \boldsymbol{\mathcal{C}}_i \dot{\boldsymbol{x}}_i + \boldsymbol{\mathcal{G}}_i = \boldsymbol{F}_i, \tag{8.52}$$

In this equation, $\boldsymbol{M}_i(\boldsymbol{x}_i)$ is the mass matrix, $\boldsymbol{\mathcal{C}}_i \dot{\boldsymbol{x}}_i$ represents the Christoffel matrix, $\boldsymbol{\mathcal{G}}_i$ accounts for the gravitational forces, and $\boldsymbol{F}_i$ is the vector of external forces acting on the limb. The mass matrix $\boldsymbol{M}_i(\boldsymbol{x}_i)$ can be directly derived from the kinetic energy of each limb. Specifically, the kinetic energy of each limb is given by:

$$K_i = \frac{1}{2} \dot{\boldsymbol{x}}_i^T \boldsymbol{M}_i \dot{\boldsymbol{x}}_i.$$

(8.53)

The kinetic energy $K_i$ of each limb is determined by summing the linear and angular kinetic energies. This can be expressed as:

$$K_i = \frac{1}{2} \boldsymbol{v}_{c_i}^T m_i \boldsymbol{v}_{c_i} + \frac{1}{2} \boldsymbol{\omega}_i^T \, I_{c_i} \, \boldsymbol{\omega}_i.$$

(8.54)

Here, $\boldsymbol{v}_{c_i}$ is the linear velocity of the center of mass of the limb, and $m_i$ is the mass of the limb. The term $\boldsymbol{\omega}_i$ represents the angular velocity of the limb, and $I_{c_i}$ is the moment of inertia about the center of mass.

Now, let's represent the velocity vector of the limb's center of mass as a function of the local generalized coordinate $\dot{\boldsymbol{x}}_i$. By recalling Equation 7.26 for linear velocity and expressing it in terms of the limb's angular velocity vector $\boldsymbol{\omega}_i$, we get:

$$\boldsymbol{v}_{c_i} = \dot{L}_i \hat{\boldsymbol{S}}_i + \frac{1}{2} L_i \boldsymbol{\omega}_i \times \hat{\boldsymbol{S}}_i.$$

(8.55)

Furthermore, both $L_i$ and $\boldsymbol{\omega}_i$ can be represented as functions of $\dot{\boldsymbol{x}}_i$ through the following derivations:

$$\dot{L}_i = \hat{\boldsymbol{S}}_i^T \dot{\boldsymbol{x}}_i \quad \Rightarrow \dot{L}_i \hat{\boldsymbol{S}}_i = \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T \dot{\boldsymbol{x}}_i$$

(8.56)

$$L_i \boldsymbol{\omega}_i = \hat{\boldsymbol{S}}_i \times \dot{\boldsymbol{x}}_i \quad \Rightarrow \boldsymbol{\omega}_i = \frac{1}{L_i} \hat{\boldsymbol{S}}_{i_\times} \dot{\boldsymbol{x}}_i$$

(8.57)

$$L_i \boldsymbol{\omega}_i \times \hat{\boldsymbol{S}}_i = (\hat{\boldsymbol{S}}_i \times \dot{\boldsymbol{x}}_i) \times \hat{\boldsymbol{S}}_i \quad = -\hat{\boldsymbol{S}}_i \times (\hat{\boldsymbol{S}}_i \times \dot{\boldsymbol{x}}_i) = -\boldsymbol{S}_{i_\times}^2 \dot{\boldsymbol{x}}_i.$$

(8.58)

Thus, the linear velocity of the limb can be written as:

$$\boldsymbol{v}_{c_i} = \left( \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T - \frac{1}{2} \hat{\boldsymbol{S}}_{i_\times}^2 \right) \dot{\boldsymbol{x}}_i.$$

(8.59)

Given that the motion of the system is planar, all rotations occur along the $\hat{\boldsymbol{z}}$ direction. Therefore, the only component in the inertia matrix contributing to the angular momentum is $I_{zz_i}$, which is simply denoted by $I_{c_i}$, as discussed in Section 7.4. Following a similar derivation to Equation 8.57, we find:

$$\begin{aligned}\boldsymbol{\omega}_i^T I_{c_i} \boldsymbol{\omega}_i &= I_{c_i} \boldsymbol{\omega}_i^T \boldsymbol{\omega}_i \\ &= \frac{1}{L_i^2} I_{c_i} \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^T \hat{\boldsymbol{S}}_{i_\times} \dot{\boldsymbol{x}}_i \\ &= -\frac{1}{L_i^2} I_{c_i} \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^2 \dot{\boldsymbol{x}}_i.\end{aligned}$$

(8.60)

Substituting these equations into Equation 8.54 and simplifying, we obtain:

$$K_i = \frac{1}{2} \boldsymbol{x}_i^T \left[ m_i \left( \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T - \frac{1}{2} \hat{\boldsymbol{S}}_{i_\times}^2 \right) - \frac{1}{L_i^2} I_{c_i} \hat{\boldsymbol{S}}_{i_\times}^2 \right] \boldsymbol{x}_i.$$

(8.61)

Substituting $m_i = \rho_m L_i$ and $I_{c_i} = \frac{\rho_m}{12} L_i^3$ and further simplifying Equation 8.61, we identify the limb mass matrix as:

$$\boldsymbol{M}_i = \rho_m L_i \left( \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T - \frac{7}{12} \hat{\boldsymbol{S}}_{i_\times}^2 \right).$$

(8.62)

As an alternative representation, we use the following relation:

$$\hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T = \boldsymbol{I}_{3\times 3} + \hat{\boldsymbol{S}}_{i_\times}^2,$$

(8.63)

Using this relation, the mass matrix can be derived as:

$$\boldsymbol{M}_i = \rho_m L_i \left( \boldsymbol{I}_{3\times 3} + \frac{5}{12} \hat{\boldsymbol{S}}_{i_\times}^2 \right).$$

(8.64)

The next step involves evaluating the gravity vector $\boldsymbol{\mathcal{G}}_i$. This term can be determined by either taking the partial derivative of the potential energy with respect to the generalized coordinate $\boldsymbol{x}_i$ or by applying the principle of virtual work to the gravitational forces. In this case, the first method is used. The potential energy of the limbs is given by the equation:

$$P_i = -\boldsymbol{g}^T m_i c_i \hat{\boldsymbol{S}}_i = \frac{1}{2} \boldsymbol{g}^T \rho_m L_i^2 \hat{\boldsymbol{S}}_i.$$

(8.65)

Here, $\boldsymbol{g}$ represents the gravitational vector, which is assumed to act in an arbitrary direction relative to the plane of motion. The fixed attachment point $\boldsymbol{A}_i$ is chosen as the reference point for zero potential energy. To find the partial derivative of the potential energy with respect to the generalized coordinate $\boldsymbol{x}_i$, we first take the differential of the potential energy $dP$ and express all differential variables in terms of the generalized coordinate.

$$dP_i = -\frac{\rho_m}{2}\boldsymbol{g}^T\left(2L_i dL_i \hat{\boldsymbol{S}}_i + L_i^2 d\hat{\boldsymbol{S}}_i\right).$$

(8.66)

To convert $dP$ into a function of generalized coordinates, we need to determine $d\hat{\boldsymbol{S}}_i$ and $dL_i\hat{\boldsymbol{S}}_i$ as functions of $d\boldsymbol{x}_i$. These terms are derived as follows:

$$\begin{aligned}\boldsymbol{x}_i &= \boldsymbol{a}_i + L_i\hat{\boldsymbol{S}}_i \\ d\boldsymbol{x}_i &= L_i d\hat{\boldsymbol{S}}_i + dL_i\hat{\boldsymbol{S}}_i.\end{aligned}$$

(8.67)

Thus,

$$d\hat{\boldsymbol{S}}_i = \frac{1}{L_i}(d\boldsymbol{x}_i - dL_i\hat{\boldsymbol{S}}_i).$$

(8.68)

To eliminate $d\hat{\boldsymbol{S}}_i$, dot multiply Equation 8.67 by $\hat{\boldsymbol{S}}_i$ and note that $\hat{\boldsymbol{S}}_i \cdot d\hat{\boldsymbol{S}}_i = 0$:

$$\hat{\boldsymbol{S}}_i \cdot d\boldsymbol{x}_i = dL_i\hat{\boldsymbol{S}}_i \cdot \hat{\boldsymbol{S}}_i = dL_i.$$

(8.69)

Hence,

$$dL_i\hat{\boldsymbol{S}}_i = (\hat{\boldsymbol{S}}_i \cdot d\boldsymbol{x}_i)\hat{\boldsymbol{S}}_i = \hat{\boldsymbol{S}}_i\hat{\boldsymbol{S}}_i^T d\boldsymbol{x}_i.$$

(8.70)

Next, cross-multiply Equation 8.67 by $\hat{\boldsymbol{S}}_i$ to eliminate $dL_i$:

$$\hat{\boldsymbol{S}}_i \times d\boldsymbol{x}_i = L_i(\hat{\boldsymbol{S}}_i \times d\hat{\boldsymbol{S}}_i).$$

(8.71)

To proceed, cross-multiply both sides of the equation by $\hat{\boldsymbol{S}}_i$:

$$\begin{aligned}\hat{\boldsymbol{S}}_i \times (\hat{\boldsymbol{S}}_i \times d\boldsymbol{x}_i) &= L_i\hat{\boldsymbol{S}}_i \times (\hat{\boldsymbol{S}}_i \times d\hat{\boldsymbol{S}}_i) \\ &= L_i\left(\hat{\boldsymbol{S}}_i(\hat{\boldsymbol{S}}_i \cdot d\hat{\boldsymbol{S}}_i) - d\hat{\boldsymbol{S}}_i(\hat{\boldsymbol{S}}_i \cdot \hat{\boldsymbol{S}}_i)\right) \\ &= -L_i d\hat{\boldsymbol{S}}_i.\end{aligned}$$

(8.72)

From this, it follows that:

$$d\hat{\boldsymbol{S}}_i = -\frac{1}{L_i}\hat{\boldsymbol{S}}_{i_\times}^2 d\boldsymbol{x}_i.$$

(8.73)

Next, substitute Equations 8.73 and 8.70 into the differential of potential energy as given in Equation 8.66, and simplify to obtain:

$$dP_i = \rho_m L_i \boldsymbol{g}^T \left( \frac{1}{2} \hat{\boldsymbol{S}}_{i_\times}^2 - \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T \right) d\boldsymbol{x}_i.$$

(8.74)

To determine the partial derivative of the potential energy $P_i$ with respect to the generalized coordinate $\boldsymbol{x}_i$, we use the following equation. Although the potential energy itself is a scalar, its gradient $\partial P_i / \partial \boldsymbol{x}_i$, which equals the gravity vector $\boldsymbol{\mathcal{G}}_i$, results in an $n \times 1$ column vector. Therefore, we must transpose Equation 8.74 to achieve the correct dimensions:

$$\begin{aligned}
\frac{\partial P_i}{\partial \boldsymbol{x}_i} = \boldsymbol{\mathcal{G}}_i \quad &= \rho_m L_i \left[ \boldsymbol{g}^T \left( \frac{1}{2} \hat{\boldsymbol{S}}_{i_\times}^2 - \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T \right) \right]^T \\
&= \rho_m L_i \left( \frac{1}{2} \hat{\boldsymbol{S}}_{i_\times}^2 - \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T \right)^T \boldsymbol{g} \\
&= \rho_m L_i \left( \frac{1}{2} \left( \hat{\boldsymbol{S}}_{i_\times}^2 \right)^T - \left( \hat{\boldsymbol{S}}_i^T \right)^T \hat{\boldsymbol{S}}_i \right) \boldsymbol{g} \\
&= \rho_m L_i \left( \frac{1}{2} \hat{\boldsymbol{S}}_{i_\times}^2 - \hat{\boldsymbol{S}}_i \hat{\boldsymbol{S}}_i^T \right) \boldsymbol{g}.
\end{aligned}$$

(8.75)

This process of substituting these expressions into the differential form of the potential energy helps in accurately determining the gravity vector $\boldsymbol{\mathcal{G}}_i$ in the context of the manipulator's dynamic model.

The final term that needs to be calculated is the Coriolis and centrifugal force vector, denoted as $\boldsymbol{\nu}_i = \boldsymbol{\mathcal{C}}_i \dot{\boldsymbol{x}}_i$. This vector represents the forces in the manipulator system resulting from rotational effects and can be computed directly from the time derivative of the Lagrangian function. Specifically, this involves using the formulation given by Lagrange's Equation 8.15. Alternatively, it can also be determined using the time derivative of the mass matrix and skew-symmetric matrices previously discussed.

To compute $\boldsymbol{\nu}_i$, we will use the time derivative of the Lagrangian function, following the general approach outlined in Equation 8.25. For a limb with generalized coordinates, this can be expressed as:

$$\boldsymbol{\nu}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) = \dot{\boldsymbol{M}}_i(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T \boldsymbol{M}_i(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i\right). \tag{8.76}$$

To compute the Coriolis and centrifugal force vector from this equation, we need the time derivative of the mass matrix as well as the partial derivative of the kinetic energy with respect to the generalized coordinate. Calculating these terms involves deriving several intermediate variables as functions of the generalized coordinate $\boldsymbol{x}_i$. We will first develop the necessary relations and then use them to simplify Equation 8.76.

To begin, we need to derive the following relations, which will aid in simplifying the second term of Equation 8.76:

$$\frac{\partial L_i}{\partial \boldsymbol{x}_i} = \hat{\boldsymbol{S}}_i; \quad \frac{\partial}{\partial \boldsymbol{x}_i}\left(\frac{1}{L_i^2}\right) = -\frac{2}{L_i^3}\hat{\boldsymbol{S}}_i; \quad \frac{\partial \hat{\boldsymbol{S}}_i}{\partial \boldsymbol{x}_i} = -\frac{1}{L_i}\hat{\boldsymbol{S}}_{i_\times}^2.$$

(8.77)

Moreover, with some manipulations, the following equations can be derived:

$$\begin{aligned}
\frac{d}{dt}\left(\hat{\boldsymbol{S}}_{i_\times}^2\right) &= -\frac{1}{L_i}\hat{\boldsymbol{S}}_i\dot{\boldsymbol{x}}_i\hat{\boldsymbol{S}}_{i_\times}^2 - \frac{1}{L_i}\hat{\boldsymbol{S}}_{i_\times}^2\dot{\boldsymbol{x}}_i\hat{\boldsymbol{S}}_i^T, \\
\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{S}}_i\hat{\boldsymbol{S}}_i^T\dot{\boldsymbol{x}}_i\right) &= -\frac{2}{L_i}\hat{\boldsymbol{S}}_{i_\times}^2\dot{\boldsymbol{x}}_i\hat{\boldsymbol{S}}_i^T\dot{\boldsymbol{x}}_i, \\
\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{S}}_{i_\times}^2\dot{\boldsymbol{x}}_i\right) &= -\frac{2}{L_i}\dot{L}_i\hat{\boldsymbol{S}}_{i_\times}^2\dot{\boldsymbol{x}}_i.
\end{aligned}$$

(8.78)

The first term of the Coriolis and centrifugal vector in Equation 8.76 is simplified as follows:

$$\begin{aligned}
\dot{\boldsymbol{M}}_i &= \frac{d}{dt}\left(\rho_m L_i\left(\boldsymbol{I}_{3\times 3} + \frac{5}{12}\hat{\boldsymbol{S}}_{i_\times}^2\right)\right) \\
&= \rho_m\dot{L}_i\left(\boldsymbol{I}_{3\times 3} + \frac{5}{12}\hat{\boldsymbol{S}}_{i_\times}^2\right) - \frac{5}{12}\rho_m L_i\left(\frac{1}{L_i}\hat{\boldsymbol{S}}_i\dot{\boldsymbol{x}}_i\hat{\boldsymbol{S}}_{i_\times}^2 - \frac{1}{L}\right.
\end{aligned}$$

(8.79)

Additionally, observe that:

$$\dot{L}_i = \frac{\partial L_i}{\partial \boldsymbol{x}_i} \cdot \dot{\boldsymbol{x}}_i = \hat{\boldsymbol{S}}_i \cdot \dot{\boldsymbol{x}}_i = \hat{\boldsymbol{S}}_i^T \dot{\boldsymbol{x}}_i.$$

(8.80)

Thus, the term $\hat{\boldsymbol{S}}_{i_\times}^2 \dot{\boldsymbol{x}}_i, \hat{\boldsymbol{S}}_i^T \dot{\boldsymbol{x}}_i$ simplifies to $\dot{L}_i \hat{\boldsymbol{S}}_{i_\times}^2 \dot{\boldsymbol{x}}_i$. Substituting this into Equation 8.79 and simplifying, we get:

$$\begin{aligned}\dot{\boldsymbol{M}}_i \dot{\boldsymbol{x}}_i &= \left( \rho_m \dot{L}_i + \frac{5}{12} \rho_m \dot{L}_i \hat{\boldsymbol{S}}_{i_\times}^2 - \frac{5}{12} \rho_m \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^2 - \frac{5}{12} \rho_m \dot{L} \right. \\ &= \rho_m \dot{L}_i \dot{\boldsymbol{x}}_i - \frac{5}{12} \rho_m \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^2 \dot{\boldsymbol{x}}_i.\end{aligned}$$

(8.81)

The second term of the Coriolis and centrifugal vector given in Equation 8.76 can be simplified as follows:

$$\begin{aligned}-\frac{1}{2} \frac{\partial}{\partial \boldsymbol{x}_i} \left( \dot{\boldsymbol{x}}_i^T \boldsymbol{M}_i(\boldsymbol{x}_i) \dot{\boldsymbol{x}}_i \right) &= -\frac{1}{2} \frac{\partial}{\partial \boldsymbol{x}_i} \left[ \rho_m L_i \dot{\boldsymbol{x}}_i^T \left( \boldsymbol{I}_{3\times 3} + \frac{5}{12} \hat{\boldsymbol{S}} \right. \right. \\ &= -\frac{\rho_m}{2} \left[ \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T \left( \boldsymbol{I}_{3\times 3} + \frac{5}{12} \hat{\boldsymbol{S}}_{i_\times}^2 \right) \dot{\boldsymbol{x}}_i \right. \\ &= -\frac{\rho_m}{2} \left( \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T + \frac{5}{12} \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^2 - \frac{5}{6} \right.\end{aligned}$$

(8.82)

By combining Equations 8.81 and 8.82 and performing the necessary simplifications, the Christoffel matrix can be expressed as:

$$\boldsymbol{\mathcal{V}}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) = \rho_m \left( \dot{L}_i - \frac{1}{2} \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T - \frac{5}{8} \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^2 + \frac{5}{12} \dot{L}_i \boldsymbol{S}_{i_\times}^2 \right) \dot{\boldsymbol{x}}_i \tag{8.83}$$

Therefore, the matrix $\boldsymbol{\mathcal{C}}_i$ that satisfies the relationship $\boldsymbol{\mathcal{V}}_i = \boldsymbol{\mathcal{C}}_i \dot{\boldsymbol{x}}_i$ is given by:

$$\boldsymbol{\mathcal{C}}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) = \rho_m \left( \dot{L}_i - \frac{1}{2} \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T - \frac{5}{8} \hat{\boldsymbol{S}}_i \dot{\boldsymbol{x}}_i^T \hat{\boldsymbol{S}}_{i_\times}^2 + \frac{5}{12} \dot{L}_i \boldsymbol{S}_{i_\times}^2 \right). \tag{8.84}$$

This matrix $\boldsymbol{\mathcal{C}}_i$, encapsulates the Coriolis and centrifugal effects as a function of the generalized coordinate $\boldsymbol{x}_i$ and its derivative $\dot{\boldsymbol{x}}_i$. It plays a crucial role in accurately modeling the dynamics of the system, accounting for the complex interactions between linear and rotational components of the motion.

### 8.4.2 Dynamic Formulation of the Moving Platform

In this section, we will derive the dynamic formulation for the moving platform by first addressing its mass matrix $\boldsymbol{M}_p$

, followed by the Christoffel matrix $\boldsymbol{\mathcal{C}}_p$, and then the gravity vector $\boldsymbol{\mathcal{G}}_p$.

To begin, we focus on the mass matrix, which is a crucial component for calculating the kinetic energy of the moving platform. The kinetic energy of the moving platform can be expressed as

$$K_p = \frac{1}{2} M \boldsymbol{v}_G^2 + \frac{1}{2} I_m \dot{\phi}^2,$$

(8.85)

where $\boldsymbol{v}_G$ represents the linear velocity of the center of mass of the moving platform, and $\dot{\phi}$ denotes the angular velocity. In this equation, $M$ is the mass of the moving platform, and $I_m$ is its moment of inertia. To express this in matrix form, we use the manipulator generalized coordinate vector $\boldsymbol{\chi} = [x_G, y_G, \phi]^T$. Substituting these coordinates into the equation yields:

$$K_p = \frac{1}{2} \dot{\boldsymbol{\chi}}^T \boldsymbol{M}_p \dot{\boldsymbol{\chi}},$$

(8.86)

where the mass matrix $\boldsymbol{M}_p$ is defined as:

$$\boldsymbol{M}_p = \begin{bmatrix} M & 0 & 0 \\ 0 & M0 & \\ 0 & 0 & I_m \end{bmatrix}.$$

(8.87)

This mass matrix $\boldsymbol{M}_p$ is a constant diagonal matrix, indicating that the mass and inertia properties of the moving platform are independent of its configuration. Consequently, the time derivative of $\boldsymbol{M}_p$ is zero, and therefore,

$$\dot{\boldsymbol{M}}_p\dot{\boldsymbol{\chi}} = \mathbf{0}; \quad -\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\chi}}\left(\dot{\boldsymbol{\chi}}^T \boldsymbol{M}_p \dot{\boldsymbol{\chi}}\right) = \mathbf{0}.$$

(8.88)

This result implies that the Coriolis and centrifugal forces, represented by the Christoffel matrix $\boldsymbol{\nu}_p(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$, are zero.

Next, we need to determine the gravity vector $\boldsymbol{\mathcal{G}}_p$ for the moving platform. This can be obtained either by computing the partial derivative of the potential energy with respect to the generalized coordinate $\boldsymbol{\chi}$ or by applying the principle of virtual work. Here, we use the potential energy approach. The potential energy $P_p$ of the moving platform, considering the gravitational force $\boldsymbol{g}$, is given by:

$$P_p = -M\boldsymbol{g} \cdot \begin{bmatrix} x_G \\ y_G \\ 0 \end{bmatrix}.$$

(8.89)

Since the potential energy is independent of the orientation $\phi$, its partial derivative with respect to $\phi$ is zero. Thus, the gravity vector $\boldsymbol{\mathcal{G}}_p$ can be calculated as:

$$\boldsymbol{\mathcal{G}}_p = \frac{\partial P_p}{\partial \boldsymbol{\chi}} = -M\boldsymbol{g} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -M \begin{bmatrix} \boldsymbol{g} \cdot \hat{\boldsymbol{x}} \\ \boldsymbol{g} \cdot \hat{\boldsymbol{y}} \\ 0 \end{bmatrix}. \tag{8.90}$$

In this expression, $\boldsymbol{g} \cdot \hat{\boldsymbol{x}}$ and $\boldsymbol{g} \cdot \hat{\boldsymbol{y}}$ represent the components of the gravitational force vector along the $x$ and $y$ axes, respectively. This completes the derivation of the necessary components for the dynamic formulation of the moving platform.

### 8.4.3 Dynamic Formulation of the Whole Manipulator

To develop the closed-form dynamic formulation for the entire manipulator, it is essential to perform a coordinate transformation that maps the intermediate generalized coordinates $\boldsymbol{x}_i$ to the primary generalized coordinates $\boldsymbol{\chi}$ [193]. This transformation is crucial because it enables the integration of the equations governing the individual limbs and the moving platform into a cohesive dynamic model for the entire system.

To achieve this, we need to define a Jacobian matrix $\boldsymbol{J}_i$ that links the intermediate coordinates $\boldsymbol{x}_i$ with the

principal generalized coordinates $\boldsymbol{\chi}$. This Jacobian matrix captures how changes in the generalized coordinates affect the velocities of the intermediate points. The relationship between these coordinates can be expressed as:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{J}_i \dot{\boldsymbol{\chi}},$$

(8.91)

where $\dot{\boldsymbol{\chi}} = [\dot{x}_G, \dot{y}_G, \dot{\phi}]^T$ represents the velocities of the principal generalized coordinates.

The velocity of the moving attachment point $\boldsymbol{x}_i$ is influenced both by the linear velocity $\boldsymbol{v}_G$ of the center of mass and by the rotational effects represented by $(\hat{\boldsymbol{K}} \times \boldsymbol{E}_i)$. Here, $\hat{\boldsymbol{K}}$ is a unit vector in the $\boldsymbol{z}$ direction, which is orthogonal to the plane of motion. This can be written as:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{v}_G + (\hat{\boldsymbol{K}} \times \boldsymbol{E}_i)\dot{\phi},$$

(8.92)

To represent this relationship in matrix form, we rearrange the equation to:

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \hat{\boldsymbol{K}} \times \boldsymbol{E}_i \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} \dot{x}_G \\ \dot{y}_G \\ \dot{\phi} \end{bmatrix}.$$

(8.93)

This matrix form shows how each component of the velocity vector $\dot{\boldsymbol{\chi}}$ contributes to the velocity of $\dot{\boldsymbol{x}}_i$. Consequently, the Jacobian matrix $\boldsymbol{J}_i$ that relates the intermediate coordinates to the generalized coordinates is:

$$\boldsymbol{J}_i = \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \hat{\boldsymbol{K}} \times \boldsymbol{E}_i \\ 0 & 0 & \\ & & \end{bmatrix}.$$

(8.94)

By using this Jacobian matrix, we can effectively map the velocities from the intermediate coordinate system to the primary coordinate system, allowing us to combine the dynamic equations for the different components of the manipulator into a unified formulation.

To derive the closed-form dynamic formulation for the entire manipulator, it is essential to calculate the time derivative of the Jacobian matrix $\boldsymbol{J}_i$. This derivative can be expressed as:

$$\dot{\boldsymbol{J}}_i = \left[\mathbf{0}_{3\times 2} \quad \hat{\boldsymbol{K}} \times \left(\hat{\boldsymbol{K}} \times \boldsymbol{E}_i\right)\dot{\phi}\right] = \left[\mathbf{0}_{3\times 2} \quad -\boldsymbol{E}_i\dot{\phi}\right]$$

(8.95)

In this equation, $\hat{\boldsymbol{K}} \times \left( \hat{\boldsymbol{K}} \times \boldsymbol{E}_i \right)$ represents the cross product involving the unit vector $\hat{\boldsymbol{K}}$ and $\boldsymbol{E}_i$, and $\dot{\phi}$ is the angular velocity. This result simplifies to $-\boldsymbol{E}_i \dot{\phi}$.

Next, substitute $\dot{\boldsymbol{x}}_i$ from Equation 8.93 into Equation 8.52 to incorporate the velocity relationship derived from the Jacobian. Then, left-multiply both sides of the resulting equation by the transpose of the Jacobian matrix $\boldsymbol{J}_i^T$. This manipulation yields:

$$\begin{aligned} &\boldsymbol{J}_i{}^T \boldsymbol{M}_i \left( \boldsymbol{J}_i \ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{\chi}} \right) + \quad \boldsymbol{J}_i{}^T \boldsymbol{\mathcal{C}}_i \boldsymbol{J}_i \dot{\boldsymbol{\chi}} + \boldsymbol{J}_i{}^T \boldsymbol{\mathcal{G}}_i = \boldsymbol{J}_i{}^T \boldsymbol{F}_i \\ &(\boldsymbol{J}_i{}^T \boldsymbol{M}_i \boldsymbol{J}_i) \ddot{\boldsymbol{\chi}} + (\boldsymbol{J}_i{}^T \boldsymbol{M}_i \dot{\boldsymbol{J}}_i \quad + \boldsymbol{J}_i{}^T \boldsymbol{\mathcal{C}}_i \boldsymbol{J}_i) \dot{\boldsymbol{\chi}} + \boldsymbol{J}_i{}^T \boldsymbol{\mathcal{G}}_i = \boldsymbol{J}_i{}^T \boldsymbol{F}_i. \end{aligned} \tag{8.96}$$

This equation represents the dynamic model for each limb in the manipulator, taking into account the contributions of the moving platform and other components.

Rewriting this equation in a more compact closed-form notation gives us:

$$\boldsymbol{M}_{li} \ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}_{li} \dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}_{li} = \boldsymbol{F}_{li}, \tag{8.97}$$

where:

$$
\begin{aligned}
\boldsymbol{M}_{li} &= \boldsymbol{J}_i^T \boldsymbol{M}_i \boldsymbol{J}_i \\
\boldsymbol{\mathcal{C}}_{li} &= \boldsymbol{J}_i^T \boldsymbol{M}_i \dot{\boldsymbol{J}}_i + \boldsymbol{J}_i^T \boldsymbol{\mathcal{C}}_i \boldsymbol{J}_i \\
\boldsymbol{\mathcal{G}}_{li} &= \boldsymbol{J}_i^T G_i.
\end{aligned}
\tag{8.98}
$$

In this notation, $\boldsymbol{M}_{li}$ is the effective mass matrix for the limb, $\boldsymbol{\mathcal{C}}_{li}$ is the effective Christoffel matrix, and $\boldsymbol{\mathcal{G}}_{li}$ is the effective gravity vector.

Combining the contributions from all limbs and the moving platform, the complete dynamic formulation for the entire manipulator is expressed as:

$$
\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{Q},
\tag{8.99}
$$

where:

$$
\begin{aligned}
\boldsymbol{M}\left(\boldsymbol{\chi}\right) &= \boldsymbol{M}_p + \sum_{i=1}^{i=4} \boldsymbol{M}_{li} \\
\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) &= \boldsymbol{\mathcal{C}}_p + \sum_{i=1}^{i=4} \boldsymbol{\mathcal{C}}_{li} \\
\boldsymbol{\mathcal{G}}\left(\boldsymbol{\chi}\right) &= \boldsymbol{\mathcal{G}}_p + \sum_{i=1}^{i=4} \boldsymbol{\mathcal{G}}_{li}.
\end{aligned}
\tag{8.100}
$$

In this formulation, $\boldsymbol{M}(\boldsymbol{\chi})$ is the total mass matrix, $\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is the total Christoffel matrix, and $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$ is the total gravity vector for the manipulator.

The term $\boldsymbol{Q}$ represents the generalized forces acting on the manipulator. According to the Lagrangian approach, internal forces such as $\boldsymbol{f}_{b_i}$ do not need to be included in this formulation. Instead, the focus is on the external forces, which include the actuator forces $\boldsymbol{\tau}$ and any disturbance wrenches $\boldsymbol{\mathcal{F}}_d$. The disturbance wrench $\boldsymbol{\mathcal{F}}_d$ is defined as:

$$\boldsymbol{\mathcal{F}}_d = \begin{bmatrix} f_{d_x} \\ f_{d_y} \\ \tau_D \end{bmatrix}. \tag{8.101}$$

The principle of virtual work is a powerful tool for determining the generalized forces acting on the manipulator, originating from two primary sources: the actuator forces and the disturbance wrench. By employing this principle, we can effectively project the actuator forces onto the center of mass of the moving platform, represented as $\boldsymbol{J}^T\boldsymbol{\tau}$. Simultaneously, the disturbance wrench $\boldsymbol{\mathcal{F}}_d$ contributes directly as a generalized force. As a result, the generalized force vector $\boldsymbol{Q}$ can be calculated by combining these two terms:

$$\boldsymbol{Q} = \boldsymbol{J}^T \boldsymbol{\tau} + \boldsymbol{\mathcal{F}}_d.$$

(8.102)

This expression provides a comprehensive view of the forces influencing the manipulator, encompassing both the forces exerted by the actuators and external disturbances.

Thus, the final equations of motion for the planar manipulator are fully derived and presented in a closed form, as shown in Equation 8.99. This closed-form representation is advantageous as it simplifies the dynamic modeling process compared to the componentwise derivation methods, which involve a detailed application of Newton-Euler and virtual work principles. While these alternative methods can indeed yield equivalent results, the closed-form approach provides a more streamlined and accessible formulation.

## 8.5 Case Study II: Stewart–Gough Platform

In this section, we provide an in-depth dynamic analysis of the Stewart–Gough platform through the application of the Euler-Lagrange formulation. This analysis builds upon the manipulator's architecture, which is thoroughly detailed in Section 4.6.1 and depicted in Figure 4.16. As previously outlined, the Stewart–Gough platform's spatial motion is facilitated by six piston–cylinder actuators that work in

tandem to control the position and orientation of the moving platform. A comprehensive kinematic analysis of the manipulator, which is essential for understanding its movement and configuration, is provided in Section 4.6. In addition, the Jacobian matrix, which plays a crucial role in relating velocities of the moving platform to those of the actuators, is analyzed in Section 5.7. Furthermore, the dynamic formulation of the Stewart–Gough platform is explored through the Newton–Euler method, as detailed in Section 7.5

### 8.5.1 Dynamic Formulation of the Limbs

To derive the dynamic formulation for the Stewart–Gough platform, we begin by breaking down the manipulator into two sets of components: the moving platform and its six individual limbs. Each limb is further divided into two segments: a cylinder and a piston. The centers of mass for the cylinder and piston are positioned at $c_{i_1}$ and $c_{i_2}$ from their respective endpoints, and their masses are represented by $m_{i_1}$ and $m_{i_2}$, respectively. For this analysis, it is assumed that the external forces on the limbs consist only of actuator and gravitational forces, with no additional disturbance wrench applied.

We denote the generalized coordinate for each limb as the position of the moving attachment point $B_i$, which is represented by $\boldsymbol{x}_i$. The objective of this section is to determine the dynamic equation for each limb in a closed

form, as provided in Equation 7.115, which we restate here for reference:

$$\boldsymbol{M}_i\ddot{\boldsymbol{x}}_i + \boldsymbol{\mathcal{C}}_i\dot{\boldsymbol{x}}_i + \boldsymbol{\mathcal{G}}_i = \boldsymbol{F}_i.$$

(8.103)

The mass matrix $\boldsymbol{M}_i(\boldsymbol{x}_i)$ for each limb can be derived from its kinetic energy. Specifically, the kinetic energy of each limb $K_i$ is given by:

$$K_i = \frac{1}{2}\dot{\boldsymbol{x}}_i^T\boldsymbol{M}_i\dot{\boldsymbol{x}}_i,$$

(8.104)

where the total kinetic energy $K_i$ is the sum of the kinetic energies of both the cylinder and the piston. This can be expressed as:

$$K_i = \frac{1}{2}\boldsymbol{v}_{i_1}^T m_{i_1}\boldsymbol{v}_{i_1} + \frac{1}{2}\boldsymbol{\omega}_i^{T\,A}I_{c_{i_1}}\boldsymbol{\omega}_i + \frac{1}{2}\boldsymbol{v}_{i_2}^T m_{i_2}\boldsymbol{v}_{i_2} + \frac{1}{2}\boldsymbol{\omega}_i^{T\,A}I_{c_{i_2}}\boldsymbol{\omega}$$

(8.105)

Here, $\boldsymbol{v}_{i_1}$ and $\boldsymbol{v}_{i_2}$ represent the velocities of the cylinder and piston center of masses, respectively, while $\boldsymbol{\omega}_i$ denotes their angular velocity. ${}^{A}I_{c_{i_1}}$ and ${}^{A}I_{c_{i_2}}$ are the moments of inertia for the cylinder and piston. This formulation allows

us to accurately capture the kinetic energy and derive the corresponding mass matrix for each limb.

To simplify the kinetic energy expression for each limb, we substitute $\boldsymbol{I}_{\text{eq}}$, defined in Equation 7.117 as the sum of ${}^{A}I_{c_{i_1}}$ and ${}^{A}I_{c_{i_2}}$, into the equation. This results in the kinetic energy of each limb being represented as follows:

$$K_i = \frac{1}{2}\boldsymbol{v}_{i_1}^T m_{i_1}\boldsymbol{v}_{i_1} + \frac{1}{2}\boldsymbol{v}_{i_2}^T m_{i_2}\boldsymbol{v}_{i_2} + \frac{1}{2}\boldsymbol{\omega}_i^T \boldsymbol{I}_{\text{eq}}\boldsymbol{\omega}_i. \tag{8.106}$$

Next, we express the velocity vectors of the cylinder and piston centers of mass as functions of the generalized coordinate $\dot{\boldsymbol{x}}_i$. According to Equations 9.63 and 9.64, these velocities are given by:

$$\boldsymbol{v}_{i_1} = -\frac{c_{i_1}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2 \dot{\boldsymbol{x}}_i, \tag{8.107}$$

$$\boldsymbol{v}_{i_2} = -\frac{c_{i_2}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2 \dot{\boldsymbol{x}}_i + \hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i. \tag{8.108}$$

Furthermore, the angular velocity $\boldsymbol{\omega}_i$ can be expressed in terms of $\dot{\boldsymbol{x}}_i$ as follows, which is also given in Equation 7.54:

$$\boldsymbol{\omega}_i = \frac{1}{\ell_i}(\hat{\boldsymbol{s}}_i \times \dot{\boldsymbol{x}}_i) = \frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}\dot{\boldsymbol{x}}_i$$

(8.109)

Assuming that the limbs are symmetric with respect to their longitudinal axes, similar to the derivation in Equation 7.93, it can be shown that:

$${}^{A}\boldsymbol{I}_{\text{eq}}\boldsymbol{\omega}_i = I_{xx_i}\boldsymbol{\omega}_i.$$

(8.110)

Substituting Equation 8.109 into this expression and simplifying, we get:

$$\begin{aligned}\boldsymbol{\omega}_i^{T\,A}\boldsymbol{I}_{\text{eq}}\boldsymbol{\omega}_i &= I_{xx_i}\boldsymbol{\omega}_i^T\boldsymbol{\omega}_i \\ &= \frac{1}{\ell_i^2}I_{xx_i}\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^T\hat{\boldsymbol{s}}_{i_\times}\dot{\boldsymbol{x}}_i \\ &= -\frac{1}{\ell_i^2}I_{xx_i}\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i.\end{aligned}$$

(8.111)

Now, substituting these results into Equation 8.105 and simplifying the kinetic energy expression becomes:

$$K_i = \frac{1}{2}\boldsymbol{x}_i^T\left(\frac{1}{\ell_i^2}m_{i_1}c_{1_1}^2\hat{\boldsymbol{s}}_{i_\times}^4 + m_{i_2}\left(-\frac{1}{\ell_i}c_{i_2}\hat{\boldsymbol{s}}_{i_\times}^2 + \hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\right)^2 - \frac{1}{\ell_i^2}I$$

(8.112)

This expression can be further simplified using the following properties of $\hat{\boldsymbol{s}}_{i_\times}$:

$$\hat{\boldsymbol{s}}_{i_\times}^4 = -\hat{\boldsymbol{s}}_{i_\times}^2$$

(8.113)

$$\hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T = \boldsymbol{I}_{3\times 3} + \hat{\boldsymbol{s}}_{i_\times}^2$$

(8.114)

$$\hat{\boldsymbol{s}}_{i_\times}^2 \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T = \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T \hat{\boldsymbol{s}}_{i_\times}^2 = \boldsymbol{0}.$$

(8.115)

Therefore, the mass matrix for each limb simplifies to:

$$\begin{aligned} \boldsymbol{M}_i &= -\frac{1}{\ell_i^2} m_{i_1} c_{i_1}^2 \hat{\boldsymbol{s}}_{i_\times}^2 - \frac{1}{\ell_i^2} m_{i_2} c_{i_2}^2 \hat{\boldsymbol{s}}_{i_\times}^2 + m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T - \frac{1}{\ell_i^2} I_{xx_i} \hat{\boldsymbol{s}}_{i_\times}^2 \\ &= m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T - m_{c_e} \hat{\boldsymbol{s}}_{i_\times}^2 - \frac{1}{\ell_i^2} I_{xx_i} \hat{\boldsymbol{s}}_{i_\times}^2, \end{aligned}$$

(8.116)

where $m_{c_e}$ is defined in Equation 7.117. This derived mass matrix is consistent with the one computed using the Newton-Euler formulation, as given in Equation 7.116. This

confirms the validity of the closed-form dynamic formulation for each limb in the Stewart-Gough platform.

The gravity vector $\boldsymbol{\mathcal{G}}_i$ is the next term that needs to be evaluated. This vector quantifies the gravitational force acting on the system and can be determined through one of two methods: either by calculating the partial derivative of the potential energy with respect to the generalized coordinate $\boldsymbol{x}_i$, or by applying the principle of virtual work to the gravitational forces. For this calculation, we use the first method.

The potential energy associated with the limbs of the system is given by the following equation:

$$P_i = -\boldsymbol{g}^T \left[ m_{i_1} c_{i_1} \hat{\boldsymbol{s}}_i + m_{i_2} (\ell_i - c_{i_2}) \hat{\boldsymbol{s}}_i \right]. \tag{8.117}$$

In this equation, $m_{i_1}$ and $m_{i_2}$ represent the masses of different segments of the limb, $c_{i_1}$ and $c_{i_2}$ are specific distances associated with these segments, and $\hat{\boldsymbol{s}}_i$ is the unit vector in the direction of the limb. The fixed attachment point $A_i$ serves as the reference for zero potential energy.

To determine the partial derivative of the potential energy $P_i$ with respect to the generalized coordinate $\boldsymbol{x}_i$, we first take the differential of $P_i$, denoted as $dP_i$. This differential expression must account for changes in the generalized coordinate $\boldsymbol{x}_i$. Hence, we express all differential variables in terms of $\boldsymbol{x}_i$:

$$dP_i = -\boldsymbol{g}^T \left(m_{i_1} c_{i_1} + m_{i_2}(\ell_i - c_{i_2})\right) d\hat{\boldsymbol{s}}_i - m_{i_2} \boldsymbol{g}^T d\ell_i \hat{\boldsymbol{s}}_i.$$

(8.118)

To express $dP_i$ in terms of $\boldsymbol{x}_i$, we need to find the differential expressions $d\hat{\boldsymbol{s}}_i$ and $d\ell_i \hat{\boldsymbol{s}}_i$ in terms of $d\boldsymbol{x}_i$. These relationships are derived as follows:

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{a}_i + \ell_i \hat{\boldsymbol{s}}_i \\ d\boldsymbol{x}_i &= \ell_i d\hat{\boldsymbol{s}}_i + d\ell_i \hat{\boldsymbol{s}}_i. \end{aligned}$$

(8.119)

In this context, $\boldsymbol{a}_i$ is the base position of the limb. Rearranging the above equation allows us to express the differential of the unit vector $\hat{\boldsymbol{s}}_i$:

$$d\hat{\boldsymbol{s}}_i = \frac{1}{\ell_i}(d\boldsymbol{x}_i - d\ell_i \hat{\boldsymbol{s}}_i).$$

(8.120)

To eliminate $d\hat{\boldsymbol{s}}_i$, dot multiply Equation 8.119 by $\hat{\boldsymbol{s}}_i$ and note that $\hat{\boldsymbol{s}}_i \cdot d\hat{\boldsymbol{s}}_i = 0$, which simplifies the calculation: $d\hat{\boldsymbol{s}}_i$, and note that $\hat{\boldsymbol{s}}_i \cdot d\hat{\boldsymbol{s}}_i = 0$:

$$\hat{\boldsymbol{s}}_i \cdot d\boldsymbol{x}_i = d\ell_i \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{s}}_i = d\ell_i.$$

(8.121)

From this, we obtain:

$$d\ell_i \hat{\boldsymbol{s}}_i = (\hat{\boldsymbol{s}}_i \cdot d\boldsymbol{x}_i)\hat{\boldsymbol{s}}_i = \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T d\boldsymbol{x}_i.$$

(8.122)

To eliminate $d\ell_i$, cross multiply Equation 8.119 by $\hat{\boldsymbol{s}}_i$:

$$\hat{\boldsymbol{s}}_i \times d\boldsymbol{x}_i = \ell_i(\hat{\boldsymbol{s}}_i \times d\hat{\boldsymbol{s}}_i).$$

(8.123)

Cross multiplying both sides by $\hat{\boldsymbol{s}}_i$ again, we have:

$$\begin{aligned}\hat{\boldsymbol{s}}_i \times (\hat{\boldsymbol{s}}_i \times d\boldsymbol{x}_i) &= \ell_i \hat{\boldsymbol{s}}_i \times (\hat{\boldsymbol{s}}_i \times d\hat{\boldsymbol{s}}_i) \\ &= \ell_i \left(\hat{\boldsymbol{s}}_i(\hat{\boldsymbol{s}}_i \cdot d\hat{\boldsymbol{s}}_i) - d\hat{\boldsymbol{s}}_i(\hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{s}}_i)\right) = -\ell_i d\hat{\boldsymbol{s}}_i.\end{aligned}$$

(8.124)

Thus, we find:

$$d\hat{\boldsymbol{s}}_i = -\frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2 d\boldsymbol{x}_i.$$

(8.125)

Substituting the expressions from Equations 8.125 and 8.122 into Equation 8.118, and simplifying, we obtain:

$$\begin{aligned}dP_i &= \frac{1}{\ell_i}(m_{i_1}c_{i_1} + m_{i_2}(\ell_i - c_{i_2}))\boldsymbol{g}^T \hat{\boldsymbol{s}}_{i_\times}^2 d\boldsymbol{x}_i - m_{i_2}\boldsymbol{g}^T \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T d\boldsymbol{x}_i \\ &= \boldsymbol{g}^T \left(m_{g_e}\hat{\boldsymbol{s}}_{i_\times}^2 - m_{i_2}\hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T\right) d\boldsymbol{x}_i\end{aligned}$$

(8.126)

where $m_{g_e}$ is defined in Equation 7.117. This result provides the differential of the potential energy in terms of the generalized coordinates.

To find the partial derivative of potential energy $P$ with respect to $\boldsymbol{x}_i$, we need to transpose Equation 8.126 to obtain the correct dimensionality. The potential energy $P$ itself is a scalar, but the partial derivative $\frac{\partial P_i}{\partial \boldsymbol{x}_i}$, which equals the gravity vector $\boldsymbol{\mathcal{G}}_i$, must be an $n \times 1$ column vector. Thus:

$$\begin{aligned}
\frac{\partial P_i}{\partial \boldsymbol{x}_i} = \boldsymbol{\mathcal{G}}_i \quad &= \left[\boldsymbol{g}^T \left(m_{g_e} \hat{\boldsymbol{s}}_{i_\times}^2 - m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T\right)\right]^T \\
&= \left(m_{g_e} \hat{\boldsymbol{s}}_{i_\times}^2 - m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T\right)^T \boldsymbol{g} \\
&= \left(m_{g_e} \left(\hat{\boldsymbol{s}}_{i_\times}^2\right)^T - m_{i_2} (\hat{\boldsymbol{s}}_i^T)^T \hat{\boldsymbol{s}}_i\right) \boldsymbol{g} \\
&= \left(m_{g_e} \hat{\boldsymbol{s}}_{i_\times}^2 - m_{i_2} \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i^T\right) \boldsymbol{g}.
\end{aligned}$$

(8.127)

This derived expression for the gravity vector $\boldsymbol{\mathcal{G}}_i$ is consistent with the result obtained using the Newton–Euler method, as described in Equation 7.116. This confirms the accuracy and validity of the derived formulation.

The final component to be evaluated is the Coriolis and centrifugal vector, denoted as $\boldsymbol{\mathcal{V}}_i = \boldsymbol{\mathcal{C}}_i \dot{\boldsymbol{x}}_i$. To determine this vector, we can either directly compute it from the time

derivative of the Lagrangian function, according to the Lagrange equation 8.15, or alternatively, derive it using relations involving the time derivative of the mass matrix and symmetric matrices discussed previously.

In this approach, we will use the time derivative of the Lagrangian function to calculate the Coriolis and centrifugal vector $\boldsymbol{\nu}_i$. This can be expressed using the general formulation provided in Equation 8.25. For a limb-generalized coordinate, this formulation is rewritten as follows:

$$\boldsymbol{\nu}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) = \dot{\boldsymbol{M}}_i(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T \boldsymbol{M}_i(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i\right). \tag{8.128}$$

In this equation, it is essential to calculate the time derivative of the mass matrix $\dot{\boldsymbol{M}}_i(\boldsymbol{x}_i)$, as well as the derivative of the kinetic energy with respect to the generalized coordinate $\boldsymbol{x}_i$. The kinetic energy is given by the term $\dot{\boldsymbol{x}}_i^T \boldsymbol{M}_i(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i$, and its derivative involves differentiating with respect to $\boldsymbol{x}_i$. To perform these calculations, we need to express the derivatives of various intermediate variables as functions of the generalized coordinate $\boldsymbol{x}_i$.

We start by deriving the required formulations and simplifying Equation 8.128. To do so, we first revisit the mass matrix $\boldsymbol{M}_i$ derived in Equation 8.116. Let's rewrite

this mass matrix in a more convenient form to facilitate further calculations:

$$\begin{aligned}\boldsymbol{M}_i &= m_{i_2}\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T - m_{c_e}\hat{\boldsymbol{s}}_{i_\times}^2 - \frac{1}{\ell_i^2}I_{xx_i}\hat{\boldsymbol{s}}_{i_\times}^2 \\ &= m_{i_2}(\boldsymbol{I}_{3\times 3} + \hat{\boldsymbol{s}}_{i_\times}^2) - m_{c_e}\hat{\boldsymbol{s}}_{i_\times}^2 - \frac{1}{\ell_i^2}I_{xx_i}\hat{\boldsymbol{s}}_{i_\times}^2 \\ &= \left(m_{i_2} - m_{c_e} - \frac{1}{\ell_i^2}I_{xx_i}\right)\hat{\boldsymbol{s}}_{i_\times}^2 + m_{i_2}\boldsymbol{I}_{3\times 3}.\end{aligned} \tag{8.129}$$

To simplify the second term of Equation 8.128, we first identify and use the following relations:

$$\frac{\partial \ell_i}{\partial \boldsymbol{x}_i} = \hat{\boldsymbol{s}}_i; \quad \frac{\partial}{\partial \boldsymbol{x}_i}\left(\frac{1}{\ell_i^2}\right) = -\frac{2}{\ell_i^3}\hat{\boldsymbol{s}}_i; \quad \frac{\partial \hat{\boldsymbol{s}}_i}{\partial \boldsymbol{x}_i} = -\frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2. \tag{8.130}$$

These expressions represent the partial derivatives of various terms with respect to the generalized coordinate $\boldsymbol{x}_i$ and will be essential for simplifying the calculations.

Additionally, we compute the derivative of the term $m_{c_e}$ with respect to $\boldsymbol{x}_i$ as follows:

$$
\begin{aligned}
\dot{m}_{c_e} &= \frac{\partial}{\partial \boldsymbol{x}_i}\left(\frac{1}{\ell_i^2}(m_{i_1}c_{i_1}^2 + m_{i_2}c_{i_2}^2)\right)\dot{\boldsymbol{x}}_i \\
&= \frac{1}{\ell_i^4}\left(-2\ell_i(m_{i_1}c_{i_1}^2 + m_{i_2}c_{i_2}^2)\hat{\boldsymbol{s}}_i + 2m_{i_2}c_{i_2}\ell_i^2\hat{\boldsymbol{s}}_i\right)\dot{\boldsymbol{x}}_i \\
&= \left(-\frac{2}{\ell_i}m_{c_e}\hat{\boldsymbol{s}}_i + \frac{2}{\ell_i^2}m_{i_2}c_{i_2}\hat{\boldsymbol{s}}_i\right)\dot{\boldsymbol{x}}_i,
\end{aligned}
\tag{8.131}
$$

Using these results, we derive additional necessary equations:

$$
\begin{aligned}
\frac{d}{dt}\left(\hat{\boldsymbol{s}}_{i_\times}\right) &= -\frac{1}{\ell_i}\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times} - \frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_i^T, \\
\frac{d}{dt}\left(\hat{\boldsymbol{s}}_{i_\times}^2\right) &= -\frac{1}{\ell_i}\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2 - \frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_i^T, \\
\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i\right) &= -\frac{2}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i\hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i, \\
\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i\right) &= -\frac{2}{\ell_i}\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i.
\end{aligned}
\tag{8.132}
$$

To simplify the first term of the Coriolis and centrifugal vector in Equation 8.128, we use these equations. The first term can be expressed as follows:

$$
\begin{aligned}
\dot{\boldsymbol{M}}_i\dot{\boldsymbol{x}}_i &= \frac{d}{dt}\left((m_{i_2} - m_{c_e} - \frac{1}{\ell_i^2}I_{xx_i})\hat{\boldsymbol{s}}_{i_\times}^2 + m_{i_2}\boldsymbol{I}_{3\times 3}\right)\dot{\boldsymbol{x}}_i \\
&= -\frac{1}{\ell_i}(m_{i_2} - m_{c_e} - \frac{1}{\ell_i^2}I_{xx_i})(\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2 + \hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i\hat{\boldsymbol{s}}_i^T)\dot{\boldsymbol{x}}_i \\
&\quad + \left(\left(\frac{2}{\ell_i}m_{c_e}\hat{\boldsymbol{s}}_i - \frac{2}{\ell_i^2}m_{i_2}c_{i_2}\hat{\boldsymbol{s}}_i\right)\dot{\ell}_i + \frac{2}{\ell_i^3}\dot{\ell}_i I_{xx_i}\right)\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i \\
&= -\frac{1}{\ell_i}(m_{i_2} - m_{c_e} - \frac{1}{\ell_i^2}I_{xx_i})(\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i + \hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i\hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i \\
&\quad + \frac{2}{\ell_i}\left(m_{i_2}c_{i_2} - \frac{1}{\ell_i^2}(I_{xx_i} + \ell_i^2 m_{c_e})\right)\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i.
\end{aligned}
\tag{8.133}
$$

To further simplify Equation 8.133, observe that:

$$
\dot{\ell}_i = \frac{\partial \ell_i}{\partial \boldsymbol{x}_i}\cdot\dot{\boldsymbol{x}}_i = \hat{\boldsymbol{s}}_i\cdot\dot{\boldsymbol{x}}_i = \hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i.
\tag{8.134}
$$

This allows us to simplify the term $\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i\hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i$ to $\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i$. Substituting this result into Equation 8.133 and simplifying further yields:

$$\begin{aligned}\dot{\boldsymbol{M}}_i\dot{\boldsymbol{x}}_i &= \frac{1}{\ell_i}(m_{c_e} - m_{i_2} + \frac{1}{\ell_i^2}I_{xx_i})\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i \\ &\quad + \frac{1}{\ell_i}\left(3m_{c_e} - m_{i_2} - \frac{2}{\ell_i}m_{i_2}c_{i_2} + \frac{3}{\ell_i^2}I_{xx_i}\right)\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i.\end{aligned} \tag{8.135}$$

The second term of the Coriolis and centrifugal vector, as derived in Equation 8.128, can be simplified by expanding and differentiating it as follows:

$$\begin{aligned}-\frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}_i}\left(\dot{\boldsymbol{x}}_i^T\boldsymbol{M}_i(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i\right) &= -\frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}_i}\left[\dot{\boldsymbol{x}}_i^T\left(m_{i_2}\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T - m_{c_e}\hat{\boldsymbol{s}}_{i_\times}^2\right.\right. \\ &= \frac{1}{\ell_i}\left(\frac{1}{\ell_i}m_{i_2}(\ell_i - c_{i_2}) - \frac{1}{\ell_i}m_{c_e} - \right. \\ &\quad + \frac{1}{\ell_i}\left(m_{i_2} - m_{c_e} - \frac{1}{\ell_i^2}I_{xx_i}\right)\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\end{aligned} \tag{8.136}$$

Combining this result with the previously simplified first term from Equation 8.135, we obtain the full expression for the Coriolis and centrifugal vector:

$$
\begin{aligned}
\boldsymbol{\mathcal{V}}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) &= \frac{1}{\ell_i}\left(\frac{1}{\ell_i}(\ell_i - c_{i_2})m_{i_2} - m_{i_2}\right)\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i \\
&\quad + \frac{2}{\ell_i}\left(m_{c_e} - \frac{1}{\ell_i}m_{i_2}c_{i_2} + \frac{1}{\ell_i^2}I_{xx_i}\right)\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i \\
&= \left(-\frac{1}{\ell_i^2}m_{i_2}c_{i_2}\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i - \frac{2}{\ell_i}m_{c_o}\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2\right)\dot{\boldsymbol{x}}_i,
\end{aligned}
\tag{8.137}
$$

where $m_{c_o}$ is defined in Equation 7.117. Therefore,

$$
\boldsymbol{\mathcal{C}}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i) = -\frac{1}{\ell_i^2}m_{i_2}c_{i_2}\hat{\boldsymbol{s}}_i\dot{\boldsymbol{x}}_i^T\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i - \frac{2}{\ell_i}m_{c_o}\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}^2,
\tag{8.138}
$$

which is consistent with the result obtained using the Newton-Euler method, as detailed in Equation 7.116. This demonstrates the coherence between the Lagrangian and Newton-Euler formulations in calculating the Christoffel matrix.

### 8.5.2 Dynamic Formulation of the Moving Platform

In this section, we derive the dynamic formulation for the moving platform, including the mass matrix $\boldsymbol{M}_p$, the Christoffel matrix $\boldsymbol{\mathcal{C}}_p$, and the gravity vector $\boldsymbol{\mathcal{G}}_p$. We start

by examining the mass matrix and then derive the kinetic energy of the moving platform. The kinetic energy of the moving platform is given by:

$$K_p = \frac{1}{2}\boldsymbol{v}_p^T m \boldsymbol{v}_p + \frac{1}{2}\boldsymbol{\omega}^T \boldsymbol{I}_p \boldsymbol{\omega},$$

(8.139)

where $\boldsymbol{v}_p$ represents the linear velocity of the center of mass of the moving platform, $\boldsymbol{\omega}$ denotes the angular velocity of the moving platform, and $m$ and $\boldsymbol{I}_p$ are the mass and inertia matrix of the moving platform, respectively. We can express this equation in matrix form using the derivative of the manipulator generalized coordinate $\dot{\boldsymbol{\chi}} = [\boldsymbol{v}_p, \boldsymbol{\omega}]^T$. This results in:

$$K_p = \frac{1}{2}\dot{\boldsymbol{\chi}}^T \begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_p \end{bmatrix} \dot{\boldsymbol{\chi}},$$

(8.140)

where $\boldsymbol{I}_{3\times3}$ is the $3 \times 3$ identity matrix and $\boldsymbol{0}_{3\times3}$ is the $3 \times 3$ zero matrix. Consequently, the mass matrix for the moving platform is given by:

$$\begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_p \end{bmatrix}.$$

(8.141)

Comparing this equation with Equation 7.120, it is evident that the formulation presented here matches the one obtained using the Newton–Euler method.

Deriving the Coriolis and centrifugal vector $\boldsymbol{\nu}_p(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is notably more challenging than deriving the mass matrix. This complexity arises because it involves differentiating both the mass matrix with respect to time and the kinetic energy with respect to the generalized coordinate:

$$\boldsymbol{\nu}_e(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \dot{\boldsymbol{M}}_e \dot{\boldsymbol{\chi}} - \frac{\partial}{\partial \boldsymbol{\chi}} K_p. \tag{8.142}$$

The first term in this equation can be computed as follows:

$$\dot{\boldsymbol{M}}_p \dot{\boldsymbol{\chi}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{d}{dt}\boldsymbol{I}_p \end{bmatrix} \dot{\boldsymbol{\chi}}. \tag{8.143}$$

The time derivative of the moving platform inertia matrix $\dot{\boldsymbol{I}}_p$ needs to be evaluated in the reference frame $\{\mathscr{A}\}$. By applying Equation 7.100 and using the fact that $\dot{}^{A}\boldsymbol{R}_B = \boldsymbol{\omega}_{\times}\, {}^{A}\boldsymbol{R}_B$, the time derivative is calculated as:

$$
\begin{aligned}
\dot{\boldsymbol{I}}_p &= \frac{d}{dt}({}^A\boldsymbol{R}_B\,{}^B\boldsymbol{I}_p\,{}^A\boldsymbol{R}_B^T) \\
&= .{}^A\boldsymbol{R}_B\,{}^B\boldsymbol{I}_p\,{}^A\boldsymbol{R}_B^T + {}^A\boldsymbol{R}_B\,{}^B\boldsymbol{I}_p.{}^A\boldsymbol{R}_B^T \\
&= \boldsymbol{\omega}_\times\,{}^A\boldsymbol{R}_B\,{}^B\boldsymbol{I}_p\,{}^A\boldsymbol{R}_B^T + {}^A\boldsymbol{R}_B\,{}^B\boldsymbol{I}_p\,{}^A\boldsymbol{R}_B^T\boldsymbol{\omega}_\times^T \\
&= \boldsymbol{\omega}_\times\boldsymbol{I}_p + \boldsymbol{I}_p\boldsymbol{\omega}_\times^T \\
&= \boldsymbol{\omega}_\times\boldsymbol{I}_p - \boldsymbol{I}_p\boldsymbol{\omega}_\times.
\end{aligned}
\tag{8.144}
$$

Thus:

$$
\dot{M}_p\dot{\boldsymbol{\chi}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{\omega}_\times\boldsymbol{I}_p - \boldsymbol{I}_p\boldsymbol{\omega}_\times \end{bmatrix}\dot{\boldsymbol{\chi}}.
\tag{8.145}
$$

Following a similar approach as previously described, the second term in Equation 8.142 can be derived as:

$$
\frac{\partial}{\partial\boldsymbol{\chi}}K_p = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -\boldsymbol{I}_p\boldsymbol{\omega}_\times \end{bmatrix}\dot{\boldsymbol{\chi}}.
\tag{8.146}
$$

In this context, $\frac{\partial}{\partial\boldsymbol{\chi}}K_p$ represents the gradient of the kinetic energy with respect to the generalized coordinates $\boldsymbol{\chi}$. This term captures how changes in the generalized coordinates influence the kinetic energy of the system, specifically

through the angular velocity component $\boldsymbol{\omega}$ and the inertia matrix $\boldsymbol{I}_p$.

Thus, the Coriolis and centrifugal vector $\boldsymbol{\nu}_p(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is expressed as:

$$\boldsymbol{\nu}_p(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \boldsymbol{C}_p \dot{\boldsymbol{\chi}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{\omega}_\times \boldsymbol{I}_p \end{bmatrix} \dot{\boldsymbol{\chi}}. \tag{8.147}$$

Here, $\boldsymbol{C}_p$ represents the Christoffel matrix, which describes the forces arising due to the rotation and motion of the platform. The term $\boldsymbol{\omega}_\times \boldsymbol{I}_p$ in the matrix reflects the contribution of angular velocity to these forces, showing how the angular motion of the platform affects the overall dynamics.

Comparing this result with Equation 7.120 confirms that the derivation of $\boldsymbol{C}_p(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is consistent with the results obtained using the Newton–Euler method. This agreement verifies that the approach taken here aligns with established methods in dynamics analysis, ensuring that the calculations for the Coriolis and centrifugal forces are accurate and reliable.

The gravitational term for the moving platform can be determined through two primary methods: by computing the partial derivative of the moving platform's potential energy with respect to the generalized coordinate $\boldsymbol{\chi}$, or by utilizing the principle of virtual work. For this derivation, we will employ the potential energy approach.

The potential energy of the moving platform is expressed as:

$$P_p = -m\boldsymbol{g} \cdot \boldsymbol{x},$$

(8.148)

where $\boldsymbol{x}$ denotes the position vector of the moving platform, and $\boldsymbol{g}$ represents the gravitational acceleration vector. This formulation indicates that the potential energy is directly proportional to the height of the platform in the gravitational field.

Since the potential energy of the moving platform depends only on the position and not on the orientation component of the generalized coordinate $\boldsymbol{\chi}$, the derivative with respect to $\boldsymbol{\chi}$ focuses solely on the translational aspect. Therefore, the partial derivative of the potential energy with respect to $\boldsymbol{\chi}$ is given by:

$$\boldsymbol{\mathcal{G}}_p = \frac{\partial P_p}{\partial \boldsymbol{\chi}} = \begin{bmatrix} -m\boldsymbol{g} \\ \mathbf{0}_{3\times 1} \end{bmatrix}.$$

(8.149)

In this expression, the first term, $-m\boldsymbol{g}$, reflects the contribution of gravitational forces in the direction of the platform's position vector, while the second term, $\mathbf{0}_{3\times 1}$, represents the fact that the gravitational force does not influence the rotational components of the generalized coordinates.

By comparing this result with Equation 7.120, it is evident that the derivation of $\mathcal{G}_p$ aligns perfectly with the result obtained using the Newton–Euler method.

### 8.5.3 Dynamic Formulation of the Whole Manipulator

To finalize the dynamic formulation for the entire manipulator, it is essential to convert the intermediate generalized coordinates $\boldsymbol{x}_i$ into the principal generalized coordinates $\boldsymbol{\chi}$. This conversion is achieved through the Jacobian matrix $\boldsymbol{J}_i$, as detailed in Equation 7.122. This transformation process integrates the intermediate structural components into a cohesive framework that defines the manipulator's overall dynamic behavior. Following this, a procedure similar to that outlined in Section 7.5.4 should be applied to combine the structural matrices of both the moving platform and the limb. This process will yield the complete manipulator mass matrix $\boldsymbol{M}$, Coriolis and centrifugal matrix $\boldsymbol{\mathcal{C}}$, and gravity vector $\boldsymbol{\mathcal{G}}$, as specified in Equation 7.130.

At this stage, the only component left to determine is the generalized forces acting on the manipulator. In the Lagrangian approach, internal forces like $\boldsymbol{f}_{b_i}$ are not included in the dynamic formulation. Instead, the focus is on external forces such as the actuator forces $\boldsymbol{\tau}$ and the disturbance wrench $\boldsymbol{\mathcal{F}}_d$. To incorporate these into the dynamics, the principle of virtual work is utilized. This principle helps in directly calculating the resulting

generalized forces of the manipulator from these two components. Specifically, the projection of actuator forces on the center of mass of the moving platform can be expressed as $\boldsymbol{J}^T\boldsymbol{\tau}$, while the disturbance wrench is directly considered as part of the generalized forces.

Therefore, the generalized force vector $\boldsymbol{Q}$ is given by:

$$\boldsymbol{Q} = \boldsymbol{J}^T\boldsymbol{\tau} + \boldsymbol{\mathcal{F}}_d. \tag{8.150}$$

A thorough examination of the components of $\boldsymbol{\mathcal{F}}$, as detailed in Equation 7.130, reveals that the generalized forces obtained from Equation 8.150 are identical to those derived using the Newton–Euler method. This confirmation is critical as it validates the final component of the dynamic formulation. The agreement between the results of these two methods demonstrates that despite using different approaches—one focusing on potential energy and the other on Newton–Euler principles—the derived dynamic formulations for the Stewart–Gough platform are fully consistent and equivalent. This consistency reinforces the robustness of the model and its applicability in accurately representing the manipulator's dynamics across different analytical methods.

## Problems

8.1 Review the five-bar mechanism depicted in Figure 4.27 and analyzed in Problems 4.1 and 5.1. Apply the Euler-Lagrange method to resolve Problem 7.1 for this mechanism. Then, compare the dynamic formulation obtained through this method with the formulation derived using the Newton–Euler approach.

8.2 Examine again the five-bar mechanism illustrated in Figure 4.28 and outlined in Problems 4.2 and 5.2. Apply the Euler-Lagrange method to address Problem 7.2 for this mechanism. Additionally, contrast the dynamic formulation derived using this approach with the one obtained through the Newton–Euler method.

8.3 Review the planar pantograph mechanism depicted in Figure 4.29 and detailed in Problems 4.3 and 5.3. Utilize the Euler-Lagrange method to tackle Problem 7.3 for this mechanism. Additionally, evaluate the dynamic formulation derived from this method and compare it with the formulation achieved using the Newton–Euler approach.

8.4 Reevaluate the planar three-degrees-of-freedom 3RRR parallel manipulator illustrated in Figure 4.30 and discussed in Problems 4.4 and 5.4. Apply the Euler-Lagrange method to solve Problem 7.4 for this manipulator. Additionally, compare the dynamic formulation derived through this method with the formulation obtained using the Newton–Euler approach.

8.5 Reassess the planar redundant 4RRR parallel manipulator depicted in Figure 4.31 and detailed in Problems 4.5 and 5.5. Use the Euler-Lagrange method to address Problem 7.5 for this manipulator. Additionally, compare the dynamic formulation derived from this approach with the one obtained using the Newton–Euler method.

8.6 Reexamine the spatial parallel manipulator depicted in Figure 4.32 and outlined in Problems 4.6 and 5.6. Apply the Euler-Lagrange method to solve Problem 7.6 for this manipulator. Additionally, compare the dynamic formulation derived using this method with the formulation obtained through the Newton–Euler approach.

8.7 Revisit the 3UPU parallel manipulator illustrated in Figure 4.33 and detailed in Problems 4.7 and 5.7. Employ the Euler-Lagrange method to resolve Problem 7.7 for this manipulator. Additionally, compare the dynamic formulation derived through this method with the formulation obtained using the Newton–Euler approach.

8.8 Review the three-degrees-of-freedom 3UPS manipulator featured in [51], as shown in Figure 4.34 and described in Problems 4.8 and 5.8. Apply the Euler-Lagrange method to solve Problem 7.8 for this manipulator. Furthermore, compare the dynamic

formulation derived through this approach with the formulation obtained via the Newton–Euler method.

8.9 Revisit the three-degrees-of-freedom 3RPS manipulator illustrated in Figure 4.35 and detailed in Problems 4.9 and 5.9. Utilize the Euler-Lagrange method to address Problem 7.9 for this manipulator. Additionally, compare the dynamic formulation obtained through this method with the formulation derived using the Newton–Euler approach.

8.10 Reexamine the three-degrees-of-freedom 3PRC translational parallel manipulator depicted in Figure 4.35 and outlined in Problems 4.10 and 5.10. Apply the Euler-Lagrange method to solve Problem 7.10 for this manipulator. Additionally, compare the dynamic formulation derived using this approach with the formulation obtained through the Newton–Euler method.

8.11 Reassess the Delta robot illustrated in Figure 4.38 and detailed in Problems 4.11 and 5.11. Use the Euler-Lagrange method to solve Problem 7.11 for this robot. Additionally, compare the dynamic formulation derived through this approach with the one obtained using the Newton–Euler method.

8.12 Examine the Quadrupteron robot shown in Figure 4.39 and outlined in Problems 4.12 and 5.12. Apply the Euler-Lagrange method to solve Problem 7.12 for this robot. Additionally, compare the dynamic formulation

derived from this approach with the one obtained using the Newton–Euler method.

# 9

# *Dynamics Analysis: Virtual Work Approach*

DOI: 10.1201/9781003590408-9

This chapter focuses on the dynamics of parallel manipulators using the virtual work approach, emphasizing its efficiency in deriving equations of motion. It begins with a detailed introduction to D’Alembert's principle and the principle of virtual work, establishing a foundation for the dynamic formulations presented later. Closed-form formulations are then introduced, demonstrating their importance in simplifying and organizing dynamic analysis for parallel manipulators. The chapter uses several case studies to illustrate the practical implementation of the virtual work approach.

The first case study analyzes a planar cable robot, exploring implicit dynamics and verifying the derived formulations. The second case study investigates the dynamics of the Stewart–Gough platform, a widely used parallel manipulator in precision robotics. The final case

study examines the 3UPU manipulator, presenting a thorough analysis of its structure, kinematics, Jacobian characteristics, acceleration, and dynamics. The case studies highlight the method's applicability to a variety of manipulator configurations.

The chapter concludes with a collection of problems designed to strengthen readers’ understanding of the virtual work approach and its use in dynamic analysis. These exercises encourage the application of theoretical knowledge to practical challenges, fostering a deeper understanding of robotic system dynamics.

## 9.1 Introduction

One of the most practical methods for deriving the dynamic formulation of a parallel manipulator is based on the principle of virtual work. This method, which leverages d’Alembert's principle, provides an alternative approach frequently discussed in robotics literature. This approach focuses on analyzing the acceleration of the centers of mass of the manipulator's links. To obtain these accelerations, you can differentiate the velocity of the center of mass directly. However, a more efficient technique involves using the derivatives of the Jacobian matrices associated with these points, which helps to reduce the computational complexity involved.

In this method, the process begins by calculating the inertial forces and moments for each rigid body within the manipulator, based on their linear and angular accelerations. These inertial forces are then used to establish the manipulator's static equilibrium. By treating the system as if it is in static equilibrium, the principle of virtual work is applied to compute the necessary actuator forces [181]. This principle essentially states that the work done by the actuator forces, in virtual displacements, must balance the virtual work done by the inertial forces and moments.

A significant advantage of the virtual work method is that it does not require the calculation of internal forces, which simplifies the process and enhances computational efficiency. In contrast, the Newton-Euler method necessitates determining these internal forces, which can be more computationally demanding. However, while the virtual work method is computationally more efficient, it demands a thorough understanding of classical mechanics to apply effectively, which may be more challenging compared to the Euler-Lagrange method, which is often considered more intuitive for dynamic analysis.

## 9.2 D’Alembert's Principle

The Newton-Euler laws are fundamentally based on the principle of balancing the summation of external forces and

torques acting on a rigid body with the inertial terms that arise from the body's linear or angular accelerations. As elaborated earlier, to correctly apply force and torque balance, it is essential to derive accelerations with respect to a fixed reference frame.

When the motion of the rigid body is observed from a moving reference frame attached to its center of mass, the body appears to be at rest or in a state of equilibrium. In such a scenario, an observer who is accelerating with the moving frame might conclude that a *fictitious* inertial force and moment are acting on the rigid body to counterbalance the *real* external forces and moments. This interpretation allows the dynamic problem to be approached using the methods of statics, representing an extension of the work of d'Alembert as presented in his seminal book *Traité de Dynamique*, published in 1743 [208].

D'Alembert's principle transforms the Newton and Euler laws into an equation where the summation of all real external forces and moments, along with the corresponding fictitious inertial forces and moments, equals zero. This transformation simplifies the analysis of the motion of the rigid body by introducing a fictitious force that accounts for the acceleration of the reference frame, thereby allowing the use of static equilibrium equations in a dynamic context. The mathematical representation of this principle is given by:

$$\sum \boldsymbol{f}_{\text{ext}} - m\boldsymbol{a}_c = \mathbf{0}$$

(9.1)

$$\sum {}^{c}\boldsymbol{n}_{\text{ext}} - ({}^{c}\boldsymbol{I}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times ({}^{c}\boldsymbol{I}\,\boldsymbol{\omega})) = \mathbf{0}.$$

(9.2)

In these equations, $\sum \boldsymbol{f}_{ext}$ represents the sum of all external forces, $m$ is the mass of the rigid body, $\boldsymbol{a}_c$ denotes the linear acceleration of the center of mass, $\boldsymbol{\omega}$ represents the angular velocity of the rigid body, $\dot{\boldsymbol{\omega}}$ is the angular acceleration, $\sum {}^{c}\boldsymbol{n}_{ext}$ is the sum of all external torques, and ${}^{c}\boldsymbol{I}$ is the moment of inertia of the rigid body relative to the moving frame attached to the center of mass. Within this framework, the inertia term $-m\boldsymbol{a}_c$ can be viewed as a fictitious force $\boldsymbol{f}^* = -m\boldsymbol{a}_c$. This fictitious force represents the resistance of the rigid body to changes in its linear momentum as perceived from the accelerating reference frame.

Similarly, the other inertial terms, ${}^{c}\boldsymbol{n}^* = -({}^{c}\boldsymbol{I}, \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times ({}^{c}\boldsymbol{I}, \boldsymbol{\omega}))$, can be considered as a fictitious torque. This fictitious torque accounts for the resistance to changes in the body's angular momentum, effectively balancing out the real torques in the accelerating frame. By introducing these fictitious forces and torques, we can simplify the dynamic analysis of the rigid body by converting it into a problem of static equilibrium.

$$\sum \hat{\boldsymbol{f}}_{\text{ext}} = \mathbf{0}$$

(9.3)

$$\sum {}^{c}\hat{\boldsymbol{n}}_{\text{ext}} = \mathbf{0},$$

(9.4)

where

$$\hat{\boldsymbol{f}}_{\text{ext}} = \boldsymbol{f}_{\text{ext}} + \boldsymbol{f}^{*}$$

(9.5)

$${}^{c}\hat{\boldsymbol{n}}_{\text{ext}} = {}^{c}\boldsymbol{n}_{\text{ext}} + {}^{c}\boldsymbol{n}^{*}.$$

(9.6)

In this context, $\hat{\boldsymbol{f}}_{\text{ext}}$ represents the total effective force acting on the rigid body, which includes both the real external forces $\boldsymbol{f}_{\text{ext}}$ and the fictitious force $\boldsymbol{f}^{*}$. Similarly, ${}^{c}\hat{\boldsymbol{n}}_{ext}$ represents the total effective torque, combining the real external torques ${}^{c}\boldsymbol{n}_{\text{ext}}$ and the fictitious torque ${}^{c}\boldsymbol{n}^{*}$. This approach allows the problem of analyzing the dynamics of the rigid body in a non-inertial frame to be handled using the simpler techniques of static equilibrium analysis, thus providing a powerful method for solving complex dynamic problems in mechanics.

The apparent conversion of a dynamic problem into a static one is known as the *d’Alembert principle*. The basic definitions provided above might appear to offer a straightforward way of transforming a dynamic equation into its static counterpart, and it might seem that this approach doesn’t significantly simplify the equations of motion. However, in practical applications, treating inertial terms as fictitious forces and moments greatly simplifies the derivation of dynamic formulations, especially when the principle of virtual work is applied to the problem.

This transformation allows for a more intuitive understanding of the forces and moments acting on a system, making it easier to visualize and analyze complex interactions. Furthermore, this approach facilitates the use of established static analysis techniques, such as those used in structural mechanics and material science, to solve dynamic problems. This can lead to more efficient computational models and simulations, as well as providing a clearer conceptual framework for understanding the behavior of dynamic systems.

## 9.3 Principle of Virtual Work

In this section, we demonstrate how the principle of virtual work can be utilized to derive the dynamic equation of motion for a parallel robot. The implementation of d’Alembert's principle in this context transforms the

dynamic system into a statically balanced one. According to d’Alembert's principle, the inertial force and moment acting on a rigid body are defined as the forces and moments exerted at the center of mass, given by:

$$\boldsymbol{f}^* = -m\boldsymbol{a}_c$$

(9.7)

$${}^c\boldsymbol{n}^* = -\left({}^c\boldsymbol{I}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \left({}^c\boldsymbol{I}\,\boldsymbol{\omega}\right)\right).$$

(9.8)

These forces and moments act in the direction opposite to the linear and angular accelerations. Thus, by introducing these fictitious forces and moments into the system, we can consider the rigid body to be in static equilibrium. When the system is in static equilibrium and a virtual displacement $\delta(\cdot)$ is introduced, the principle of virtual work can be used to determine the input forces of the manipulator.

The principle of virtual work has its roots in the work-energy relation, which states that for a system with negligible energy loss due to friction, the relationship between work and energy is given by:

$$dW = dK + dP,$$

(9.9)

where $dW$ represents the total work done by all *active* forces acting on the system during an infinitesimal displacement change, and $dK$ and $dP$ denote the changes in the kinetic and potential energies of the system, respectively.

For a general multibody interconnected system, the infinitesimal changes in kinetic and potential energies can be represented as:

$$dK = \sum_i m_i \boldsymbol{a}_{c_i} \cdot d\boldsymbol{x}_i + \sum_i {}^{c_i}\dot{\boldsymbol{H}}_i \cdot d\boldsymbol{\theta}_i \tag{9.10}$$

$$dP = \sum_i W_i \, dh_i + \sum_i k_i \, x_i \, dx_i \tag{9.11}$$

In this formulation, each body within the system has a mass $m_i$ and an angular momentum ${}^{c_i}\boldsymbol{H}_i$. The terms $d\boldsymbol{x}_i$ and $d\boldsymbol{\theta}_i$ represent the infinitesimal linear and angular displacements of each body's center of mass, respectively. Additionally, $h_i$ signifies the vertical motion of the center of mass of representative bodies with weight $W_i$, and $x_i$ denotes the elastic deformation of a representative elastic member (such as a spring) with stiffness $k_i$.

When applying Equation 9.9 to mechanical systems, particularly those where the configuration is an unknown

function of a prescribed motion and for systems with multiple degrees of freedom, it is often useful to introduce the concept of *virtual work*. This approach simplifies the analysis of such systems by providing a framework for dealing with the complex interactions between forces, moments, and displacements.

The notion of virtual displacement and the associated virtual work were initially developed to solve equilibrium configurations for static systems of interconnected bodies. These concepts were then extended to dynamic systems using the d'Alembert principle, which allows the dynamic analysis of systems to be treated similarly to static equilibrium problems.

A virtual displacement $\delta(\cdot)$ refers to any assumed, arbitrary displacement, whether linear or angular, from the system's current or natural position. The term "virtual" is used to distinguish it from actual motion. During actual motion, the geometric structure of the robot, the magnitudes of forces and torques applied to the environment, and the torques exerted by the actuators change. However, in virtual displacement, it is assumed that the geometry of the systems of interconnected bodies and the distribution of forces remain unchanged despite this infinitesimal movement. Therefore, these virtual displacements align with the system's kinematic constraints. By applying a set of virtual displacements that adhere to these constraints, one can derive the correct relationship between the coordinates that define the

system's configuration to that of the final moving platform motion, using the work-energy relation 9.9.

In this context, the differential symbol $d$ is used to denote real, infinitesimal changes in displacements, while $\delta$ is used for infinitesimal, assumed, or virtual changes. When applying work-energy principles to a system of rigid bodies, it is crucial to construct an active-force diagram to isolate the system and identify all forces that do work on it.

Consider first a static system of interconnected bodies. In static equilibrium, the sum of forces and moments applied to the system is zero. Consequently, the virtual work done on the system is also zero:

$$\delta W = \sum_i \boldsymbol{f}_i \cdot \delta \boldsymbol{x}_i + \sum_i \boldsymbol{n}_i \cdot \delta \boldsymbol{\theta}_i = 0. \tag{9.12}$$

Here, $\boldsymbol{f}_i$ represents the external forces, such as gravitational, spring, damping, or any other disturbance forces, and $\boldsymbol{n}_i$ denotes the external moments applied to the system. It is important to note that the constraint forces and moments, which ensure that the virtual displacements are consistent with the system's kinematic constraints, do not contribute to the work because they are orthogonal to the direction of the virtual displacement.

To facilitate calculations and simplify notation, the representation of a wrench is employed. As defined before, this allows the generalized forces and moments to be

combined into a single $6 \times 1$ array $\boldsymbol{\mathcal{F}}_i = \begin{bmatrix} \boldsymbol{f}_i & \boldsymbol{n}_i \end{bmatrix}^T$, and the linear and angular displacements into a twist notation $\boldsymbol{\chi}_i = \begin{bmatrix} \boldsymbol{x}_i & \boldsymbol{\theta}_i \end{bmatrix}^T$. Thus, the expression for virtual work can be compactly written as:

$$\delta W = \sum_i \boldsymbol{\mathcal{F}}_i \cdot \delta \boldsymbol{\chi}_i = 0.$$

(9.13)

Similarly, in the analysis of dynamic systems, we employ the d'Alembert principle by treating the inertial wrench as if it were an external wrench applied to the system. This approach is formulated in the following equation:

$$\boldsymbol{\mathcal{F}}^* = \begin{bmatrix} \boldsymbol{f}^* \\ {}^c\boldsymbol{n}^* \end{bmatrix} = \begin{bmatrix} -m\boldsymbol{a}_c \\ -\left({}^c\boldsymbol{I}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \left({}^c\boldsymbol{I}\,\boldsymbol{\omega}\right)\right) \end{bmatrix},$$

(9.14)

Here, $\boldsymbol{\mathcal{F}}^*$ represents the inertial wrench, which is derived by augmentation of $\boldsymbol{f}^*$, the inertial force, and ${}^c\boldsymbol{n}^*$, the inertial moment. By incorporating these inertial effects, the principle of virtual work can be applied to analyze the system:

$$\delta W = \sum_i (\boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_i^*) \cdot \delta \boldsymbol{\chi}_i = 0.$$

(9.15)

In this equation, $\boldsymbol{\mathcal{F}}_i$ and $\boldsymbol{\mathcal{F}}_i^*$ denote the real external and inertial wrenches acting on the system. To isolate the actuator forces, denoted by $\boldsymbol{\tau}$, from the other forces and moments, the principle of virtual work is modified as follows:

$$\delta W = \boldsymbol{\tau} \cdot \delta \boldsymbol{q} + \sum_i (\boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_i^*) \cdot \delta \boldsymbol{\mathcal{X}}_i = 0, \tag{9.16}$$

where $\boldsymbol{\mathcal{F}}_i$ includes all external forces (excluding actuator forces) and $\boldsymbol{\tau}$ represents the forces applied by the actuators.

Applying the principle of virtual work to a parallel robot with multiple limbs and a moving platform, we first isolate the external and inertial wrenches affecting both the limbs and the moving platform. The resulting equation is:

$$\delta W = \boldsymbol{\tau} \cdot \delta \boldsymbol{q} + (\boldsymbol{\mathcal{F}}_p + \boldsymbol{\mathcal{F}}_p^*) \cdot \delta \boldsymbol{\mathcal{X}}_p + \sum_i (\boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_i^*) \cdot \delta \boldsymbol{\mathcal{X}}_i = 0, \tag{9.17}$$

where $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_m \end{bmatrix}^T$ represents the vector of active joint variables, and $\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \dots & \tau_m \end{bmatrix}^T$ denotes the vector of actuator force/torques. The terms $\boldsymbol{\mathcal{F}}_p$ and $\boldsymbol{\mathcal{F}}_p^*$ refer to the external and inertial wrenches applied to the center of mass of the moving platform, respectively, while $\boldsymbol{\mathcal{F}}_i$ and $\boldsymbol{\mathcal{F}}_i^*$ are the external and inertial wrenches applied

to the centers of mass of the individual limbs. Additionally, $\boldsymbol{\chi}_p$ and $\boldsymbol{\chi}_i$ represent the virtual displacements of the moving platform and limb centers of mass, respectively.

To streamline notation and facilitate calculations, we use d'Alembert's shorthand notation as defined in Equation 9.5. This involves defining $\hat{\boldsymbol{\mathcal{F}}}_p = \boldsymbol{\mathcal{F}}_p + \boldsymbol{\mathcal{F}}_p^*$ for the moving platform, and $\hat{\boldsymbol{\mathcal{F}}}_i = \boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_i^*$, for the limbs. The virtual work Equation 9.17 can then be expressed in a compact matrix form as:

$$\delta \boldsymbol{q}^T \boldsymbol{\tau} + \delta \boldsymbol{\chi}_p^T \hat{\boldsymbol{\mathcal{F}}}_p + \sum_i \delta \boldsymbol{\chi}_i^T \hat{\boldsymbol{\mathcal{F}}}_i = 0. \tag{9.18}$$

This matrix form not only simplifies the calculations but also provides a clear framework for understanding the interplay between actuator forces, external forces, and inertial effects in a dynamic system. As previously discussed, the virtual displacements in Equation 9.17 must adhere to the kinematic constraints set by the closed-loop chains in the system. Therefore, it is crucial to express these virtual displacements in terms of a set of independent generalized virtual displacements.

In parallel manipulators, the generalized coordinate can conveniently be chosen as the coordinate of the moving platform, $\delta \boldsymbol{\chi}_p$. This is because the virtual displacement of the limbs, $\delta \boldsymbol{q}$, is typically related to the virtual displacement

of the moving platform, $\delta \boldsymbol{\chi}_p$, via the manipulator Jacobian matrix $\boldsymbol{J}$:

$$\delta \boldsymbol{q} = \boldsymbol{J}\, \delta \boldsymbol{\chi}_p.$$

(9.19)

Additionally, the virtual displacement of the center of mass for each limb, $\delta \boldsymbol{\chi}_i$, can be related to $\delta \boldsymbol{\chi}_p$ through a corresponding Jacobian matrix specific to each limb, denoted as $\boldsymbol{J}_i$:

$$\delta \boldsymbol{\chi}_i = \boldsymbol{J}_i\, \delta \boldsymbol{\chi}_p.$$

(9.20)

By substituting Equations 9.19 and 9.20 into Equation 9.17, we obtain:

$$\delta \boldsymbol{\chi}_p^T \left( \boldsymbol{J}^T \boldsymbol{\tau} + \hat{\boldsymbol{\mathcal{F}}}_p + \sum_i \boldsymbol{J}_i^T \hat{\boldsymbol{\mathcal{F}}}_i \right) = 0.$$

(9.21)

Since this equality must hold for any arbitrary virtual displacement $\delta \boldsymbol{\chi}_p$, it follows that:

$$\boldsymbol{J}^T \boldsymbol{\tau} + \hat{\boldsymbol{\mathcal{F}}}_p + \sum_i \boldsymbol{J}_i^T \hat{\boldsymbol{\mathcal{F}}}_i = \boldsymbol{0}.$$

(9.22)

Equation 9.22 provides a general dynamic formulation for a parallel manipulator using the principle of virtual work. In this formulation, all wrenches are considered relative to the center of mass of each link and the moving platform. This consideration is crucial because if an external wrench is applied at a point other than the center of mass, it must first be transformed into the coordinate frame of the center of mass before it can be substituted into this equation.

Moreover, this equation is expressed in task space coordinates, which simplifies the interpretation of the results in terms of the manipulator's moving platform. The actuator forces appear in these equations through the Jacobian transpose mapping. Specifically, $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$ represents the projection of the actuator forces onto the moving platform. From Equation 9.22, this projection can be uniquely determined.

In the context of a fully parallel manipulator, where the Jacobian matrix $\boldsymbol{J}$ is square and nonsingular, the actuator forces can be uniquely determined by the equation $\boldsymbol{\tau} = \boldsymbol{J}^{-T} \boldsymbol{\mathcal{F}}$. This is due to the fact that, in such systems, the number of actuators matches the number of degrees of freedom, ensuring a one-to-one correspondence between actuator forces and the resulting wrench on the moving platform.

However, in redundant manipulators, where the number of actuators exceeds the number of degrees of freedom, there are infinitely many possible solutions for $\boldsymbol{\tau}$ that can

result in the same projection $\mathcal{F}$. To resolve this redundancy, the minimum norm solution is commonly used. This solution is derived from the pseudo-inverse of $\boldsymbol{J}^T$ and is given by $\boldsymbol{\tau} = \boldsymbol{J}^{T\dagger}\mathcal{F}$. This approach provides a solution with the smallest norm among all possible actuator force vectors. Alternatively, other optimization techniques may be used to determine the actuator forces that project to $\mathcal{F}$ while minimizing a user-defined cost function, such as energy consumption under bounded actuator force/torques. These optimization techniques can be tailored to specific application needs as discussed in detail in Section 6.6.

## 9.4 Closed-Form Formulation

As outlined, the dynamic formulation for a general parallel robot with several links and a moving platform is encapsulated in Equation 9.22. To derive the dynamic equations using the principle of virtual work, it is crucial to break down the individual components of the wrenches and Jacobian matrices present in this equation. Assuming no frictional forces at the base and no external disturbances affecting the limbs, the wrenches applied to both the moving platform and the limbs at their centers of mass can be computed with respect to the base coordinate frame $\{\mathscr{A}\}$ [209].

$$\hat{\boldsymbol{\mathcal{F}}}_p = \begin{bmatrix} \hat{\boldsymbol{f}}_p \\ \hat{\boldsymbol{n}}_p \end{bmatrix} = - \begin{bmatrix} m_p \left( {}^A\boldsymbol{a}_p - {}^A\boldsymbol{g} \right) - \boldsymbol{f}_d \\ {}^A\boldsymbol{I}_p \, {}^A\dot{\boldsymbol{\omega}} + {}^A\boldsymbol{\omega}_p \times \left( {}^A\boldsymbol{I}_p \, {}^A\boldsymbol{\omega} \right) - \boldsymbol{n}_d \end{bmatrix},$$

(9.23)

$$\hat{\boldsymbol{\mathcal{F}}} i = \begin{bmatrix} \hat{\boldsymbol{f}}_i \\ \hat{\boldsymbol{n}}_i \end{bmatrix} = - \begin{bmatrix} m_i \left( {}^A\boldsymbol{a}_i - {}^A\boldsymbol{g} \right) \\ {}^A\boldsymbol{I}_{c_i} \, {}^A\dot{\boldsymbol{\omega}}_i + {}^A\boldsymbol{\omega}_i \times \left( {}^A\boldsymbol{I}_{c_i} \, {}^0\boldsymbol{\omega}_i \right) \end{bmatrix}.$$

(9.24)

Here, ${}^A\boldsymbol{a}_p$ represents the acceleration of the moving platform's center of mass in the base reference frame $\{\mathscr{A}\}$, ${}^A\boldsymbol{\omega}$ denotes the moving platform's angular velocity, and ${}^A\boldsymbol{I}p$ is the moment of inertia of the moving platform relative to its center of mass, as observed from frame $\{\mathscr{A}\}$. Similarly, ${}^A\boldsymbol{a}_i$ refers to the acceleration of the center of mass of each limb in the base reference frame $\{\mathscr{A}\}$, ${}^A\boldsymbol{\omega}$ denotes the angular velocity of the limb, and ${}^A\boldsymbol{I}_{c_i}$ is the moment of inertia of the limb relative to its center of mass, also represented in frame $\{\mathscr{A}\}$. In these relations, the total wrench applied to the moving platform is derived from both the inertial and external wrenches: $\hat{\boldsymbol{\mathcal{F}}}_p = \boldsymbol{\mathcal{F}}_p + \boldsymbol{\mathcal{F}}_p^*$, while the wrench applied to the limb is given by $\hat{\boldsymbol{\mathcal{F}}}_i = \boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_i^*$.

The wrench applied to the moving platform comprises two main components: forces and moments, denoted as $\hat{\boldsymbol{f}}_p$ and $\hat{\boldsymbol{n}}_p$, respectively. The force components include the inertial force $-m_p \, {}^A\boldsymbol{a}_p$, the gravitational force $m_p \, {}^A\boldsymbol{g}$, and

any disturbance force $\boldsymbol{f}_d$. The torque components include the inertial torque $-{}^A\boldsymbol{I}_p\,{}^A\dot{\boldsymbol{\omega}}_p - {}^A\boldsymbol{\omega}_p \times \left({}^A\boldsymbol{I}_p\,{}^A\boldsymbol{\omega}_p\right)$ and any disturbance torque $\boldsymbol{n}_d$. Similarly, the wrench applied to the limbs includes analogous components but without any disturbance forces or torques.

Next, consider the Jacobian matrix of the robot as defined in Equation 9.19, and the Jacobian matrix for the limbs as indicated in Equation 9.20. To facilitate analysis, let us decompose the limb Jacobian matrix into its linear and angular parts:

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{J}_{v_i} \\ \boldsymbol{J}_{\omega_i} \end{bmatrix} \tag{9.25}$$

Using this decomposition, the dynamic formulation derived from the principle of virtual work, as expressed in Equation 9.22, can be reformulated into an implicit form:

$$\boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{\mathcal{F}} = -\left(\boldsymbol{\mathcal{F}}_p + \sum_{i=1}^{k} \boldsymbol{J}_{v_i}^T\boldsymbol{f}_i + \sum_{i=1}^{k} \boldsymbol{J}_{\omega_i}^T\boldsymbol{n}_i\right). \tag{9.26}$$

Equation 9.26 represents the implicit dynamic formulation of the parallel manipulator. In this formulation, the projection of the actuator forces/torques, $\boldsymbol{\tau}$, onto the moving platform is denoted by $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$. The implicit

nature of this formulation poses challenges for direct use in dynamic simulations and model-based control algorithms.

To obtain a closed-form dynamic formulation using the principle of virtual work, we start by reconsidering the implicit dynamic formulation given in Equation 9.26. This involves redefining the kinematics of the limbs:

$$
\begin{aligned}
{}^{A}\boldsymbol{v}_i &= \boldsymbol{J}_{v_i}\, \dot{\boldsymbol{\chi}}_p, \\
\end{aligned}
\tag{9.27}
$$

$$
\begin{aligned}
{}^{A}\boldsymbol{\omega}_i &= \boldsymbol{J}_{\omega_i}\, \dot{\boldsymbol{\chi}}_p. \\
\end{aligned}
\tag{9.28}
$$

Consequently, the linear and angular accelerations of the limbs' centers of mass can be derived as:

$$
\begin{aligned}
{}^{A}\boldsymbol{a}_i &= \boldsymbol{J}_{v_i}\ddot{\boldsymbol{\chi}}_p + \dot{\boldsymbol{J}}_{v_i}\dot{\boldsymbol{\chi}}_p, \\
\end{aligned}
\tag{9.29}
$$

$$
\begin{aligned}
{}^{A}\dot{\boldsymbol{\omega}}_i &= \boldsymbol{J}_{\omega_i}\ddot{\boldsymbol{\chi}}_p + \dot{\boldsymbol{J}}_{\omega_i}\dot{\boldsymbol{\chi}}_p. \\
\end{aligned}
\tag{9.30}
$$

Using Equations 9.28 and 9.29 and applying the anti-commutative property of the cross product, Equation 9.23 can be rewritten as:

$$\hat{\boldsymbol{\mathcal{F}}}_i = \begin{bmatrix} \hat{\boldsymbol{f}}_i \\ \hat{\boldsymbol{n}}_i \end{bmatrix} = \begin{bmatrix} m_i(\boldsymbol{J}_{v_i}\ddot{\boldsymbol{\chi}}_p + \dot{\boldsymbol{J}}_{v_i}\dot{\boldsymbol{\chi}}_p - {}^A\boldsymbol{g}) \\ {}^A\boldsymbol{I}_i(\boldsymbol{J}_{\omega_i}\ddot{\boldsymbol{\chi}}_p + \dot{\boldsymbol{J}}_{\omega_i}\dot{\boldsymbol{\chi}}_p) + (\boldsymbol{J}_{\omega_i}\dot{\boldsymbol{\chi}}_p) \times {}^A\boldsymbol{I}_i\ (\boldsymbol{J}_{\omega_i}\dot{\boldsymbol{\chi}}_1 \end{bmatrix}$$

(9.31)

Substitute Equation 9.31 into Equation 9.26, and let us defining the disturbance wrench $\boldsymbol{\mathcal{F}}_d = [\boldsymbol{f}_d, \boldsymbol{n}_d]^T$, obtain the dynamic formulation of the manipulator in closed form:

$$\boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{\mathcal{F}} + \boldsymbol{\mathcal{F}}_d = \boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}}_p + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}_p, \dot{\boldsymbol{\chi}}_p)\boldsymbol{\chi}_p + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}_p),$$

(9.32)

where

$$\boldsymbol{M}(\boldsymbol{\chi}_p) \quad = \boldsymbol{M}_p + \sum_{i=1}^{k} \boldsymbol{M}_i,$$

(9.33)

$$\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}_p, \dot{\boldsymbol{\chi}}_p) \quad = \boldsymbol{\mathcal{C}}_p(\boldsymbol{\chi}_p, \dot{\boldsymbol{\chi}}_p) + \sum_{i=1}^{k} \boldsymbol{\mathcal{C}}_i,$$

(9.34)

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\chi}_p) \quad = \boldsymbol{\mathcal{G}}_p + \sum_{i=1}^{k} \boldsymbol{\mathcal{G}}_i.$$

(9.35)

In this closed-form formulation, the dynamic terms for the moving platform are given by:

$$\boldsymbol{M}_p = \begin{bmatrix} m_p \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & {}^A\boldsymbol{I}_p \end{bmatrix},$$

(9.36)

$$\boldsymbol{\mathcal{C}}_p = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & {}^A\boldsymbol{\omega}^A_\times \boldsymbol{I}_p \end{bmatrix},$$

(9.37)

$$\boldsymbol{\mathcal{G}}_p = \begin{bmatrix} -m_p {}^A\boldsymbol{g} \\ \boldsymbol{0}_{3\times 1} \end{bmatrix},$$

(9.38)

in which, $\boldsymbol{I}_{3\times 3}$ denotes the $3 \times 3$ identity matrix. Moreover, the closed-form dynamic terms of the limb $i$ are defined as:

$$\boldsymbol{M}_i = m_i \, \boldsymbol{J}_{v_i}^T \boldsymbol{J}_{v_i} + \boldsymbol{J}_{\omega_i}^T {}^A\boldsymbol{I}_i \, \boldsymbol{J}_{\omega_i},$$

(9.39)

$$\boldsymbol{\mathcal{C}}_i = m_i \boldsymbol{J}_{v_i}^T \dot{\boldsymbol{J}}_{v_i} + \boldsymbol{J}_{\omega_i}^T \, {}^A\boldsymbol{I}_i \, \dot{\boldsymbol{J}}_{\omega_i} + \boldsymbol{J}_{\omega_i}^T (\boldsymbol{J}_{\omega_i} \dot{\boldsymbol{\chi}}_p)_\times {}^A\boldsymbol{I}_i \boldsymbol{J}_{\omega_i},$$

(9.40)

$$\boldsymbol{\mathcal{G}}_i \quad = -m_i \, \boldsymbol{J}_{v_i}^{T\,A} \boldsymbol{g}.$$

(9.41)

These equations provide a comprehensive representation of the manipulator's dynamics, incorporating both the moving platform and limb contributions, and facilitate more straightforward application in simulations and control algorithms.

It is valuable to compare the three methods discussed in this book in terms of computational cost. The computational cost for dynamic formulation methods for parallel robots differs significantly among the Newton-Euler, Euler-Lagrange, and Virtual Work approaches. The Newton-Euler method requires consideration of unwanted internal forces and torques in the formulation, making it the least efficient in terms of computational complexity. For parallel robots with multiple interconnected limbs, the complexity increases substantially due to the need to handle multiple branches simultaneously.

In contrast, the Euler-Lagrange method is the most efficient in terms of computational complexity since it does not involve internal forces or acceleration analysis. However, it demands moderate computations due to the need to compute and differentiate the Lagrangian, which includes both kinetic and potential energy terms. Although

these computations can be performed by using symbolic manipulation software.

The virtual work method falls between the two in terms of complexity, being generally more computationally efficient than the Newton-Euler method. It effectively manages constraints and does not require the determination of internal forces. However, it necessitates acceleration analysis for deriving the limb Jacobians and their derivatives. The need to calculate and handle Jacobian matrices can be computationally demanding, especially for complex parallel structures.

Despite the computational efficiency of the virtual work formulation, accurately deriving the Jacobian matrices for both the moving platform and the limbs is crucial. These matrices are essential for translating between task space and actuator space, and errors in their calculation can significantly impact the accuracy of the derived actuator forces. The following sections will provide a detailed derivation of these Jacobian matrices, with comprehensive details and analyses for three case studies to illustrate the formulation process and address the challenges associated with deriving and applying the Jacobian matrices in practical scenarios.

## 9.5 Case Study I: Planar Cable Robot

In this section, the dynamic formulation of the planar 4RPR cable robot, whose architecture is detailed in Section 4.4.1 and illustrated in Figure 4.4, is carried out using the principle of virtual work. As previously mentioned, the moving platform in this manipulator is supported by four limbs with identical RP Rkinematic structures. A comprehensive Jacobian and singularity analysis of this manipulator is provided in Section 5.5. Additionally, the dynamic analysis of this manipulator using Newton-Euler and Euler-Lagrange methods are discussed in Sections 7.4, and 8.4, respectively.

### 9.5.1 Implicit Dynamics

To derive the dynamic formulation of the manipulator using the principle of virtual work, the external and inertial generalized forces acting on the moving platform $\hat{\boldsymbol{\mathcal{F}}}_p$ and on each limb $\hat{\boldsymbol{\mathcal{F}}}_i$ are first obtained. Subsequently, the Jacobian matrices of the limbs are determined, which are essential for relating the velocities and forces in the system.

The external and inertial wrenches applied to the moving platform are expressed as follows:

$$\hat{\boldsymbol{\mathcal{F}}}_p = \begin{bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{n}} \end{bmatrix} = \begin{bmatrix} f_{d_x} + M(\boldsymbol{g} \cdot \hat{\boldsymbol{x}}) - M\,\ddot{x}_G \\ f_{d_y} + M(\boldsymbol{g} \cdot \hat{\boldsymbol{y}}) - M\,\ddot{y}_G \\ \tau_d - I_m\,\ddot{\phi} \end{bmatrix}.$$

(9.42)

Here, $f_{d_x}$ and $f_{d_y}$ represent the external forces in the $x$ and $y$ directions, respectively, while $M$ denotes the mass of the moving platform, $\boldsymbol{g}$ is the gravitational acceleration, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are the unit vectors in the $x$ and $y$ directions, and $\ddot{x}_G$ and $\ddot{y}_G$ are the accelerations of the moving platform's center of mass. The term $\tau_d$ represents the external torque, and $I_m$ is the diagonal $z$ component of the moving platform's moment of inertia about the center of mass, with $\ddot{\phi}$ being the angular acceleration.

Furthermore, the external and inertial wrenches applied to limb $i$ are given by:

$$\hat{\boldsymbol{\mathcal{F}}}_i = \begin{bmatrix} \hat{\boldsymbol{f}}_i \\ \hat{\boldsymbol{n}}_i \end{bmatrix} = \begin{bmatrix} m_i(\boldsymbol{g} - \boldsymbol{a}_{c_i}) \\ -I_{c_i}\ddot{\alpha}_i \end{bmatrix},$$

(9.43)

where $m_i$ and $I_{c_i}$ represent the mass and the diagonal $z$ component of the moment of inertia of the limb about its center of mass, respectively. The term $\boldsymbol{a}_{c_i}$ represents the acceleration of the center of mass of the limb, and $\ddot{\alpha}_i$ denotes the angular acceleration of the limb.

Writing Equation 9.43 component-wise, in the cable directions $[\hat{\boldsymbol{S}}_i, \hat{\boldsymbol{N}}_i, \hat{\boldsymbol{K}}]^T$, and using the acceleration of the center of mass of the limbs found in Equation 7.27, the following relation is obtained after some manipulation:

$$\hat{\boldsymbol{\mathcal{F}}}_i = \rho_m \begin{bmatrix} L_i g^S - L_i \ddot{L}_i + \frac{1}{2}(L_i \dot{\alpha}_i)^2 \\ L_i g^N - \frac{1}{2} L_i^2 \ddot{\alpha}_i - \frac{3}{2} L_i \dot{L}_i \dot{\alpha}_i \\ \frac{1}{12} L_i^3 \ddot{\alpha}_i \end{bmatrix},$$

(9.44)

where $g^S$ and $g^N$ are the components of the gravity vector $\boldsymbol{g}$, along the $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{N}}$ directions, respectively.

The next step is to derive the Jacobians, which are fundamental to analyzing the manipulator's kinematics and dynamics. The manipulator Jacobian matrix $\boldsymbol{J}$ has been derived earlier and is provided in Equation 5.26. The limb Jacobians $\boldsymbol{J}_i$ could be derived from the linear and angular velocities of the center of mass of the limbs:

$$\dot{\boldsymbol{\chi}}_i = \begin{bmatrix} \boldsymbol{v}_{c_i} \\ \dot{\alpha}_i \hat{\boldsymbol{K}} \end{bmatrix} = \begin{bmatrix} \dot{L}_i \hat{\boldsymbol{S}}_i \\ \frac{1}{2} L_i \dot{\alpha}_i \hat{\boldsymbol{N}}_i \\ \dot{\alpha}_i \hat{\boldsymbol{K}} \end{bmatrix}.$$

(9.45)

This equation captures the velocities of the limb components, where $\boldsymbol{v}_{c_i}$ represents the linear velocity of the limb's center of mass, and $\dot{\alpha}_i \hat{\boldsymbol{K}}$ denotes the limb's angular velocity. By substituting Equations 5.25 and 5.28 into Equation 9.45, the limbs' Jacobian $\boldsymbol{J}_i$ can be derived to satisfy the relation $\dot{\boldsymbol{\chi}}_i = \boldsymbol{J}_i \boldsymbol{\chi}_p$ as follows:

$$\boldsymbol{J}_i = \begin{bmatrix} S_{i_x} & S_{i_y} & (E_{i_x}S_{i_y} - E_{i_y}S_{i_x}) \\ -\frac{1}{2}S_{i_y} & \frac{1}{2}S_{i_x} & \frac{1}{2}(E_{i_x}S_{i_x} + E_{i_y}S_{i_y}) \\ -\frac{1}{L_i}S_{i_y} & \frac{1}{L_i}S_{i_x} & \frac{1}{L_i}(E_{i_x}S_{i_x} + E_{i_y}S_{i_y}) \end{bmatrix}. \tag{9.46}$$

This formulation is critical for the mapping relation between the velocities and forces in the manipulator limbs from joint space into task space.

Finally, consider the actuator forces as $\boldsymbol{\tau} = \begin{bmatrix} -\tau_1 & -\tau_2 & -\tau_3 & -\tau_4 \end{bmatrix}$, representing positive actuator forces for tensions. By substituting Equations 5.25, 9.42, 9.44, and 9.5.1 into Equation 9.22, the governing equation of motion of the manipulator can be derived after some manipulation:

$$f_{d_x} + M(\boldsymbol{g} \cdot \hat{\boldsymbol{x}}) - M\,\ddot{x}_G - \sum_{i=1}^{4} \left\{ (\tau_i - P_i)S_{i_x} + Q_i S_{i_y} \right\} = 0 \tag{9.47}$$

$$f_{d_y} + M(\boldsymbol{g} \cdot \hat{\boldsymbol{y}}) - M\,\ddot{y}_G - \sum_{i=1}^{4} \left\{ (\tau_i - P_i)S_{i_y} - Q_i S_{i_x} \right\} = 0 \tag{9.48}$$

$$\tau_d - I_m \ddot{\phi} - \sum_{i=1}^{4} \Big\{ (\tau_i - P_i)(E_{i_x} S_{i_y} - E_{i_y} S_{i_x}) - Q_i (\boldsymbol{E}_i \cdot \hat{\boldsymbol{S}}_i$$

(9.49)

where

$$P_i \quad = \rho_m \left( L_i g^S - L_i \ddot{L}_i + \frac{1}{2} (L_i \dot{\alpha}_i)^2 \right),$$

(9.50)

$$Q_i \quad = \rho_m \left( \frac{1}{2} L_i g^N - \frac{3}{4} L_i \dot{L}_i \dot{\alpha}_i - \frac{1}{3} L_i^2 \ddot{\alpha}_i \right).$$

(9.51)

In these equations, $P_i$ and $Q_i$ represent the internal dynamics of each limb, capturing the effects of gravity, inertia, and angular motion. The terms $\tau_i$ represent the actuator forces, which, when adjusted for tensions, play a crucial role in balancing the manipulator's dynamics.

Equation 9.47 addresses the force equilibrium in the *x*-direction, where the sum of the external force components, inertial forces, and the contributions from the actuator and internal forces must equal zero. Similarly, Equation 9.48 ensures force equilibrium in the *y*-direction, and Equation 9.49 ensures torque equilibrium about the center of mass.

The derived equations incorporate the implicit manipulator's dynamics comprehensively, considering the contributions from both external and internal forces, as well as the actuator inputs. Moreover, the detailed expressions for $P_i$ and $Q_i$ highlight the complex interplay between the limbs' movements and the manipulator's overall dynamics.

### 9.5.2 Formulation Verification

To validate the accuracy and reliability of the dynamic formulation developed for this manipulator, this section undertakes a comparative analysis of two distinct methods: the Newton-Euler approach and the principle of virtual work. Specifically, we will overlay the dynamic formulations derived from the Newton-Euler method, detailed in Equations 7.42 through 7.44, with those obtained using the principle of virtual work, as given in Equations 9.47 through 9.49. While these methods are fundamentally different in their theoretical underpinnings, they should produce equivalent results when applied correctly.

Through this comparison, we aim to demonstrate that despite the different frameworks and computational strategies employed by the Newton-Euler and virtual work methods, the final dynamic formulations are consistent with each other. By comparing term by term, it can be shown that the results from both methods align precisely.

To demonstrate the validity and consistency of the dynamic formulations derived from different methods, we will examine and compare the *x*-component equations from both the Newton-Euler and virtual work approaches. Specifically, we will analyze Equations 7.42 and 9.47.

First, let's substitute $F_{B_i}^N$ and $F_{B_i}^S$ from Equations 7.38 and 7.39, respectively, into Equation 7.42:

$$M\ddot{\boldsymbol{x}}_G - M(\boldsymbol{g}\cdot\hat{\boldsymbol{x}}) - f_{d_x} + \sum_{i=1}^{4}\left(F_{B_i}^S S_{i_x} - F_{B_i}^N S_{i_y}\right) = 0$$

$$\Rightarrow M\ddot{\boldsymbol{x}}_G - M(\boldsymbol{g}\cdot\hat{\boldsymbol{x}}) - f_{d_x} + \sum_{i=1}^{4}\left(\tau_i + \rho_m\left(-L_i g^S + L_i\ddot{L}_i\right.\right.$$

$$\left.\left. - \sum_{i=1}^{4}\rho_m\left(-\frac{1}{2}L_i g^N + \frac{3}{4}L_i\dot{L}_i\dot{\alpha}_i + \frac{1}{3}L_i^2\ddot{\alpha}_i\right)S_{i_y} = 0.\right.\right.$$

Rearranging this equation, we obtain:

$$M\ddot{\boldsymbol{x}}_G - M(\boldsymbol{g}\cdot\hat{\boldsymbol{x}}) - f_{d_x} + \sum_{i=1}^{4}\left(\tau_i - \rho_m\left(L_i g^S - L_i\ddot{L}_i + \frac{1}{2}\right.\right.$$

$$\left.\left. + \sum_{i=1}^{4}\rho_m\left(\frac{1}{2}L_i g^N - \frac{3}{4}L_i\dot{L}_i\dot{\alpha}_i - \frac{1}{3}L_i^2\ddot{\alpha}_i\right)S_{i_y} = 0.\right.\right. \tag{9.52}$$

Next, consider Equation 9.47. To facilitate comparison, negate this equation:

$$M\ddot{x}_G - M(\boldsymbol{g}\cdot\hat{\boldsymbol{x}}) - f_{d_x} + \sum_{i=1}^{4}\left((\tau_i - P_i)S_{i_x} + Q_i S_{i_y}\right) = 0$$

Substitute $P_i$ and $Q_i$ from Equations 9.50 and 9.51:

$$M\ddot{x}_G - M(\boldsymbol{g}\cdot\hat{\boldsymbol{x}}) - f_{d_x} + \sum_{i=1}^{4}\left(\tau_i - \rho_m\left(L_i g^S - L_i\ddot{L}_i + \frac{1}{2}\right.\right.$$

$$\left.\left.+\sum_{i=1}^{4}\rho_m\left(\frac{1}{2}L_i g^N - \frac{3}{4}L_i\dot{L}_i\dot{\alpha}_i - \frac{1}{3}L_i^2\ddot{\alpha}_i\right)S_{i_y} = 0.\right.\right.$$

(9.53)

It is clear that Equations 9.52 and 9.53 are term-by-term identical. This indicates that the *x*-component results from both the Newton-Euler and virtual work methods are consistent with each other.

To ensure the thoroughness of this comparison, similar procedures can be applied to the *y* and *z* components of the dynamic equations. By performing these comparisons, we can confirm that the equations of motion in these directions are also identical across both methods. This consistency across different components reinforces the validity of the dynamic formulations and demonstrates that both

approaches effectively describe the manipulator's dynamics in a coherent manner.

## 9.6 Case Study II: Stewart–Gough Platform

In this section, we explore the dynamic analysis of the Stewart–Gough platform, employing the principle of virtual work. The architecture of the platform is outlined in Section 4.6.1 and illustrated in Figure 4.16. The kinematic details of this manipulator are thoroughly covered in Section 4.6, while the Jacobian analysis is elaborated in Section 5.7. For a comprehensive discussion of the dynamic aspects of this manipulator, please refer to Section 7.5.

To derive the equations of motion for the Stewart–Gough platform using the principle of virtual work, we start by determining the external and inertial wrenches that act on the moving platform, denoted as $\hat{\boldsymbol{\mathcal{F}}}_p$, as well as on each limb, represented as $\hat{\boldsymbol{\mathcal{F}}}_i$. Each limb is composed of a cylinder and a piston, so the wrenches are split into two components: $\hat{\boldsymbol{\mathcal{F}}}_i = \hat{\boldsymbol{\mathcal{F}}}_{i_1} + \hat{\boldsymbol{\mathcal{F}}}_{i_2}$. Here, $\hat{\boldsymbol{\mathcal{F}}}_{i_1}$ refers to the forces related to the cylinder, while $\hat{\boldsymbol{\mathcal{F}}}_{i_2}$ pertains to the piston.

To apply the principle of virtual work, we substitute these components into the formulation. The resulting equation captures the balance of virtual work for the entire manipulator, expressed as:

$$\boldsymbol{J}^T\boldsymbol{\tau} + \hat{\boldsymbol{\mathcal{F}}}_p + \sum_i \left( \boldsymbol{J}_{i_1}^T \hat{\boldsymbol{\mathcal{F}}}_{i_1} + \boldsymbol{J}_{i_2}^T \hat{\boldsymbol{\mathcal{F}}}_{i_2} \right) = \mathbf{0}. \tag{9.54}$$

In this equation, $\boldsymbol{J}$ denotes the Jacobian matrix for the moving platform, and $\boldsymbol{\tau}$ represents the actuator forces. The term $\hat{\boldsymbol{\mathcal{F}}}_p$ encompasses the external and inertial wrenches acting on the platform, while the summation term accounts for the contributions of each limb's cylinder and piston.

The Jacobian matrix for the manipulator is derived in Equation 5.56, allowing us to compute the Cartesian wrench, which projects the actuator forces into Cartesian space, as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$. This is expressed by:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau} = \sum_i \begin{bmatrix} \hat{\boldsymbol{s}}_i \boldsymbol{\tau}_i \\ \boldsymbol{b}_{i_\times} \hat{\boldsymbol{s}}_i \boldsymbol{\tau}_i \end{bmatrix}, \tag{9.55}$$

where $\boldsymbol{b}_{i_\times}$ represents the skew-symmetric matrix derived from the vector elements $\boldsymbol{b}_i$, as defined in Equation 7.109.

Next, we determine the external and inertial wrenches acting on the moving platform:

$$\hat{\boldsymbol{\mathcal{F}}}_p = \begin{bmatrix} \hat{\boldsymbol{f}} \\ {}^c\hat{\boldsymbol{n}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_d + m(\boldsymbol{g} - \boldsymbol{a}_p) \\ \boldsymbol{n}_d - {}^A\boldsymbol{I}_P\dot{\boldsymbol{\omega}} - \boldsymbol{\omega} \times {}^A\boldsymbol{I}_P\boldsymbol{\omega} \end{bmatrix},$$

(9.56)

where $\boldsymbol{a}_p$ is the acceleration of point $P$ on the moving platform, as given in Equation 7.59. Here, $m$ denotes the mass of the moving platform, and ${}^{A}\boldsymbol{I}_P$ represents the moment of inertia about the center of mass of the platform. Additionally, $\boldsymbol{\omega}$ is the angular velocity, $m\boldsymbol{g}$ is the gravitational force, and $\boldsymbol{\mathcal{F}}_d = [\boldsymbol{F}_d, \boldsymbol{n}_d]^T$ represents the external disturbance wrench applied to the platform.

The external and inertial wrenches for each limb segment are calculated using the following equations:

$$\hat{\boldsymbol{\mathcal{F}}}_{i_1} = \begin{bmatrix} \hat{\boldsymbol{f}}_{i_1} \\ \hat{\boldsymbol{n}}_{i_1} \end{bmatrix} = \begin{bmatrix} m_{i_1}(\boldsymbol{g} - \boldsymbol{a}_{i_1}) \\ -{}^{A}\boldsymbol{I}_{c_{i_1}}\dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i \times {}^{A}\boldsymbol{I}_{c_{i_1}}\boldsymbol{\omega}_i \end{bmatrix},$$

(9.57)

and

$$\hat{\boldsymbol{\mathcal{F}}}_{i_2} = \begin{bmatrix} \hat{\boldsymbol{f}}_{i_2} \\ \hat{\boldsymbol{n}}_{i_2} \end{bmatrix} = \begin{bmatrix} m_{i_2}(\boldsymbol{g} - \boldsymbol{a}_{i_2}) \\ -{}^{A}\boldsymbol{I}_{c_{i_2}}\dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i \times {}^{A}\boldsymbol{I}_{c_{i_2}}\boldsymbol{\omega}_i \end{bmatrix}.$$

(9.58)

In these equations, ${}^{A}\boldsymbol{I}_{c_{i_1}}$ and ${}^{A}\boldsymbol{I}_{c_{i_2}}$ are the inertia matrices of the cylinder and piston, respectively, evaluated at their respective centers $c_{i_1}$ and $c_{i_2}$. Additionally, $\boldsymbol{a}_{i_1}$ and $\boldsymbol{a}_{i_2}$ represent the accelerations of the centers of mass of the cylinder and piston, as described in Equations 7.66 and

7.67. Substituting these accelerations simplifies the equations to:

$$\hat{\boldsymbol{\mathcal{F}}}_{i_1} = \begin{bmatrix} m_{i_1}\left(\boldsymbol{g} - c_{i_1}(\dot{\boldsymbol{\omega}}_{i_\times}\hat{\boldsymbol{s}}_i) - c_{i_1}|\boldsymbol{\omega}_i|_2^2\hat{\boldsymbol{s}}_i\right) \\ -{}^A\boldsymbol{I}_{c_{i_1}}\dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_{i_\times}{}^A\boldsymbol{I}_{c_{i_1}}\boldsymbol{\omega}_i \end{bmatrix},$$

(9.59)

and

$$\hat{\boldsymbol{\mathcal{F}}}_{i_2} = \begin{bmatrix} m_{i_2}\boldsymbol{g} - m_{i_2}(\ell_i - c_{i_2})\left(\dot{\boldsymbol{\omega}}_{i_\times}\hat{\boldsymbol{s}}_i - |\boldsymbol{\omega}_i|_2^2\hat{\boldsymbol{s}}_i + 2\dot{\ell}_i\boldsymbol{\omega}_{i_\times}\hat{\boldsymbol{s}}_i + \ddot{\ell}_i\right. \\ -{}^A\boldsymbol{I}_{c_{i_2}}\dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_{i_\times}{}^A\boldsymbol{I}_{c_{i_2}}\boldsymbol{\omega}_i \end{bmatrix}$$

(9.60)

The next step involves deriving the Jacobians for the limb segments. The Jacobians $\boldsymbol{J}_{i_1}$ and $\boldsymbol{J}_{i_2}$ for the cylinder and piston, respectively, are derived from their linear and angular velocities as follows:

$$\begin{aligned} \dot{\boldsymbol{\chi}}_{i_1} &= \boldsymbol{J}_{i_1}\dot{\boldsymbol{\chi}} \\ \begin{bmatrix} \boldsymbol{v}_{i_1} \\ \boldsymbol{\omega}_i \end{bmatrix} &= \boldsymbol{J}_{i_1}\begin{bmatrix} \boldsymbol{v}_p \\ \boldsymbol{\omega} \end{bmatrix} \end{aligned}$$

(9.61)

$$
\begin{aligned}
\dot{\boldsymbol{\chi}}_{i_2} &= \boldsymbol{J}_{i_2}\dot{\boldsymbol{\chi}} \\
\begin{bmatrix} \boldsymbol{v}_{i_2} \\ \boldsymbol{\omega}_i \end{bmatrix} &= \boldsymbol{J}_{i_2} \begin{bmatrix} \boldsymbol{v}_p \\ \boldsymbol{\omega} \end{bmatrix}.
\end{aligned}
$$

(9.62)

The linear velocity of the center of mass of the cylinder, $\boldsymbol{v}_{i_1}$, can be expressed as follows, using $\boldsymbol{\omega}_i$ from Equation 7.110 and $\dot{\boldsymbol{x}}_i$ from Equation 7.121:

$$
\begin{aligned}
\boldsymbol{v}_{i_1} = c_{i_1}(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) &= -c_{i_1}(\hat{\boldsymbol{s}}_i \times \boldsymbol{\omega}_i) \\
&= -c_{i_1}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{\omega}_i \\
&= -\frac{c_{i_1}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i \\
&= -\frac{c_{i_1}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2\begin{bmatrix} I_{3\times 3} & -\boldsymbol{b}_{i_\times} \end{bmatrix}\dot{\boldsymbol{\chi}} \\
&= -\frac{c_{i_1}}{\ell_i}\begin{bmatrix} \hat{\boldsymbol{s}}_{i_\times}^2 & -\hat{\boldsymbol{s}}_{i_\times}^2\boldsymbol{b}_{i_\times} \end{bmatrix}\dot{\boldsymbol{\chi}}.
\end{aligned}
$$

(9.63)

Similarly, the linear velocity of the center of mass of the piston, $\boldsymbol{v}_{i_2}$, is given by:

$$\begin{aligned}
\boldsymbol{v}_{i_2} = c_{i_2}(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) + \dot{l}\hat{\boldsymbol{s}}_i \quad &= -c_{i_2}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{\omega}_i + \dot{l}\hat{\boldsymbol{s}}_i \\
&= -\frac{c_{i_2}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2\dot{\boldsymbol{x}}_i + \hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\dot{\boldsymbol{x}}_i \\
&= \left(-\frac{c_{i_2}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2 + \hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\right)[I_{3\times 3} \quad -\boldsymbol{b}_{i_\times}]\boldsymbol{\chi} \\
&= \left[-\frac{c_{i_2}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2 + \hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T \quad \frac{c_{i_2}}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}^2\boldsymbol{b}_{i_\times} - \hat{\boldsymbol{s}}
\end{aligned} \tag{9.64}$$

Furthermore, the angular velocity of the limb $\boldsymbol{\omega}_i$ can be related to the generalized coordinates of the moving platform $\dot{\boldsymbol{\chi}}$ by:

$$\begin{aligned}
\boldsymbol{\omega}_i \quad &= \frac{1}{\ell_i}\left(\hat{\boldsymbol{s}}_i \times \boldsymbol{v}_p + \hat{\boldsymbol{s}}_i \times (\boldsymbol{\omega} \times \boldsymbol{b}_i)\right) \\
&= \frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{v}_p - \frac{1}{\ell_i}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{b}_{i_\times}\boldsymbol{\omega} \\
&= \frac{1}{\ell_i}[\hat{\boldsymbol{s}}_{i_\times} - \hat{\boldsymbol{s}}_{i_\times}\boldsymbol{b}_{i_\times}]\begin{bmatrix}\boldsymbol{v}_p \\ \boldsymbol{\omega}\end{bmatrix}.
\end{aligned} \tag{9.65}$$

Thus, the final Jacobians for the limb segments are determined as:

$$\boldsymbol{J}_{i_1} = \frac{1}{\ell_i}\begin{bmatrix} -c_{i_1}\hat{\boldsymbol{s}}_{i_\times}^2 & c_{i_1}\hat{\boldsymbol{s}}_{i_\times}^2\boldsymbol{b}_{i_\times} \\ \hat{\boldsymbol{s}}_{i_\times} & -\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{b}_{i_\times} \end{bmatrix},$$

(9.66)

and

$$\boldsymbol{J}_{i_2} = \frac{1}{\ell_i}\begin{bmatrix} -c_{i_2}\hat{\boldsymbol{s}}_{i_\times}^2 + \ell_i\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T & c_{i_2}\hat{\boldsymbol{s}}_{i_\times}^2\boldsymbol{b}_{i_\times} - \ell_i\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^T\boldsymbol{b}_{i_\times} \\ \hat{\boldsymbol{s}}_{i_\times} & -\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{b}_{i_\times} \end{bmatrix}.$$

(9.67)

The final dynamic model of the Stewart-Gough platform can be expressed implicitly by substituting the appropriate terms into Equation 9.54. Specifically:

- The actuator wrench $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$ is substituted with Equation 9.55.

- The wrench on the moving platform, $\hat{\boldsymbol{\mathcal{F}}}_p$, is substituted with Equation 9.56.

- The wrenches applied to the cylinder and piston, $\hat{\boldsymbol{\mathcal{F}}}_{i_1}$ and $\hat{\boldsymbol{\mathcal{F}}}_{i_2}$, are substituted with Equations 9.59 and 9.60, respectively.

- The corresponding Jacobian matrices, $\boldsymbol{J}_{i_1}$ and $\boldsymbol{J}_{i_2}$, are substituted with Equations 9.66 and 9.67, respectively.

Substituting these expressions into the principle of virtual work yields a complete dynamic formulation of the Stewart-Gough platform. This formulation integrates all the actuator forces, external and inertial wrenches, and the detailed Jacobians of the limbs. The resulting dynamic equations can be further simplified, though deriving each term individually and comparing the simplified result with those obtained using the Newton-Euler method can be a highly intricate process. Furthermore, to validate the accuracy of these formulations, inverse and forward dynamic simulations can be performed.

---

## 9.7 Case Study III: The 3UPU Manipulator

Unlike the case studies examined in earlier chapters, this chapter introduces a fourth case study focused on deriving the closed-form dynamic formulation for a different type of parallel manipulator, namely, the 3UPU translational parallel manipulator (TPM). This manipulator uses two universal joints on both the base and the moving platform. These joints introduce angular velocity components for each limb, which are often overlooked in existing research to simplify the robot's kinematic and dynamic analysis [210, 211].

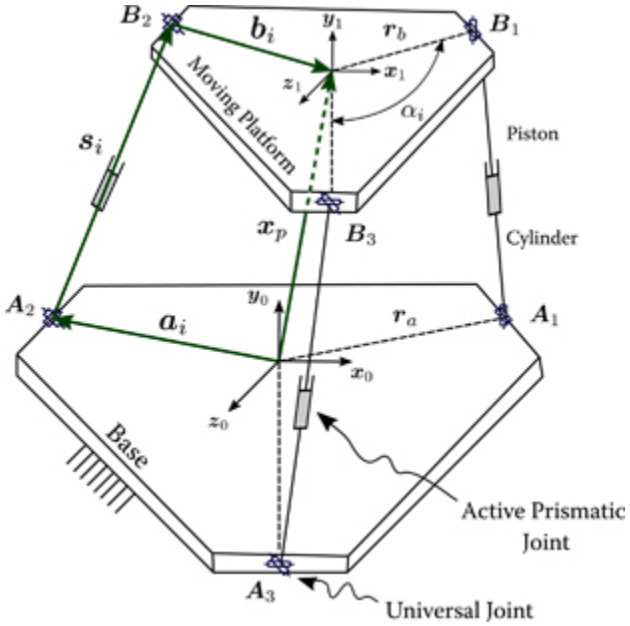
While simplifying assumptions make kinematic analysis more straightforward, they can compromise the accuracy of the dynamic model. To address this issue, the case study

begins by thoroughly identifying and detailing all kinematic terms affecting the 3UPU manipulator. Following this, the principle of virtual work is used to extract the implicit dynamics of the 3UPU manipulator. The section then proceeds to derive the closed-form dynamic formulation of the robot using the Jacobians of the moving platform and limbs, along with their derivatives.

Additionally, in this section the skew-symmetry property of the dynamic matrices is examined, which is crucial for verifying the consistency and correctness of the dynamic model. By conducting a comprehensive analysis that includes detailed kinematic terms and dynamic formulations, this case study aims to fill gaps left by previous case studies to provide closed-form formulation for a parallel manipulator using the virtual work approach.

### 9.7.1 Manipulator Structure and Kinematics

The 3UPU translational parallel manipulator discussed here features a symmetrical architecture composed of three kinematic chains. Each chain includes a cylinder and a piston actuator, connected to both the base and the moving platform via universal joints. These universal joints can be kinematically modeled as two revolute joints with intersecting, perpendicular axes. In this design, the universal joints are arranged at an angle of $\alpha = 120^\circ$ relative to one another, as illustrated in the schematic of the 3UPU manipulator shown in Figure 9.1.




▶ Long Description for Figure 9.1

**FIGURE 9.1**

Schematic of a 3UPU manipulator. ⏎

Moreover, it is assumed that the attachment points $A_i$ and $B_i$ are situated on the $x_0 - z_0$ and $x_1 - z_1$ planes, respectively. Therefore, with respect to the base frame coordinates, the position vectors $A_i$ and $B_i$ can be expressed as $\boldsymbol{a}_i = \boldsymbol{R}_y(\alpha_i)[0,\, 0,\, r_a]^T$ and $\boldsymbol{b}_i = \boldsymbol{R}_y(\alpha_i)[0,\, 0,\, r_b]^T$, respectively. Here, $\boldsymbol{x}_p = [x_p,\, y_p,\, z_p]^T$ represents the position of the moving platform, while $\boldsymbol{q} = [d_1,\, d_2,\, d_3]^T$ denotes the robot's active joint variables.

Furthermore, it is assumed that the base frame and the moving platform are rigidly connected through these kinematic chains. The moving platform is designed for purely translational movement, which simplifies the control and analysis of the manipulator. This translational

movement is ensured when the axes of the two universal joints in each limb are perpendicular to the axis of the prismatic joint and are parallel to each other [8].

The design of the 3UPU manipulator also offers several advantages, such as a high degree of symmetry and a simplified control structure. This can lead to more straightforward analytical solutions and potentially easier implementation of control algorithms. However, the complexity of the underlying kinematics necessitates a thorough understanding and accurate modeling to fully leverage these benefits.

Referring to Figure 9.1, for $i = 1, 2, 3$, apply the loop-closure method to each limb.

$$\boldsymbol{s}_i = \ell_i \hat{\boldsymbol{s}}_i = \boldsymbol{x}_p + \boldsymbol{b}_i - \boldsymbol{a}_i,$$

(9.68)

where $\hat{\boldsymbol{s}}_i$ is a unit vector in the direction of limb $i$. Consequently, the inverse kinematics of the manipulator can be determined easily. Using this approach, the active joint variable, $\ell_i$, can be calculated as:

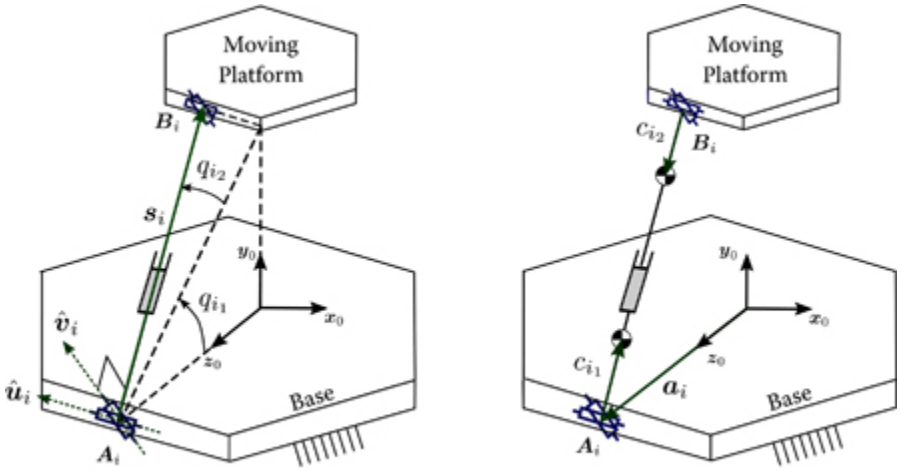
$$\ell_i = \sqrt{\left[\boldsymbol{x}_p + \boldsymbol{b}_i - \boldsymbol{a}_i\right]^T \left[\boldsymbol{x}_p + \boldsymbol{b}_i - \boldsymbol{a}_i\right]}.$$

(9.69)

Moreover, in each closed kinematic chain, there are two passive angles, $q_{i_1}$ and $q_{i_2}$, associated with the revolute

joints of each universal joint. As shown in Figure 9.2(a), these angles can be determined as follows:

$$
\begin{aligned}
q_{i_1} &= \operatorname{atan2}\left({}^{i}\boldsymbol{s}_{y_i}, -{}^{i}\boldsymbol{s}_{z_i}\right), \\
q_{i_2} &= \sin^{-1}\left({}^{i}\boldsymbol{s}_{x_i}/\ell_i\right),
\end{aligned}
$$




(a) Passive angles for each limb. (b) Center of mass positions.

▶ Long Description for Figure 9.2

**FIGURE 9.2**

Schematics and geometry of the 3UPU parallel manipulator. ⏎

where ${}^{i}\boldsymbol{s}_i$ is defined as ${}^{i}\boldsymbol{s}_i = \boldsymbol{R}_y^T(\alpha_i)\,\boldsymbol{s}_i$. Furthermore, the unit vector for each universal joint can be derived as follows:

$$
\begin{aligned}
\hat{\boldsymbol{u}}_i &= \boldsymbol{R}_y(\alpha_i)[-1,\, 0,\, 0]^T, \\
\hat{\boldsymbol{v}}_i &= \boldsymbol{R}_y(\alpha_i)[0,\, \cos(q_{i_1}),\, \sin(q_{i_1})]^T, \\
\hat{\boldsymbol{w}}_i &= \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{v}}_i.
\end{aligned}
\tag{9.70}
$$

Thus, the rotation matrix for each limb with respect to the base coordinate system can be expressed as:

$$
{}^0\boldsymbol{R}_i = \begin{bmatrix}
\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{s}}_i & \hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{v}}_i & \hat{\boldsymbol{x}} \cdot \left(\hat{\boldsymbol{s}}_i \times \hat{\boldsymbol{v}}_i\right) \\
\hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{s}}_i & \hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{v}}_i & \hat{\boldsymbol{y}} \cdot \left(\hat{\boldsymbol{s}}_i \times \hat{\boldsymbol{v}}_i\right) \\
\hat{\boldsymbol{z}} \cdot \hat{\boldsymbol{s}}_i & \hat{\boldsymbol{z}} \cdot \hat{\boldsymbol{v}}_i & \hat{\boldsymbol{z}} \cdot \left(\hat{\boldsymbol{s}}_i \times \hat{\boldsymbol{v}}_i\right)
\end{bmatrix},
\tag{9.71}
$$

where $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ denote the unit axes of the base coordinate system $\{0\}$.

Finally, as shown in Figure 9.2(b), the position of the center of mass for both the cylinder and the piston of each limb can be written as:

$$
\begin{aligned}
\boldsymbol{c}_{i_1} &= \boldsymbol{a}_i + c_{i_1}\hat{\boldsymbol{s}}_i, \\
\boldsymbol{c}_{i_2} &= \boldsymbol{a}_i + (\ell_i - c_{i_2})\hat{\boldsymbol{s}}_i.
\end{aligned}
\tag{9.72}
$$

### 9.7.2 Jacobian Analysis

To obtain the Jacobian matrix of the manipulator, differentiate both sides of the loop closure Equation 9.68 with respect to time, to derive the following velocity loop closure equation.

$$\dot{\boldsymbol{x}}_p = \dot{\ell}_i \hat{\boldsymbol{s}}_i + \ell_i \left( \boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i \right).$$

(9.73)

In this equation, $\dot{\ell}_i$ represents the rate of change in limb length and $\boldsymbol{\omega}_i$ denotes the angular velocity of each limb. To solve for $\dot{\ell}_i$, dot product both sides of Equation 9.73 with $\hat{\boldsymbol{s}}_i$:

$$\dot{\ell}_i = {\hat{\boldsymbol{s}}_i}^T \dot{\boldsymbol{x}}_p.$$

(9.74)

Therefore, the Jacobian matrix for the moving platform of the 3U<u>P</u>U manipulator can be expressed as:

$$\boldsymbol{J} = \begin{bmatrix} \hat{\boldsymbol{s}}_1 & \hat{\boldsymbol{s}}_2 & \hat{\boldsymbol{s}}_3 \end{bmatrix}^T.$$

(9.75)

Similarly, by cross-multiplying both sides of Equation 9.73 with $\hat{\boldsymbol{s}}_i$, we can derive the perpendicular component of the angular velocity for each limb as follows:

$$\boldsymbol{\omega}_{n_i} = \frac{\hat{\boldsymbol{s}}_{i_\times}}{\ell_i} \dot{\boldsymbol{x}}_p,$$

(9.76)

where $\hat{\boldsymbol{s}}_{i_\times}$ denotes the $3 \times 3$ skew-symmetric matrix derived from the vector $\hat{\boldsymbol{s}}_i$. It is important to highlight that $\boldsymbol{\omega}_{n_i}$ is not equal to $\boldsymbol{\omega}_i$; rather, it represents only one component of the total angular velocity $\boldsymbol{\omega}_i$. This distinction is a common source of error in the dynamic analysis of parallel robots with universal joints, as discussed in references [191, 212].

The accurate representation of the angular velocity of the limbs comprises two components. The first component, $\boldsymbol{\omega}_{n_i}$, is perpendicular to the limb axis, while the second component, $\omega_{s_i}$, aligns with the direction of the limb axis. Thus, the angular velocity can be expressed as:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{n_i} + \omega_{s_i} \hat{\boldsymbol{s}}_i.$$

(9.77)

Furthermore, $\boldsymbol{\omega}_i$ can be conceptualized as a vector split into two components along the unit vectors $\hat{\boldsymbol{u}}_i$ and $\hat{\boldsymbol{v}}_i$ of the universal joint, as illustrated in Figure 9.2(a):

$$\boldsymbol{\omega}_i = \omega_{u_i} \hat{\boldsymbol{u}}_i + \omega_{v_i} \hat{\boldsymbol{v}}_i.$$

(9.78)

Given that the angular velocity of each limb is considered within the $\hat{\boldsymbol{u}}_i - \hat{\boldsymbol{v}}_i$ plane, it follows that $\hat{\boldsymbol{w}}_i \cdot \boldsymbol{\omega}_i = 0$ holds true. Therefore, to determine the $\omega_{s_i}$ component of the angular velocity, it is sufficient to perform a dot product of

both sides of Equation 9.77 with $\hat{\boldsymbol{w}}_i$, the unit vector perpendicular to the directions of the unit vectors $\hat{\boldsymbol{u}}_i$ and $\hat{\boldsymbol{v}}_i$ given in Equation 9.70. Thus,

$$\omega_{s_i} = \frac{-\hat{\boldsymbol{w}}_i \cdot \boldsymbol{\omega}_{n_i}}{\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{s}}_i},$$

(9.79)

Now, consider the Jacobian matrix relation for the limb

$$\boldsymbol{\omega}_i = \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p.$$

(9.80)

Equation 9.77 can be rewritten in terms of $\dot{\boldsymbol{x}}_p$, allowing us to derive $\boldsymbol{J}_{\omega_i}$ as follows:

$$\boldsymbol{J}_{\omega_i} = \boldsymbol{J}_{\omega_{n_i}} + \hat{\boldsymbol{s}}_i \, \boldsymbol{J}_{\omega_{s_i}},$$

(9.81)

where $\boldsymbol{J}_{\omega_{n_i}}$ and $\boldsymbol{J}_{\omega_{s_i}}$ are defined as:

$$\boldsymbol{J}_{\omega_{n_i}} \quad = \frac{\hat{\boldsymbol{s}}_{i_\times}}{\ell_i}, \quad \in \mathbb{R}^{3\times 3},$$

(9.82)

$$\boldsymbol{J}_{\omega_{s_i}} \quad = \left( \frac{-\hat{\boldsymbol{w}}_i{}^T}{\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{s}}_i} \right) \boldsymbol{J}_{\omega_{n_i}}, \quad \in \mathbb{R}^{1\times 3}.$$

(9.83)

In this way, the angular component of the limb's Jacobian matrix is fully determined.

Let's derive the scalar values of $\omega_{u_i}$ and $\omega_{v_i}$ for use in calculating the derivatives of the limbs' Jacobian matrices. To do this, substitute Equation 9.78 into Equation 9.73 and then perform dot products of both sides with $\hat{\boldsymbol{v}}_i$ and $\hat{\boldsymbol{u}}_i$, respectively.

$$\omega_{u_i} = \frac{\hat{\boldsymbol{v}}_i \cdot \dot{\boldsymbol{x}}_p - \hat{\boldsymbol{v}}_i \cdot \hat{\boldsymbol{s}}_i \, \dot{\ell}_i}{\ell_i \, \hat{\boldsymbol{v}}_i \cdot \left( \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{s}}_i \right)}, \quad \omega_{v_i} = \frac{\hat{\boldsymbol{u}}_i \cdot \dot{\boldsymbol{x}}_p - \hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i \, \dot{\ell}_i}{\ell_i \, \hat{\boldsymbol{u}}_i \cdot \left( \hat{\boldsymbol{v}}_i \times \hat{\boldsymbol{s}}_i \right)}.$$

(9.84)

Consider the vector product properties:

$$\begin{aligned}
&\hat{\boldsymbol{v}}_i \cdot \left( \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{s}}_i \right) = -\hat{\boldsymbol{s}}_i \cdot \left( \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{v}}_i \right), \\
&\hat{\boldsymbol{u}}_i \cdot \left( \hat{\boldsymbol{v}}_i \times \hat{\boldsymbol{s}}_i \right) = \hat{\boldsymbol{s}}_i \cdot \left( \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{v}}_i \right), \\
&\left( \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{v}}_i \right) = \hat{\boldsymbol{w}}_i.
\end{aligned}$$

Using these properties, in Equation 9.84 to derive:

$$\omega_{u_i} = -\left(\frac{\hat{\boldsymbol{v}}_i \cdot \dot{\boldsymbol{x}}_p - \hat{\boldsymbol{v}}_i \cdot \hat{\boldsymbol{s}}_i \, \dot{\ell}_i}{\ell_i \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{w}}_i}\right) = \boldsymbol{J}_{\omega_{u_i}} \dot{\boldsymbol{x}}_p,$$
$$\omega_{v_i} = +\left(\frac{\hat{\boldsymbol{u}}_i \cdot \dot{\boldsymbol{x}}_p - \hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i \, \dot{\ell}_i}{\ell_i \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{w}}_i}\right) = \boldsymbol{J}_{\omega_{v_i}} \dot{\boldsymbol{x}}_p.$$

(9.85)

Therefore, $\boldsymbol{J}_{\omega_{u_i}}$ and $\boldsymbol{J}_{\omega_{v_i}}$ can be derived by substituting $\dot{\ell}_i = \hat{\boldsymbol{s}}_i{}^T \dot{\boldsymbol{x}}_p$ into Equation 9.85 and performing some manipulations, resulting in:

$$\boldsymbol{J}_{\omega_{u_i}} = \frac{\hat{\boldsymbol{v}}_i{}^T \left(-\boldsymbol{I}_{3\times 3} + \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i{}^T\right)}{\ell_i \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{w}}_i}, \quad \in \mathbb{R}^{1\times 3},$$
$$\boldsymbol{J}_{\omega_{v_i}} = \frac{\hat{\boldsymbol{u}}_i{}^T \left(+\boldsymbol{I}_{3\times 3} - \hat{\boldsymbol{s}}_i \hat{\boldsymbol{s}}_i{}^T\right)}{\ell_i \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{w}}_i}, \quad \in \mathbb{R}^{1\times 3}.$$

(9.86)

Here, $\boldsymbol{I}_{3\times 3}$ represents the $3 \times 3$ identity matrix.

Now, let us derive the linear velocity components of the limb's Jacobian matrix. To do this, differentiate both sides of the position relations of the limb's center of masses given in Equation 9.72. Thus, the linear velocity of the center of mass of the cylinder and the piston of each limb can be represented by:

$$
\begin{aligned}
\boldsymbol{v}_{c_{i_1}} &= c_{i_1}\left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right), \\
\boldsymbol{v}_{c_{i_2}} &= \left(\ell_i - c_{i_2}\right)\left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right) + \dot{\ell}_i \hat{\boldsymbol{s}}_i.
\end{aligned}
\tag{9.87}
$$

Utilizing Equations 9.74, 9.77, and the anti-commutative property of the cross product, these linear velocities can be expressed as functions of $\dot{\boldsymbol{x}}_p$ as follows:

$$
\begin{aligned}
\boldsymbol{v}_{c_{i_1}} &= -c_{i_1}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{\omega}_i = \left(-c_{i_1}\hat{\boldsymbol{s}}_{i_\times}\,\boldsymbol{J}_{\omega_i}\right)\dot{\boldsymbol{x}}_p, \\
\boldsymbol{v}_{c_{i_2}} &= -\left(\left(\ell_i - c_{i_2}\right)\hat{\boldsymbol{s}}_{i_\times}\,\boldsymbol{J}_{\omega_i}\right)\dot{\boldsymbol{x}}_p + \left(\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i{}^T\right)\dot{\boldsymbol{x}}_p.
\end{aligned}
$$

Consequently, the linear component of each limb's Jacobian matrix can be derived as follows:

$$
\boldsymbol{J}_{v_{i_1}} = -c_{i_1}\hat{\boldsymbol{s}}_{i_\times}\,\boldsymbol{J}_{\omega_i},
\tag{9.88}
$$

$$
\boldsymbol{J}_{v_{i_2}} = -\left(\ell_i - c_{i_2}\right)\hat{\boldsymbol{s}}_{i_\times}\,\boldsymbol{J}_{\omega_i} + \hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i{}^T.
\tag{9.89}
$$

Finally, the Jacobian matrix for the limb's cylinder and piston can be expressed as:

$$\boldsymbol{J}_{i_1} = \begin{bmatrix} \boldsymbol{J}_{v_{i_1}} \\ \boldsymbol{J}_{\omega_i} \end{bmatrix}, \quad \boldsymbol{J}_{i_2} = \begin{bmatrix} \boldsymbol{J}_{v_{i_2}} \\ \boldsymbol{J}_{\omega_i} \end{bmatrix}.$$

(9.90)

### 9.7.3 Acceleration Analysis

To derive $\ddot{\ell}_i$, take the time derivative of Equation 9.74 as follows:

$$\ddot{\ell}_i = \hat{\boldsymbol{s}}_i{}^T \ddot{\boldsymbol{x}}_p + (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)^T \dot{\boldsymbol{x}}_p.$$

(9.91)

Thus, it is evident that $\dot{\boldsymbol{J}}$ can be derived as:

$$\dot{\boldsymbol{J}} = \begin{bmatrix} (\boldsymbol{\omega}_1 \times \hat{\boldsymbol{s}}_1) & (\boldsymbol{\omega}_2 \times \hat{\boldsymbol{s}}_2) & (\boldsymbol{\omega}_3 \times \hat{\boldsymbol{s}}_3) \end{bmatrix}^T$$

(9.92)

Moreover, by taking the time derivative of both sides of Equation 9.73 we can derive the acceleration loop closure equation for each limb:

$$\ddot{\boldsymbol{x}}_p \ddot{\ell}_i \hat{\boldsymbol{s}}_i + 2\dot{\ell}_i \left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right) + \ell_i \left(\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i\right) + \ell_i \left(\boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right)\right).$$

(9.93)

Next, let's derive the time derivative of the limb Jacobians. To achieve this, and to derive $\dot{\boldsymbol{\omega}}_{n_i}$, we need to cross multiply both sides of Equation 9.93 by $\hat{\boldsymbol{s}}_i$:

$$\begin{aligned}\dot{\boldsymbol{\omega}}_{n_i} &= \frac{\hat{\boldsymbol{s}}_i \times \left(\ddot{\boldsymbol{x}}_p - 2\dot{\ell}_i \left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right) - \ell_i \left(\boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i\right)\right)\right)}{\ell_i} \\ &= \boldsymbol{J}_{\omega_{n_i}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_{n_i}} \dot{\boldsymbol{x}}_p.\end{aligned} \tag{9.94}$$

Utilizing the anti-commutative property of the cross product and Equation 9.77, $\dot{\boldsymbol{J}}_{\omega_{n_i}}$ can be derived as follows:

$$\dot{\boldsymbol{J}}_{\omega_{n_i}} = \frac{2\,\dot{\ell}_i\,{\hat{\boldsymbol{s}}_i}_\times^{\,2}\,\boldsymbol{J}_{\omega_i} + \ell_i\,\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{\omega}_{i_\times}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{J}_{\omega_i}}{\ell_i}. \tag{9.95}$$

To calculate the scalar value of $\dot{\omega}_{s_i}$, differentiate Equations 9.77 and 9.78:

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{n_i} + \dot{\omega}_{s_i}\hat{\boldsymbol{s}}_i = \boldsymbol{J}_{\omega_i}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_i}\dot{\boldsymbol{x}}_p, \tag{9.96}$$

$$\dot{\boldsymbol{\omega}}_i = \dot{\omega}_{u_i}\hat{\boldsymbol{u}}_i + \dot{\omega}_{v_i}\hat{\boldsymbol{v}}_i + \omega_{u_i}\omega_{v_i}\hat{\boldsymbol{w}}_i. \tag{9.97}$$

Noting that $\hat{\boldsymbol{v}}_i \cdot \hat{\boldsymbol{s}}_i = 0$, first calculate $\dot{\omega}_{u_i}$ by taking the dot product of both sides of Equation 9.97 with $\hat{\boldsymbol{u}}_i$, and then calculate $\dot{\omega}_{s_i}$ by taking the dot product of both sides of Equation 9.96 with $\hat{\boldsymbol{s}}_i$, as follows:

$$\dot{\omega}_{u_i} = \left(\dot{\boldsymbol{\omega}}_{n_i} + \dot{\omega}_{s_i}\hat{\boldsymbol{s}}_i\right) \cdot \hat{\boldsymbol{u}}_i,$$

(9.98)

$$\dot{\omega}_{s_i} = \dot{\omega}_{u_i}\left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right) + \omega_{u_i}\omega_{v_i}\left(\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{s}}_i\right).$$

(9.99)

Thus, $\dot{\omega}_{s_i}$ can be derived in closed-form by substituting Equation 9.98 into Equation 9.99, as follows:

$$\dot{\omega}_{s_i} = \frac{\left(\dot{\boldsymbol{\omega}}_{n_i} \cdot \hat{\boldsymbol{u}}_i\right)\left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right) + \omega_{u_i}\omega_{v_i}\left(\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{s}}_i\right)}{1 - \left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)^2}.$$

(9.100)

On the other hand, $\dot{\omega}_{s_i}$ can be expressed as $\dot{\omega}_{s_i} = \boldsymbol{J}_{\omega_{s_i}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_{s_i}}\dot{\boldsymbol{x}}_p$. To derive $\boldsymbol{J}_{\omega_{s_i}}$ and $\dot{\boldsymbol{J}}_{\omega_{s_i}}$, substitute Equation 9.7.3 into Equation 9.100 and, with some manipulations, $\dot{\omega}_{s_i}$ can be computed as follows:

$$\begin{aligned}\dot{\boldsymbol{\omega}}_{s_i} &= \frac{\left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)\hat{\boldsymbol{u}}_i{}^T \left(\boldsymbol{J}_{\omega_{n_i}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_{n_i}}\ddot{\boldsymbol{x}}_p\right) + \left(\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{s}}_i\right)\left(\boldsymbol{J}_{\omega_{u_i}}\dot{\boldsymbol{x}}_p\, \boldsymbol{J}\right.}{1 - \left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)^2} \\ &= \boldsymbol{J}_{\omega_{s_i}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_{s_i}}\dot{\boldsymbol{x}}_p.\end{aligned}$$

Therefore, $\boldsymbol{J}_{\omega_{s_i}}$ and $\dot{\boldsymbol{J}}_{\omega_{s_i}}$ can be derived as follows:

$$\boldsymbol{J}_{\omega_{s_i}} = \frac{\left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)\hat{\boldsymbol{u}}_i{}^T}{1 - \left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)^2}\, \boldsymbol{J}_{\omega_{n_i}},$$

(9.101)

$$\dot{\boldsymbol{J}}_{\omega_{s_i}} = \frac{\left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)\hat{\boldsymbol{u}}_i{}^T \dot{\boldsymbol{J}}_{\omega_{n_i}} + \left(\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{s}}_i\right)\left(\boldsymbol{J}_{\omega_{u_i}}\dot{\boldsymbol{x}}_p\, \boldsymbol{J}_{\omega_{v_i}}\right)}{1 - \left(\hat{\boldsymbol{u}}_i \cdot \hat{\boldsymbol{s}}_i\right)^2}.$$

(9.102)

It should be noted that $\boldsymbol{J}_{\omega_{s_i}}$ can be derived from both Equations 9.83 and 9.101, and the equality of these two expressions can be easily verified. Finally, by rewriting Equation 9.96 as $\dot{\boldsymbol{\omega}}_i = \boldsymbol{J}_{\omega_i}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_i}\dot{\boldsymbol{x}}_p$, where $\boldsymbol{J}_{\omega_i}$ is provided in Equation 9.81, $\dot{\boldsymbol{J}}_{\omega_i}$ can be expressed as:

$$\dot{\boldsymbol{J}}_{\omega_i} = \dot{\boldsymbol{J}}_{\omega_{n_i}} + \hat{\boldsymbol{s}}_i\, \dot{\boldsymbol{J}}_{\omega_{s_i}},$$

(9.103)

where $\dot{\boldsymbol{J}}_{\omega_{n_i}}$ and $\dot{\boldsymbol{J}}_{\omega_{s_i}}$ are given by Equations 9.95 and 9.102, respectively.

The linear acceleration vectors of the center of masses of the piston and the cylinder can be obtained by differentiating Equation 9.87:

$$\begin{aligned}\boldsymbol{a}_{c_{i_1}} &= c_{i_1}\left(\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)\right) \\ &= \boldsymbol{J}_{v_{i_1}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_1}}\dot{\boldsymbol{x}}_p,\end{aligned}$$

(9.104)

$$\begin{aligned}\boldsymbol{a}_{c_{i_2}} &= (\ell_i - c_{i_2})\left(\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i)\right) + 2\dot{\ell}_i(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) - \\ &= \boldsymbol{J}_{v_{i_2}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_2}}\dot{\boldsymbol{x}}_p.\end{aligned}$$

(9.105)

By applying the anti-commutative property of the cross product, along with Equations 9.77, 9.91 and 9.96, various terms in Equations 9.104 and 9.105 can be expressed as functions of $\dot{\boldsymbol{x}}_p$ and $\ddot{\boldsymbol{x}}_p$ as follows:

$$
\begin{aligned}
&\dot{\boldsymbol{\omega}}_i \times \hat{\boldsymbol{s}}_i = -\hat{\boldsymbol{s}}_{i_\times}(\boldsymbol{J}_{\omega_i}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_i}\dot{\boldsymbol{x}}_p),\\
&\boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) = -\boldsymbol{\omega}_{i_\times}\ \hat{\boldsymbol{s}}_{i_\times}\ \boldsymbol{\omega}_i = -\boldsymbol{\omega}_{i_\times}\ \hat{\boldsymbol{s}}_{i_\times}\ (\boldsymbol{J}_{\omega_i}\dot{\boldsymbol{x}}_p),\\
&2\dot{\ell}_i(\boldsymbol{\omega}_i \times \hat{\boldsymbol{s}}_i) = -2\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}\ (\boldsymbol{J}_{\omega_i}\dot{\boldsymbol{x}}_p),\\
&\hat{\boldsymbol{s}}_i\ddot{\ell}_i = \hat{\boldsymbol{s}}_i({\hat{\boldsymbol{s}}_i}^T\ddot{\boldsymbol{x}}_p + \left(\boldsymbol{\omega}_{i\times}\hat{\boldsymbol{s}}_i\right)^T\dot{\boldsymbol{x}}_p).
\end{aligned}
$$

Therefore, Equations 9.104 and 9.105 can be rewritten as follows:

$$
\begin{aligned}
\boldsymbol{a}_{c_{i_1}} &= -\left(c_{i_1}\hat{\boldsymbol{s}}_{i_\times}\boldsymbol{J}_{\omega_i}\right)\ddot{\boldsymbol{x}}_p - c_{i_1}\left(\hat{\boldsymbol{s}}_{i_\times}\dot{\boldsymbol{J}}_{\omega_i} + \boldsymbol{\omega}_{i_\times}\ \hat{\boldsymbol{s}}_{i_\times}\ \boldsymbol{J}_{\omega_i}\right)\dot{\boldsymbol{x}}_p\\
&= \boldsymbol{J}_{v_{i_1}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_1}}\dot{\boldsymbol{x}}_p,
\end{aligned}
\tag{9.106}
$$

$$
\begin{aligned}
\boldsymbol{a}_{c_{i_2}} &= \left(-(\ell_i - c_{i_2})\hat{\boldsymbol{s}}_{i_\times}\ \boldsymbol{J}_{\omega_i} + \hat{\boldsymbol{s}}_i{\hat{\boldsymbol{s}}_i}^T\right)\ddot{\boldsymbol{x}}_p +\\
&\quad \left(-(\ell_i - c_{i_2})\left(\hat{\boldsymbol{s}}_{i_\times}\ \dot{\boldsymbol{J}}_{\omega_i} + \boldsymbol{\omega}_{i_\times}\ \hat{\boldsymbol{s}}_{i_\times}\ \boldsymbol{J}_{\omega_i}\right) - 2\dot{\ell}_i\hat{\boldsymbol{s}}_{i_\times}\ \boldsymbol{J}_{\omega_i} + \hat{\boldsymbol{s}}\right.\\
&= \boldsymbol{J}_{v_{i_2}}\ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_2}}\dot{\boldsymbol{x}}_p.
\end{aligned}
\tag{9.107}
$$

It should be noted that $\boldsymbol{J}_{v_{i_1}}$ and $\boldsymbol{J}_{v_{i_2}}$ are given in Equation 9.88, and $\dot{\boldsymbol{J}}_{v_{i_1}}$ while $\dot{\boldsymbol{J}}_{v_{i_2}}$ can be derived as:

$$
\dot{\boldsymbol{J}}_{v_{i_1}} = -c_{i_1}\left(\hat{\boldsymbol{s}}_{i_\times}\ \dot{\boldsymbol{J}}_{\omega_i} + \boldsymbol{\omega}_{i_\times}\ \hat{\boldsymbol{s}}_{i_\times}\ \boldsymbol{J}_{\omega_i}\right),
$$

(9.108)

$$\dot{\boldsymbol{J}}_{v_{i_2}} = -(\ell_i - c_{i_2}) \left( \hat{\boldsymbol{s}}_{i_\times} \dot{\boldsymbol{J}}_{\omega_i} + \boldsymbol{\omega}_{i_\times} \hat{\boldsymbol{s}}_{i_\times} \boldsymbol{J}_{\omega_i} \right) - 2\dot{\ell}_i \hat{\boldsymbol{s}}_{i_\times} \boldsymbol{J}_{\omega_i} + \hat{\boldsymbol{s}}_i ($$

(9.109)

### 9.7.4 Implicit Dynamics

In this section, the implicit dynamic formulation of the 3UPU manipulator is initially analyzed using the principle of virtual work. Subsequently, in the upcoming section the closed-form expressions for the dynamics are derived and their properties are verified.

Referring to Equation 9.22, the dynamic formulation of the 3UPU manipulator, based on the principle of virtual work, can be derived as follows:

$$\boldsymbol{J}^T \tau + \boldsymbol{f}_p + \sum_{i=1}^{3} \left( {\boldsymbol{J}_{i_1}}^T \boldsymbol{f}_{i_1} + {\boldsymbol{J}_{i_2}}^T \boldsymbol{f}_{i_2} \right) = 0,$$

(9.110)

Here, $\boldsymbol{J}$ represents the Jacobian matrix of the manipulator, $\boldsymbol{\tau}$ denotes the generalized forces applied by the actuators, and $\boldsymbol{J}_{i_1}$ and $\boldsymbol{J}_{i_2}$ refer to the Jacobian matrices of the cylinder and piston of each limb, respectively. Additionally, $\boldsymbol{f}_p$ and $\boldsymbol{f}_i$ signify the inertia forces acting at the center of mass of the cylinder and piston of each limb. Assuming that

friction effects are negligible and that the moving platform of the 3UPU manipulator does not experience rotational motion, these inertial forces can be derived as follows [213]:

$$\begin{aligned}
\boldsymbol{f}_p &= -m_p\left(\ddot{\boldsymbol{x}}_p - {}^0\boldsymbol{g}\right) + \boldsymbol{f}_d, \\
\boldsymbol{f}_{i_1} &= -\begin{bmatrix} m_{i_1}\left(\boldsymbol{a}_{c_{i_1}} - {}^0\boldsymbol{g}\right) \\ {}^0\boldsymbol{I}_{i_1}\,\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \left({}^0\boldsymbol{I}_{i_1}\,\boldsymbol{\omega}_i\right) \end{bmatrix}, \\
\boldsymbol{f}_{i_2} &= -\begin{bmatrix} m_{i_2}\left(\boldsymbol{a}_{c_{i_2}} - {}^0\boldsymbol{g}\right) \\ {}^0\boldsymbol{I}_{i_2}\,\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \left({}^0\boldsymbol{I}_{i_2}\,\boldsymbol{\omega}_i\right) \end{bmatrix}.
\end{aligned} \tag{9.111}$$

Here, ${}^0\boldsymbol{I}_{i_1}$ and ${}^0\boldsymbol{I}_{i_2}$ represent the moments of inertia of the cylinder and piston of each limb about their centers of mass with respect to the base coordinate system $\{0\}$. Additionally, $\boldsymbol{f}_d$ denotes the external disturbance wrench applied to the moving platform. Consequently, the inverse dynamic model of the 3UPU manipulator can be expressed as:

$$\boldsymbol{J}^T\tau = \boldsymbol{f} = -\boldsymbol{f}_p - \sum_{i=1}^{3}\left(\boldsymbol{J}_{i_1}^T\boldsymbol{f}_{i_1} + \boldsymbol{J}_{i_2}^T\boldsymbol{f}_{i_2}\right), \tag{9.112}$$

Here, $\boldsymbol{f} = \boldsymbol{J}^T\boldsymbol{\tau}$ represents the projection of the actuator forces $\boldsymbol{\tau}$ onto the moving platform. In this formulation, the

Jacobian matrices for the limbs, $\boldsymbol{J}_{i_1}$ and $\boldsymbol{J}_{i_2}$, can be computed using Equation 9.90, while the Jacobian matrix of the moving platform is given by Equation 9.75.

This formulation provides the implicit dynamic model of the 3UPU manipulator. It shows how the actuator forces $\boldsymbol{\tau}$ are implicitly related to the forces and torques acting on the moving platform. According to this model, the actuator forces $\boldsymbol{\tau}$ can be uniquely determined by the inverse relation $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{f}$, provided the robot is in a non-singular configuration. This allows for precise control and analysis of the manipulator's behavior under various conditions.

Furthermore, this implicit dynamic formulation is essential for tasks such as trajectory planning, control, and real-time monitoring of the manipulator's performance. By understanding how the actuator forces are projected onto the moving platform and how they influence the overall dynamics, one can develop more effective control strategies and ensure the accurate operation of the 3UPU manipulator in practical applications.

### 9.7.5 Closed-Form Formulation

To obtain the closed-form expression for the dynamic formulations using the principle of virtual work, the implicit dynamic formulation given in Equation 9.112 is revisited.

As previously discussed, the velocity and acceleration of the center of mass for the cylinder and piston in each limb can be expressed using their respective Jacobian matrices.

$$
\begin{aligned}
&\boldsymbol{v}_{c_{i_1}} = \boldsymbol{J}_{v_{i_1}} \dot{\boldsymbol{x}}_p, && \boldsymbol{v}_{c_{i_2}} = \boldsymbol{J}_{v_{i_2}} \dot{\boldsymbol{x}}_p, \\
&\boldsymbol{a}_{c_{i_1}} = \boldsymbol{J}_{v_{i_1}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_1}} \dot{\boldsymbol{x}}_p, && \boldsymbol{a}_{c_{i_2}} = \boldsymbol{J}_{v_{i_2}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_2}} \dot{\boldsymbol{x}}_p, \\
&\boldsymbol{\omega}_i = \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p, && \dot{\omega}_i = \boldsymbol{J}_{\omega_i} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_i} \dot{\boldsymbol{x}}_p,
\end{aligned}
\tag{9.113}
$$

Substituting Equation 9.113 into Equation 9.111 yields:

$$
\begin{aligned}
\boldsymbol{f}_p &= -m_p \left( \ddot{\boldsymbol{x}}_p - {}^0\boldsymbol{g} \right) + \boldsymbol{f}_d, \\
\boldsymbol{f}_{i_1} &= - \begin{bmatrix} m_{i_1} \left( \boldsymbol{J}_{v_{i_1}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_1}} \dot{\boldsymbol{x}}_p - {}^0\boldsymbol{g} \right) \\ {}^0\boldsymbol{I}_{i_1} \left( \boldsymbol{J}_{\omega_i} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_i} \dot{\boldsymbol{x}}_p \right) + \left( \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p \right)^0_\times \boldsymbol{I}_{1_i} \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p \end{bmatrix}, \\
\boldsymbol{f}_{i_2} &= - \begin{bmatrix} m_{i_2} \left( \boldsymbol{J}_{v_{i_2}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_2}} \dot{\boldsymbol{x}}_p - {}^0\boldsymbol{g} \right) \\ {}^0\boldsymbol{I}_{i_2} \left( \boldsymbol{J}_{\omega_i} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{\omega_i} \dot{\boldsymbol{x}}_p \right) + \left( \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p \right)^0_\times \boldsymbol{I}_{i_2} \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p \end{bmatrix}.
\end{aligned}
$$

By applying the inertial forces from Equation 9.7.5, Equation 9.112 can be rewritten as:

$$
\begin{aligned}
\boldsymbol{J}^T\tau = \boldsymbol{f} + \boldsymbol{f}_d = \quad & \sum_{i=1}^{3} m_{i_1} \boldsymbol{J}_{v_{i_1}}^T \left( (\boldsymbol{J}_{v_{i_1}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_1}} \dot{\boldsymbol{x}}_p) - {}^0\boldsymbol{g} \right) \\
& + \sum_{i=1}^{3} m_{i_2} \boldsymbol{J}_{v_{i_2}}^T \left( (\boldsymbol{J}_{v_{i_2}} \ddot{\boldsymbol{x}}_p + \dot{\boldsymbol{J}}_{v_{i_2}} \dot{\boldsymbol{x}}_p) - {}^0\boldsymbol{g} \right) \\
& + \sum_{i=1}^{3} \boldsymbol{J}_{\omega_i}^T \left( \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i_2} \right) (\boldsymbol{J}_{\omega_i} \ddot{\boldsymbol{x}}_p + \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p) \right) \\
& + \sum_{i=1}^{3} \boldsymbol{J}_{\omega_i}^T \left( (\boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p)_\times \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i_2} \right) \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p \right) \\
& + m_p (\ddot{\boldsymbol{x}}_p - {}^0\boldsymbol{g}).
\end{aligned}
\tag{9.114}
$$

Therefore, the closed-form dynamic formulation of the 3U<u>P</u>U manipulator can be expressed as:

$$
\boldsymbol{J}^T\tau = \boldsymbol{f} + \boldsymbol{f}_d = \boldsymbol{M}(\boldsymbol{x}_p)\ddot{\boldsymbol{x}}_p + \boldsymbol{\mathcal{C}}(\boldsymbol{x}_p, \dot{\boldsymbol{x}}_p)\dot{\boldsymbol{x}}_p + \boldsymbol{\mathcal{G}}(\boldsymbol{x}_p),
\tag{9.115}
$$

where $\boldsymbol{M}(\boldsymbol{x}_p)$, $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_p, \dot{\boldsymbol{x}}_p)$, and $\boldsymbol{\mathcal{G}}(\boldsymbol{x}_p)$ represent the mass matrix, the Christoffel matrix, and the gravity vector of the robot, respectively. These can be determined as follows:

$$
\begin{aligned}
\boldsymbol{M}(\boldsymbol{x}_p) &= m_p + \sum_{i=1}^{3} \Big( m_{i_1} \boldsymbol{J}_{v_{i_1}}^T \boldsymbol{J}_{v_{i_1}} + m_{i_2} \boldsymbol{J}_{v_{i_2}}^T \boldsymbol{J}_{v_{i_2}} + \boldsymbol{J}_{\omega_i}^T \left( {}^0\boldsymbol{I}_i \right. \\
\boldsymbol{\mathcal{C}}(\boldsymbol{x}_p, \dot{\boldsymbol{x}}_p) &= \sum_{i=1}^{3} \Big( m_{i_1} \boldsymbol{J}_{v_{i_1}}^T \dot{\boldsymbol{J}}_{v_{i_1}} + m_{i_2} \boldsymbol{J}_{v_{i_2}}^T \dot{\boldsymbol{J}}_{v_{i_2}} + \boldsymbol{J}_{\omega_i}^T \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i} \right. \\
&\qquad + \boldsymbol{J}_{\omega_i}^T ( \boldsymbol{J}_{\omega_i} \dot{\boldsymbol{x}}_p \\
\boldsymbol{\mathcal{G}}(\boldsymbol{x}_p) &= -m_p {}^0\boldsymbol{g} - \sum_{i=1}^{3} \left( m_{i_1} \boldsymbol{J}_{v_{i_1}}^T + m_{i_2} \boldsymbol{J}_{v_{i_2}}^T \right) {}^0\boldsymbol{g}.
\end{aligned}
$$

The closed-form formulation is the governing dynamic equation of the manipulator with some advantageous properties for designing model-based controllers. The properties of the dynamic matrices is fully elaborated in Chapter 8 To verify the skew-symmetric property of dynamic matrices, determine the time derivative of the mass matrix $\dot{\boldsymbol{M}}(\boldsymbol{x}_p, \dot{\boldsymbol{x}}_p)$:

$$
\begin{aligned}
\dot{\boldsymbol{M}} = \; & m_{i_1} \left( \dot{\boldsymbol{J}}_{v_{i_1}}^T \boldsymbol{J}_{v_{i_1}} + \boldsymbol{J}_{v_{i_1}}^T \dot{\boldsymbol{J}}_{v_{i_1}} \right) + m_{i_2} \left( \dot{\boldsymbol{J}}_{v_{i_2}}^T \boldsymbol{J}_{v_{i_2}} + \boldsymbol{J}_{v_{i_2}}^T \dot{\boldsymbol{J}}_{v_{i_2}} \right) \\
& + \boldsymbol{J}_{\omega_i}^T \left( \boldsymbol{\omega}_{i_\times} \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i_2} \right) + \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i_2} \right) \boldsymbol{\omega}_{i_\times}^T \right) \boldsymbol{J}_{\omega_i} \\
& + \dot{\boldsymbol{J}}_{\omega_i}^T \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i_2} \right) \boldsymbol{J}_{\omega_i} + \boldsymbol{J}_{\omega_i}^T \left( {}^0\boldsymbol{I}_{i_1} + {}^0\boldsymbol{I}_{i_2} \right) \dot{\boldsymbol{J}}_{\omega_i},
\end{aligned}
\tag{9.116}
$$

Substitute the Jacobian matrices and their derivatives into this equation, and with some manipulation, it can be

confirmed that the following skew-symmetric property of the dynamic matrices is satisfied.

$$(\dot{\boldsymbol{M}} - 2\boldsymbol{C})^T + (\dot{\boldsymbol{M}} - 2\boldsymbol{C}) = 0. \tag{9.117}$$

The closed-form dynamics of the manipulator provide a detailed and comprehensive representation of the manipulator's behavior by incorporating the contributions from both the moving platform and the limbs. This formulation not only captures the full scope of dynamic interactions but also simplifies its application in practical scenarios. It enables more efficient and accurate simulations, facilitating the development of sophisticated model-based control algorithms and optimization strategies.

## Problems

9.1 Revisit the five-bar mechanism illustrated in Figure 4.27 and detailed in Problems 4.1 and 5.1. Derive the dynamic formulation of this manipulator using the virtual work method. Additionally, compare this dynamic formulation with those obtained using the Newton–Euler approach, as discussed in Problem 7.1, and the Euler-Lagrange method, explored in Problem 8.1.

9.2 Examine the five-bar mechanism depicted in Figure 4.28 and described in Problems 4.2 and 5.1. Derive the dynamic formulation of this mechanism using the principle of virtual work. Then, compare the results with those obtained through the Newton–Euler formulation presented in Problem 7.2 and the Euler-Lagrange method discussed in Problem 8.2.

9.3 Review the planar pantograph mechanism shown in Figure 4.27 and analyzed in Problems 4.3 and 5.3. Derive its dynamic formulation using the virtual work method. Subsequently, compare this formulation with those derived using the Newton–Euler method in Problem 7.3 and the Euler-Lagrange approach discussed in Problem 8.3.

9.4 Revisit the planar three-degree-of-freedom 3RRR parallel manipulator illustrated in Figure 4.30 and detailed in Problems 4.4 and 5.4. Develop its dynamic formulation using the virtual work method. Then, compare this formulation with those obtained through the Newton–Euler method as outlined in Problem 7.4 and the Euler-Lagrange approach covered in Problem 8.4.

9.5 Revisit the planar redundant 4RRR parallel manipulator depicted in Figure 4.31 and analyzed in Problems 4.5 and 5.5. Formulate its dynamics using the virtual work method. Afterward, compare this formulation with those derived using the Newton–Euler

approach described in Problem 7.5 and the Euler-Lagrange method discussed in Problem 8.5.

9.6 Reexamine the spatial parallel manipulator illustrated in Figure 4.32 and detailed in Problems 4.6 and 5.6. Derive its dynamics using the virtual work method. Then, compare this formulation with those obtained using the Newton–Euler approach outlined in Problem 7.6 and the Euler-Lagrange method covered in Problem 8.6.

9.7 Revisit the three-degrees-of-freedom 3UPS manipulator presented in Reference [51], depicted in Figure 4.34, and discussed in Problems 4.8 and 5.8. Develop its dynamic formulation using the virtual work method. Following that, compare this formulation with those derived through the Newton–Euler approach detailed in Problem 7.8 and the Euler-Lagrange method discussed in Problem 8.8.

9.8 Reexamine the three-degrees-of-freedom 3RPS manipulator depicted in Figure 4.35 and discussed in Problems 4.9 and 5.9. Formulate its dynamics using the virtual work method. Then, compare this formulation with those obtained through the Newton–Euler approach described in Problem 7.9 and the Euler-Lagrange method outlined in Problem 8.9.

9.9 Reevaluate the three-degrees-of-freedom 3PRC translational parallel manipulator shown in Figure 4.35 and detailed in Problems 4.10 and 5.10. Develop its

dynamics using the virtual work method. Furthermore, compare this formulation with those derived using the Newton–Euler method as outlined in Problem 7.10 and the Euler-Lagrange approach discussed in Problem 8.10.

9.10 Revisit the Delta robot illustrated in Figure 4.38 and detailed in Problems 4.11 and 5.11. Formulate its dynamics using the virtual work method. Then, compare this formulation with those obtained through the Newton–Euler approach described in Problem 7.11 and the Euler-Lagrange method discussed in Problem 8.11.

9.11 Examine the Quadrupteron robot depicted in Figure 4.39 and detailed in Problems 4.12 and 5.12. Derive its dynamics using the virtual work method. Following this, compare the resulting formulation with those obtained via the Newton–Euler method in Problem 7.12 and the Euler-Lagrange method covered in Problem 8.12.

# 10

# *Dynamic Simulation and Calibration*

DOI: 10.1201/9781003590408-10

This chapter focuses on dynamic simulation and calibration techniques in the context of robotic manipulators. It begins with an introduction to motion planning, covering key methods such as cubic polynomials, cubic polynomials with via points, quintic polynomials, and linear trajectories with parabolic blends. The chapter also explores trajectory planning in joint space, offering a range of methods for optimizing the movement of robotic systems.

Following this, motion simulation techniques are discussed, including forward dynamics and inverse dynamics. These methods are applied to practical case studies, beginning with the planar cable robot, where both forward and inverse dynamics simulations are performed. The chapter also includes simulations involving closed-loop dynamics, demonstrating their application to robot control. Case studies continue with the Stewart–Gough platform,

which serves as a further example for forward, closed-loop, and inverse dynamics simulations. These case studies illustrate how dynamic simulations can be applied to evaluate system behavior under different conditions and configurations.

A significant aspect of robot design is dynamic calibration, ensuring that the predicted motion aligns precisely with the actual robot behavior. The chapter introduces dynamic calibration methods, including linear regression models, the least squares method, and the evaluation of calibrated models. These tools are essential for improving the accuracy and reliability of robotic systems, helping to ensure precise performance in real-world applications. The chapter concludes with a set of problems designed to reinforce the concepts discussed and allow readers to practice applying dynamic simulation and calibration techniques in various contexts.

## 10.1 Introduction

Simulating robot motion is a widely used technique by designers before the actual implementation and construction of the robot. By leveraging advanced engineering software, designers can closely mimic the motion behavior of robots, allowing for thorough investigation, optimization, and evaluation of design concepts. These simulations provide insights into the

performance, feasibility, and potential issues of a robot design, enabling designers to make informed decisions and refinements prior to physical prototyping.

Two prevalent methods are commonly employed in robot motion simulation. In the first approach, the simulation relies on the dynamic formulation of the robot, with the numerical solution of the governing differential equations implemented through suitable software. This method directly incorporates the dynamic equations specific to the robot under consideration, providing a detailed and accurate representation of the robot's behavior. Given the computational complexities associated with determining the dynamic equations governing robot motion, several highly efficient engineering software programs have been developed. These programs are equipped with advanced algorithms and solvers to handle the intricate calculations required for dynamic simulations.

In the second approach, these programs simulate robot motion using only the kinematic structure and dynamic characteristics of its components, without explicitly deriving the dynamic formulation. This method is advantageous for visualizing and analyzing robot motion quickly, allowing designers to evaluate the kinematic performance and feasibility of different designs. However, it does not provide dynamic formulations to the designer, making it impractical for incorporating robot dynamics into model-based control methods. Despite the usefulness of these simulation methods, this chapter focuses on

simulation approaches grounded in the robot's dynamic model, enabling a thorough examination and analysis of the robot's dynamic behavior.

The first step in robot motion simulation involves defining the robot's motion trajectory, a process referred to as motion trajectory planning in robotics literature. This can take place either in joint space or task space. Since the dynamic equations of parallel robots are typically established in task space, it is logical for trajectory generation to occur in this space as well. The upcoming sections of this chapter will explore various methods of trajectory generation in robots, discussing their advantages, limitations, and applications.

Just as we examined both forward and inverse kinematics in robot kinematic analysis, in dynamic simulation, these processes take the form of forward and inverse dynamic simulations. Forward dynamic simulation involves computing the robot's motion given a set of actuator forces or torques, while inverse dynamic simulation determines the required actuator forces or torques to achieve a desired motion. The subsequent sections will elaborate on the characteristics of each of these methods, provide details related to dynamic simulation in robots, and explore various applications of these simulations in design optimization, control strategy development, and performance analysis.

Another crucial aspect of practical robot design is kinematic and dynamic calibration. The accuracy of a

robot's dynamic model in predicting its motion depends on the precise identification of model parameters. In robotics, this process is known as robot calibration. There are numerous calibration methods for robots, with extensive research on the subject, making it impractical to cover all cases in this brief overview. This chapter will thoroughly elaborate on implementing robot calibration using linear regression models and determining model parameters through least squares methods. Furthermore, by introducing a consistency measure, the determined model will be evaluated, ensuring that the parameters obtained are reliable and accurate. This evaluation process will also include a discussion on the validation of the calibrated model against experimental data and the exploration of potential discrepancies.

In summary, the chapter will provide a comprehensive guide to simulating robot motion using dynamic models, emphasizing the importance of accurate trajectory planning, dynamic simulation methods, and calibration techniques. This holistic approach aims to equip designers and engineers with the knowledge and tools necessary to develop high-performance robotic systems that meet the desired specifications and operate reliably in real-world scenarios.

## 10.2 Motion Planning

In this section, we will explore methods for designing and generating motion trajectories in parallel robots. To start, it's crucial to establish a precise definition of the motion path and its distinction to the trajectory. Consider the scenario where we aim to move the robot's moving platform through three-dimensional space, navigating from the initial point $\boldsymbol{\chi}_0$ to the final point $\boldsymbol{\chi}_f$. The path from the starting point to the endpoint can be defined by the function connecting these two points. Various functions such as linear, circular, or helical, among others, can be employed to traverse this path. Regardless of the chosen function, the path is expressed in terms of spatial variables $x$,$y$,$z$ and the angular variables $\theta_x, \theta_y, \theta_z$, and does not explicitly depend on time. This is in contrast to the motion trajectory, which incorporates time, and considers the speed and acceleration of movement along the desired path. In the design of robot motion trajectories, the objective is to determine the task space variables $\boldsymbol{\chi}(t)$ or the joint space variables $\boldsymbol{q}(t)$ over time.

With this differentiation, it becomes evident that, whether in the joint or task space of the robot, the initial step in designing a trajectory is to create a preferred path that ensures the robot avoids collisions with its surrounding objects. Subsequently, the temporal aspects of the motion path need to be taken into account, and the duration of the journey from the starting point to the endpoint must be determined. Numerous methods have been developed for robot path planning, particularly in

mobile robots, to circumvent obstacles [214, 215, 216]. Due to the breadth of these methods, we avoid detailing them here.

In the context of motion planning for industrial robots, it is evident that utilizing a teaching pendant to establish the robot's motion points serves to simplify the path generation process. The teaching pendant allows for the manual movement of the robot moving platform to a specified pose while recording its position and orientation. Consequently, based on the desired path for the robot's moving platform, the key motion points are identified, and the corresponding active joint information is documented. These recorded points then facilitate trajectory planning, wherein suitable time functions can be determined to smoothly guide the robot through these specified points [217].

For this purpose, let's consider the scenario where we aim to transition from the initial point $\boldsymbol{\chi}(t_0) = \boldsymbol{\chi}_0$ to the final point $\boldsymbol{\chi}(t_f) = \boldsymbol{\chi}_f$ within the time interval $[t_0, t_f]$. Additionally, let's assume that the velocities at the initial and final points are given as $\dot{\boldsymbol{\chi}}(t_0) = \dot{\boldsymbol{\chi}}_0$ and $\dot{\boldsymbol{\chi}}(t_f) = \dot{\boldsymbol{\chi}}_f$, respectively. This implies that designing the path in the robot involves identifying a suitable time function $\boldsymbol{\chi}(t)$ that satisfies these specified boundary conditions. Through the application of interpolation methods, it is possible to determine various functions that meet these criteria. In the subsequent section, we will concentrate on and explore this issue using three distinct methods.

### 10.2.1 Cubic Polynomials

Assume the initial and final conditions of the trajectory of the robot's moving platform are given.

$$\boldsymbol{\chi}(t_0) = \boldsymbol{\chi}_0 \quad \boldsymbol{\chi}(t_f) = \boldsymbol{\chi}_f$$
$$\dot{\boldsymbol{\chi}}(t_0) = \dot{\boldsymbol{\chi}}_0 \quad \dot{\boldsymbol{\chi}}(t_f) = \dot{\boldsymbol{\chi}}_f$$

In the following equation, the index $i$ is used to introduce the motion variable for each of the moving platform motion components.

$$\boldsymbol{\chi}_0 = \begin{bmatrix} \mathscr{X}_{1_0} & \mathscr{X}_{2_0} & \cdots & \mathscr{X}_{n_0} \end{bmatrix} \quad \boldsymbol{\chi}_f = \begin{bmatrix} \mathscr{X}_{1_f} & \mathscr{X}_{2_f} & \cdots & \mathscr{X}_{n_f} \end{bmatrix}$$
$$\dot{\boldsymbol{\chi}}_0 = \begin{bmatrix} \dot{\mathscr{X}}_{1_0} & \dot{\mathscr{X}}_{2_0} & \cdots & \dot{\mathscr{X}}_{n_0} \end{bmatrix} \quad \dot{\boldsymbol{\chi}}_f = \begin{bmatrix} \dot{\mathscr{X}}_{1_f} & \dot{\mathscr{X}}_{2_f} & \cdots & \dot{\mathscr{X}}_{n_f} \end{bmatrix}$$

By using cubic polynomials as a function of time, it is possible to design a continuous function for traversing this trajectory for each of the motion components, denoted as $\mathscr{X}_i(t)$.

$$\mathscr{X}_i(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3 \tag{10.1}$$

$$\dot{\mathscr{X}}_i(t) = p_1 + 2p_2 t + 3p_3 t^2 \tag{10.2}$$

In order to satisfy the motion boundary conditions at the initial and final times, four algebraic relations are obtained in determining the four associated parameters $p_i$ for the desired cubic polynomial. These relations can be rewritten in the form of the following matrix equation.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} \mathscr{X}_{i_0} \\ \mathscr{X}_{i_f} \\ \dot{\mathscr{X}}_{i_0} \\ \dot{\mathscr{X}}_{i_f} \end{bmatrix} \tag{10.3}$$

If $t_0 \neq t_f$, the matrix in 10.3 is invertible, and by determining its inverse, the coefficients of the cubic polynomial, in general, can be obtained.

$$\begin{aligned} p_0 &= \mathscr{X}_{i_0} \quad , \quad p_1 = \dot{\mathscr{X}}_{i_0} \\ p_2 &= \frac{3}{t_f^2}\left(\mathscr{X}_{i_f} - \mathscr{X}_{i_0}\right) - \frac{1}{t_f}\left(2\dot{\mathscr{X}}_{i_0} + \dot{\mathscr{X}}_{i_f}\right) \\ p_3 &= -\frac{2}{t_f^3}\left(\mathscr{X}_{i_f} - \mathscr{X}_{i_0}\right) + \frac{1}{t_f^2}\left(\dot{\mathscr{X}}_{i_0} + \dot{\mathscr{X}}_{i_f}\right) \end{aligned} \tag{10.4}$$

In many applications, it is common to initiate the robot's motion at the starting point with zero velocity and come to a gradual stop at the endpoint. In this case, $\dot{\mathscr{X}}_{i_0} = \dot{\mathscr{X}}_{i_f} = 0$,

and the cubic polynomial trajectory simplifies to the following equation.

$$\mathscr{X}_i(t) = \mathscr{X}_{i_0} + \frac{3}{t_f^2}\big(\mathscr{X}_{i_f} - \mathscr{X}_{i_0}\big)t^2 - \frac{2}{t_f^3}\big(\mathscr{X}_{i_f} - \mathscr{X}_{i_0}\big)t^3 \tag{10.5}$$

If this motion trajectory is utilized in the robot's moving platform, a continuous and smooth motion will be achieved in each component of the translational and rotational motion of the moving platform, as it moves from the initial point to the endpoint. Consequently, the robot's active joint motions will also be smooth and continuous.

**Example 10.1 (Cubic Trajectory Planning)** ⏎ *For the planar 4RPR cable robot, the objective is for the robot's moving platform to achieve the final state* $\boldsymbol{\chi}_f = [100,\, -50,\, 45^\circ]^T$ *within five minutes, starting from the initial state* $\boldsymbol{\chi}_0 = [10,\, -10,\, 0]^T$. *Develop the desired motion trajectory using a cubic polynomial, and plot the position, velocity, and acceleration of the moving platform over time.*

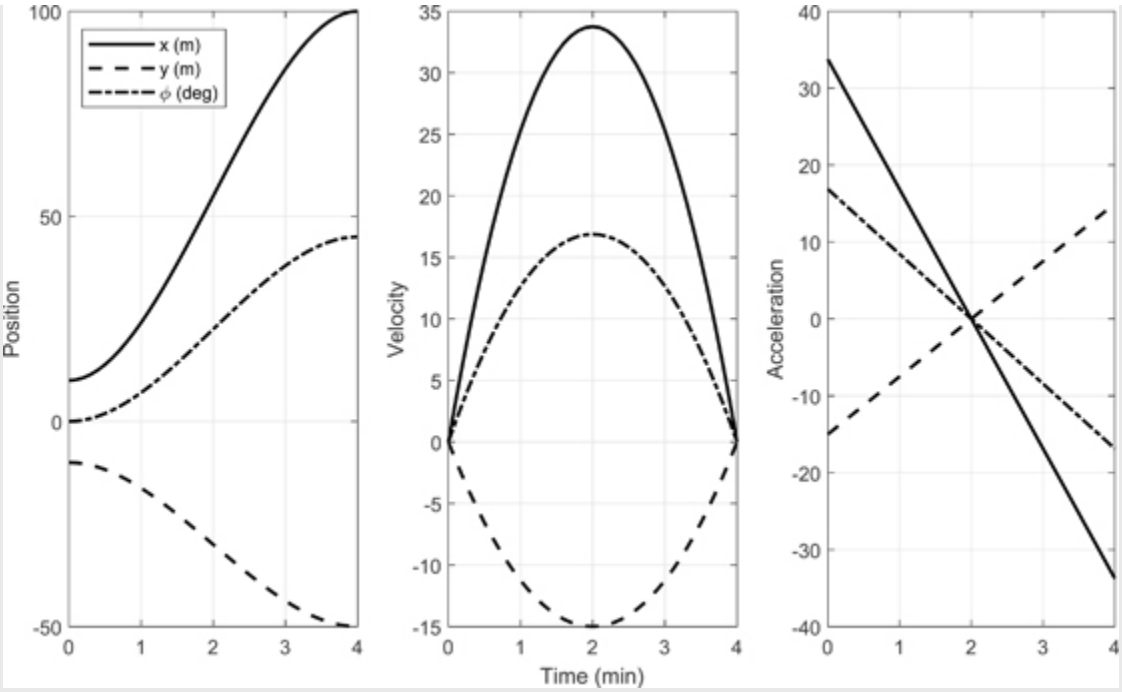
**Solution:** Using the initial and final states of the motion, the trajectories of motion in $x_G$, $y_G$, and $\phi$ variables are obtained using the Equation 10.5.

$$
\begin{aligned}
x_{_G} &= 10 + 10.8t^2 - 1.44t^3 \\
y_{_G} &= -10 - 4.8t^2 + 0.64t^3 \\
\phi &= 5.4t^2 - 0.72t^3
\end{aligned}
$$

To determine the relationships between the velocity and acceleration of the joints, differentiate these equations with respect to time:

$$
\begin{aligned}
\dot{x}_{_G}(t) = 21.6t - 4.32t^2 \quad &\rightarrow \ddot{x}_{_G}(t) = 21.6 - 8.64t \\
\dot{x}_{_G}(t) = -9.6t + 1.92t^2 \quad &\rightarrow \ddot{y}_{_G}(t) = -9.6 + 3.84t \\
\dot{\phi}(t) = -10.8t - 2.16^2 \quad &\rightarrow \ddot{\phi}(t) = 10.8 - 4.32t
\end{aligned}
$$

The motion trajectories are illustrated in Figure 10.1. As observed in this figure, the motion of the moving platform in the $x$ direction, represented by a solid line, in the $y$ direction by a dashed line, and the rotational motion $\phi$, shown by a dash-dotted line, smoothly transitions from their initial desired values to their final desired angles over the course of four minutes. The translational motion in the $x$ direction is positive, ranging from 10 to 100 meters, the motion in the $y$ direction is negative, ranging from $-10$ to $-50$ meters, while the orientation of the moving platform is positive, varying from zero to 45 degrees smoothly over time based on the cubic polynomial function.




▶ Long Description for Figure 10.1

**FIGURE 10.1**

The designed trajectory of motion for 4RPR moving platform $\boldsymbol{\chi}$, along with the velocity and acceleration. ⏎

The velocity in the $x$ direction, depicted by the solid line in the middle graph, is also smooth and consistently positive, forming a parabolic function with respect to time. Its maximum value occurs at 2 minutes, reaching 33.7 meters per minute. Meanwhile, the velocity in the $y$ direction, illustrated by the dashed line, is consistently negative, with its minimum value occurring at 2 minutes and reaching $-15$ meters per minute. The angular velocity of the moving platform $\omega$, depicted by a dash-dotted line, is also always positive, with its maximum value about 16.8 occurring at 2 minutes. All the

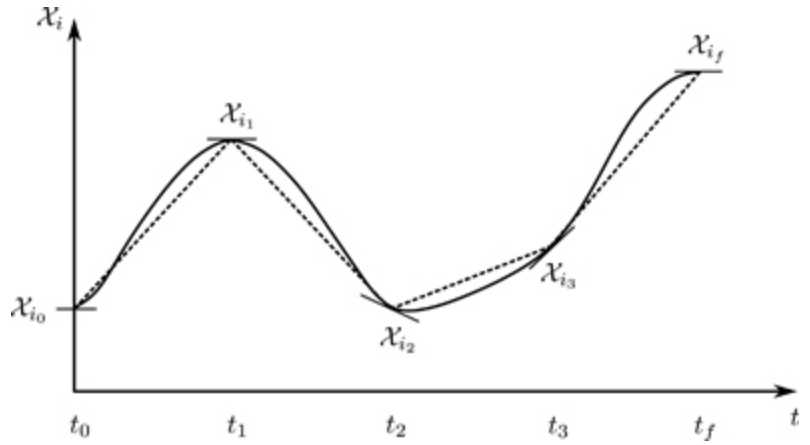
acceleration trajectories, shown in the rightmost graph, are linear with the first one reaching a maximum value of 33.75 meters per minute squared, the second one having a minimum value of $-15$ meters per minute square, and the angular acceleration starting with the maximum value of 16.875 radians per minutes square at the initial time.

### 10.2.2 Cubic Polynomials with Via Points

So far, the trajectory design from the initial point to the final point has been studied using cubic polynomials. However, in the trajectory design for robot motion, passing through multiple intermediate points is often desired. In the trajectory design using intermediate points, it is usually not necessary to consider zero velocity at via points. If non-zero velocity is desired at intermediate points, the general trajectory equation with non-zero velocity, given in 10.4, can be used, requiring the specification of the velocity at intermediate points. Although the user can have complete freedom in choosing the velocity at intermediate points and determine them at their discretion, sometimes, especially in trajectories with multiple via points, this freedom can be cumbersome for the user, and it is convenient to automatically make these selections.

For automatic selection of the trajectory velocity at via points, the average speed between two consecutive points on the motion path, as illustrated in Figure 10.2, can be

used. To do this, connect the path points with straight lines and choose the average speed between two via points as the appropriate velocity for intermediate points. This simple method facilitates the user's selection of velocity at intermediate points and allows for its automatic execution.




▶ Long Description for Figure 10.2

**FIGURE 10.2**
Determination of the velocity at via points in robot trajectory planning. ⏎

For a better understanding of the subject, let's assume, according to Figure 10.3, we want to design a cubic motion trajectory for a robot's motion with an intermediate point. If the initial and final conditions are given as before, and the motion state at the intermediate point, denoted by index $v$, the velocity at the intermediate point is automatically determined based on the following equation.

$$\dot{\mathscr{X}}_v = \dot{\mathscr{X}}_v = \frac{1}{2}\left(\frac{\mathscr{X}_v - \mathscr{X}_0}{t_v - t_0} + \frac{\mathscr{X}_f - \mathscr{X}_v}{t_f - t_v}\right) \tag{10.6}$$

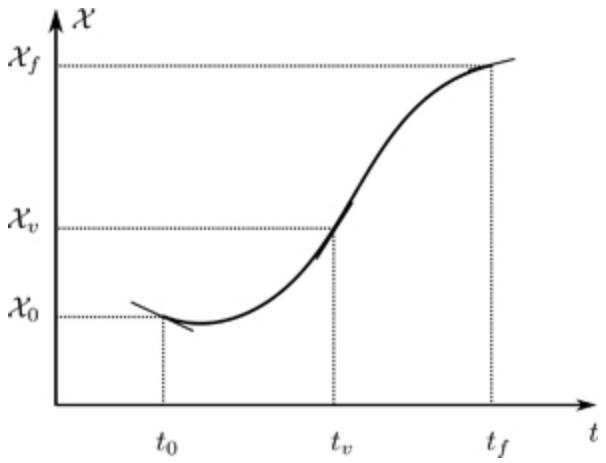



**FIGURE 10.3**
A segment of the trajectory with one via point. ⏎

In this way, the initial, intermediate, and final conditions of the motion trajectory are given as follows.

$$\begin{aligned} \mathscr{X}(t_0) &= \mathscr{X}_0 \quad \mathscr{X}(t_v) = \mathscr{X}_v \mathscr{X}(t_f) = \mathscr{X}_f \\ \dot{\mathscr{X}}(t_0) &= \dot{\mathscr{X}}_0 \quad \dot{\mathscr{X}}(t_v) = \dot{\mathscr{X}}_v \dot{\mathscr{X}}(t_f) = \dot{\mathscr{X}}_f \end{aligned}$$

With this description, the trajectory consists of two segments described by a piecewise function composed of two cubic polynomials.

$$\mathscr{X}(t) = \begin{cases} p_{1_0} + p_{1_1} t + p_{1_2} t^2 + p_{1_3} t^3 & t_0 < t \leq t_v \\ p_{2_0} + p_{2_1} t + p_{2_2} t^2 + p_{2_3} t^3 & t_v < t \leq t_f \end{cases}$$

The parameters of each polynomial in this equation can be obtained based on the general relation given in 10.4. In these calculations, it is sufficient to substitute the initial and final conditions as well as the time of traversing the trajectory for each segment into this equation. Similarly, if we use multiple via points in trajectory design, we just need to first calculate the velocities at these via points and then determine the parameters of the polynomial for each segment of the trajectory.

**Example 10.2 (Cubic Trajectory Planning with Via Point)** *For the motion in the x direction of the planar 4UPU manipulator, as outlined in Example 10.1, design a trajectory that transitions from an initial position of $x_0 = 10$ meters to a final position of $x_f = 100$ meters over a duration of four minutes. This trajectory must also pass through a via point at $x_v = 150$ meters at the two-minute mark. Given that the velocity at the start and end of the trajectory is zero, determine the appropriate velocity at the via point. Subsequently, define the motion using a cubic polynomial trajectory and plot the resulting trajectory over time.*

**Solution:** Initially, we calculate an appropriate velocity at the via point using Equation 10.6.

$$\dot{x}_v = \frac{1}{2}\left(\frac{150-10}{2} + \frac{100-150}{2}\right) = 45\ m/s$$

Using the initial, midpoint and final states of the motion, the trajectories of motion in *x* direction is obtained using the Equation 10.4.

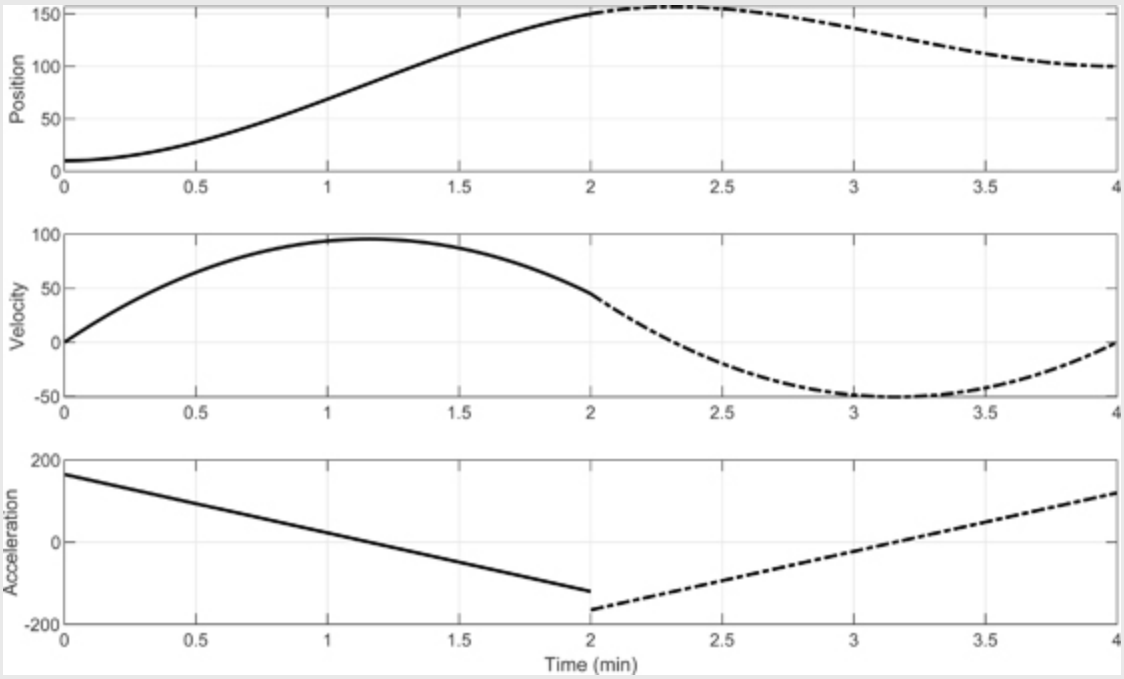
$$x_{_G}(t) = \begin{cases} 10 + 82.5t^2 - 23.75t^3 & 0 < t \leq 2 \\ 150 + 45t - 82.5t^2 + 23.75t^3 & 2 < t \leq 4 \end{cases}$$

By differentiating this equation, we can derive the relationships for velocity and acceleration as follows:

$$\begin{aligned} \dot{x}_{_G} &= \begin{cases} 165t - 71.25t^2 & 0 < t \leq 2 \\ 45 - 165t + 71.25t^2 & 2 < t \leq 4 \end{cases} \\ \ddot{x}_{_G} &= \begin{cases} 165 - 142t & 0 < t \leq 2 \\ -165 + 142t & 2 < t \leq 4 \end{cases} \end{aligned}$$

The motion path and its first and second derivatives are depicted in Figure 10.4. The top graph illustrates the smooth trajectory curve, tracing the path from 10 to 100 meters and passing through an intermediate point at 150 meters. The middle graph shows the velocity profile, which is a continuous function, with the motion

beginning and ending at zero velocity, peaking at 45 meters per minute at the 2-minute mark.




▶ Long Description for Figure 10.4

**FIGURE 10.4**
Cubic trajectory planning with one via point. ⏎

The acceleration profile in the bottom graph reveals linear acceleration across both segments of the path. However, it also shows that the acceleration is not continuous, with discontinuities occurring at the initial, intermediate, and final points. To mitigate these discontinuities, the use of higher-degree polynomials, as will be discussed in the next section, is recommended.

### 10.2.3 Quintic Polynomials

The use of a cubic polynomial is the simplest smooth function that can be chosen for robot motion trajectory planning. However, as demonstrated in the previous section, with only four adjustable parameters in this polynomial, it can only ensure interpolation with four conditions at the initial and final points. Therefore, continuity in acceleration and its derivatives is not achievable with this polynomial.

If we aim to satisfy the necessary continuity conditions in acceleration, at least a fifth-degree polynomial must be used. Furthermore, if continuity in the derivative of acceleration (jerk) is required, a polynomial of at least the seventh degree should be used. In this section, we explore how to design a trajectory based on a quintic polynomial, while similarly, this concept can be extended to higher-degree polynomials.

Suppose the initial and final conditions for the motion trajectory, velocity, and acceleration of a robot joint are given with the following values.

$$\begin{aligned} \mathscr{X}(t_0) &= \mathscr{X}_0 \qquad & \mathscr{X}(t_f) &= \mathscr{X}_f \\ \dot{\mathscr{X}}(t_0) &= \dot{\mathscr{X}}_0 \qquad & \dot{\mathscr{X}}(t_f) &= \dot{\mathscr{X}}_f \\ \ddot{\mathscr{X}}(t_0) &= \ddot{\mathscr{X}}_0 \qquad & \ddot{\mathscr{X}}(t_f) &= \ddot{\mathscr{X}}_f \end{aligned}$$

And we want to design a continuous function over time using fifth-degree polynomials for traversing this motion trajectory.

$$
\begin{aligned}
\mathscr{X}_i(t) &= p_0 + p_1 t + p_2 t^2 + p_3 t^3 + p_4 t^4 + p_5 t^5 \\
\dot{\mathscr{X}}_i(t) &= p_1 + 2p_2 t + 3p_3 t^2 + 4p_4 t^3 + 5p_5 t^4 \\
\ddot{\mathscr{X}}_i(t) &= 2p_2 t + 6p_3 t + 12p_4 t^2 + 20p_5 t^3
\end{aligned}
\tag{10.7}
$$

In order to satisfy the motion, velocity, and acceleration conditions at the initial and final times, six algebraic equations are derived to determine the six parameters $p_i$ of the quintic polynomial. These equations can be rewritten in matrix form as follows:

$$
\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\
1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\
0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\
0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\
0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\
0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3
\end{bmatrix}
\begin{bmatrix}
p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5
\end{bmatrix}
=
\begin{bmatrix}
\mathscr{X}_0 \\ \mathscr{X}_f \\ \dot{\mathscr{X}}_0 \\ \dot{\mathscr{X}}_f \\ \ddot{\mathscr{X}}_0 \\ \ddot{\mathscr{X}}_f
\end{bmatrix}
\tag{10.8}
$$

If $t_0 \neq t_f$, the matrix in Equation 10.8 is invertible, and the coefficients of the quintic polynomial can be obtained in its general form, by using its inverse.

$$
\begin{aligned}
p_0 &= \mathscr{X}_0, \quad p_1 = \omega_0, \quad p_2 = \frac{1}{2}\alpha_0 \\
p_3 &= \frac{10}{t_f^3}(\mathscr{X}_f - \mathscr{X}_0) - \frac{2}{t_f^2}(2\dot{\mathscr{X}}_f + 3\dot{\mathscr{X}}_0) + \frac{1}{2t_f}\left(3\ddot{\mathscr{X}}_f - \ddot{\mathscr{X}}_0\right) \\
p_4 &= -\frac{15}{t_f^4}(\mathscr{X}_f - \mathscr{X}_0) + \frac{1}{t_f^3}(7\dot{\mathscr{X}}_f + 8\dot{\mathscr{X}}_0) - \frac{1}{2t_f^2}\left(3\ddot{\mathscr{X}}_f - 2\right. \\
p_5 &= \frac{6}{t_f^5}(\mathscr{X}_f - \mathscr{X}_0) - \frac{3}{t_f^4}(\dot{\mathscr{X}}_f + \dot{\mathscr{X}}_0) + \frac{1}{2t_f^3}\left(\ddot{\mathscr{X}}_f - \ddot{\mathscr{X}}_0\right)
\end{aligned}
\tag{10.9}
$$

Since in many applications, the robot's motion starts and ends with zero velocity and acceleration, meaning $\dot{\mathscr{X}}_0 = \dot{\mathscr{X}}_f = 0$ and $\ddot{\mathscr{X}}_0 = \ddot{\mathscr{X}}_f = 0$, the quintic polynomial for the motion trajectory simplifies as follows:

$$
\mathscr{X}(t) = \mathscr{X}_0 + \frac{10}{t_f^3}(\mathscr{X}_f - \mathscr{X}_0)t^3 - \frac{15}{t_f^4}(\mathscr{X}_f - \mathscr{X}_0)t^4 + \frac{6}{t_f^5}(\mathscr{X}_f
\tag{10.10}
$$

If this motion trajectory is used in the robot motion components, continuous and smooth motion will be achieved at the moving platform, leading to continuous and differentiable motion of each limb active variable.

As mentioned, in trajectory design using via points, it is not necessary to consider zero velocity and acceleration at

these points. To find the desired quintic polynomial trajectory, it is sufficient to use the general relationship for trajectory design with non-zero velocity and acceleration, given in Equation 10.9. To use this equation, the velocity and acceleration at the intermediate points need to be specified. Similar to the method of automatically determining the velocity of the motion trajectory at via points, the average velocity and acceleration between two consecutive points on the motion trajectory can be used.

**Example 10.3 (Quintic Trajectory Planning)** *For the planar 4R<u>P</u>R cable robot, the objective is for the robot's moving platform to achieve the final state* $\boldsymbol{\chi}_f = [100,\, -50,\, 45^\circ]^T$ *within four minutes, starting from the initial state* $\boldsymbol{\chi}_0 = [10,\, -10,\, 0]^T$. *Develop the desired motion trajectory using a quintic polynomial, and plot the position, velocity, and acceleration of the moving platform over time, and compare it with the motion trajectory based on cubic polynomials.*

**Solution:** With the initial and final joint given, the motion trajectory of the joints can be obtained using the Equation 10.10.

$$
\begin{aligned}
x_{_G}(t) &= 10 + 14.06t^3 - 5.27t^4 + 0.53t^5 \\
y_{_G}(t) &= -10 - 6.25t^3 + 2.34t^4 - 0.23t^5 \\
\phi(t) &= 7.03t^3 - 2.64t^4 + 0.26t^5
\end{aligned}
$$

By taking derivatives of these equations, we can determine the relationships for the angular velocity and acceleration of the joints:

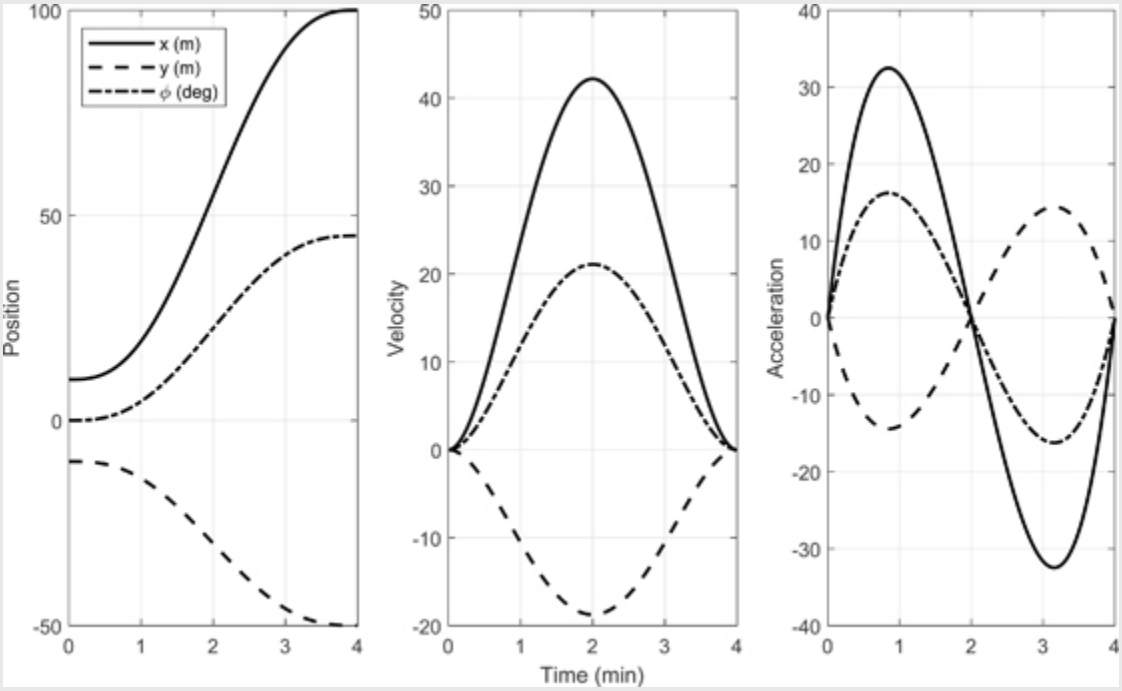
$$\begin{aligned}\dot{x}_G(t) &= 42.19t^2 - 21.09t^3 + 2.64t^4 \\ \dot{y}_G(t) &= -18.75t^2 + 9.375t^3 - 1.17t^4 \\ \dot{\phi}(t) &= 21.09t^2 - 10.55t^3 + 1.32t^4\end{aligned}$$

and,

$$\begin{aligned}\ddot{x}_G(t) &= 84.375t - 63.28t^2 + 105.5t^3 \\ \ddot{y}_G(t) &= -37.50t + 28.125t^2 - 46.874t^3 \\ \ddot{\phi}(t) &= 42.19t - 31.64t^2 + 52.73t^3\end{aligned}$$

The motion trajectory and its first and second derivatives are illustrated in Figure 10.5. As depicted, the motion of the moving platform in the $x$ direction is represented by a solid line, the $y$ direction by a dashed line, and the rotational motion $\phi$ by a dash-dotted line. These motions smoothly transition from their initial desired values to their final desired states over the course of four minutes. The translational motion in the $x$ direction is positive, increasing from 10 to 100 meters, while the motion in the $y$ direction is negative, decreasing from $-10$ to $-50$ meters. The orientation of the moving platform, $\phi$, progresses positively from zero

to 45 degrees, all following a quintic polynomial function.




▶ Long Description for Figure 10.5

**FIGURE 10.5**
The designed motion trajectory of the joints using quintic polynomials and their angular velocity and acceleration. ⏎

The velocity profile in the $x$ direction, shown in the middle graph with a solid line, is continuous, consistently positive, and follows a fourth-degree polynomial function over time, reaching its peak at 2 minutes with a speed of 42.19 meters per second. The velocity in the $y$ direction, represented by the dashed line, remains negative, with a minimum velocity of $-18.75$ meters per minute. The angular velocity $\omega$ of the

moving platform, depicted by the dash-dotted line, is also always positive, peaking at around 21.09 at the two-minute mark.

The acceleration trajectories, shown in the rightmost graph, are quadratic. The acceleration in the $x$ direction reaches a maximum value of 32.47 meters per minute squared, while the $y$ direction reaches a minimum of $-14.43$ meters per minute squared. The angular acceleration starts at a maximum of 16.23 radians per minute squared. Compared to the motion described in Example 10.1, this trajectory begins more gradually and smoothly decelerates to rest, achieving zero acceleration at the end. The slope of this motion is steeper, leading to a higher maximum speed.

### 10.2.4 Linear Trajectory with Parabolic Blends

A commonly employed technique in trajectory planning involves the utilization of a blended trajectory. The simplest way to move from one point to another is linear interpolation. Although it is possible to use the maximum possible speed of the robot moving platform along this trajectory, due to the choice of a first-degree polynomial trajectory, the motion speed at the beginning and end of the trajectory will be discontinuous and undesirable. On the other hand, if trajectories of third and higher-degree polynomials are used, the motion acceleration will be a linear or higher-order function of time, varying along the

trajectory. Thus, the maximum possible applied wrench component, proportional to the maximum motion acceleration, cannot be used along the path, and the trajectory cannot be traversed in the shortest possible time.

To generate a simple trajectory and utilize the maximum speed and acceleration of the moving platform, a combination of a linear trajectory over time and smoothing the angular motion at the beginning and end of the trajectory using parabolic functions is used. As seen in Figure 10.6, the start of the motion is blended, and from time zero to $t_b$, positive constant acceleration is designed based on a parabolic polynomial. Then, along the trajectory and just before the end point a linear trajectory is used with constant speed. Finally, from time $t_f - t_b$ to $t_f$, a decelerating blended trajectory with constant deceleration is used to smoothly come to rest at zero speed.

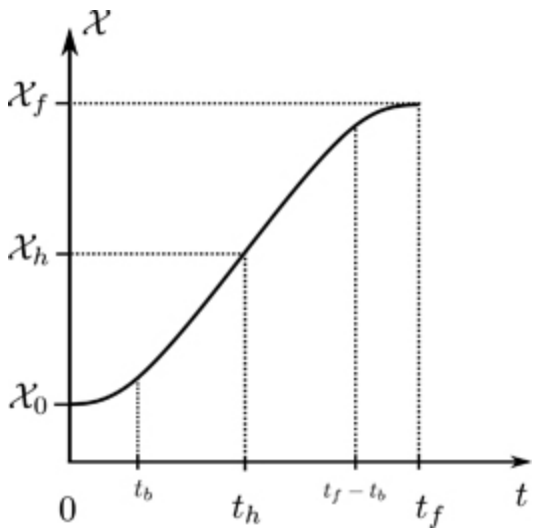



**FIGURE 10.6**

Details of the linear trajectory planning with parabolic blends. ⏎

In designing this trajectory, assume that the acceleration time at the beginning of the trajectory lasting until $t_b$, is equal to the deceleration time at the end of the trajectory. Furthermore, assume that the motion speed at the beginning and end of the trajectory is zero, $\dot{\mathscr{X}}_0 = \dot{\mathscr{X}}_f = 0$. With this description, the following piecewise function can be used in trajectory planning.

$$\mathscr{X}(t) = \begin{cases} \mathscr{X}_0 + \frac{1}{2}\ddot{\mathscr{X}} t^2 \\ \mathscr{X}(t_b) + \dot{\mathscr{X}}(t - t_b) \\ \mathscr{X}(t_f - t_b) + \dot{\mathscr{X}}(t - t_f + t_b) - \frac{1}{2}\ddot{\mathscr{X}}(t - t_f + t_b)^2 \end{cases} \tag{10.11}$$

Where $\dot{\mathscr{X}}$ is the speed of movement along the linear trajectory and $\ddot{\mathscr{X}}$ is the acceleration along the blended trajectory. In order to create a continuous and smooth motion at time $t_b$, the speeds of the two blended and linear trajectory segments at the blending time $t_b$ must be equal.

$$\dot{\mathscr{X}} = \ddot{\mathscr{X}}\, t_b \quad \rightarrow \quad t_b = \frac{\dot{\mathscr{X}}}{\ddot{\mathscr{X}}} \tag{10.12}$$

The trajectory planning in this method is possible in two ways. In the first case, suppose that in addition to $\mathscr{X}_0$ and $\mathscr{X}_f$, the time to traverse the trajectory $t_f$ is also given in advance. In this case, one can consider the speed and acceleration limits in the joint actuators, and if the maximum speed and acceleration of the moving platform are higher than the required limits, the trajectory can be traversed within the given time. To determine the speed limit of the moving platform, it is sufficient to assume that the motion is entirely linear. In this case, the necessary speed limit for traversing the trajectory is obtained as follows

$$\Omega_{max} \quad > \quad \frac{\mathscr{X}_f - \mathscr{X}_0}{t_f} \tag{10.13}$$

In this equation, the minimum speed required to traverse the trajectory at a constant speed is determined, and the necessary speed limit for the moving platform is determined based on that. To determine the necessary acceleration limit of the moving platform, assume that the motion is entirely blended over half of the trajectory. In this case:

$$\ddot{\mathscr{X}}_{max} > 2\frac{(\mathscr{X}_f - \mathscr{X}_0)/2}{t_f^2/4} = 4\frac{\mathscr{X}_f - \mathscr{X}_0}{t_f^2}$$

(10.14)

In this equation, the minimum acceleration required to traverse half of the trajectory at a constant acceleration is determined, and the necessary acceleration limit for the moving platform is calculated based on that. If these conditions are satisfied, the piecewise trajectory function 10.11 can be designed as follows. By determining the blending time $t_b$ from the Equation 10.12, $\mathscr{X}(t_b)$ can be determined as follows.

$$\mathscr{X}(t_b) = q_0 + \frac{1}{2}\dot{m}Xt_b$$

Similarly, $\mathscr{X}(t_f - t_b)$ can be obtained in a similar manner.

$$\begin{aligned}\mathscr{X}(t_f - t_b) &= \mathscr{X}(t_b) + \dot{\mathscr{X}}(t_f - t_b) \\ &= \mathscr{X}_0 + \left(t_f - \frac{3}{2}t_b\right)\dot{\mathscr{X}}\end{aligned}$$

Substitute these relations into the Equation 10.11.

$$\mathscr{X}(t) = \begin{cases} \mathscr{X}_0 + \frac{1}{2}\ddot{\mathscr{X}}t^2 & 0 \leq t \\ \mathscr{X}_0 + \dot{\mathscr{X}}\left(t - \frac{1}{2}t_b\right) & t_b \leq t \leq \\ \mathscr{X}_0 + \dot{\mathscr{X}}\left(t - \frac{1}{2}t_b\right) - \frac{1}{2}\ddot{\mathscr{X}}(t - t_f + t_b)^2 & t_f - t_b \end{cases}$$

(10.15)

Thus, the velocity along the trajectory can also be determined from the following piecewise function.

$$\dot{\mathscr{X}}(t) = \begin{cases} \ddot{\mathscr{X}} t & 0 \leq t \leq t_b \\ \dot{\mathscr{X}} & t_b \leq t \leq t_f - t_b \\ \dot{\mathscr{X}} - \ddot{\mathscr{X}}(t - t_f + t_b) & t_f - t_b \leq t \leq t_f \end{cases} \tag{10.16}$$

In these equations, $t_b$, $\dot{\mathscr{X}}$, $\ddot{\mathscr{X}}$ must be chosen in such a way that, while satisfying the maximum speed and acceleration conditions, the trajectory is traversed within the time $t_f$. To determine these quantities appropriately, first express the velocity based on the length of trajectory and determine the blending time $t_b$ based on the movement to half of the trajectory.

$$\mathscr{X}_h = \frac{1}{2}(\mathscr{X}_f + \mathscr{X}_0) = q_0 + \dot{\mathscr{X}}(t_h - \frac{1}{2}t_b) \rightarrow \dot{\mathscr{X}} = \frac{\mathscr{X}_f - \mathscr{X}_0}{t_f - t_b} \tag{10.17}$$

If $t_b$ is given, $\dot{\mathscr{X}}$ can be easily determined based on this relationship. If the moving platform's speed and acceleration limits are provided, an appropriate option is to set the ratio of speed to maximum acceleration.

$$t_b = \frac{\dot{\mathscr{X}}_{max}}{\ddot{\mathscr{X}}_{max}}$$

(10.18)

However, keep in mind that this is not the only option and various values can be selected for $t_b$. Nevertheless, it is crucial to ensure the condition $t_b \leq \frac{1}{2} t_f$ is satisfied when choosing an alternative. By selecting $t_b$, the motion speed $\dot{\mathscr{X}}$ is obtained from the relationship in 10.17, and the motion acceleration is chosen as follows.

$$\ddot{\mathscr{X}} = \frac{\dot{\mathscr{X}}}{t_b}$$

(10.19)

By determining these values, the piecewise function in 10.15 can easily be used in trajectory planning, and the function in 10.16 can be utilized to determine the speed profile.

The second method of trajectory planning using the linear trajectory with parabolic blends is when the time to traverse the trajectory is not specified, and we want to determine the minimum time to traverse the trajectory based on the maximum speed and acceleration constraints of the actuator. In this case, the motion speed and

acceleration are set equal to their possible maximum values in the moving platform.

$$\dot{\mathscr{X}} = \dot{\mathscr{X}}_{max} \quad , \quad \ddot{\mathscr{X}} = \ddot{\mathscr{X}}_{max}$$

The blending time can be obtained using the Equation 10.12.

$$t_b = \frac{\dot{\mathscr{X}}}{\ddot{\mathscr{X}}}$$

Using these values, the optimum time to cover half of the trajectory, $t_h = t_f/2$, is calculated with the following equation.

$$\frac{1}{2}(\mathscr{X}_f + \mathscr{X}_0) = \mathscr{X}_0 + \dot{\mathscr{X}}\left(t_h - \frac{\dot{\mathscr{X}}}{2\ddot{\mathscr{X}}}\right)$$

By this means, the optimal time to cover the entire trajectory, $t_f$, is determined.

$$t_f = \frac{1}{\dot{\mathscr{X}}}(\mathscr{X}_f - \mathscr{X}_0) + \frac{\dot{\mathscr{X}}}{\ddot{\mathscr{X}}} \tag{10.20}$$

By determining these values, one can easily use the piecewise function 10.15 for trajectory planning and the function 10.16 to determine the trajectory velocity.

**Example 10.4 (Linear Trajectory with Parabolic Blends)** *For the planar 4R<u>P</u>R cable robot, the aim is for the robot's moving platform to reach the final state* $\boldsymbol{\chi}_f = [100,\, -50,\, 45^\circ]^T$ *within four minutes, starting from the initial state* $\boldsymbol{\chi}_0 = [10,\, -10,\, 0]^T$. *Given that the maximum speed of motion is 30 meters per minute and the maximum acceleration is 40 meters per square minute, design and plot the desired joint motion trajectory using linear paths with parabolic blends by selecting a suitable time* $t_b$.

**Solution:** First, considering the time required to traverse the trajectory and the speed and acceleration limits for the joints, we assess the feasibility of designing the trajectory according to conditions 10.13 and 10.14. Since the range of motion is greater in the $x$ direction, it is sufficient to verify these conditions for this motion.

$$\begin{aligned} \Omega_{max} &> \frac{100 - 10}{4} = 22.5 \\ \alpha_{max} &> 4 \cdot \frac{100 - 10}{16} = 22.5 \end{aligned}$$

As the speed and acceleration limits of the joint actuators are higher than the necessary values, it is feasible to design the trajectory within four minutes. Now, using Equation 10.12, calculate the blending time.

$$t_b = \frac{30}{40} = 0.75\ (min)$$

With these values and the initial and final positions, the trajectories are derived using Equation 10.15.

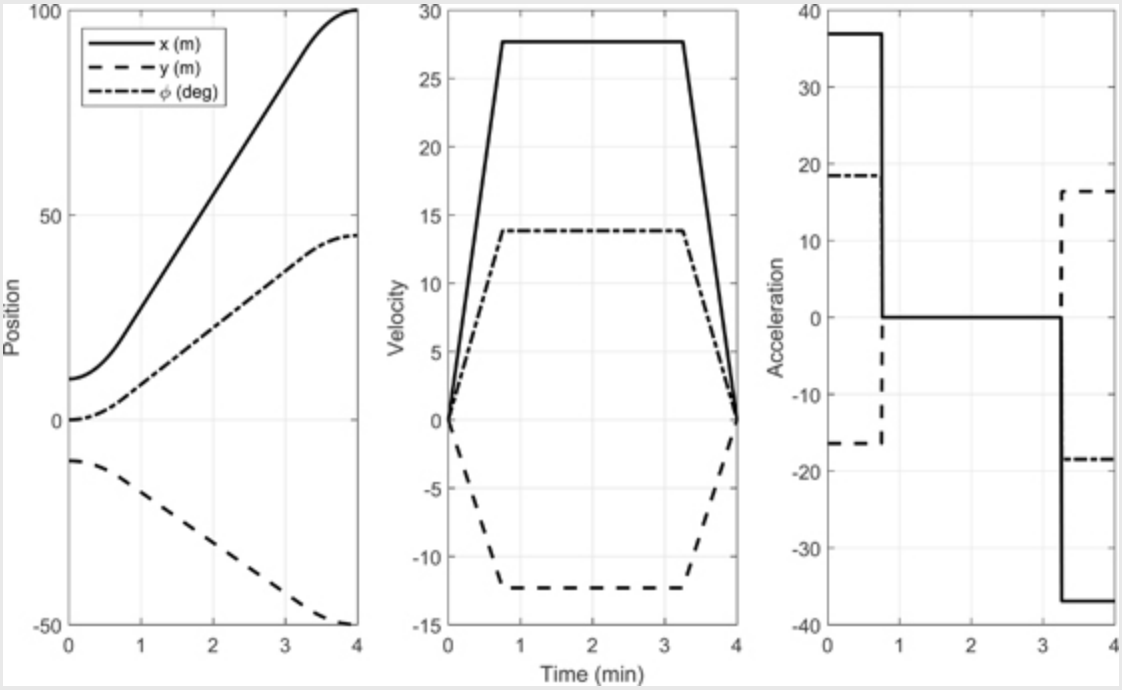
$$
\begin{aligned}
x_G(t) &= \begin{cases} 10 + 18.46t^2 & \\ 10 + 27.69(t - 0.375) & 0. \\ 10 + 27.69(t - 0.375) - 18.46(t - 3.25)^2 & \end{cases} \\
y_G(t) &= \begin{cases} -10 - 8.21t^2 & \\ -10 - 12.31\,(t - 0.375) & ( \\ -10 - 12.31\,(t - 0.375) + 8.21(t - 3.25)^2 & \end{cases} \\
\phi(t) &= \begin{cases} 9.23t^2 & 0 \le t \\ 13.85\,(t - 0.375) & 0.75 \le \\ 13.85\,(t - 0.375) - 9.23(t - 3.25)^2 & 3.25 \le \end{cases}
\end{aligned}
$$

The velocity of motion is calculated using Equation 10.16.

$$
\begin{aligned}
\dot{x}_G(t) &= \begin{cases} 36.92t & 0 \le t \le 0.75 \\ 27.69 & 0.75 \le t \le 3.25 \\ 27.69 - 36.92(t - 3.25) & 3.25 \le t \le 4 \end{cases} \\
\dot{x}_G(t) &= \begin{cases} -16.41t & 0 \le t \le 0.75 \\ -12.31 & 0.75 \le t \le 3.25 \\ -12.31 + 16.41(t - 3.25) & 3.25 \le t \le 4 \end{cases} \\
\dot{\phi}(t) &= \begin{cases} 18.41t & 0 \le t \le 0.75 \\ 13.85 & 0.75 \le t \le 3.25 \\ 13.85 - 18.46(t - 3.25) & 3.25 \le t \le 4 \end{cases}
\end{aligned}
$$

Figure 10.7 shows the motion trajectories of the moving platform along with their velocity and acceleration. The trajectory in the *x* direction is represented by the solid line, the trajectory in the *y* direction by the dashed line, and the orientation trajectory by the dash-dotted line. All trajectories smoothly reach their final positions within four minutes, starting from their initial values. The velocity in the *x* direction, depicted by the solid line in the middle plot, begins with increasing linear speed, maintains a constant speed in the middle, and then decelerates linearly toward the end. This pattern is also observed in the angular acceleration shown in the rightmost plot. The acceleration in the *x* direction is positive and constant at the start, becomes zero in the middle, and then becomes constant and negative

towards the end. A similar pattern is noted for the other two motion components.




▶ Long Description for Figure 10.7

**FIGURE 10.7**
Linear trajectory with parabolic blends: position, velocity, and acceleration. ⏎

**Example 10.5 (Linear-Parabolic Trajectory with Optimal Time)** *For the planar 4RPR cable robot, the aim is for the robot's moving platform to reach the final state* $\boldsymbol{\chi}_f = [100, -50, 45^\circ]^T$ *with a minimum time, starting from the initial state* $\boldsymbol{\chi}_0 = [10, -10, 0]^T$. *Given that the maximum speed of motion is 30 meters per minute and the maximum acceleration is 40 meters per*

*square minute, determine the optimal time of motion $t_f$, design the optimal linear trajectory with parabolic blend, and plot it.*

**Solution:** First, considering the maximum velocity and acceleration constraints, the minimum time of motion for each trajectory is determined based on Equation 10.20:

$$\begin{aligned} t_{f_1} &= \frac{1}{30}(100-10) + \frac{30}{40} = 3.75 \\ t_{f_2} &= \frac{1}{-30}(-50+10) + \frac{30}{40} = 0.58 \\ t_{f_3} &= \frac{1}{30}(45+0) + \frac{30}{40} = 2.25 \end{aligned}$$

It is necessary to select a minimum time of 3.75 minutes to ensure all trajectories complete their motion successfully. Therefore, the velocity and acceleration in the $x$ direction are set to the maximum actuator constraints, $\Omega_1 = 30$ and $\alpha_1 = 40$. Considering the required motion ratios $\gamma_2$ and $\gamma_3$ for the $y$ direction and $\phi$, respectively, these values are adjusted accordingly:

$$\begin{aligned} \gamma_2 &= \frac{-50+10}{100-10} = -0.44, & \gamma_3 &= \frac{45+0}{100-10} = 0.5, \\ v_2 &= \gamma_2\, v_{max} = -13.33, & \omega_3 &= \gamma_3\, v_{max} = 15.0, \\ a_2 &= \gamma_2\, a_{max} = -17.78, & \alpha_3 &= \gamma_3\, a_{max} = 20.0. \end{aligned}$$

Next, the blend time is determined based on the joint motions using Equation 10.18:

$$t_b = \frac{30}{40} = 0.75$$

With these values and the initial and final joint states, the motion trajectory is obtained using Equation 10.15:

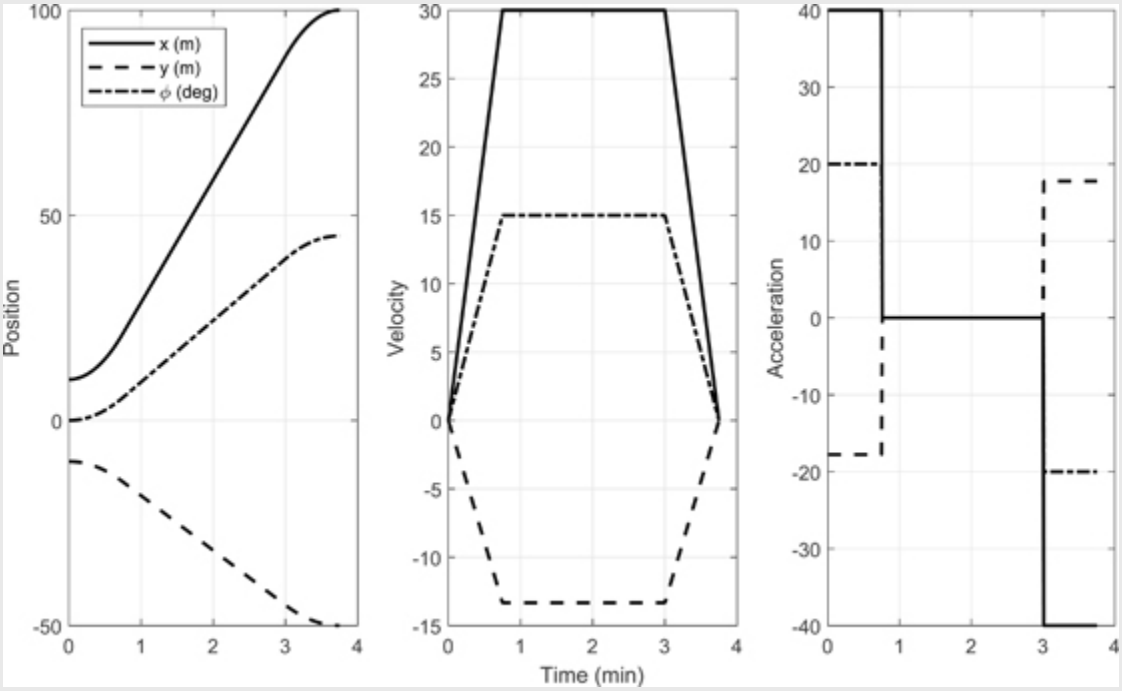
$$
\begin{aligned}
x_G(t) &= \begin{cases} 10 + 20t^2 & 0 \leq t \\ 10 + 30(t - 0.375) & 0.75 \leq \\ 10 + 30(t - 0.375) - 20(t - 3.25)^2 & 3.25 \leq \end{cases} \\
y_G(t) &= \begin{cases} -10 - 8.89t^2 & \\ -10 - 13.33\,(t - 0.375) & ( \\ -10 - 13.33\,(t - 0.375) + 8.89(t - 3.25)^2 & \end{cases} \\
\phi(t) &= \begin{cases} 10t^2 & 0 \leq t \leq 0. \\ 15\,(t - 0.375) & 0.75 \leq t \leq 3 \\ 15\,(t - 0.375) - 10(t - 3.25)^2 & 3.25 \leq t \leq \end{cases}
\end{aligned}
$$

The velocity of motion is calculated using Equation 10.16:

$$
\begin{aligned}
\dot{x}_G(t) &= \begin{cases} 40t & 0 \le t \le 0.75 \\ 30 & 0.75 \le t \le 3.25 \\ 30 - 40(t - 3.25) & 3.25 \le t \le 4 \end{cases} \\
\dot{x}_G(t) &= \begin{cases} -17.78t & 0 \le t \le 0.75 \\ -13.33 & 0.75 \le t \le 3.25 \\ -13.33 + 17.78(t - 3.25) & 3.25 \le t \le 4 \end{cases} \\
\dot{\phi}(t) &= \begin{cases} 20t & 0 \le t \le 0.75 \\ 15 & 0.75 \le t \le 3.25 \\ 15 - 20(t - 3.25) & 3.25 \le t \le 4 \end{cases}
\end{aligned}
$$

Figure 10.8 shows the motion trajectories of the moving platform along with their velocity and acceleration. The trajectory in the $x$ direction is represented by the solid line, the trajectory in the $y$ direction by the dashed line, and the orientation trajectory by the dash-dotted line. All trajectories smoothly reach their final positions within the minimum time of 3.75 minutes, starting from their initial values. The velocity in the $x$ direction, depicted by the solid line in the middle plot, begins with increasing linear speed, maintains the constant and maximum allowable speed in the middle, and then decelerates linearly towards the end. This pattern is also observed in the angular acceleration shown in the rightmost plot, where the velocity limit is not hit. Additionally, the acceleration in the $x$ direction starts at the maximum allowable value at the beginning of the trajectory. In this

way, the optimal time trajectory has been successfully designed.




▶ Long Description for Figure 10.8

**FIGURE 10.8**

The linear-parabolic optimal time trajectory for moving platform motions and their velocity and acceleration. ⏎

## 10.2.5 Trajectory Planning in Joint Space

The techniques presented so far in planning the trajectory of robot motion have been implemented in the task space of the robot, specifically focusing on the moving platform motion variables $\boldsymbol{\chi}(t)$. Given that robot motion control commonly employs a closed-loop structure within the task space, it is reasonable for trajectory planning to occur in

this space as well. In this methodology, constraints, on the actuator's maximum speed and acceleration, intricately used for optimal time trajectory planning, will not be directly implementable.

On the other hand, in some applications, it is desirable for the robot's active joints to follow a specific trajectory, such as a linear or circular trajectory. In such cases, all trajectory planning methods can be implemented on the joint space variables $\boldsymbol{q}(t)$. It should be noted that in this case, if a control structure based on task space is used in the robot, the designed trajectories in the joint space must be mapped to the task space using the robot's forward kinematics. Naturally, the computational workload for this mapping process should be considered, and trajectory design in the joint space should only be utilized when necessary.

In the manual movement of robots using the teaching pendant, the linear-parabolic trajectory generation method is usually adopted in joint space. In this type of movement, by actuating the lever or motion key of each limb motion, the command to initiate the movement of that limb is prompted. In practice, upon receiving the motion command from the operator, the actuator of that joint is briefly stimulated with constant acceleration until the joint's movement speed reaches the desired speed (and less than the maximum speed). While the stimulation button remains engaged, the joint maintains its motion at a consistent speed. Once the operator disengages the motion command,

the robot undergoes maximum deceleration to swiftly come to a stop.

On the other hand, this method can also be used in optimal-time motion trajectory planning. In this application, the movement of the joint begins with the maximum possible torque in the initial motion using the maximum acceleration command. This command will continue until either the robot's speed reaches its maximum, or the movement time reaches half of $t_f$. In the first case, the robot will continue its movement at its maximum speed, and in the second case, the deceleration with the maximum deceleration will occur. In both cases, the motor along the trajectory will exhibit either the maximum acceleration or speed, resulting in optimal motion time.

Despite the advantages mentioned in this trajectory planning approach, sudden acceleration and deceleration will occur in the robot. It should be noted that although the application of this method will minimize the robot's motion time, the movement of the robot arm will lack the expected continuity and smoothness compared to what happens in trajectory design methods based on high-order polynomial functions. Although, due to the mechanical structure of the robot, acting as a low-pass mechanical filter, sudden accelerations in the robot are somewhat reduced, still the presence of such discontinuous acceleration movements will lead to a reduction in the useful life of moving parts and the robot's power transmission system.

## 10.3 Motion Simulation

By deriving the dynamic formulation of the robot and designing appropriate motion trajectories, the robot's motion can be simulated by applying forces/torques to the joint actuators. Simulating the dynamic motion of the robot is crucial for analyzing its movement and designing a suitable controller. Through the simulation of dynamic robot motion prior to construction, it is possible to analyze the dynamic behavior and motion characteristics of robot components.

This dynamic formulation of the robot can be used in two main ways:

**Forward Dynamics:** In this approach, it is assumed that the actuator forces/torques $\boldsymbol{\tau}(t)$ are known, and the objective is to determine the motion trajectory $\boldsymbol{\chi}(t)$ of the manipulator. This type of analysis is used in dynamics simulations. If the closed-form formulation of the robot dynamics is not available, integrating the implicit dynamics of the robot is plausible. However, due to the implicit nature of the dynamic equations, specialized integration routines that can handle implicit differential equations are required for simulations.

**Inverse Dynamics:** Here, the goal is to calculate the actuator forces $\boldsymbol{\tau}$ needed to achieve a prescribed trajectory $\boldsymbol{\chi}(t)$ in the presence of any disturbance wrench $\boldsymbol{\mathcal{F}}_d$ applied to the robot. This form of dynamic

formulation is used to evaluate the forces required for desired motion. Additionally, the inverse dynamics formulation is crucial in the design of model-based control laws, which are discussed in Chapter 12.

The method of simulating dynamic robot motion using the robot's forward dynamics will be explained further in section 10.3.1, where such simulations are advantageous in the selection of sensors and actuators for the robot. Furthermore, the simulation of the robot's inverse dynamics will be elaborated in section 10.3.2. The robot's inverse dynamics is also fundamentally utilized in the design of model-based controllers, whose intricacies will be detailed in Chapter 12.

### 10.3.1 Forward Dynamics

In the analysis of forward dynamics, it is assumed that the values of the actuator forces/torques, and friction forces/torques in the active joints, as well as any external wrench applied to the robot moving platform, are known, and it is desirable to determine the dynamic motion of the robot and the performed trajectory of its motion. If we denote the robot's actuator forces/torques as $\boldsymbol{\tau}$ and encapsulate all the other disturbance wrenches applied to the robot's moving platform by $\boldsymbol{\mathcal{F}}_d$, the governing dynamic equation for the robot's motion can be rewritten from Equation 9.32, as follows.

$$\begin{aligned} \boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{\mathcal{F}} &= \boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) + \boldsymbol{\mathcal{F}}_d, \\ \boldsymbol{\mathcal{F}}_d &= \boldsymbol{\mathcal{F}}_e + \boldsymbol{\mathcal{F}}_f + \cdots \end{aligned} \tag{10.21}$$

In this equation, $\boldsymbol{\mathcal{F}}_f$ represents the projection of the friction torques in the robot's active joints into the task space, $\boldsymbol{J}$ is the Jacobian matrix of the robot's moving platform, and $\boldsymbol{\mathcal{F}}_e$ is any external wrench applied to the robot's moving platform. Accordingly, as depicted in Figure 10.9, the forward dynamics of the robot can be introduced as the process of determining the motion trajectory of the robot based on the applied input actuator torques and external disturbances.

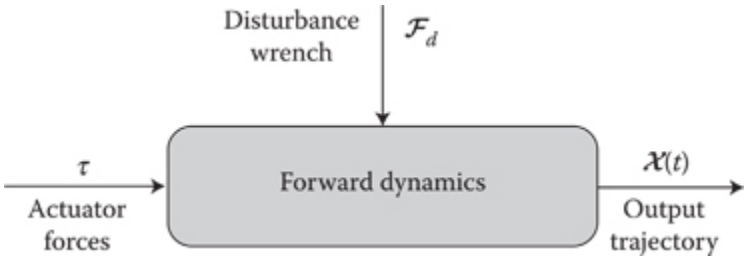



**FIGURE 10.9**
Block diagram of forward dynamics simulations. ⏎

Given that the governing equations for robot dynamics are a set of nonlinear differential equations, determining the motion trajectory using algebraic methods is not feasible. Analytical solution of forward dynamics, if possible, is often challenging and not readily achievable except in very simple cases. Therefore, numerical solutions

are commonly employed for solving these differential equations.

Numerical solutions for differential equations have been extensively addressed in various references and engineering software [218, 219]. A general approach for solving such systems of differential equations involves transforming them into state-space form and using numerical integration methods such as Euler or Runge-Kutta methods [220]. In these methods, the derivatives of state variables are approximated using their previous values over small time intervals, and the recursive methods are used to calculate the solution to the differential equation with sufficient accuracy. In software like MATLAB®, this approach can be implemented using the ode family of commands.

Transforming the robot's dynamic equations into state-space form is also straightforward for parallel robots if the closed-form formulation of system dynamics is available. Considering the invertibility property of the mass matrix discussed in section 8.3.1, the dynamic equations of parallel robots can be easily transformed into state-space form in two steps. Firstly, using the inverse of the mass matrix, the Equation 10.21 can be explicitly written in terms of $\ddot{\boldsymbol{\chi}}$ as follows.

$$\ddot{\boldsymbol{\chi}} = \boldsymbol{M}^{-1}(\boldsymbol{\chi}) \left( \boldsymbol{J}^T \boldsymbol{\tau} - \boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \dot{\boldsymbol{\chi}} - \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) \right)$$

(10.22)

This equation represents a second-order system of differential equations. This system can be transformed by introducing a state variable:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\chi} \\ \dot{\boldsymbol{\chi}} \end{bmatrix}$$

(10.23)

into a state-space representation.

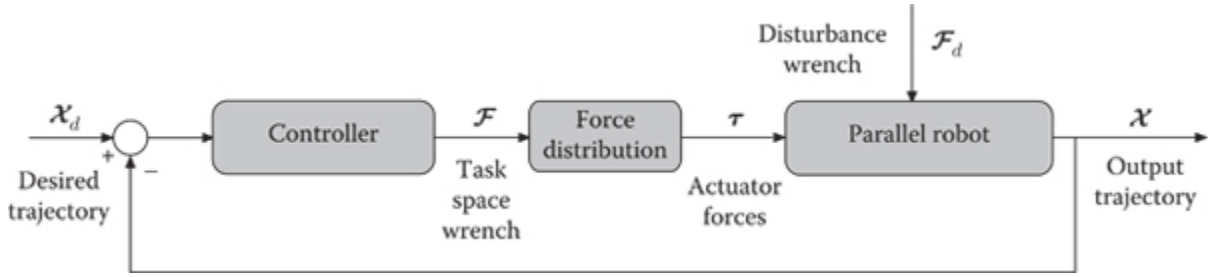
$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \dot{\boldsymbol{\chi}} &&= \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_2 &= \ddot{\boldsymbol{\chi}} &&= \boldsymbol{M}^{-1}(\boldsymbol{q}) \left( \boldsymbol{J}^T \boldsymbol{\tau} - \boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \dot{\boldsymbol{\chi}} - \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) \right) \end{aligned}$$

(10.24)

In engineering software, numerically solving these state-space equations with specified initial conditions $\boldsymbol{x}(0) = [\boldsymbol{x}_1(0), \boldsymbol{x}_2(0)]^T$ is easily accomplished using numerical differential equation solvers such as the Runge-Kutta method. An example of such a numerical solution for a robot will be performed in the continuation of this section, and the results will be analyzed.

Forward dynamics can also be utilized in the simulation of closed-loop control of robots. If motion tracking is the goal, feedback loops are commonly used in robot motion

control, as illustrated in the block diagram in Figure 10.10. As seen in this diagram, by employing feedback loops and designing an appropriate controller, the desired motion tracking of the moving platform can be achieved. In this diagram, the desired robot motion trajectory is generated by the trajectory planner algorithms, for which some trajectory planning methods are presented in section 10.2.




▶ Long Description for Figure 10.10

**FIGURE 10.10**
Block diagram of the forward dynamics implementation in closed loop. ⏎

The primary characteristic of this diagram lies in its utilization of a feedback loop. This loop dynamically compares the real trajectory $\boldsymbol{\mathcal{X}}(t)$ of the robot with the desired trajectory $\boldsymbol{\mathcal{X}}_d(t)$ in real-time, and applies the tracking error to the robot's controller block. The robot controller, utilizing various methods as a function of the tracking error, issues wrench commands in task space to be applied to the moving platform. The wrench commands are mapped into the actuator torques through the force distribution block. For force distribution one may use the

Jacobian transpose mapping or redundance resolution techniques, fully elaborated in Chapter 6, and in Section 6.6. Consequently, this approach enables the continuous online monitoring and control of the tracking error.

Irrespective of the specific type and design of control law implemented in the robot, details of which will be elaborated on in upcoming chapters, the closed-loop structure stands out as the fundamental element of motion control. It ensures the achievement of the desired tracking even in the presence of disturbances and model uncertainties. Adjusting the controller parameters within this structure typically involves simulating the closed-loop dynamics of the robot, as illustrated in the block diagram presented in Figure 10.10. The numerical simulation of this structure will be conducted on a robot in the subsequent sections of this chapter, and the results will be evaluated.

### 10.3.2 Inverse Dynamics

In the analysis of inverse dynamics for a robot, it is assumed that the trajectory of the robot's motion, denoted as $\boldsymbol{\chi}(t)$, and its derivatives are given, and the objective is to determine the actuator forces/torques that produce such a predefined motion. Thus, the inverse dynamics of the robot can be described as the process of determining the necessary actuator forces/torques to produce a motion along a predefined trajectory, as shown in Figure 10.11.

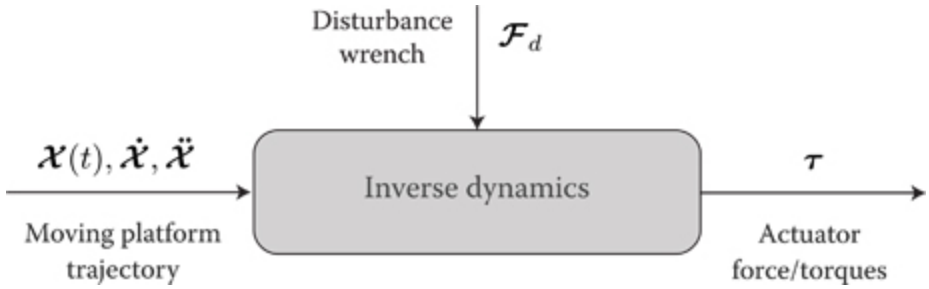



**FIGURE 10.11**
Block diagram of the inverse dynamics process in robots. ⏎

Despite the governing equations of robot dynamics being a set of nonlinear differential equations, assuming knowledge of the motion trajectory and its derivatives eliminates the need for integration in determining the required actuator torques. In this case, it is sufficient to first calculate the following algebraic relationship.

$$\mathcal{F} = M(\mathcal{X})\ddot{\mathcal{X}} + \mathcal{C}(\mathcal{X}, \dot{\mathcal{X}})\mathcal{X} + \mathcal{G}(\mathcal{X}) + \mathcal{F}_d,$$

The actuator forces/torques required to achieve the desired trajectory are derived by projecting the calculated wrench $\mathcal{F}$ from the task space to the joint space. This projection is achieved through the Jacobian transpose matrix $J^T$, which yields the relationship $\mathcal{F} = J^T\tau$. When the manipulator is fully parallel and does not have actuation redundancy, $J^T$ is a square matrix, allowing for straightforward computation of actuator forces with $\tau = J^{-T}\mathcal{F}$, provided that $J$ is nonsingular.

For redundant manipulators, which are equipped with more actuators than the minimum required, there are infinitely many solutions for the actuator forces $\boldsymbol{\tau}$ that can produce the same Cartesian wrench $\boldsymbol{\mathcal{F}}$. To address this redundancy, various algorithms have been developed to optimally utilize the extra degrees of actuation, as detailed in Section 6.6. As a simple solution, the Moore–Penrose pseudo inverse of the Jacobian transpose, denoted $\boldsymbol{J}^{T^{\dagger}}$, can be used for this calculation used in this simulation. This method provides a minimum norm solution for the actuator forces $\boldsymbol{\tau}_{\min}$, ensuring that the wrench projection $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^{T}\boldsymbol{\tau}_{\min}$ is satisfied. Although the Moore–Penrose pseudo inverse method is widely used and effective for many applications, it is not ideal for cable-driven manipulators because it may lead to undesirable compression forces on the cable-driven actuators. A comprehensive analysis of this issue is provided in redundancy resolution techniques extensively studied in Chapter 6.

Through inverse dynamics simulation, one can leverage the simulation results to select appropriate sensors and actuators for the robot. To achieve this, the required motions of the robot in the workspace are pre-planned, and the inverse dynamics calculates the necessary actuator forces/torques to produce such a desired motion. Considering that the calculation of these forces/torques assumes precise robot motion without tracking errors, in the absence of noise and disturbances, it is necessary to

adjust the required actuator forces/torques with a suitable and greater-than-one scaling factor. This ensures that the maximum power required by the actuators and their maximum speed and torque during motion are determined. Based on the maximum power requirement, the suitable actuator type is selected, and based on the maximum torque and speed requirements, the appropriate combination of actuator and transmission system is determined.

Inverse dynamics of the robot are widely used in model-based control methods. Since the robot's dynamic formulation is nonlinear and coupled, using the inverse dynamics of the system in robot control allows these equations to be linearized through feedback. This control method, known as *inverse dynamics control*, has served as the fundamental basis for the development of model-based control methods in robots. The application of inverse dynamics control and enhanced methods based on it will be explored in Chapter 12.

## 10.4 Case Study I: Planar Cable Robot

The dynamic formulation of the planar cable robot has been meticulously developed using various methods, as detailed in previous chapters. Chapter 7 provides an exhaustive analysis of the robot's dynamics through the Newton-Euler Method, offering a comprehensive understanding of the

forces and moments acting on the robot. The implicit dynamic formulation, which deals with the robot's dynamics without explicitly solving for all the accelerations, is presented in section 7.4.

Furthermore, Chapter 8 elaborates on the closed-form dynamics of the robot, presenting a more straightforward and potentially computationally efficient representation of the system. The governing equations of motion, which form the foundation of the robot's dynamic behavior, are detailed in Section 8.4. In the current section, the dynamic simulation of the manipulator is performed to validate and compare the theoretical formulations.

### 10.4.1 Forward Dynamics Simulations

As previously discussed, the dynamic formulation of the planar 4RPR manipulator can be utilized for forward dynamics simulations. In forward dynamics, as illustrated in Figure 10.9, the manipulator's actuator forces and external disturbances are specified, and the objective is to determine the resulting trajectory of the moving platform. This involves calculating the platform's position, velocity, and acceleration over time based on the given inputs. The implicit dynamic formulation of the robot is detailed in section 7.4, with the essential equations for forward dynamic simulation being Equations 7.42 through 7.44, which must be integrated using an appropriate integration routine.

Due to the implicit nature of the dynamic formulation, standard numerical integration methods such as the Runge–Kutta methods cannot be directly applied. Instead, a specialized integration routine capable of handling implicit functions is required. For instance, the `ode15i` function in MATLAB or equivalent functions in Python, like those available in the SciPy library, can be utilized for this purpose.

The process begins by determining the initial conditions for accelerations $\ddot{\boldsymbol{\chi}}(0)$ by solving the implicit equations at the initial time. This initial step is crucial as it sets the foundation for accurate simulation of the manipulator's dynamics. Once the initial accelerations are computed, the dynamic formulation, expressed in vector form, is integrated iteratively over time. Throughout the integration process, it is essential to monitor the stability and accuracy of the simulation. Adjustments may be needed to the integration parameters to ensure convergence and to handle any numerical challenges that arise due to the implicit nature of the equations.

Numerous simulations have been conducted to verify the consistency and precision of the dynamic formulation. The geometric and inertial parameters of the manipulator used in these simulations are provided in Table 10.1. For these simulations, it is assumed that the manipulator's motion occurs in a horizontal $x-y$ plane with gravitational force acting in the $-\hat{\boldsymbol{K}}$ direction.

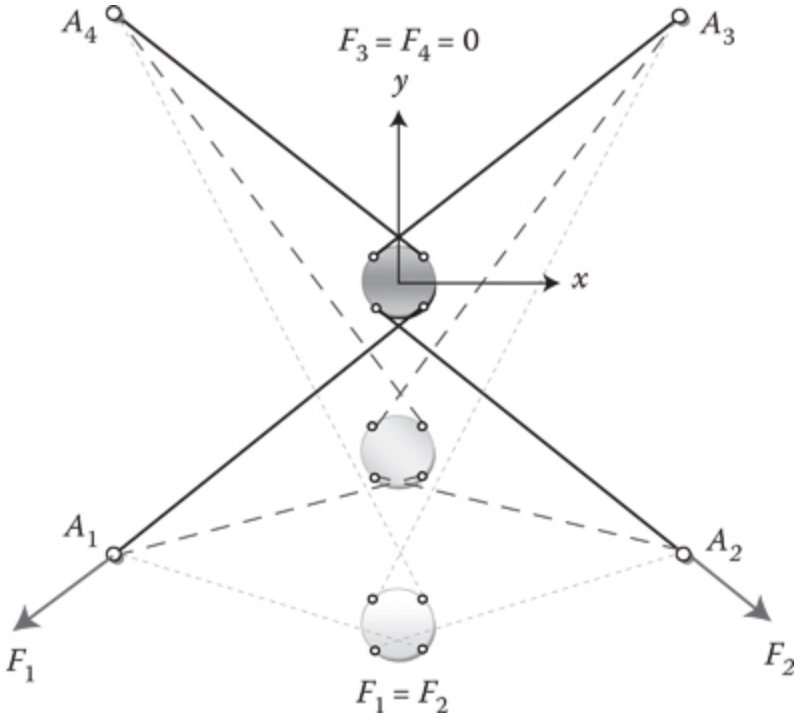
**TABLE 10.1**

Geometric and Inertial Parameters of the 4RPRP arallel Manipulator ⏎

| Description | Quantity |
|---|---|
| $R_A$: Radius of the fixed base points $A_i$'s | 900m |
| $R_B$: Radius of the moving platform points $B_i$'s | 10m |
| $\theta_{A_i}$: Angle of the fixed base points$A_i$'s | $[-135^\circ, -45^\circ, 45^\circ, 135^\circ]$ |
| $\theta_{B_i}$: Angle of the moving platform points$B_i$'s | $[-45^\circ, -135^\circ, 135^\circ, 45^\circ]$ |
| $M$: The moving platform mass | 2500kg |
| $I_m$: The moving platform moment of inertia | $3.5 \times 10^5 \text{kg} \cdot \text{m}^2$ |
| $\rho_m$: The limb density per length | $0.215 \text{kg}/\text{m}$ |

In the first simulation, the dynamic equations for the planar 4RPR manipulator are solved using a forward dynamics approach, with the specified geometric and inertial parameters from Table 10.1. In this setup, both the disturbance wrench and the initial conditions for all states are set to zero. The actuator torques are defined as $\boldsymbol{\tau} = [1, 1, 0, 0]^T$ kN, and the gravity vector is $\boldsymbol{g} = [0, 0, -9.81]^T$.

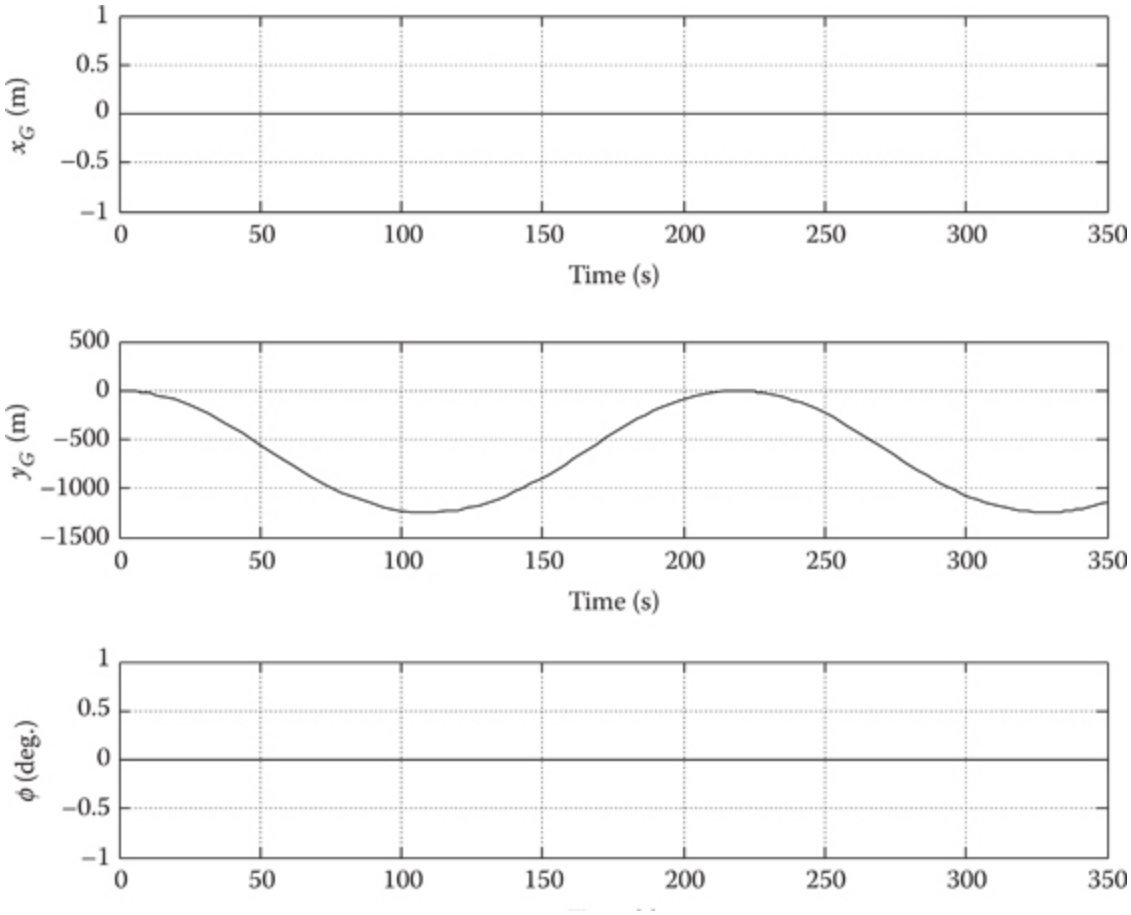
As shown in Figure 10.12, the actuator forces result in a net force in the $-y$ direction, causing the manipulator to move solely along the $-y$ axis, with no motion in the $x$ or $\phi$ directions. The simulation is run over a period of 350 seconds, and the resulting trajectory of the moving platform is depicted in Figure 10.13. This figure demonstrates that the position $x(t)$ and the orientation $\phi(t)$ of the moving platform remain constant at zero throughout the simulation. In contrast, the position $y(t)$ exhibits oscillatory motion. The amplitude of this oscillation is such that the center of mass of the moving platform crosses the line $A_1A_2$ as expected. Additionally, the platform returns to its initial position after approximately 220 seconds. These results confirm that the output trajectories align perfectly with the anticipated system behavior, validating the accuracy of the simulation.



▶ Long Description for Figure 10.12

**FIGURE 10.12**

First simulation scenario for the planar 4RPR manipulator with $\boldsymbol{\tau} = [1, 1, 0, 0]^T kN$. ⏎



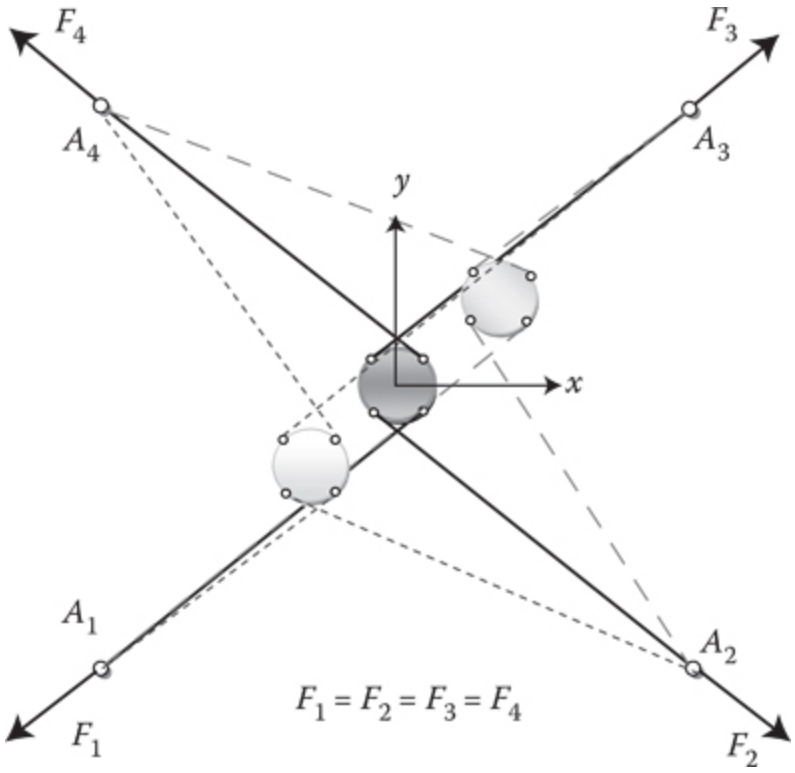

▶ Long Description for Figure 10.13

**FIGURE 10.13**

The output trajectory of the first simulation scenario for the planar 4RPR manipulator. ⏎

In the second simulation, as shown in Figure 10.14, the actuator forces are set to $\boldsymbol{\tau} = [1, 1, 1, 1]^T$ kN, and the

gravity vector is $\boldsymbol{g} = [0, 0, -9.81]^T$. This configuration leads to a zero net force on the moving platform during its planar motion. If all initial states are set to zero, the platform is expected to remain stationary. However, when the initial conditions are set to $\boldsymbol{\chi}(0) = [1, 1, 0]^T$ (with initial displacements of one in both the $x$ and $y$ directions and no initial orientation), the simulation predicts no angular motion but oscillatory behavior in the $x$ and $y$ directions. As depicted in Figure 10.14, the oscillation amplitude corresponds to the initial displacements in the $x$ and $y$ directions.

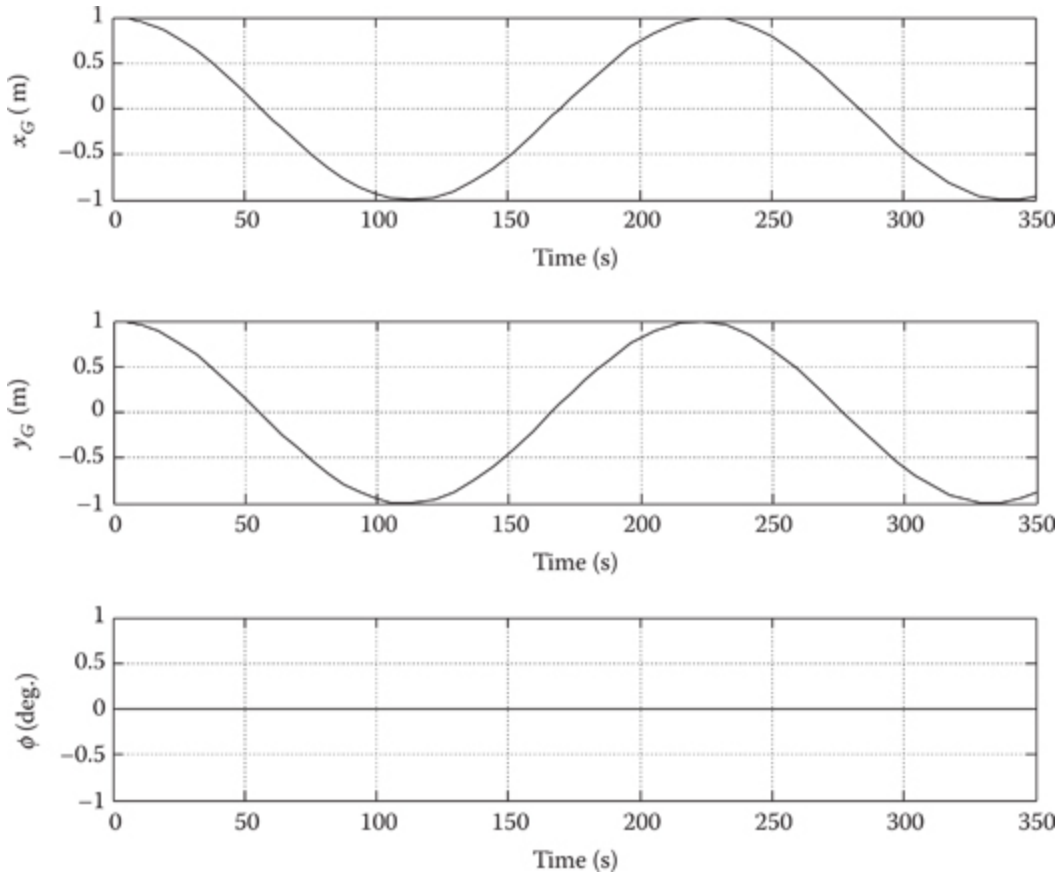



▶ Long Description for Figure 10.14

**FIGURE 10.14**

Second simulation scenario for the planar 4RPR manipulator with $\boldsymbol{\tau} = [1, 1, 1, 1]^T kN$, and $\boldsymbol{X}(0) = [1, 1, 0]^T$ . ⏎

For the simulations, the governing dynamic equations for the planar 4RPR manipulator are solved, with the disturbance wrench set to zero and the specified initial conditions and actuator forces applied. The simulation runs for 350 seconds, and the resulting trajectory of the moving platform is detailed in Figure 10.15.




▶ Long Description for Figure 10.15

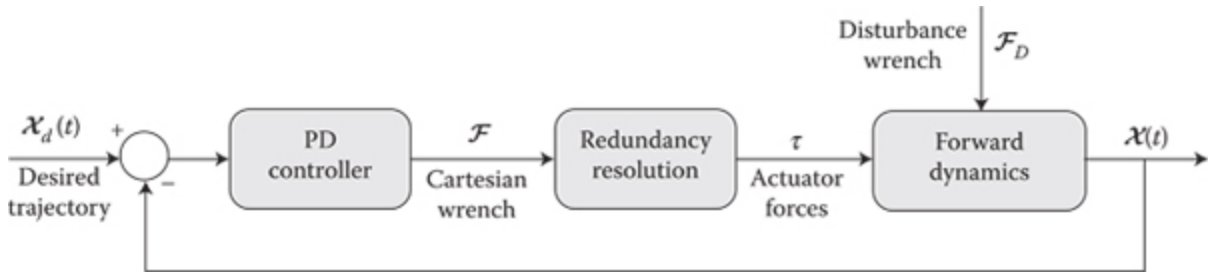
**FIGURE 10.15**

The output trajectory of the second simulation scenario for the planar 4RPR manipulator. ⏎

The figure shows that while the orientation $\phi(t)$ of the moving platform remains constant at zero throughout the simulation, the positions $x(t)$ and $y(t)$ exhibit oscillatory motion with an amplitude of one. Remarkably, the amplitude of these oscillations stays consistently at one, and the moving platform returns to its initial position after approximately 220 seconds. These results demonstrate a precise match with the expected system behavior, confirming the accuracy of the simulation and the reliability of the dynamic model used.

### 10.4.2 Closed-Loop Dynamics Simulations

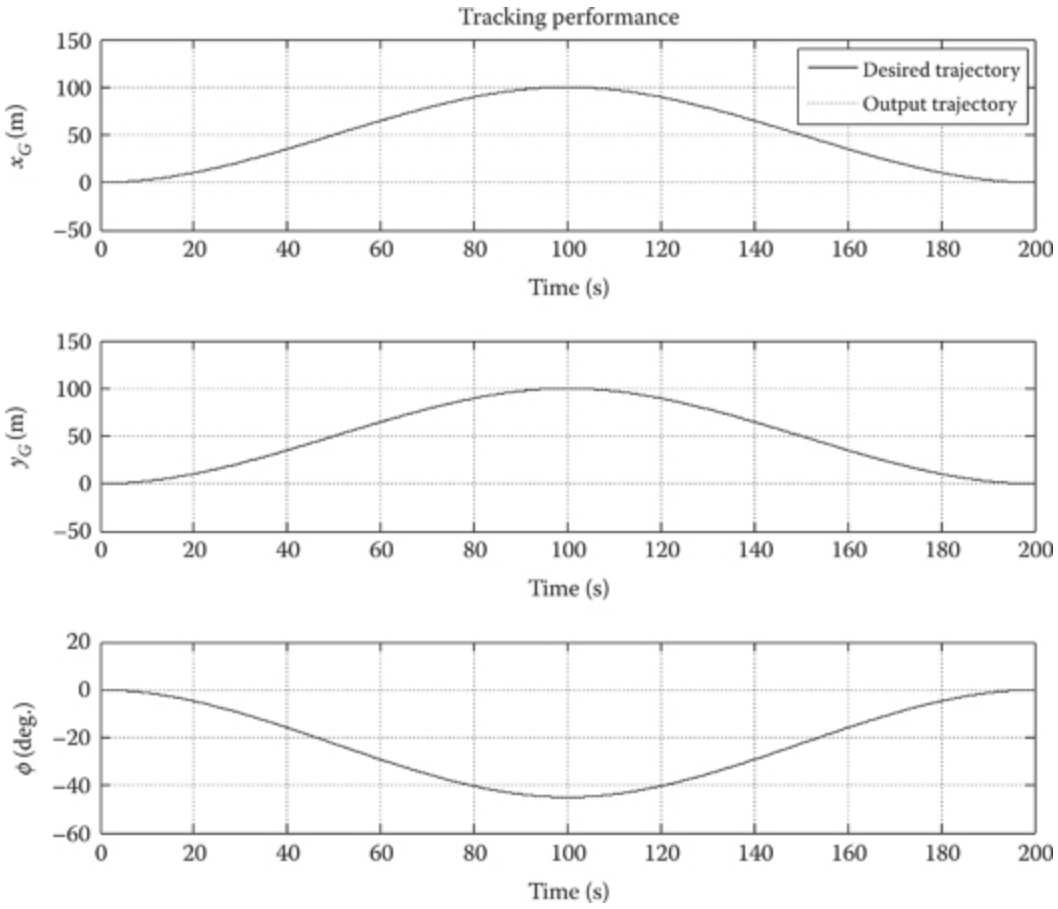
In the third simulation, forward dynamics are applied within a closed-loop control framework, as illustrated in Figure 10.16. This simulation aims to follow a specified trajectory for the motion of the manipulator's moving platform, as shown in Figure 10.17. To achieve this trajectory, a closed-loop control system continuously adjusts the actuator forces based on real-time feedback, ensuring precise adherence to the desired path.



▶ Long Description for Figure 10.16

**FIGURE 10.16**

Block diagram of the forward dynamics implementation of 4RPR manipulator in closed loop. ⏎




▶ Long Description for Figure 10.17

**FIGURE 10.17**

The desired and output trajectories of the planar 4RPR manipulator in closed loop. ⏎

In this setup, the actual trajectory of the moving platform, denoted as $\boldsymbol{\chi}(t)$, is measured and compared to

the desired trajectory $\boldsymbol{\chi}_d(t)$. The difference between these trajectories, known as the tracking error $\boldsymbol{e}(t) = \boldsymbol{\chi}_d(t) - \boldsymbol{\chi}(t)$, is fed into a control system. This system determines the appropriate wrench $\boldsymbol{\mathcal{F}}$ to be applied to the moving platform to correct deviations and achieve the desired motion.

Various control strategies can be used to compute this wrench, as discussed in Chapters 11 and 12. In this particular simulation, a basic Proportional-Derivative (PD) controller is utilized. The PD controller calculates the required wrench $\boldsymbol{\mathcal{F}}$ by addressing each component of the tracking error $\boldsymbol{e}(t)$ separately, generating the necessary actuator forces to drive the manipulator's moving platform.

The actuator forces required to achieve the desired trajectory are derived by projecting the calculated wrench $\boldsymbol{\mathcal{F}}$ from the task space to the joint space. This projection is achieved through the Jacobian transpose matrix $\boldsymbol{J}^T$, as described in Equation 6.9, through the relationship $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$. When the manipulator is fully parallel and lacks actuation redundancy, $\boldsymbol{J}^T$ is a square matrix, allowing for straightforward computation of actuator forces with $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$, provided that $\boldsymbol{J}$ is nonsingular. For redundant manipulators, which have more actuators than necessary, there are infinitely many solutions for the actuator forces $\boldsymbol{\tau}$ that can produce the same Cartesian wrench $\boldsymbol{\mathcal{F}}$. For simplicity, the Moore–Penrose pseudo inverse of the Jacobian transpose, denoted $\boldsymbol{J}^{T^\dagger}$, is used in this simulation.

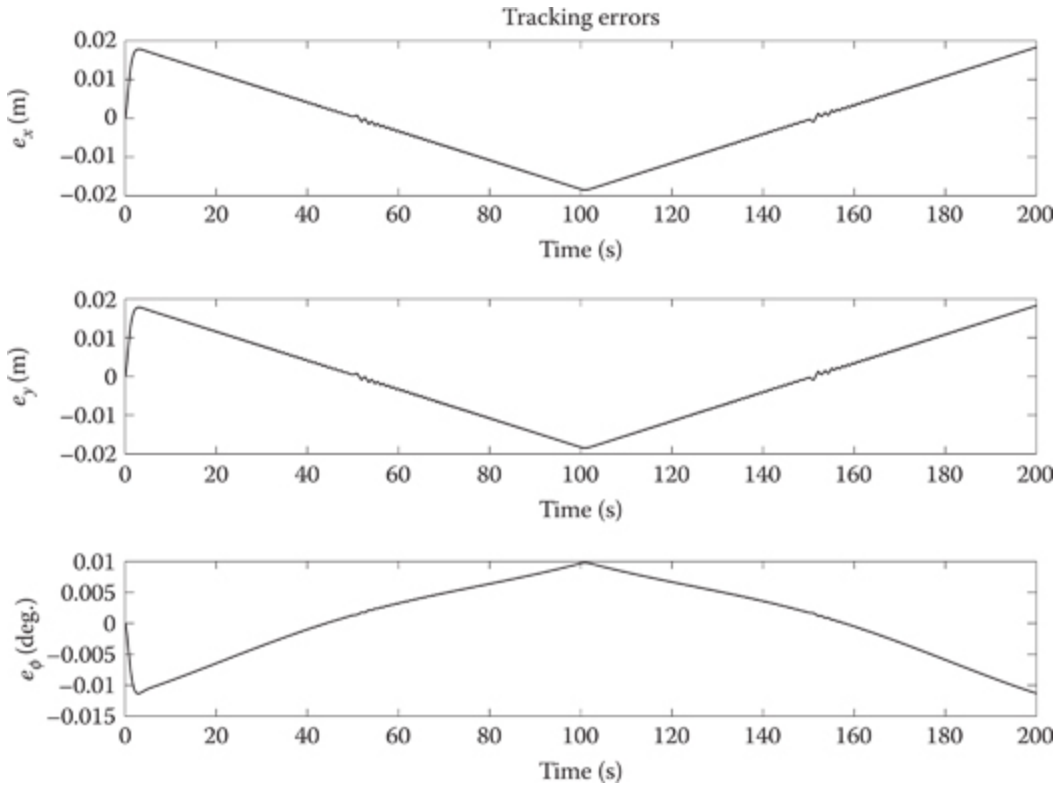
For this simulation, the geometric and inertial parameters of the manipulator are outlined in Table 10.1. Both the disturbance wrench and the initial conditions for all states are initialized to zero, ensuring that any deviations observed during the simulation arise solely from the manipulator's dynamics and the control system's response. Additionally, the gravity vector is set to $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$, representing standard gravitational acceleration in the negative z-direction.

The desired trajectory for the moving platform is defined by a smooth cubic polynomial function for the position coordinates $x_d$, $y_d$, and the orientation $\phi_d$, as illustrated in Figure 10.17. The simulation is executed over a period of 200 seconds, with close monitoring of the system's behavior. The proportional-derivative (PD) controller gains used in this simulation are set to $K_P = K_D = 10^4 \cdot \mathrm{diag}[1, 1, 100]$. These gains are selected to balance the responsiveness and stability of the control system. High gains are often employed in simple linear control structures to ensure that tracking errors remain within acceptable levels while making the response less sensitive to disturbance torques. However, high gains can lead to a stiffer closed-loop system, complicating numerical integration [221]. This challenge is exacerbated when implicit integration of a stiff dynamical system is involved [222].

To evaluate the dynamics formulation and its integration within a closed-loop control system, both implicit and

closed-form formulations of the 4RPR robot were utilized in the simulations. The simulations aimed to thoroughly compare the performance and stability of these two methods under various conditions. Although both approaches produced similar results in terms of trajectory tracking, the closed-form dynamics approach showed superior stability and was less sensitive to variation of the high controller gains. This was particularly evident when high gains led to significant fluctuations in the system's response using the implicit formulation, occasionally causing the simulation to terminate early due to excessive instability. These observations underline the advantage of using the closed-form method for ensuring a more robust and reliable simulation, particularly in environments where precision and stability are crucial.

The closed-form formulation applied in this simulation is based on the Euler-Lagrange method, as extensively detailed in Section 8.4.2. The specific dynamic equations used for the robot are presented in Equation 8.99, while the associated dynamic matrices are described in Equation 8.100. By implementing this formulation in a closed-loop control system, the simulation effectively demonstrates how well the planar 4RPR manipulator adheres to the desired trajectory. Figures 10.17 and 10.18 illustrate the results, showing how the closed-form approach supports precise and stable trajectory tracking.




▶ Long Description for Figure 10.18

**FIGURE 10.18**

The closed-loop tracking performance of the planar 4RPR manipulator. ⏎

Figure 10.17 visually represents the small discrepancy between the desired trajectory and the actual closed-loop motion of the system. The minimal deviation indicates that the manipulator closely follows the intended path with high accuracy. In Figure 10.18, a more detailed analysis reveals that the PD controller effectively reduces tracking errors to less than 0.02 meters in positional accuracy and less than 0.012° in orientation. This level of precision reflects the controller's capability to maintain exact control over the

manipulator's movements, demonstrating its effectiveness in managing and correcting errors within the system.

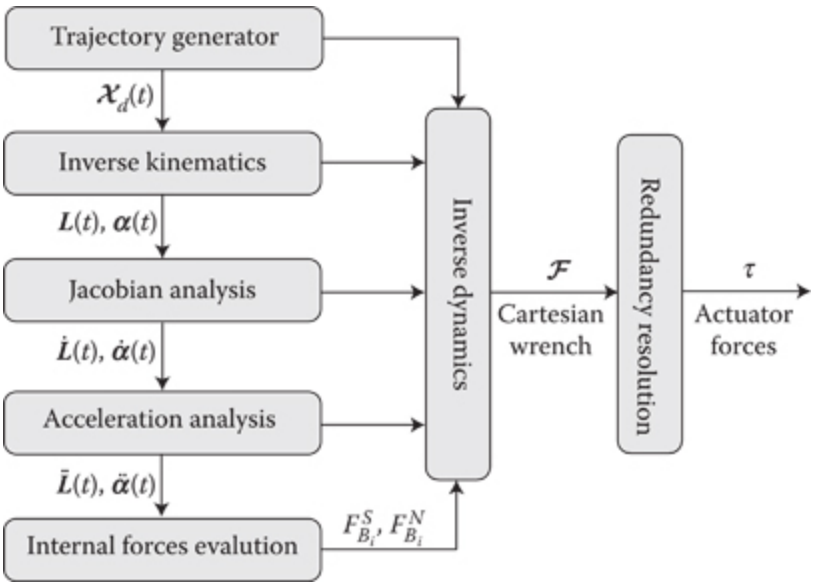
It is important to recognize that the controller used in this simulation is relatively simple, designed primarily to illustrate the fundamental principles of integrating forward dynamics in a closed-loop control system. Despite its effectiveness, the simplicity of this controller may limit its performance in more demanding or complex applications where external disturbances or dynamic uncertainties are more significant. The purpose of this example is to showcase the basic implementation of forward dynamics in a controlled environment, and not the details of the controller design and redundancy resolution.

### 10.4.3 Inverse Dynamics Simulation

The process of inverse dynamics for the robot involves calculating the required actuator forces or torques needed to achieve a specific motion along a predefined trajectory, as illustrated in Figure 10.11. In this process, the trajectory of the manipulator is considered predetermined, and the primary goal is to determine the actuator forces or torques necessary to follow this trajectory accurately.

For the 4RPR planar manipulator, the inverse dynamics process is conducted through a systematic sequence, as detailed in Figure 10.19. The first step involves defining the trajectory that the manipulator's moving platform is expected to follow. In this simulation, a cubic polynomial trajectory planner is used to specify this trajectory, which

is exemplified in Figure 10.17. Once the trajectory is established, the inverse dynamics calculations determine the necessary forces and torques that must be applied by the actuators to achieve the predefined motion.




▶ Long Description for Figure 10.19

**FIGURE 10.19**
Flowchart of inverse dynamics implementation sequence. ⏎

By defining the desired trajectory $\boldsymbol{\mathcal{X}}_d(t)$, the next step is to solve the inverse kinematics of the manipulator to determine the corresponding joint positions $\boldsymbol{L}(t)$ and angles $\boldsymbol{\alpha}(t)$ based on this trajectory. This process is carried out using Equations 4.10 and 4.11. Once the joint positions and limb angles are determined, the manipulator's Jacobian matrices $\boldsymbol{J}$ and $\boldsymbol{J}_\alpha$ are calculated using Equations 5.26 and 5.30, respectively. These Jacobian matrices are essential to

derive the time derivatives $\dot{\boldsymbol{L}}(t)$ and $\dot{\boldsymbol{\alpha}}(t)$ and to describe the manipulator's velocity.

The following step involves performing an acceleration analysis to determine the joint accelerations. Equations 7.23 and 7.25 are employed to calculate the accelerations $\ddot{\boldsymbol{L}}(t)$ and $\ddot{\boldsymbol{\alpha}}(t)$. These equations ensure that the manipulator's motion dynamics are accurately captured, considering the kinematic constraints and the desired trajectory.

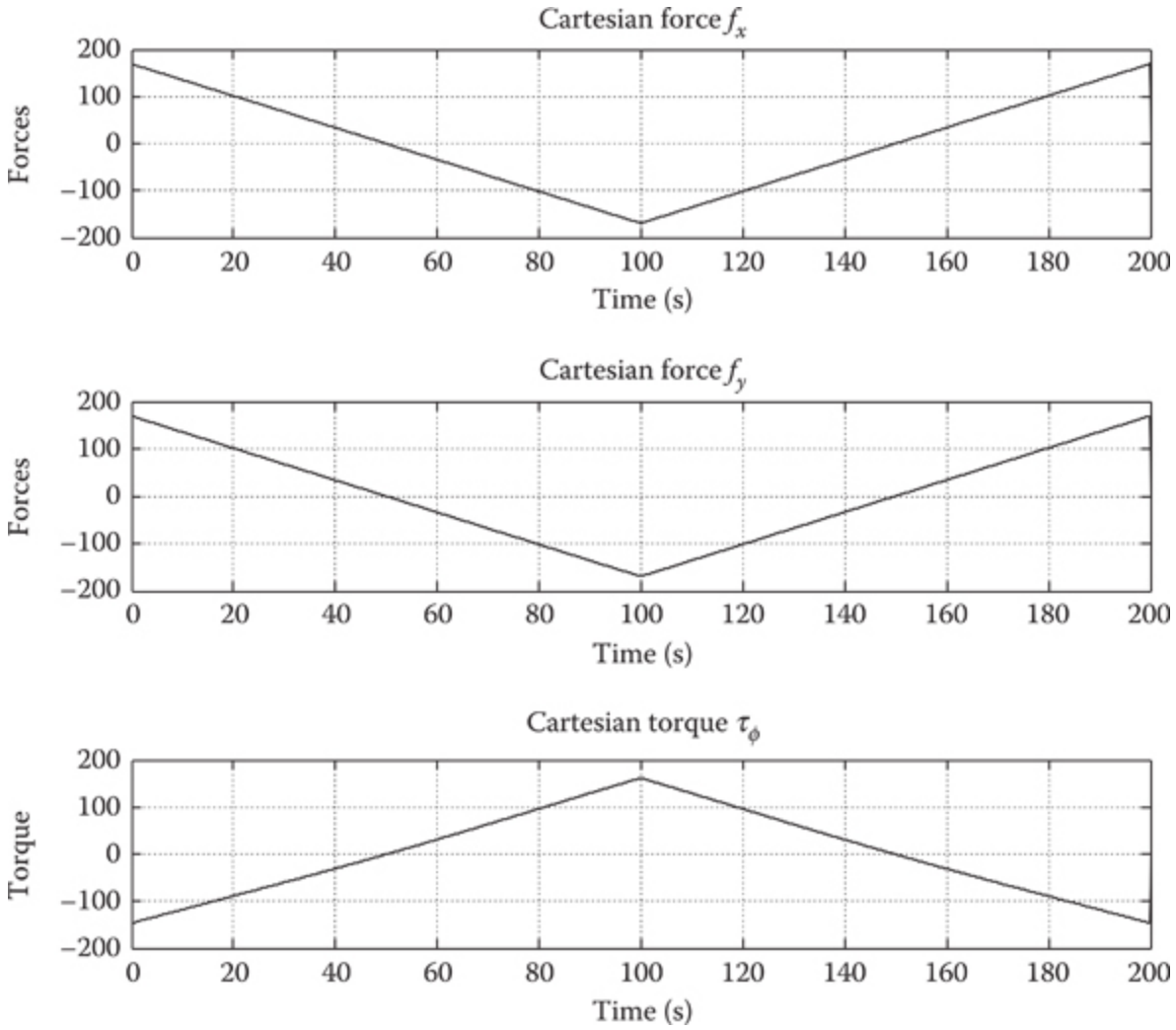
Finally, the interaction forces between the manipulator's limbs and the moving platform, denoted as $F_{B_i}^S$ and $F_{B_i}^N$, are computed using Equations 7.39 and 7.38, respectively. These forces are then substituted into the dynamic equations of the manipulator to solve for the actuator forces. Through this comprehensive sequence, the inverse dynamics simulation accurately determines the actuator forces necessary to follow the predefined trajectory.

Let us define $\boldsymbol{\mathcal{F}}$ as the resulting Cartesian wrench applied to the moving platform, which represents the combined effect of all inertial and external forces acting on the platform, excluding the actuator torques $\boldsymbol{\tau}$. This Cartesian wrench $\boldsymbol{\mathcal{F}}$ is calculated using the implicit dynamic equations provided in Equations 7.42 through 7.44. As an alternative approach, the closed-form equations given in Equation 8.99 can be used to derive the required wrench $\boldsymbol{\mathcal{F}}$.

Finally, $\boldsymbol{\mathcal{F}}$ can be expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$, meaning that $\boldsymbol{\mathcal{F}}$ is essentially the projection of the actuator

forces/torques onto the moving platform. The actuator forces/torques could be determined through a force distribution routine or redundancy resolution method. In scenarios where the manipulator does not have redundancy in its actuation, the Jacobian matrix $\boldsymbol{J}$ is square, making it possible to determine the actuator forces uniquely from the wrench using the formula $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$, provided that $\boldsymbol{J}$ is nonsingular. However, for redundant manipulators— as the one in hand in this case study—the pseudo-inverse of the Jacobian transpose, denoted $\boldsymbol{J}^{T^{\dagger}}$ can be used for the projection.

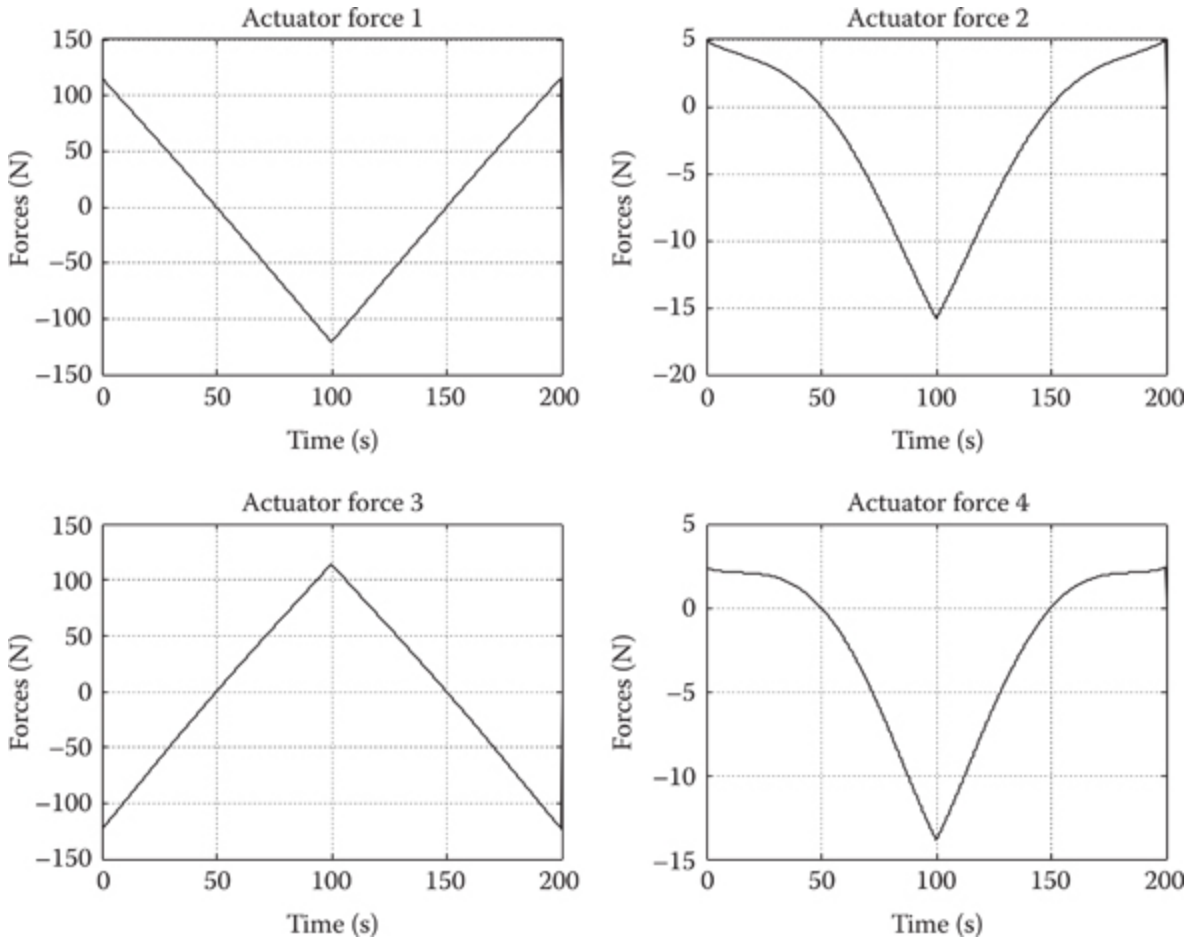
In this simulation, the inverse dynamics of the manipulator are examined with no external disturbance forces applied, meaning $\boldsymbol{\mathcal{F}}_d = \mathbf{0}$. The results of this analysis are depicted in Figures 10.20 and 10.21. The simulation employs standard third-order polynomial trajectories for the manipulator, as illustrated in Figure 10.17. Figure 10.20 shows the Cartesian wrench at the moving platform, denoted as $\boldsymbol{\mathcal{F}} = [f_x, f_y, \tau_\phi]^T$. This figure demonstrates that the Cartesian forces closely match the desired trajectory accelerations, which are linear due to the cubic trajectory profile.




▶ Long Description for Figure 10.20

**FIGURE 10.20**

The Cartesian wrench of the planar 4RPR manipulator required to generate trajectory shown in Figure 10.17. ⏎




▶ Long Description for Figure 10.21

**FIGURE 10.21**

The actuator forces of the planar 4RPR manipulator required to generate trajectory shown in Figure 10.17. ⏎

Figure 10.21 presents the actuator forces of the manipulator. It is evident that, given the manipulator's motion is predominantly in the positive $x$ and $y$ directions, the forces exerted by the first and third limbs are the most significant. This is consistent with the roles of these limbs in producing the necessary forces for achieving the specified motion. To select appropriate actuators for the

robot, one can calculate the required power for each limb by multiplying the force by the linear velocity of each joint and plotting this over time. Identifying the peak power requirement assists in selecting the correct actuator type and specifications.

It is important to note that in this simulation, redundancy resolution schemes were not utilized, resulting in actuator forces with both positive and negative values. This issue can be addressed by employing redundancy resolution techniques, as discussed in Section 6.6. These techniques will be effectively implemented in simulations in the subsequent chapters to enhance the performance and feasibility of the manipulator's operation.

## 10.5 Case Study II: Stewart–Gough Platform

The dynamic analysis of the Stewart-Gough platform has been thoroughly developed through a range of methodologies, as outlined in previous chapters. Chapter 7 provides a comprehensive analysis of the robot's dynamics using the Newton-Euler Method, detailing the forces and moments influencing the robot. Chapter 8 further elaborates on the closed-form dynamics of the robot, presenting a more direct and potentially more computationally efficient approach. Section 8.5 outlines the fundamental equations of motion that define the robot's dynamic responses. The previous section's simulation

results indicate that the closed-form formulation offers fewer challenges compared to the implicit dynamics, particularly in the context of closed-loop simulations. Consequently, this section adopts the closed-form formulation given in Section 8.5 for the reported simulation results.

### 10.5.1 Forward Dynamics Simulation

As previously discussed, dynamic formulations are essential for executing forward dynamics simulations. This approach involves forecasting the trajectory of a manipulator's moving platform based on known actuator forces and external disturbance wrenches. Figure 10.9 illustrates this process, aiming to determine the impact of these inputs on the platform's motion.

In the setup shown in Figure 10.9, the Stewart-Gough platform's closed-form dynamic equations, outlined in Equation 7.106 and elaborated upon in Equation 7.130, are used to calculate the system's response. These equations are integrated numerically using the Runge-Kutta method, which is well-regarded for its accuracy in solving complex differential equations.

The simulations focus on a 3-6 Stewart-Gough platform, depicted in Figure 6.12. This platform features six identical cylinder-piston limbs that are crucial for its operation. Accurate simulation requires detailed information on the manipulator's geometrical and inertial parameters, as specified in Table 10.2. These parameters include

dimensions, masses, and other significant physical attributes that affect the system's dynamics.

**TABLE 10.2**

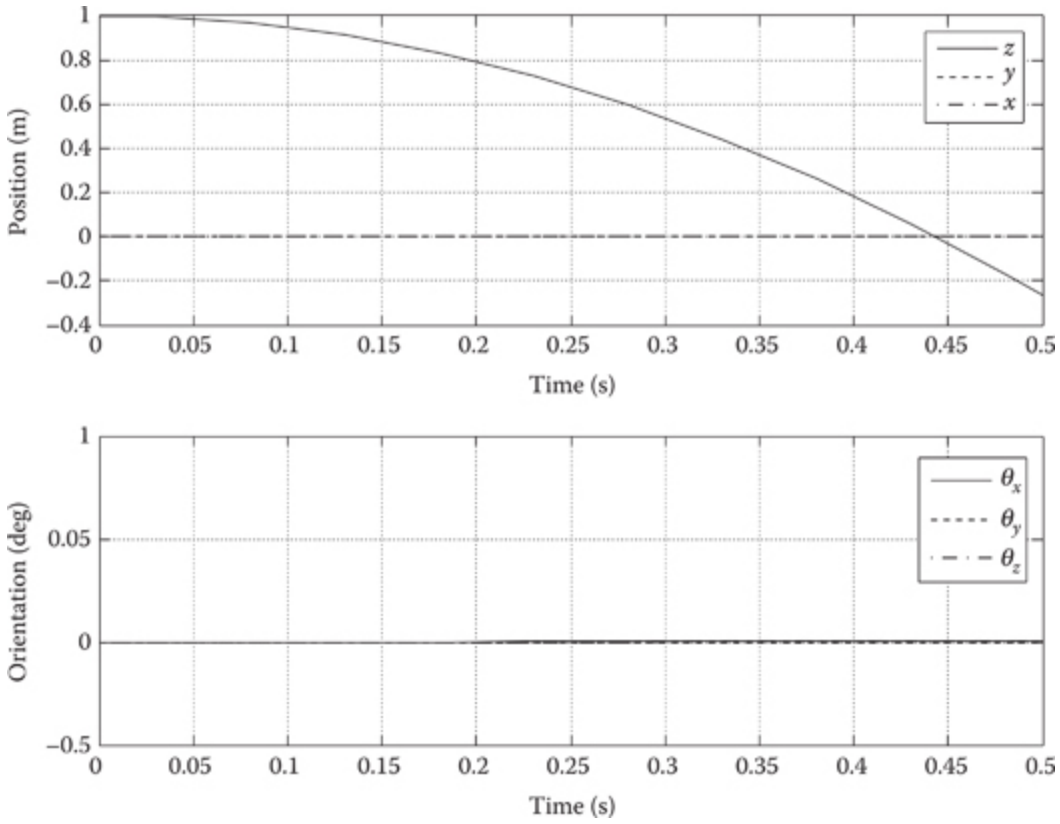
Geometric and Inertial Parameters of a 3–6 Stewart–Gough Platform ⏎

| | Description | Quantity |
|---|---|---|
| $R_A$: | Fixed base radius | 2m |
| $R_B$: | Moving platform radius | 1m |
| $\theta_{A_i}$: | Angle of fixed base points $A_i$'s | $[0^\circ, 60^\circ, 120^\circ, 180^\circ, 240^\circ, 300^\circ]$ |
| $\theta_{B_i}$: | Angle of moving platform points $B_i$'s | $[30^\circ, 30^\circ, 150^\circ, 150^\circ, 270^\circ, 270^\circ]$ |
| $m$: | Moving platform mass | 1150kg |
| $m_{i_1}$: | Cylinder mass | 85kg |
| $m_{i_2}$: | Piston mass | 22kg |
| $c_{i_1}$: | Cylinder center of | 0.75m |

| | Description | Quantity |
|---|---|---|
| | mass | |
| $c_{i_2}$: | Piston center of mass | 0.75m |
| $\boldsymbol{I}_p$: | Moving platform moment of inertia | $\text{diag}(570, 285, 285)\text{kg} \cdot \text{m}^2$ |
| $\boldsymbol{I}_{c_{i_1}}$: | Cylinder moment of inertia | $\text{diag}(16, 16, 0)\text{kg} \cdot \text{m}^2$ |
| $\boldsymbol{I}_{c_{i_2}}$: | Piston moment of inertia | $\text{diag}(4.1, 4.1, 0)\text{kg} \cdot \text{m}^2$ |

Several simulations were conducted to evaluate the precision and validity of the dynamic formulations. In the first reported simulation, the manipulator was initialized at its central equilibrium position, $\boldsymbol{\chi}_0 = [0,\, 0,\, 1,\, 0,\, 0,\, 0]^T$. In this scenario, it is assumed that all actuator forces are zero ($\boldsymbol{\tau} = \mathbf{0}$), and the only external force acting on the manipulator is gravity, $\boldsymbol{g} = [0,\, 0,\, -9.8]^T$. Given these conditions, the expected behavior is a vertical motion along the $z$-axis, with no angular rotations. The simulation results, as depicted in Figure 10.22, confirmed this expectation. The manipulator exhibited pure vertical motion, consistent with the prediction that gravity alone

would lead to movement along the $z$-axis while leaving the other degrees of freedom unaffected.



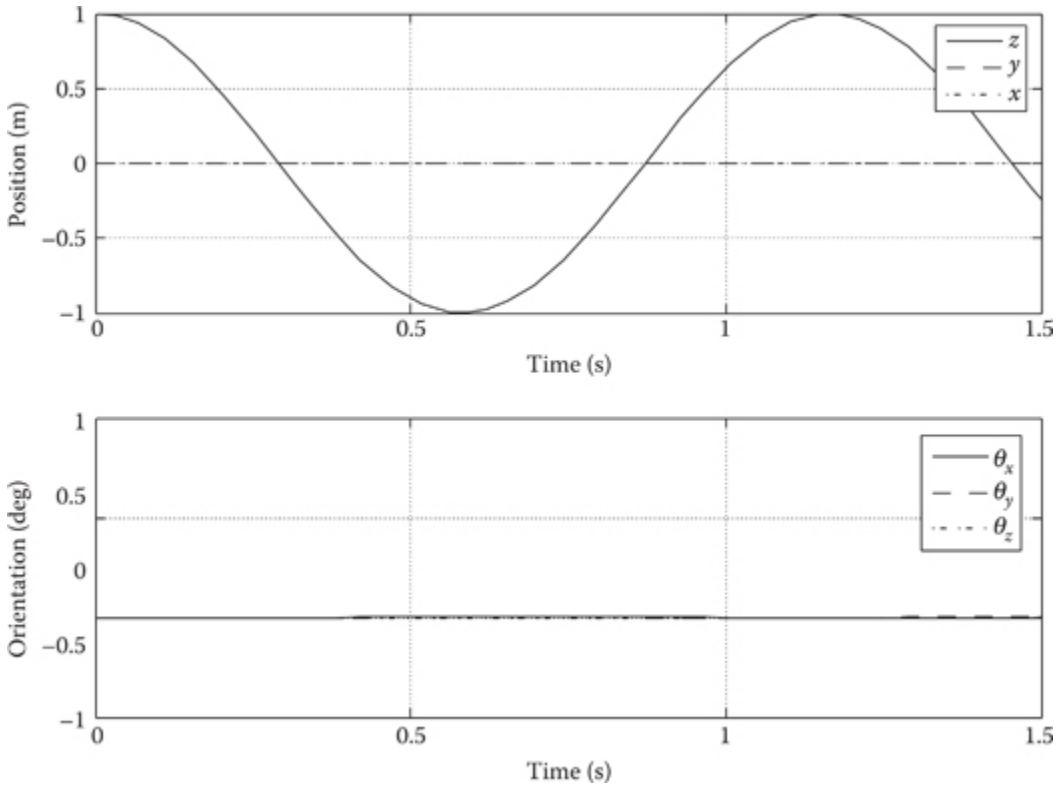

▶ Long Description for Figure 10.22

**FIGURE 10.22**

First simulation scenario for the Stewart–Gough platform with $\boldsymbol{g} = [0,\ 0,\ -9.8]^T$ and $\boldsymbol{\tau} = \mathbf{0}$. ⏎

In the second simulation, as illustrated in Figure 10.23, all actuator forces are uniformly set to $-10\,\text{kN}$, while gravitational forces are excluded ($\boldsymbol{g} = [0,\ 0,\ 0]^T$). This setup aimed to apply a force in the $-z$ direction. Without physical constraints in the simulation, the expected result is that the motion would be directed along $-z$, and would

continue even after the moving platform came into contact with the fixed base. This would lead to an oscillatory motion with constant amplitude. The simulation, which ran for 1.5 seconds, produced results shown in Figure 10.23. The figure illustrates that the positions $x(t)$ and $y(t)$, along with the orientation angles $\boldsymbol{\theta}$, remained at zero, while $z(t)$ exhibited oscillatory behavior. The oscillation amplitude was such that the moving platform's center of mass crossed the fixed base as anticipated and returned to its initial position within about 1.2 seconds. These results validate the dynamic formulation and confirm that the simulation accurately reflects the expected system behavior.



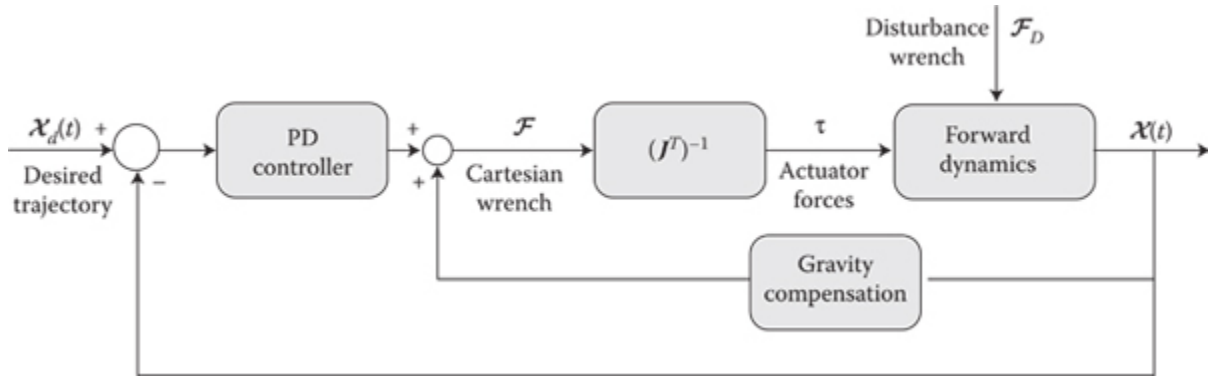

▶ Long Description for Figure 10.23

**FIGURE 10.23**

Second simulation scenario for the Stewart–Gough platform in zero gravity field, applying the following actuator forces $\boldsymbol{\tau} = [-10,\ -10,\ -10, -10,\ -10,\ -10]^T\ kN$. ⏎

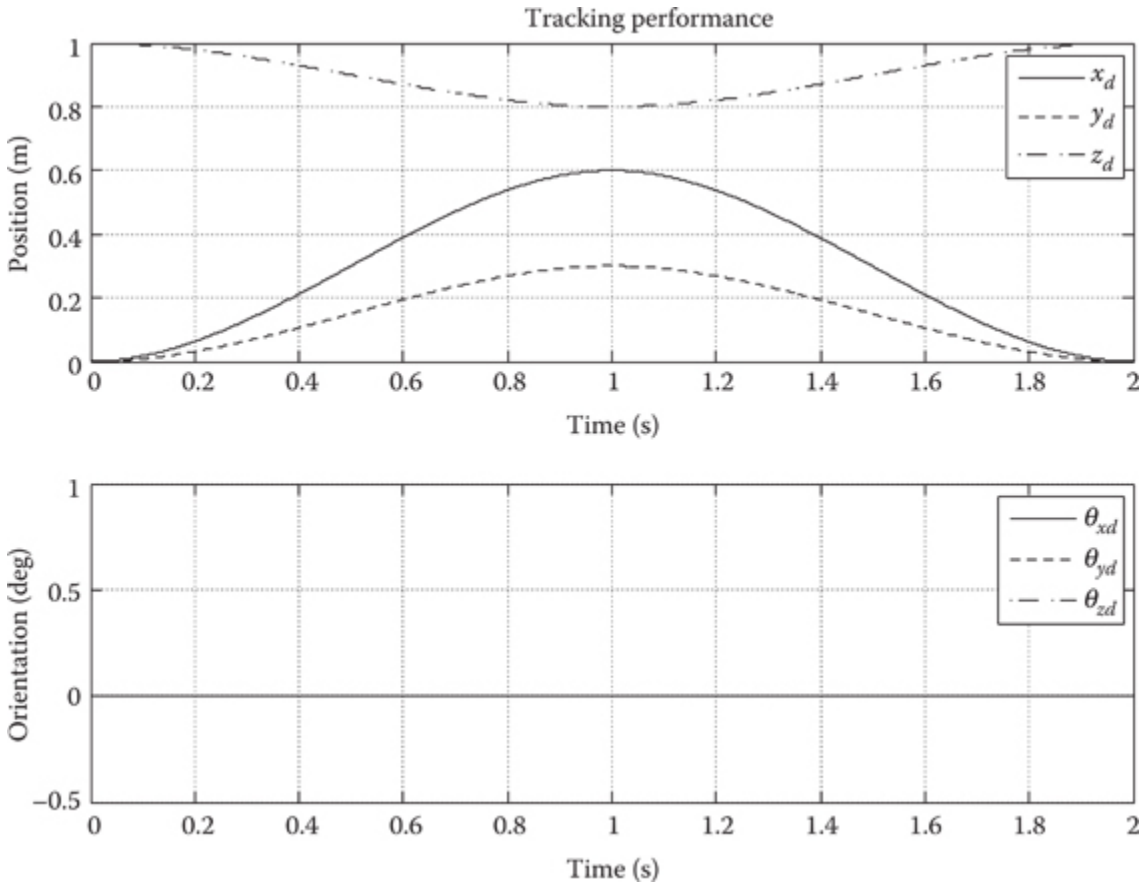
### 10.5.2 Closed-Loop Dynamics Simulation

In the third simulation, detailed in Figure 10.24, the forward dynamics of the manipulator are analyzed through the application of a closed-loop control system. This simulation is designed to execute a predefined trajectory for the moving platform of the manipulator, as shown in Figure 10.25. To achieve this trajectory, the simulation uses a closed-loop control mechanism that continuously adjusts the actuator forces based on the real-time discrepancies between the actual and desired platform motions.




▶ Long Description for Figure 10.24

**FIGURE 10.24**
Block diagram of the forward dynamics implementation for fully parallel manipulators in closed loop. ⏎




▶ Long Description for Figure 10.25

**FIGURE 10.25**
The desired and output trajectories of the Stewart–Gough manipulator in closed loop. ⏎

In this closed-loop framework, the trajectory of the moving platform, denoted as $\boldsymbol{\chi}(t)$, is consistently monitored and compared against the target trajectory, $\boldsymbol{\chi}_d(t)$. The deviation between these trajectories, referred to as the tracking error $\boldsymbol{e}(t) = \boldsymbol{\chi}_d(t) - \boldsymbol{\chi}(t)$, is input into a feedback controller. This controller processes the tracking error to compute the necessary wrench, which includes the force and torque required to be applied to the moving

platform to correct any deviations and ensure the trajectory is accurately followed.

For this simulation, a basic PD controller is used to address each component of the tracking error vector $\boldsymbol{e}(t)$. Additionally, a gravity compensation term, $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$, is incorporated into the control strategy. This term, derived from the Stewart–Gough platform's dynamic equations as specified in Equation 7.130, accounts for the gravitational forces acting on the manipulator, thereby helping to maintain accurate tracking by compensating for the effects of gravity.

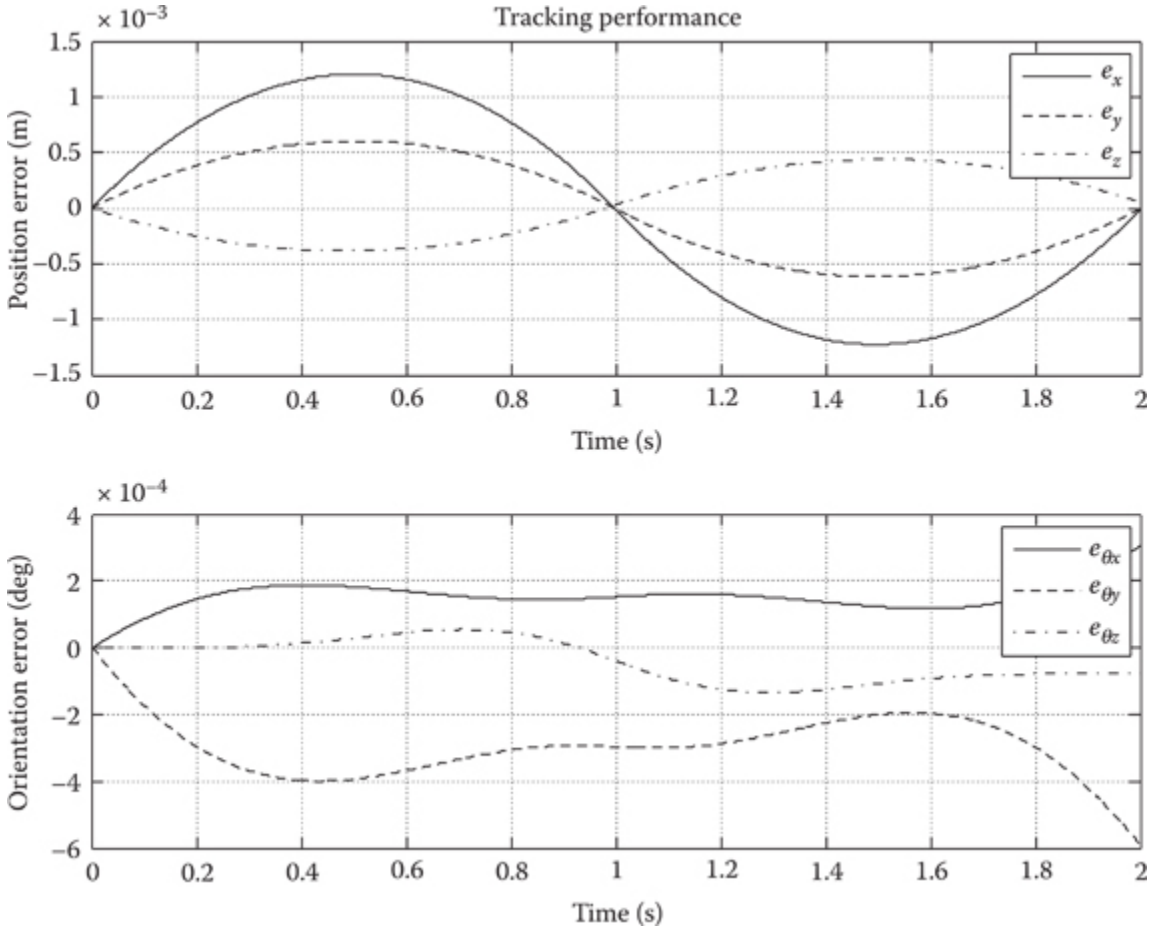
Once the necessary wrench $\boldsymbol{\mathcal{F}}$ is computed, it must be converted from Cartesian coordinates to joint space to determine the required actuator forces. This conversion is performed using the Jacobian matrix $\boldsymbol{J}^T$, as outlined in Equation 6.9, which relates the Cartesian wrench to actuator forces. In the case of the Stewart–Gough manipulator, which is a fully parallel manipulator without redundant actuators, $\boldsymbol{J}^T$ is a square matrix, and as a result, the actuator forces can be directly calculated from $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$, provided the manipulator is in a nonsingular configuration.

In this simulation, we utilized the geometric and inertial parameters specified in Table 10.2, while both the disturbance wrench and initial conditions were set to zero. The trajectory for the moving platform is defined using smooth cubic polynomial functions for the coordinates $x_d$, $y_d$, and $z_d$, as illustrated in Figure 10.17. These cubic

polynomials were chosen for their ability to create smooth, continuous paths that facilitate precise tracking. Orientation angles were fixed at zero for this particular simulation, focusing solely on translational movements.

The simulation was conducted over two seconds, PD gains set at $\boldsymbol{K}_P = 10^4 \cdot \boldsymbol{I}_6$ and $\boldsymbol{K}_D = 10^6 \cdot \boldsymbol{I}_6$, where $\boldsymbol{I}_6$ is a $6 \times 6$ identity matrix. These gain values were selected to balance responsiveness and stability, aiming to achieve accurate tracking of the predefined trajectory while maintaining control system stability. The use of high PD gains is common in control systems to minimize tracking errors and improve performance.

The results of the closed-loop tracking performance are depicted in Figures 10.25 and 10.26. Figure 10.25 shows that the actual motion of the platform closely adheres to the desired trajectory, confirming the system's effectiveness in following the planned path. The trajectory tracking is visually validated by comparing the actual and desired positions over time. Figure 10.26 provides a detailed analysis of tracking accuracy, illustrating that the PD controller, with the addition of gravity compensation, achieved exceptional performance. The tracking errors were minimized to below 1.5 *mm* in position and less than 0.001° in orientation, which underscores the controller's effectiveness in achieving high precision.




▶ Long Description for Figure 10.26

**FIGURE 10.26**

The closed-loop tracking performance of the Stewart–Gough manipulator. ⏎

The primary objective of this simulation was to investigate the practicality of implementing forward dynamics within a closed-loop control framework. By using a basic PD controller, the focus was placed on the core aspects of forward dynamics and control performance. The inclusion of gravity compensation feedback was particularly notable, as it significantly contributed to the high level of

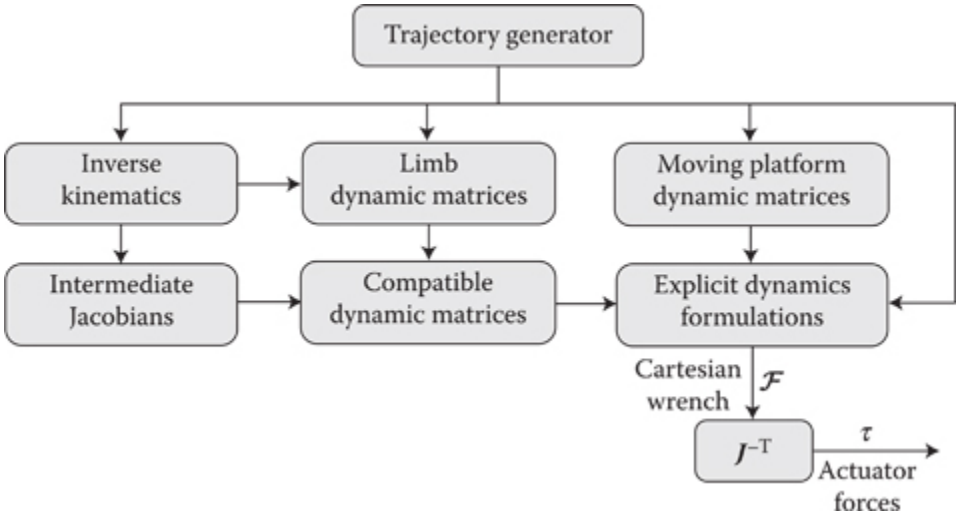
accuracy achieved. This feedback mechanism corrected for gravitational effects, thereby enhancing the overall control system performance. The results illustrate that even with a simple PD control structure, precise control and minimal tracking errors can be accomplished, provided that gravitational effects are adequately accounted for. This reinforces the effectiveness of forward dynamics in closed-loop systems and sets the stage for more complex simulations and optimizations in future studies.

### 10.5.3 Inverse Dynamics Simulation

The process of inverse dynamics for the robot involves calculating the required actuator forces or torques needed to achieve a specific motion along a predefined trajectory, as illustrated in Figure 10.11. In this process, the trajectory of the manipulator is considered predetermined, and the primary goal is to determine the actuator forces or torques necessary to follow this trajectory accurately. This approach is crucial for ensuring that the manipulator can execute complex trajectories as designed, translating high-level commands into precise mechanical actions.

For the Stewart-Gough platform, the implementation of inverse dynamics is carried out through a carefully structured series of steps, as depicted in Figure 10.27. The initial stage involves defining the trajectory that the manipulator's moving platform is expected to follow. In this simulation, a cubic polynomial trajectory planner is used to establish this path. The cubic polynomial trajectory not only

specifies the desired position $\boldsymbol{\chi}_d(t)$ but also provides its first and second time derivatives, $\dot{\boldsymbol{\chi}}_d(t)$ and $\ddot{\boldsymbol{\chi}}_d(t)$, respectively.




▶ Long Description for Figure 10.27

**FIGURE 10.27**
Flowchart of inverse dynamics implementation sequence. ⏎

Following this, the inverse kinematics of the manipulator is tackled to compute the positions and velocities of each limb. This involves calculating the linear and angular positions, velocities, and accelerations of the manipulator's limbs according to the predetermined trajectory. An integral part of this process is the computation of the manipulator's Jacobian matrix $\boldsymbol{J}$, which plays a crucial role in translating joint velocities into end-effector velocities.

Once these kinematic parameters are determined, attention shifts to evaluating the dynamic matrices for both the individual limbs and the moving platform. This

evaluation is based on the closed-form formulations detailed in Equations 7.116 and 7.120. To derive the comprehensive dynamics of the manipulator, the initial step involves calculating the intermediate Jacobian matrices $\boldsymbol{J}_i$, as detailed in Equation 7.122. Once these matrices are determined, the next task is to compute the corresponding dynamic matrices for the limbs, following the guidelines specified in Equation 7.127.

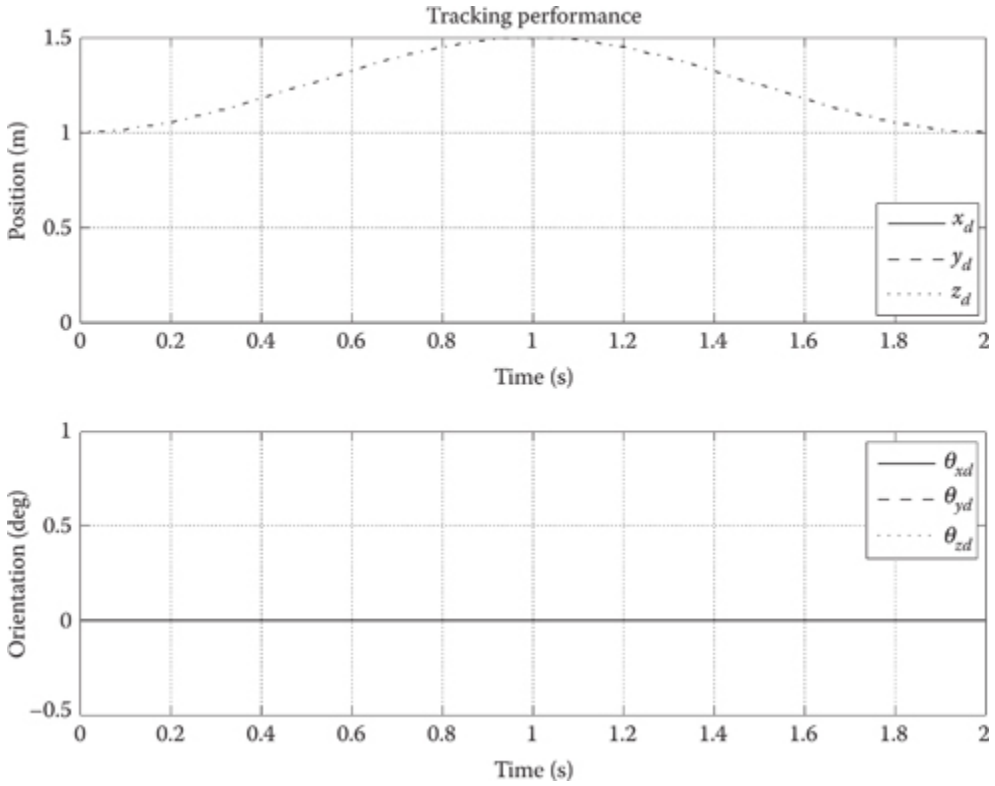
With all the necessary terms for determining the actuator forces identified, the Cartesian wrench $\boldsymbol{\mathcal{F}}$ applied to the moving platform is defined. This wrench is calculated by aggregating all inertial and external forces, while excluding the actuator torques $\boldsymbol{\tau}$ in the closed-form dynamic formulation provided in 7.106. Consequently, $\boldsymbol{\mathcal{F}}$ represents the resultant force and moment in Cartesian space resulting from these combined forces, without including the actuator contributions.

Given that the Stewart-Gough platform features no redundant actuation, the Jacobian matrix $\boldsymbol{J}$ is square, facilitating the precise calculation of actuator forces from the Cartesian wrench. Specifically, actuator forces $\boldsymbol{\tau}$ can be derived from $\boldsymbol{\mathcal{F}}$ using the inverse transpose of the Jacobian matrix, as long as $\boldsymbol{J}$ remains nonsingular. This relationship is expressed by the equation $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$. Therefore, in simulations, the actuator forces $\boldsymbol{\tau}$ are computed using the aforementioned equation.

$$\boldsymbol{\tau} = \boldsymbol{J}^{-T}\left(\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) + \boldsymbol{\mathcal{F}}_d\right).$$

(10.25)

In this section, we explore a series of simulations designed to thoroughly examine the dynamic behavior of the Stewart-Gough platform (SGP). The initial simulation focuses on a basic trajectory where the motion of the platform is restricted to movement along the $z$-axis. This scenario is illustrated in Figure 10.28, which depicts a vertical trajectory with no changes in position along the $x$ or $y$ axes and no variations in orientation. Gravity acts in the negative $z$ direction, and for this simulation, we assume the absence of any external disturbance forces ($\boldsymbol{\mathcal{F}}_D = \mathbf{0}$).

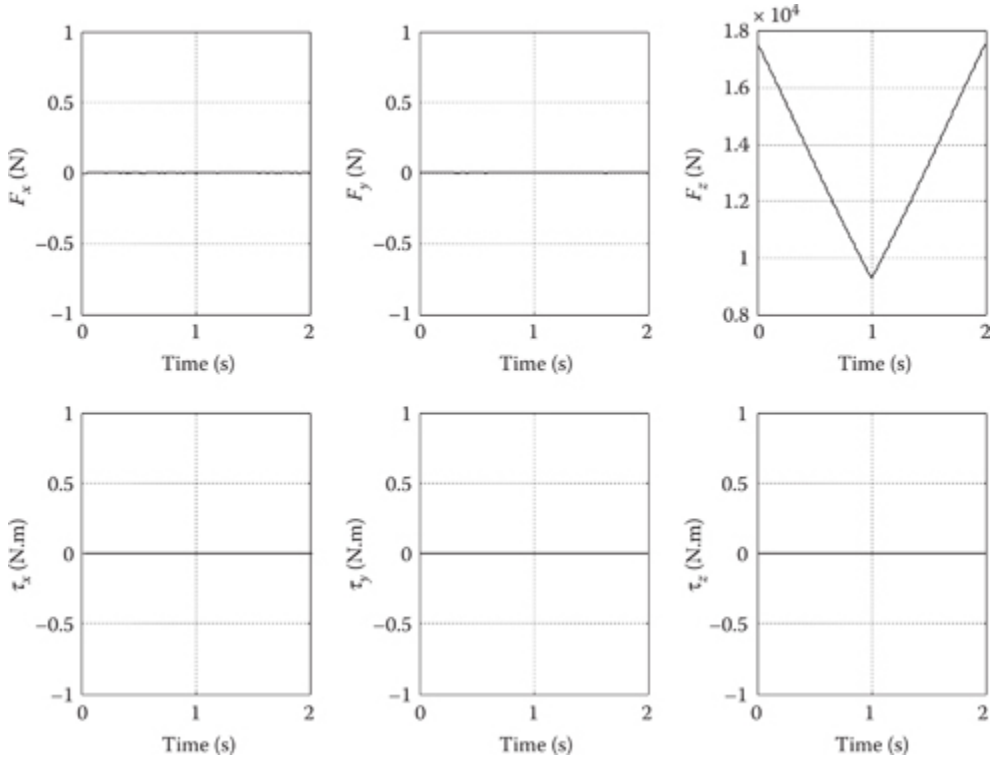



▶ Long Description for Figure 10.28

**FIGURE 10.28**

A typical trajectory that provides motion only in the $z$ direction for the Stewart–Gough Platform. ⏎

The results of this simulation are presented in Figures 10.29 and 10.30. Figure 10.29 shows the Cartesian wrench $\boldsymbol{\mathcal{F}} = [F_x, F_y, F_z, \tau_x, \tau_y, \tau_z]^T$ required to follow the specified vertical trajectory. As expected, all components of the Cartesian wrench are zero except for $F_z$, the vertical force component. This observation is consistent with the linear nature of the desired trajectory, with gravity contributing to a noticeable shift in the $F_z$ component from zero. The impact of gravity on the platform's motion is clearly reflected in this result, highlighting how gravitational forces influence the required wrench to achieve the vertical trajectory.



▶ Long Description for Figure 10.29

**FIGURE 10.29**

The Cartesian wrench $\mathcal{F}$ of the Stewart–Gough Platform required to produce the desired motion depicted in Figure 10.28. ⏎

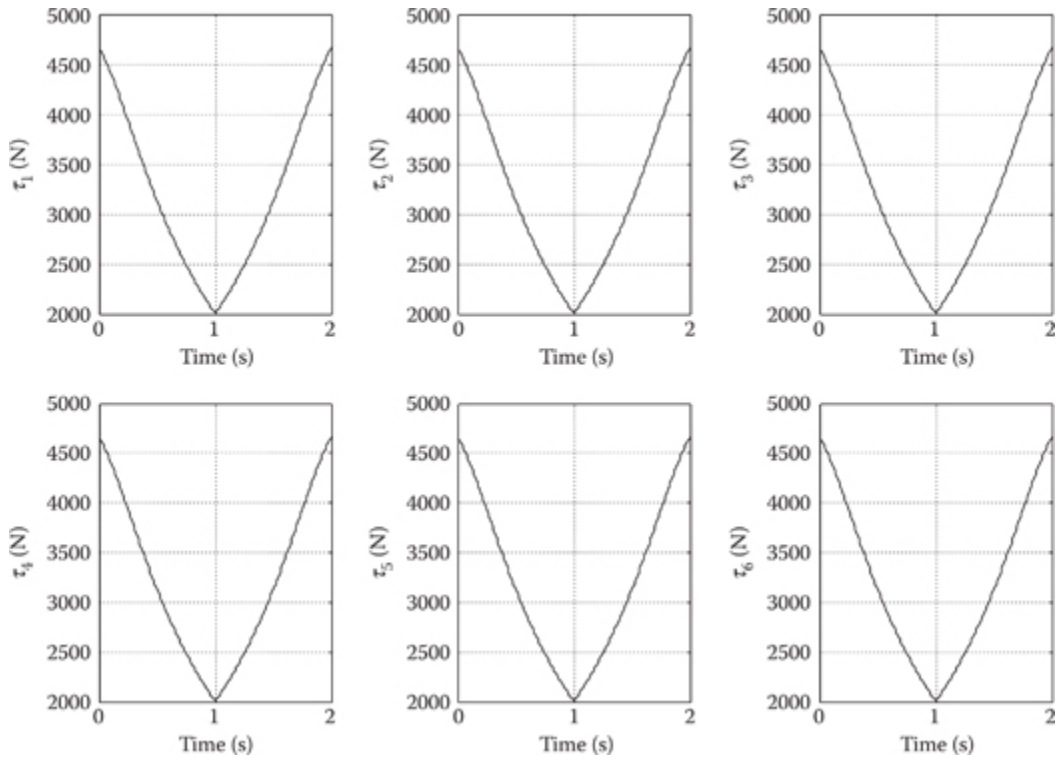


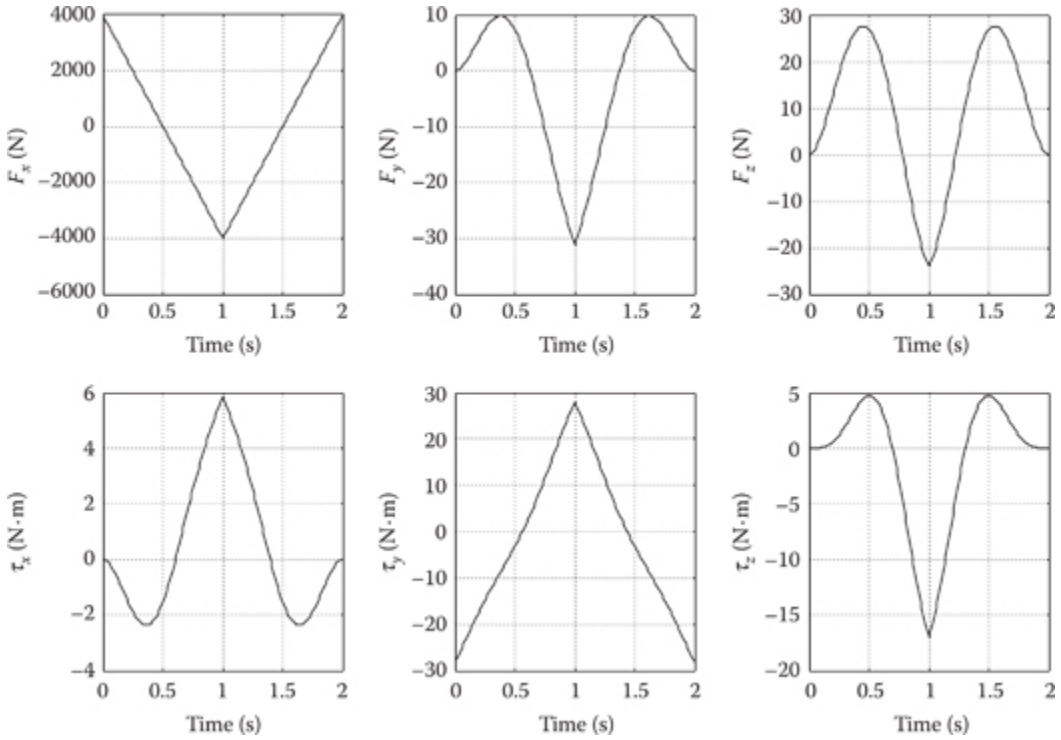

**FIGURE 10.30**

The actuator forces of the Stewart–Gough manipulator required to produce the desired motion depicted in Figure 10.28. ⏎

Figure 10.30 illustrates the actuator forces necessary to produce the vertical motion described. The symmetry of the SGP's structure is evident from the equal distribution of forces across all actuators. This symmetry ensures that

identical actuator forces are required to maintain vertical movement, demonstrating the uniformity of force distribution needed to counteract gravitational effects.

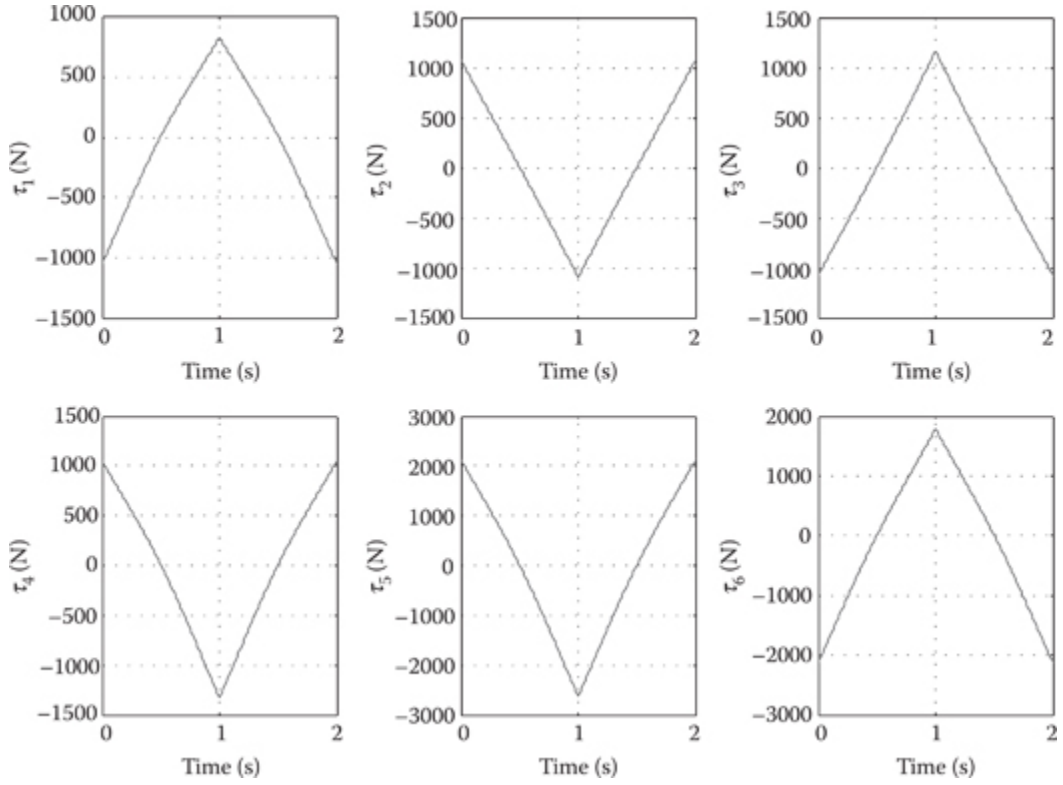
In the subsequent simulation, we focus on analyzing the inverse dynamics of the Stewart-Gough platform for a pure motion along the $x$ axis, as detailed in Figures 10.31 and 10.32. In this scenario, we deliberately exclude the effects of gravity and any external disturbance forces to isolate the impact of horizontal motion alone.




▶ Long Description for Figure 10.31

**FIGURE 10.31**

The Cartesian wrench $\mathcal{F}$ of the Stewart–Gough manipulator required to produce the desired motion in only the $x$ direction. ⏎




▶ Long Description for Figure 10.32

**FIGURE 10.32**

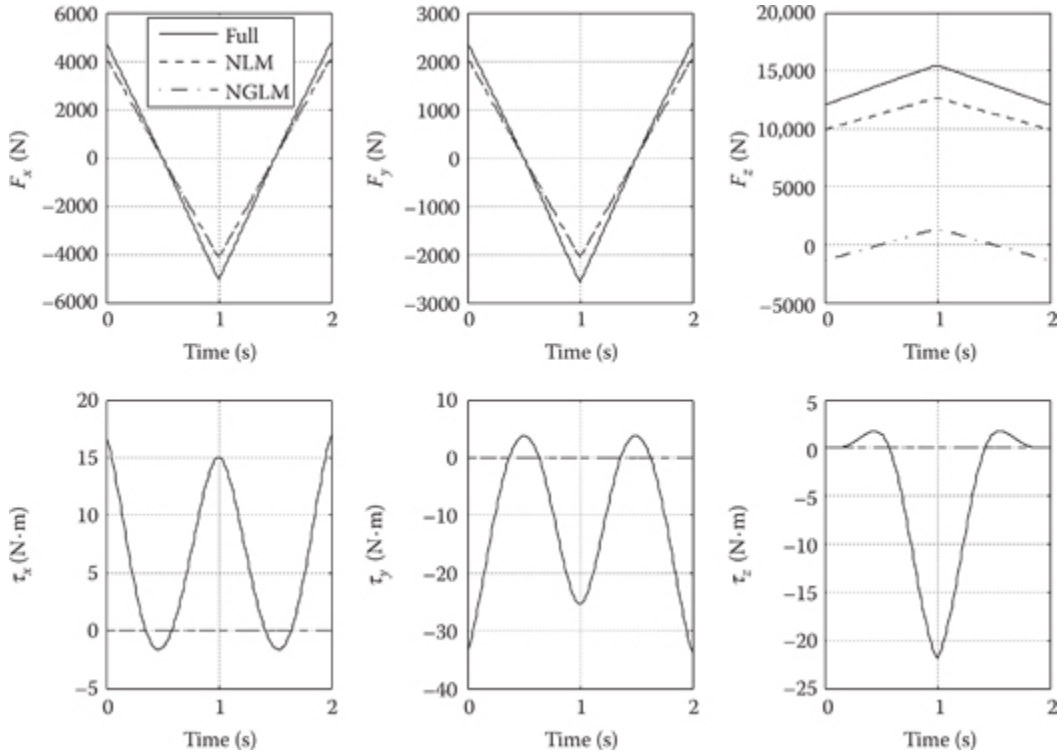
The actuator forces of the Stewart–Gough manipulator required to produce the desired motion in only the $x$ direction. ⏎

Figure 10.31 presents the Cartesian force $F_x$, which is required to achieve the desired movement in the $x$ direction. The figure illustrates that this force is in the range of several kilonewtons, indicating a substantial force requirement for horizontal displacement. Notably, the other Cartesian wrench components—such as $F_y$, $F_z$, and the torques $\tau_x$, $\tau_y$, and $\tau_z$—are relatively minor or negligible

in this context, reflecting that the simulation is focused purely on horizontal motion.

In addition, Figure 10.32 shows the actuator forces necessary to produce the horizontal motion. Unlike the vertical motion scenario, where the actuator forces were distributed symmetrically due to the nature of vertical movement and gravitational effects, the horizontal motion does not exhibit such symmetry. Instead, the forces required are more evenly distributed among all six actuators. This result highlights the impact of motion direction on the distribution of actuator forces, showing that the horizontal motion necessitates a more balanced force allocation across the actuators compared to the vertical case. Overall, these two simulations underscores how different types of motion—whether vertical or horizontal—affect the force requirements and distribution across the actuators in the Stewart-Gough platform.

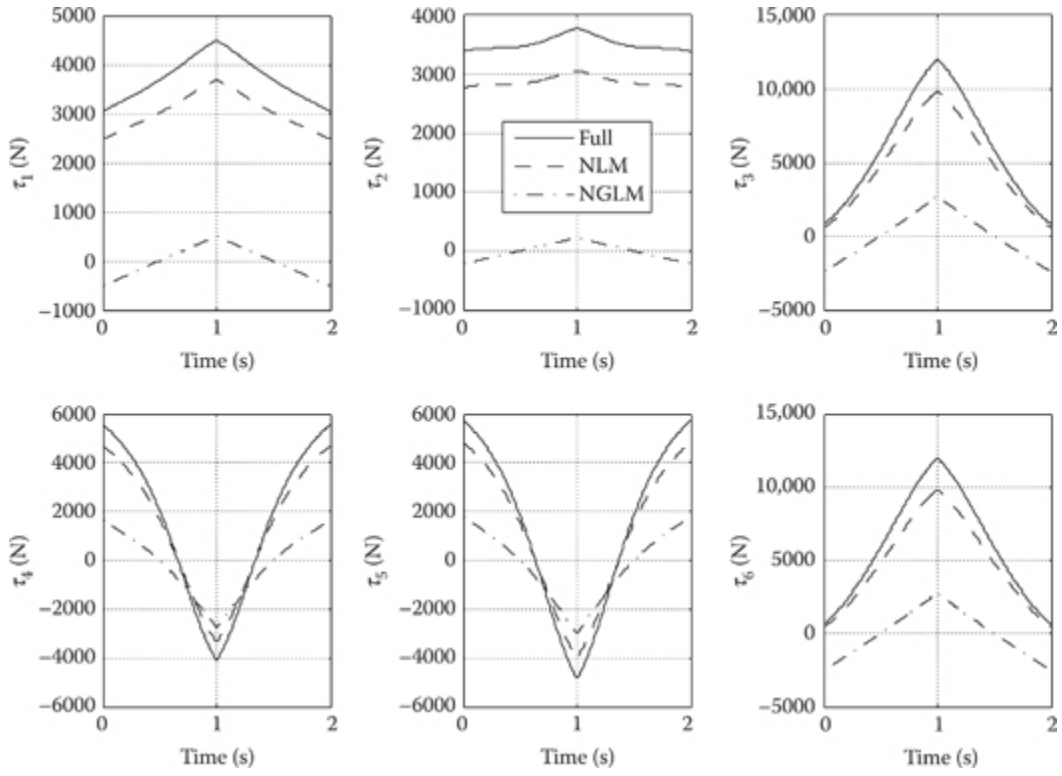
The third simulation investigates a more complex trajectory, as illustrated in Figure 10.25. This analysis aims to evaluate how gravity and limb masses influence the actuator forces required for the manipulator's operation. Figures 10.33 and 10.34 present the findings, comparing three scenarios: total forces (solid lines), forces calculated without accounting for limb masses (NLM, dashed lines), and forces excluding both gravity and limb masses (NGLM, dash-dotted lines).




▶ Long Description for Figure 10.33

**FIGURE 10.33**

The Cartesian wrench $\mathcal{F}$ of the Stewart–Gough manipulator required to produce a desired motion depicted in Figure 10.25; The total force (solid), neglecting limb masses (dashed), and neglecting gravity and limb masses (dash-dotted). ⏎




▶ Long Description for Figure 10.34

**FIGURE 10.34**

The actuator forces of the Stewart–Gough manipulator required to produce the desired motion depicted in Figure 10.25; The total force (solid), neglecting limb masses (dashed), and neglecting gravity and limb masses (dash-dotted). ⏎

In Figure 10.33, it is evident that excluding limb masses results in a 20% reduction in the vertical Cartesian force $F_z$. This observation underscores the significant impact of the cylinder and piston actuators' weights on the required vertical force. When gravity is also disregarded, the required force decreases dramatically by an order of magnitude. This dramatic reduction highlights that a

considerable amount of the power used by the manipulator is allocated to counteract its own weight. For horizontal movements in the $x$ and $y$ directions, the contribution of limb masses to the total Cartesian force is approximately 15% for the trajectory analyzed. This indicates that while gravity has a more pronounced effect, the weight of the limbs still plays a non-negligible role in determining the force requirements.

Figure 10.34 offers a detailed view of the actuator forces required under different conditions. It illustrates how the distribution of these forces varies with the inclusion or exclusion of limb masses and gravity effects. By comparing these scenarios, the figure provides valuable insights into how these factors influence the performance and efficiency of the Stewart-Gough platform. The detailed analysis reveals how the manipulator's design and operating conditions can significantly affect the force distribution and overall control strategy, providing a deeper understanding of the dynamics involved.

## 10.6 Dynamic Calibration

So far, we have examined different methods for modeling the dynamics of parallel robots. The dynamic model of robots, determined by any method, includes both kinematic and dynamic parameters, and the accuracy of predicting the dynamic behavior of a robot relies on the precise

knowledge of these parameters. In the robotics literature, the precise identification of these parameters is studied under the title of *Robot Calibration.*

Robot calibration methods have been developed for two main aspects: kinematic and dynamic parameters. While the fundamental principles for both are similar, the implementation details differ. In kinematic calibration, the focus is on the robot's geometrical and kinematic parameters. In contrast, dynamic calibration methods aim to determine mass properties, which generally consist of ten parameters for each link, including the link mass, three parameters defining the center of mass, and six parameters describing the inertia matrix. Moreover, the dynamic calibration process can also involve examining and estimating frictional forces and disturbances affecting the robot dynamics.

It should be noted that depending on how the robot moves, many of the dynamic parameters may remain unexcited and challenging to determine in practice. For example, if we consider a planar robot, the dynamic parameters of each link reduce to four parameters, namely, the link mass, two parameters defining the center of mass in the plane, and only one parameter describing the inertia matrix about out of plane motion. By reducing the general dynamic parameters to a limited number, we will be able to identify the base inertial parameters (BIP) [223]. These base inertial parameters are the fundamental inertial

parameters that can be experimentally identified in dynamic calibration methods.

The first and simplest method for estimating the kinematic and dynamic parameters of a robot is to use computer-aided design (CAD) software to approximate nominal values for these parameters. Although these software tools are capable of accurately determining these parameters in the three-dimensional model of robot components, they usually lack precision in determining the accurate parameters of complex robot structures composed of multiple moving parts [224]. These nominal values can often be used as initial values in the robot model and in dynamic simulation.

The primary methods for robot calibration are based on linear regression models. In this approach, the robot's kinematic or dynamic model is expressed as a linear regression in terms of base parameters, and least squares linear regression methods are employed to determine these parameters. To implement these methods in dynamic calibration, motion and force/torque data are recorded during a maneuver, and then these data are utilized in the least squares methods to determine the parameters of the linear regression model of the robot.

Calibration methods can be readily applied to complex robot models obtained in the form of linear regression models, as derived in Equation 8.49. A general treatment of the calibration process will be followed.

### 10.6.1 Linear Regression Models

Consider the dynamic model of the parallel robot, given in Equation 8.49, which is reiterated here for easier follow-up.

$$\boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{Y}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}},\ddot{\boldsymbol{\chi}})\boldsymbol{\theta}. \tag{10.26}$$

Although this model is a differential equation with respect to the motion variables of the robot $\boldsymbol{\chi}$, it has a linear algebraic relationship with respect to the dynamic parameters of the robot. Transforming this model into a linear regression form with calibration parameters allows for leveraging methods rooted in least squares error to effectively determine these parameters.

For this purpose, let us focus on the linear regression model with respect to the dynamic parameters $\boldsymbol{\theta}$, represented by the matrix $\boldsymbol{Y}$, and the input–output relation of this mapping through the actuator forces/torques $\boldsymbol{\tau}$. This model could be formulated in several cases, considering the practical consideration of collecting input-output data. In the first case, assume that during a typical maneuver in the robot's workspace, motion information $\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}}$ and the robot's actuator forces/torques $\boldsymbol{\tau}$ are recorded at a number of time instances $i = 1, 2, \ldots, N$ throughout the maneuver.

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_N \end{bmatrix}, \quad \boldsymbol{\chi} = \begin{bmatrix} \mathscr{X}_1 \\ \mathscr{X}_2 \\ \vdots \\ \mathscr{X}_N \end{bmatrix}, \quad \dot{\boldsymbol{\chi}} = \begin{bmatrix} \dot{\mathscr{X}}_1 \\ \dot{\mathscr{X}}_2 \\ \vdots \\ \dot{\mathscr{X}}_N \end{bmatrix}, \quad \ddot{\boldsymbol{q}} = \begin{bmatrix} \ddot{\mathscr{X}}_1 \\ \ddot{\mathscr{X}}_2 \\ \vdots \\ \ddot{\mathscr{X}}_N \end{bmatrix}, \quad i = 1, 2$$

(10.27)

In this scenario, if we represent the calibration parameters with the vector $\boldsymbol{\theta}$ and the linear regression matrix with $\boldsymbol{Y}$, similar to what has been explained in Section 8.3.3, the robot joint model can be expressed in a general linear regression form.

$$\boldsymbol{J}^T \boldsymbol{\tau} = \boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}})\boldsymbol{\theta}.$$

(10.28)

Given this relationship, we observe that the vector of calibration parameters for robot parameters consists of a limited number of base inertial parameters (BIP). This is despite the fact that, for the identification of these limited parameters denoted by $p$, one can utilize information from the motion data of the robot at $N$ time instances, where $N \gg p$.

In the second case, suppose that in logging the robot's motion data, only the motion of the moving platform $\boldsymbol{\chi}$ is measured, and their corresponding velocity and

acceleration are not directly available. This is common in industrial robots. In this case, various methods can be used to obtain the motion velocity and acceleration using motion data, and this information can be used in the regression matrix.

One effective approach in this regard is the use of derivative filters. Consider the block diagram of the simplest derivative filter in Figure 10.35. This filter, which can be easily implemented in software like MATLAB, Simulink or Python, could be represented by the following transfer function

$$G(s) = \frac{s}{\tau_f s + 1} \tag{10.29}$$

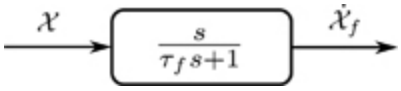



**FIGURE 10.35**
Block diagram of filtered derivative. ⏎

As observed, the Laplace transform variable $s$ in this transfer function is the derivative operator in the Laplace space. If this operator is used alone, signal noise will be amplified during differentiation, leading to undesirable results. The denominator of the transfer function shown in Figure 10.35 represents a very simple first-order low-pass filter with a time constant $t_f$. This part of the transfer

function is used to reduce noise in the differentiated signal. Choosing an appropriate time constant for the filter is dependent on the information stored in the signal and the characteristics of its noise. If we consider the cutoff frequency of the filter as $\omega_f = 1/\tau_f$, it is advisable to choose this frequency to be larger than ten times the joint motion bandwidth $\omega_{bw}$ and smaller than one-tenth of the main noise frequency $\omega_n$ in the measured signal.

$$10\,\omega_{bw} < \omega_f < 0.1\,\omega_n \tag{10.30}$$

In robotics applications, by choosing $20 < \omega_f < 10^3$ or equivalently, $10^{-3} < \tau_f < 0.05$, a suitable filter can be obtained. The implementation of this filter can be easily done in the discrete-time domain using the `filtfilt` command in MATLAB. The advantage of using this command is that it completely eliminates the inherent time delay in applying low-pass filters. However, the execution of this command may not be real-time and is only feasible on stored and recorded data.

Using this filter, the filtered joint angular velocity $\dot{q}_f$ can be obtained based on the stored joint motion data. By applying this filter again to the resulting data, the filtered joint angular acceleration $\ddot{q}_f$ can also be determined. With the determination of these quantities, the linear regression matrix $\boldsymbol{Y}$ is obtained as follows.

$$\boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}_f, \ddot{\boldsymbol{\chi}}_f)\boldsymbol{\theta}.$$

(10.31)

In both cases the linear regression form of the robot could be used to determine the base inertial parameters. The main method for this calculation is based on the least-squares method, which will be elaborated in the upcoming section.

### 10.6.2 Least Squares Method

By modeling the robot dynamics in the form of linear regression, the values of robot's parameters can be obtained using least squares error methods. For this purpose, consider the main linear regression relationship 10.28. Given that the number of calibration parameters $p$ is significantly less than the number of points where the input and output system data is recorded and stored, the regression matrix will be a rectangular matrix with the number of rows $N$, much greater than the number of its columns. This means that the number of equations is much greater than the number of unknowns, and there is no solution that satisfies all equations.

Solving the least squares error minimizes the Euclidean norm of the error in all equations. This solution is obtained from the left pseudo-inverse of the regression matrix.

$$\hat{\boldsymbol{\theta}} \quad = \boldsymbol{Y}^{\dagger}\, \boldsymbol{J}^{T} \boldsymbol{\tau}$$

(10.32)

$$\boldsymbol{Y}^{\dagger} \quad = \left(\boldsymbol{Y}^{T} \boldsymbol{Y}\right)^{-1} \boldsymbol{Y}^{T}$$

(10.33)

As seen in this relation the kinematic parameters for the robot shall be known, for direct evaluation of base inertial parameters. Otherwise, the Jacobian matrix of the robot $\boldsymbol{J}$ is unknown, this formulation could not be directly used for the robot parameter derivation.

Given the large number of rows in the regression matrix and the possibility of having more parameters than the basic inertia parameters in the general robot calibration problem, the direct calculation of the inverse matrix $\boldsymbol{Y}^{T}\boldsymbol{Y}$ is not recommended. Usually, appropriate computational methods are used when the condition number of the matrix is large. One efficient method used in the `pinv` command in MATLAB is the Householder reflection method. In this method, the matrix $\boldsymbol{Y}$ is decomposed as follows.

$$\boldsymbol{Y}_{N\times p} \boldsymbol{P}_{p\times p} = \boldsymbol{Q}_{N\times p} \boldsymbol{R}_{p\times p}$$

In this expression, $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{Q}$ is an orthogonal matrix, and $\boldsymbol{R}$ is an upper triangular matrix.

Using this decomposition, the pseudo-inverse of the regression matrix is easily calculated as follows.

$$\boldsymbol{Y}^{\dagger} = \boldsymbol{P}\boldsymbol{R}^{-1}\boldsymbol{Q}^{T} \tag{10.34}$$

In this expression, the inverse of the matrix $\boldsymbol{R}$ can be easily determined through recursive calculations.

### 10.6.3 Evaluation of Calibrated Model

Evaluation of the accuracy of the obtained model is crucial in any modeling procedure. To evaluate the accuracy of the obtained model, the calibration results can be used in multiple experiments. If the assumed model and the robot parameters are perfectly matched, the robot's trajectory of motion will not affect the accuracy of the calculations. However, both the robot dynamic model and the calibrated parameters have limited accuracy; thus, the calibration process must be repeated in several different experiments.

For this purpose, by considering various motion maneuvers in the robot's workspace, data recorded from the input and output of each of these motion maneuvers is stored as an independent experiment. Then, by modeling the robot in the form of linear regression for each of these experiments, the calibrated dynamic parameters of the robot are determined. If the calculation results in different experiments are relatively close to each other, it can be

concluded that the assumed model is close enough to reality, and the calibrated dynamic parameters of the robot are reliable.

However, due to the presence of measurement noise and structural uncertainty in the system model, the accurate match is never achieved, and the results obtained from different experiments are usually different. In this case, the average of the results obtained from the calibration of each of the parameters in different experiments can be considered as a suitable estimate of the nominal value of that parameter. In addition, the standard deviation of the results obtained for each of the parameters in different experiments can also be used as a suitable statistical quantity to assess the confidence level of the obtained result.

In robotics literature, the *Consistency Measure* (CM) is introduced for qualitative evaluation of parametric calibration results. The consistency measure is the ratio of the standard deviation to the mean of the estimated parameter in different experiments [225].

$$\%\mathrm{CM} = \frac{\text{Standard Deviation}}{\text{Mean}} \cdot 100 \tag{10.35}$$

Through multiple calibration experiments, it has been empirically demonstrated that if the value of the consistency measure in practical calibration experiments is

less than 30%, the used model and the obtained parameters are entirely reliable and can be used in the design of a linear controller. With an increase in the value of the consistency measure to around 80%, there is less confidence in the estimated parameters. This means that the model used in the dynamic calibration process is not accurate enough, and it needs refinement through engineering judgment. If the consistency measure is greater than 100%, it indicates that either the employed model or the experimental data used does not have an acceptable level of accuracy, and the obtained results are not reliable. Therefore, the process of calibrating the dynamic parameters of the robot is usually done in several stages, with the results being evaluated at each stage.

---

## Problems

10.1 Review the five-bar mechanism depicted in Figure 4.27 and analyzed in Problems 4.1 and 5.1. The dynamic formulation of this manipulator is derived using different methods in Problems 7.1, 8.1, and 9.1.

a. Provide specific numerical values for the manipulator's geometry and inertial properties, create a typical trajectory for the end effector, and write a MATLAB or Python program to solve the forward dynamics to find the actuator forces needed for this motion.

b. Generate and interpret plots of the trajectory and forces over time, and explain the results intuitively.

10.2 Consider the five-bar mechanism depicted in Figure 4.28 and outlined in Problems 4.2 and 5.2. The dynamic formulation of this manipulator is derived using different methods in Problems 7.2, 8.2, and 9.2.

a. Assign numerical values to the manipulator's geometric and inertial parameters, create a typical trajectory for the end effector, and develop a program in MATLAB or Python to compute the forward dynamics and determine the actuator forces needed for this motion.

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.3 Consider the planar pantograph mechanism shown in Figure 4.29 and detailed in Problems 4.3 and 5.3. The dynamic formulation of this manipulator is derived using different methods in Problems 7.3, 8.3, and 9.3.

a. Provide numerical values to the manipulator's geometric and inertial parameters, create a typical trajectory for the end effector, and write a program in MATLAB or Python to perform forward dynamics and determine the necessary actuator forces for this maneuver.

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.4 Consider the planar three-degrees-of-freedom 3RRR parallel manipulator shown in Figure 4.30 and described in Problems 4.4 and 5.4. The dynamic formulation of this manipulator is derived using different methods in Problems 7.4, 8.4, and 9.4.

a. Create a program in MATLAB or Python using the following numerical values for the manipulator parameters. Generate a typical trajectory for the moving platform and solve the forward dynamics to find the required Cartesian and actuator forces for the maneuver:

$$a_i = b_i = 1, (\mathrm{m}); \quad L = 5, (\mathrm{m}); \quad \ell = 1.5, (\mathrm{m})$$
$$m_{a_i} = m_{b_i} = 1, (\mathrm{kg}); \quad I_{a_i} = I_{b_i} = 0.1, (\mathrm{kg}, \mathrm{m}^2)$$
$$m_e = 5, (\mathrm{kg}); \quad I_e = 1, (\mathrm{kg}, \mathrm{m}^2).$$

Consider the manipulator trajectory as a cubic polynomial with the following initial and final positions and time:

$$\boldsymbol{x}_o = [0, 0, 0]^T; \quad \boldsymbol{x}_f = [0.2, -0.1, \pi/3]^T; \quad t_o = 0; \quad t$$

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.5 Consider the planar three-degrees-of-freedom 4RRR parallel manipulator shown in Figure 4.31 and

discussed in Problems 4.5 and 5.5. The dynamic formulation of this manipulator is derived using different methods in Problems 7.5, 8.5, and 9.5.

a. Develop a program in MATLAB or Python with the following numerical values for the manipulator parameters. Create a typical trajectory for the moving platform and solve the forward dynamics to determine the Cartesian and actuator forces required for this maneuver:

$$\theta_{A_i} = \theta_{B_i} = [-3\pi/4, -\pi/4, \pi/4, 3\pi/4]^T;$$
$$a_i = b_i = 1.5, (\mathrm{m})$$
$$m_{a_i} = m_{b_i} = 1, (\mathrm{kg}); \quad I_{a_i} = I_{b_i} = 0.1, (\mathrm{kg}, \mathrm{m}^2)$$
$$m_e = 5, (\mathrm{kg}); \quad I_e = 1, (\mathrm{kg}, \mathrm{m}^2).$$

Consider the manipulator trajectory as a cubic time polynomial with the following initial and final positions and times:

$$\boldsymbol{x}_o = [0, 0, 0]^T; \quad \boldsymbol{x}_f = [0.2, -0.1, \pi/3]^T; \quad t_o = 0; \quad t$$

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.6 Consider the spatial parallel manipulator depicted in Figure 4.32 and detailed in Problems 4.6 and 5.6. The

dynamic formulation of this manipulator is derived using different methods in Problems 7.6, 8.6, and 9.6.

a. Assign numerical values to the geometry and inertial parameters of the manipulator, create a typical trajectory for the moving platform, and write a program in MATLAB or Python to compute the forward dynamics and determine the Cartesian and actuator forces required for this maneuver.

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.7 Consider the 3UPU parallel manipulator shown in Figure 4.33 and discussed in Problems 4.7 and 5.7. The dynamic formulation of this manipulator is derived using different methods in Problems 7.7, and 8.7.

a. Provide numerical values for the manipulator's geometry and inertial parameters, create a typical trajectory for the moving platform, and develop a MATLAB or Python program to compute the forward dynamics and determine the Cartesian and actuator forces needed for this maneuver.

b. Plot and analyze the trajectory and forces over time, and explain the results intuitively.

10.8 Consider the three-degrees-of-freedom 3UPS manipulator introduced in Reference47, as shown in Figure 4.34 and detailed in Problems 4.8 and 5.8. The

dynamic formulation of this manipulator is derived using different methods in Problems 7.8, 8.8, and 9.7.

a. Assign numerical values to the manipulator's geometry and inertial parameters, create a typical trajectory for the moving platform, and develop a MATLAB or Python program to compute the forward dynamics and determine the Cartesian and actuator forces needed for this maneuver.

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.9 Consider the three-degrees-of-freedom 3RPS manipulator shown in Figure 4.35 and discussed in Problems 4.9 and 5.9. The dynamic formulation of this manipulator is derived using different methods in Problems 7.9, 8.9, and 9.8.

a. Provide numerical values for the manipulator's geometry and inertial parameters, create a typical trajectory for the moving platform, and write a MATLAB or Python program to compute the forward dynamics and find the Cartesian and actuator forces required for the maneuver.

b. Plot and analyze the trajectory and forces over time, and offer an intuitive explanation for the results.

10.10 Consider the three-degrees-of-freedom $3RPS$ manipulator shown in Figure 4.35 and discussed in Problems 4.9 and 5.9. The dynamic formulation of this

manipulator is derived using different methods in Problems 7.10, 8.10, and 9.9.

a. Provide numerical values for the manipulator's geometry and inertial parameters, create a typical trajectory for the moving platform, and write a MATLAB or Python program to compute the forward dynamics and find the Cartesian and actuator forces required for the maneuver.

b. Plot and analyze the trajectory and forces over time, and offer an intuitive explanation for the results.

10.11 Consider the Delta robot depicted in Figure 4.38 and discussed in Problems 4.11 and 5.11. The dynamic formulation of this manipulator is derived using different methods in Problems 7.12, 8.12, and 9.11. The dynamic formulation of this manipulator is derived using different methods in Problems 7.11, 8.11, and 9.10.

a. Assign numerical values to the manipulator's geometric and inertial parameters, create a typical trajectory for the moving platform, and develop a MATLAB or Python program to compute the forward dynamics and determine the Cartesian and actuator forces necessary for this maneuver.

b. Plot and analyze the trajectory and forces over time, and provide an intuitive explanation for the results.

10.12 Consider the Quadrupteron robot illustrated in Figure 4.39 and detailed in Problems 4.12 and 5.12, [55]. The dynamic formulation of this manipulator is derived using different methods in Problems 7.12, 8.12, and 9.11.

a. Provide numerical values for the manipulator's geometric and inertial parameters, create a typical trajectory for the moving platform, and write a MATLAB or Python program to compute the forward dynamics and determine the Cartesian and actuator forces required for this maneuver.

b. Plot and analyze the trajectory and forces over time, and offer an intuitive explanation for the results.

# Part IV

# Control

# 11

# *Linear Motion Control*

DOI: 10.1201/9781003590408-11

This chapter explores the context of linear motion control for robotics application, beginning with an introduction that sets the stage for a comprehensive understanding of the subject. The chapter elaborates on various robot motion control topologies, including motion control in joint and task spaces, and the integration of task space control with joint variable measurement. Each topology is carefully examined, providing insights into their applications and implementation intricacies.

A significant portion of the chapter focuses on decentralized control in parallel robots, highlighting the magic of feedback and its transformative impact. It will be demonstrated that leveraging feedback loops, regardless of the controller type, can facilitate the attainment of multiple objectives within a control system. A key reason behind the success of control systems in practical applications is the ability to achieve desirable tracking performance in the presence of disturbances and uncertainties in the system

model through online observation of system states via feedback. In the context of linear controller design, this chapter covers PD and PID control, and provides an insightful discussion on performance limitations, taking into account factors such as sensor noise and actuator saturation. The inclusion of feed-forward control and multivariable PID control further enriches the discussion, presenting a comprehensive perspective on control strategies for individual actuators.

The practical application of the concepts is illustrated through the closed-loop motion simulation of two case studies. The chapter investigates the performance of PID and PD control in this context, beginning with the planar cable robot. Furthermore, the chapter explores the implications of measurement noise and actuator saturation on the motion simulation. Case studies continue with the Stewart–Gough platform, which serves as a further example of linear controller design, and performance limitations. The chapter closes with a collection of problems, providing readers with an opportunity to put their knowledge into practice and enhance their comprehension of the discussed concepts.

## 11.1 Introduction

A seemingly lifeless robot springs to life when control systems generate desired motions into it. Acting as the

robot's software brain, the controller evaluates the positions of its limbs and moving platform using sensor data, and then sends control commands to the actuators to achieve the intended motion trajectory. As a result, the robot gives the impression of acting with autonomous intent, even though it doesn’t possess true autonomy. This is because the control system skillfully manages the tracking of the desired motion.

In today's world, as the scope of robotic applications grows, robots are being utilized in various aspects of daily life. This wide range of applications has led to the development of different control structures for robots. Robotic applications can be broadly categorized into two main groups. The first category involves robots that have minimal interaction with their environment and operate as autonomous agents for moving components and equipment. This includes many industrial robots used across various sectors such as packaging, assembly, and space exploration. In these scenarios, the primary focus is on motion control, ensuring the robot's moving platform follows the desired trajectories accurately. Although this fundamental control structure is common in most robots, additional enhancements are sometimes integrated into the robot control systems for specific applications.

The second category involves applications where the robot actively interacts with its environment, necessitating force interactions. When it is necessary for the robot's moving platform to exert a specific wrench on the

environment, relying solely on motion control becomes insufficient. In such cases, the robot's control system must incorporate various force control methods. To highlight the difference, consider an industrial robot used in two similar applications. In the first scenario, consider a parallel robot to navigate across the entire surface of a blackboard and reach any specified point using its motion control system. However, if the task requires the robot to erase or write on the blackboard, simple motion control is not enough; the robot must also apply a specific force, particularly in a direction perpendicular to the blackboard's surface.

Now, imagine the robot interacting with a human, as in applications like rehabilitation or exoskeleton robots. In these cases, the force interaction between the robot and the human must be carefully regulated to ensure safety. The design of the necessary controllers becomes more complex in such scenarios, as they must account for the delicate balance between precise motion and the application of appropriate interaction forces.

This chapter investigates the exploration and design of fundamental linear “motion control” methods specifically for parallel robots, which are crucial control techniques in the robotics domain. The following chapters will provide a detailed examination of multivariable and nonlinear control schemes, as well as force and impedance control in robots. For those interested in gaining deeper insights into advanced motion and force control techniques, additional references in this field are recommended [226].

Numerous control topologies and techniques have been developed for robotic manipulators. Even though a robotic manipulator with multiple degrees of freedom can typically be modeled as a multi-input/multi-output (MIMO) and nonlinear system, many industrial controllers for these robots use several linear controllers to manage individual active joint motions. This decentralization can be effective in practice partly because large gear reductions in robot actuators greatly diminish the coupling and nonlinear aspects of robot dynamics [1, 188]. The linear controllers are often simpler to implement and can offer robust performance under a variety of operating conditions, making them a practical choice in many industrial settings.

However, for parallel manipulators, even with gear reductions incorporated into the design, the closed-kinematic chains in their structure considerably limit the effectiveness of simple decentralized controllers. The complex interactions between the joints and the structure of parallel robots necessitate more sophisticated control approaches. Consequently, advanced techniques in nonlinear and MIMO control strategies are often employed for these manipulators [227]. These advanced techniques can better handle the inherent coupling and dynamic complexities, providing enhanced precision and stability, whose details will be discussed in the upcoming chapter.

In addition to the challenges posed by the mechanical design, parallel manipulators often operate in environments that require high accuracy and quick response times, such

as in precision machining or medical robotics. This further underscores the need for advanced control strategies. Researchers and engineers continue to explore and refine these control methods, incorporating cutting-edge algorithms and leveraging advancements in computing power to achieve optimal performance. The following sections present several such topologies and techniques, discussing their theoretical foundations as well as practical implementations and applications.

To design a motion controller for a parallel manipulator, it is crucial to first examine the possible controller implementation structures developed for this purpose and then elaborate on various techniques for designing controllers based on these structures. The subsequent sections will first introduce the different controller topologies, followed by a discussion of linear controller design methods.

## 11.2 Controller Topologies

Control topology refers to the structure and arrangement of the control system used to determine the necessary actuator forces/torques based on system state measurements, as well as the preprocessing methods employed. In the motion control of a parallel manipulator, it is assumed that the controller provides the necessary actuator forces and torques to ensure that the robot's

motion adheres to a desired position and orientation trajectory.

Let us first define the main control variable and the manipulator dynamics in the closed-loop structures. The main motion variable in the control topologies is previously defined by the twist variable $\boldsymbol{\mathcal{X}} = [\boldsymbol{x}\,\boldsymbol{\theta}]^T$, where the linear motion of the moving platform is denoted by $\boldsymbol{x} = [x,\, y,\, z]^T$, while its orientation is described using screw coordinates, $\boldsymbol{\theta} = \theta[\boldsymbol{s}_x,\, \boldsymbol{s}_y,\, \boldsymbol{s}_z]^T = [\theta_x,\, \theta_y,\, \theta_z]^T$.

Furthermore, the general closed-form dynamic formulation of a parallel robot can be represented by the following equation.

$$\boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{F}}. \tag{11.1}$$

where, $\boldsymbol{M}(\boldsymbol{\mathcal{X}})$ denotes the mass matrix, $\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})$ represents the Christoffel matrix, $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$ indicates the gravitational forces, and $\boldsymbol{\mathcal{F}}$ represents the total wrench applied to the moving platform.

If no external forces are acting on the moving platform, and the system is considered to be in free motion, the wrench can be determined by projecting the actuator forces, $\boldsymbol{\tau}$, onto the moving platform through the Jacobian transpose, represented as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$. However, when external wrenches, such as a disturbance wrench $\boldsymbol{\mathcal{F}}_d$, are applied to the moving platform, the total wrench is computed by $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau} + \boldsymbol{\mathcal{F}}_d$.

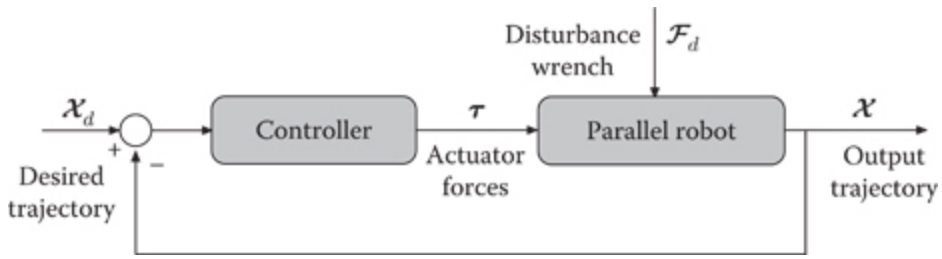
### 11.2.1 Motion Control in Task Space

For the motion control of a manipulator, the controller needs to determine the actuator forces/torques required to guide the moving platform along a specified trajectory. The desired motion of the moving platform is typically expressed by the desired twist of the manipulator, denoted as $\boldsymbol{\chi}_d$. This variable has the same dimension and structure as the manipulator's motion variable $\boldsymbol{\chi}$. In a closed-loop control system, it is essential to measure the actual motion of the manipulator using an instrumentation system. This system provides feedback on the platform's current position and orientation, which is critical for accurately tracking the actual motion, comparing it with the desired trajectory, and making necessary adjustments to ensure precise performance.

In the motion control of parallel manipulators, the emphasis is on accurately determining the final motion of the moving platform, making it essential to utilize instrumentation systems that can precisely measure the motion variable $\boldsymbol{\chi}$. These systems are specifically designed to directly measure the platform's final position and orientation with high accuracy. Typically, such instrumentation systems consist of two primary subsystems: the first may include precise accelerometers or global positioning systems (GPS) to determine the position of a point on the moving platform, while the second subsystem may use inertial sensors or laser gyros to evaluate the platform's orientation.

Although various techniques for measuring the position and orientation of moving objects in space have been developed, advancements in this area are still ongoing. High-precision commercial instrumentation systems that can provide these measurements with high accuracy are relatively rare and often expensive. If such an instrumentation system is available on a robotic platform, the control topology can be established on the task space variables.

Figure 11.1 depicts the general structure of a motion controller that uses direct measurements of the motion variable $\boldsymbol{\mathcal{X}}$ as feedback in a closed-loop system. In this configuration, the actual position and orientation of the manipulator are compared to their desired values, resulting in the motion error vector $\boldsymbol{e}_x$. The controller then processes this error information to generate appropriate commands for the actuators, aiming to reduce the tracking error. If the controller's design requires the dynamic model of the robot, the dynamic formulation in task space is typically determined. While this feedback control structure is quite standard, it is important to highlight that the feedback measurement is a vector-valued quantity, which makes the controller a MIMO system.

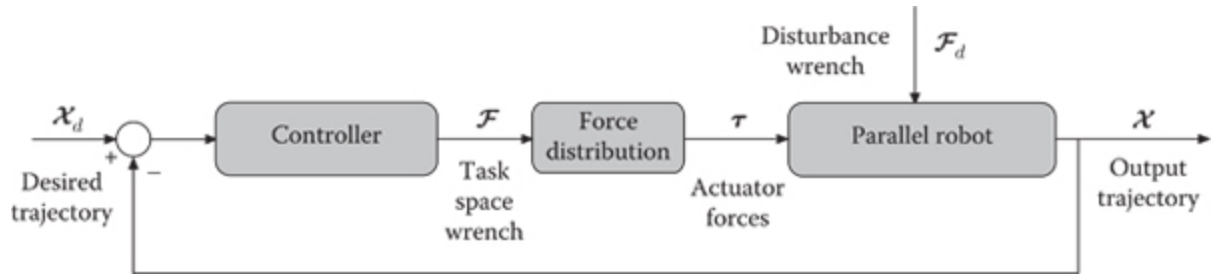


▶ Long Description for Figure 11.1

**FIGURE 11.1**

The general topology of motion feedback control in task space: motion variable $\boldsymbol{\mathcal{X}}$ is measured. ⏎

To establish a direct input-output relationship in task space, consider the topology illustrated in Figure 11.2. This topology is fundamentally similar to that shown in Figure 11.1; however, the key difference is that the controller's output is a wrench calculated in task space, denoted by $\boldsymbol{\mathcal{F}}$. This structure is preferable to the previous structure, since the controller is designed in the task space, and its output generates $\boldsymbol{\mathcal{F}}$. With this structure, a one-to-one correspondence is obtained between the components of the tracking errors, and the applied wrench to the moving platform.




▶ Long Description for Figure 11.2

**FIGURE 11.2**

The general topology of motion feedback control in task space: the motion variable $\boldsymbol{\mathcal{X}}$ is measured, and the controller output generates wrench in task space. ⏎

Additionally, a force distribution block is included, which maps the wrench $\boldsymbol{\mathcal{F}}$ to the corresponding actuator forces or torques $\boldsymbol{\tau}$. For a fully parallel manipulator, such as the Stewart–Gough platform, where the number of actuators matches the number of degrees of freedom, this mapping can be directly constructed using the Jacobian transpose of the manipulator. Specifically, as given in Equation 6.9, the mapping is expressed as $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}$. Thus, for a fully parallel manipulator, the actuator forces can be determined using $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$, assuming a nonsingular configuration.

For manipulators that are not fully parallel, additional computations may be required for the force distribution block. Consider a redundant parallel manipulator where the number of actuators, denoted by $m$, exceeds the number of degrees of freedom, denoted by $n$. Redundancy in parallel robots is often used to expand the workspace and avoid singularities. In cable-driven parallel manipulators, redundancy helps maintain tension in all cables. In such cases, the Jacobian matrix is not square, making it impossible to use the inverse of the Jacobian transpose for force distribution. Instead, the Jacobian transpose is an $m \times n$ matrix with $m > n$, resulting in infinitely many solutions for the actuator efforts $\boldsymbol{\tau}$ required to generate the desired wrench $\boldsymbol{\mathcal{F}}$ in task space.

Various algorithms have been developed to optimize the use of extra degrees of actuation, whose details are given in Section 6.6 under the title of *Redundancy Resolution.* The simplest approach to address this problem is to use the

Moore–Penrose pseudo-inverse of the Jacobian transpose, denoted as $\boldsymbol{J}^{T^{\dagger}}$. This method calculates the actuator forces as $\boldsymbol{\tau} = \boldsymbol{J}^{T^{\dagger}}\boldsymbol{\mathcal{F}}$. Among the many possible solutions that satisfy $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^{T}\boldsymbol{\tau}$, this approach finds the minimum norm force distribution, $\boldsymbol{\tau}_{min}$, that meets the wrench projection $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^{T}\boldsymbol{\tau}_{min}$. Although popular for many applications, this method is not recommended for cable-driven manipulators because it does not ensure positive tension forces in all actuators. A comprehensive overview of redundancy resolution techniques that address this requirement is provided in Section 6.6. Regardless of the technique used for redundancy resolution, a specific force distribution in the actuators is generated, which satisfies the Cartesian wrench relation $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^{T}\boldsymbol{\tau}$. This is why the block added to the control topology in Figure 11.2 is referred to as the force distribution block.

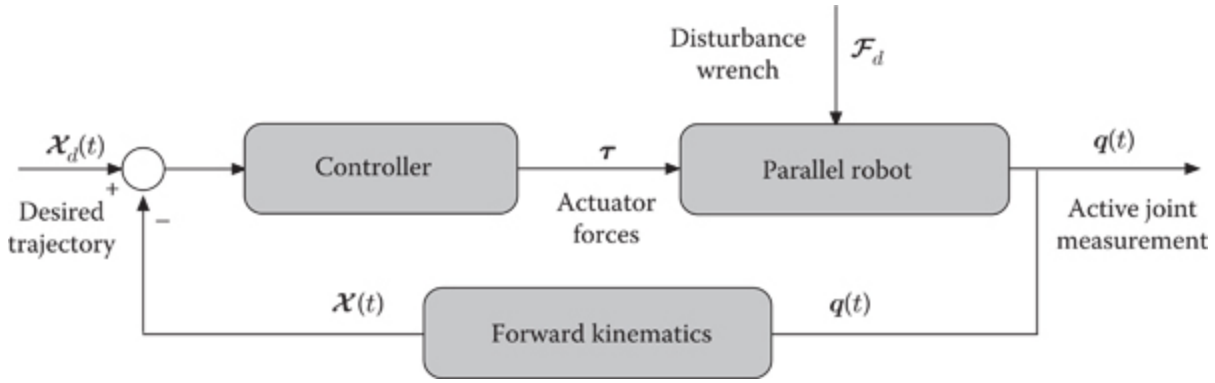
### 11.2.2 Motion Control in Joint Space

Various motion control topologies can be explored based on different methods for determining the position and orientation of the moving platform. For instance, in a general parallel manipulator with multiple limbs and both passive and active joints, as shown in Figure 1.11, it is often easier to measure the active joint variables rather than directly measuring the final position and orientation of the moving platform. Measuring these active joint variables can provide sufficient information to deduce the platform's

motion, potentially simplifying the required instrumentation system and reducing the cost of the feedback system.

As defined in Equation 9.17, let $\boldsymbol{q} = [q_1 \quad q_2 \quad \cdots \quad q_n]$ represent the joint space variables, including the motion variables of the active joints. The relationship between the joint variables $\boldsymbol{q}$ and the motion variable of the moving platform $\boldsymbol{\chi}$ has been thoroughly analyzed in Chapter 4 through forward and inverse kinematics. Additionally, the relationship between the differential motion variables $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{\chi}}$ is extensively covered in Chapter 5 via Jacobian analysis.

For the parallel manipulator under consideration, forward kinematic analysis can be utilized to compute $\boldsymbol{\chi}$ from the measured joint variables $\boldsymbol{q}$. This enables the implementation of the control topology shown in Figure 11.3. In this setup, the joint variable measurements $\boldsymbol{q}$ are used as feedback inputs, and forward kinematics are applied to calculate the moving platform's motion variable $\boldsymbol{\chi}$. Similar to the control topology depicted in Figure 11.1, the motion error in task space, $\boldsymbol{e}_x$, is determined and used in a controller designed to generate the necessary actuator forces/torques, $\boldsymbol{\tau}$.




▶ Long Description for Figure 11.3

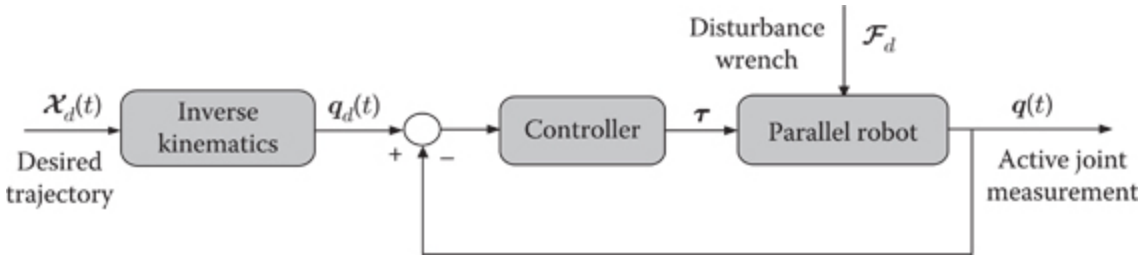
**FIGURE 11.3**

The general topology of motion feedback control: the active joint variable $\boldsymbol{q}$ is measured. ⏎

While this structure avoids the need for direct and expensive measurement of the motion variable of the robot, it relies heavily on precise knowledge of the robot's kinematics and requires kinematic calibration, which introduces new challenges in the design and implementation of motion controllers. In the control topology illustrated in Figure 11.3, forward kinematic analysis of the manipulator is essential for the feedback loop. As discussed earlier, conducting forward kinematic analysis for parallel manipulators is usually complex because it involves solving a set of nonlinear equations, and this task becomes even more demanding when a real-time solution is required. On the other hand, inverse kinematic analysis for parallel manipulators is generally more straightforward and often provides a unique, analytic solution. To overcome the challenges associated with

implementing the feedback loop shown in Figure 11.3, an alternative control topology is commonly used, which will be detailed in the upcoming section.

In the alternative topology shown in Figure 11.4, the desired motion trajectory of the robot, $\boldsymbol{\chi}_d$, is used in an inverse kinematic analysis to determine the corresponding desired joint variables, $\boldsymbol{q}_d$. The controller is then designed based on the joint space error, $\boldsymbol{e}_q$, and operates entirely within the joint space. This structure fundamentally differs from the previous ones in its approach and characteristics. Here, the controller's input is the motion error in joint space, $\boldsymbol{e}_q$, and its output is the actuator forces/torques, $\boldsymbol{\tau}$, also represented in joint space.




▶ Long Description for Figure 11.4

**FIGURE 11.4**
The general topology of motion feedback control: the active joint variable $\boldsymbol{q}$ is measured, and the inverse kinematic analysis is used. ⏎

In this topology, both the measurement of motion variables and the controller design are conducted in the joint space. As illustrated in Figure 11.4, the measurement of joint variables and their feedback form the core of the

motion control system. Thus, the controller is designed in the joint space. If the controller design requires the dynamic model of the robot, the standard dynamic model of the parallel robot must be translated into joint space, which is a significantly more complex process compared to modeling and calibrating the robot in task space.

In contrast, the earlier topologies involve the controller receiving the motion error in task space, $\boldsymbol{e}_x$, and producing outputs in joint space. This difference means the latter approach may be more conducive to using independent controllers for each joint, whereas such a structure may not be as effective for the topologies illustrated in Figures 11.1 and 11.3.

### 11.2.3 Topology Selection

To categorize the developed controllers according to the implementable topologies discussed, the focus will be on control topologies in the task space. This approach considers the conceptual control structures applicable to the task space topology illustrated in Figure 11.1, while for implementation, the topology provided in Figure 11.2 will serve as the foundation for the controller design. In both topologies, the motion variable in task space, $\boldsymbol{x}$, is measured and used for feedback, classifying these methods as motion control techniques in task space. During the implementation stage, the force distribution block, depicted in the latter topology, will be employed.

If direct measurement of the motion control variables in task space is not plausible, the controllers must be designed and implemented in joint space. For such implementation, the structure depicted in Figure 11.4 is recommended, while the focus of the design techniques is shifted into the joint space. Although this approach aligns well with the industrial implementation of controllers for parallel robots, it is not recommended for model-based control strategies, since the dynamic formulation in joint space is a prohibitive task.

As previously mentioned, numerous advanced control techniques have been developed for robotic manipulators. The following sections will be dedicated to controller design techniques in the linear domain, while more advanced methods for designing controllers for parallel robots will be covered in the upcoming chapter. This will provide a step-by-step description of the controller design techniques, starting with basic concepts and progressing to more sophisticated structures.

## 11.3 Linear Control: Concepts and Methods

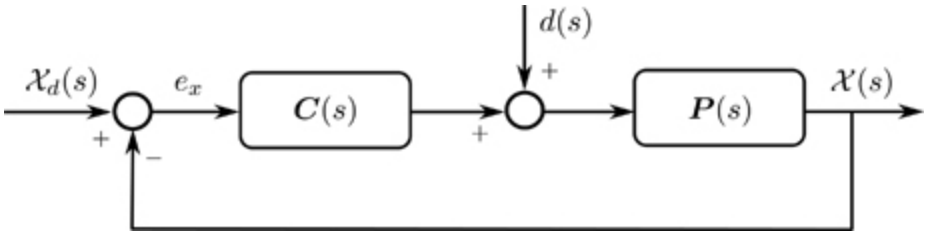
In the section, we explore the foundational principles and methodologies of linear control systems, which are widely applied in industrial robotics. It is well-documented that many industrial robots achieve exceptional precision and reliability using relatively simple linear controllers, such as

Proportional-Derivative (PD) and Proportional-Integral-Derivative (PID) controllers. Despite their simplicity, these controllers can deliver impressive performance due to their ability to manage a wide range of operational scenarios and disturbances effectively. This chapter aims to provide a comprehensive overview of the analysis and design processes for various linear controllers in the context of parallel robots. We will discuss the specific conditions and requirements that enable these controllers to achieve remarkable accuracy and robustness.

The main objective of a robot's motion control system is to ensure precise tracking of the desired motion trajectory. However, this primary is intertwined with several other critical objectives that must be met to ensure optimal performance. One of these is internal stability, which is particularly important in systems with feedback loops. The stability of the closed-loop system is a fundamental prerequisite, as instability can lead to unpredictable system behavior, potentially causing harm to both the machinery and its operators. The chapter will give insight into the criteria and methods for analyzing and ensuring stability in linear control systems.

Another essential consideration is the system's resilience to disturbances. In practice, disturbances can arise from various sources, including external forces, the effect of gravity, and parameter variations. As depicted in Figure 11.5, these disturbances can significantly impact the tracking accuracy of the closed-loop system. Therefore, one

of the key goals in controller design is to mitigate the influence of these disturbances, thereby enhancing the system's ability to maintain accurate trajectory tracking.




▶ Long Description for Figure 11.5

**FIGURE 11.5**
Block diagram illustrating robot motion in a single direction with disturbance, using a linear model. ⏎

Furthermore, the robustness of the closed-loop system to modeling inaccuracies is a crucial factor. The dynamic behavior of robots, especially parallel robots, can only be accurately captured through complex nonlinear and multivariable models. However, in practice, these models often have limitations due to simplifications and parameter uncertainties. It is crucial for the designed controllers to perform well even when there are discrepancies between the actual system and the nominal model used during the design process.

In this chapter, we will provide an in-depth exploration of the concept of feedback in control systems. Feedback is a powerful tool that allows for continuous correction of the system's behavior, making it possible to compensate for disturbances and modeling uncertainties. We will discuss

how feedback mechanisms can be effectively utilized to enhance the performance of parallel robots, ensuring that they operate reliably even in the face of external disturbances and modeling uncertainties.

### 11.3.1 Magic of Feedback

From the very beginning, control system designers have faced a significant challenge in achieving multiple objectives within linear control systems. Remarkably, it is noteworthy to acknowledge that the use of feedback loops, irrespective of the type of controller, can enable the achievement of the aforementioned multiple objectives in a control system. It could be argued that this is perhaps the most pivotal reason behind the success of control systems in practical applications. Through online observation of the system states, control systems can effectively achieve desirable tracking, mitigating the impact of disturbances and uncertainties in the system model.

Given the importance of this issue in the structure of control systems, let us explore and verify this property of feedback, which we refer to as the “magic of feedback” in this book, through simple mathematical analysis. For the sake of simplicity in writing equations and making them more general, we deliberately refrain from writing complex multivariable system, and focus on one component of the moving platform's motion. Let us mathematically examine the dynamic behavior of the tracking error for this single-valued variable in the closed-loop system, denoted as $e_x$,

considering the desired trajectory input $x_d$, disturbance input $d$, and uncertainty in the system modeling.

To achieve this, suppose that the dynamic model of the input-output system for this motion component can be separately modeled as a linear system along with the disturbance torque. In this representation, each of the moving platform motion component is modeled separately with a transfer function $\boldsymbol{P}(s)$. If the task space control topology is applied to the system by using a linear controller $\boldsymbol{C}(s)$ for motion component in a single direction, the closed-loop block diagram for each joint of the robot can be depicted as shown in Figure 11.5. For the robot controller $\boldsymbol{C}(s)$, common industrial and linear structures such as PD, PID, or state feedback can be employed, or initiatives can be taken to completely eliminate disturbance in the input using feed-forward control law or weight compensation.

The main goal within a robot's motion control system is the accurate tracking of the intended motion trajectory. However, accomplishing this objective inherently requires achieving additional crucial goals. Internal stability, especially when feedback loops are present, stands as an essential condition that must be taken into account when analyzing the dynamic behavior of the closed-loop system. Instability represents a potential threat to both the system's well-being and the users engaging with it.

On the other hand, the presence of disturbances in the system, as observed in the diagram in Figure 11.5,

undoubtedly has an impact on the tracking accuracy of the closed-loop system. Eliminating or reducing the effect of disturbances on the system's performance, which is manifested in the robot's accuracy in tracking the desired trajectory, is another crucial goal that should never be overlooked. Another goal that should be addressed in the successful implementation of control systems is the insensitivity of the closed-loop system's performance to the model used in controller design.

To start the analysis, we first derive the transfer function of the tracking error of the closed-loop system, $e_x$, with respect to the desired input $\mathscr{X}_d$. Easily and by simplifying block diagrams in linear systems, it can be demonstrated that this transfer function is equal to:

$$\boldsymbol{S}(s) = \frac{e_x(s)}{\mathscr{X}_d(s)} = \frac{1}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)} \tag{11.2}$$

In control literature, $\boldsymbol{S}(s)$ is called the sensitivity transfer function. It is evident that achieving a small tracking error $e_x \approx 0$, occurs only when the value of this transfer function is close to zero within the desired bandwidth of the system. Regardless of the control structure used, if the control is high-gain, the denominator of the transfer function $\boldsymbol{S}(s)$ becomes significantly large, resulting in a very small value for $\boldsymbol{S}(s)$ within the bandwidth.

Now, to examine the effect of disturbance on the tracking error, based on the tracking error transfer function of the closed-loop system $e_x$ with respect to the input $d(s)$, we will have:

$$\boldsymbol{D}(s) = \frac{e_x(s)}{d(s)} = \frac{\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)} = \boldsymbol{P}(s)\,\boldsymbol{S}(s)$$

(11.3)

This transfer function is denoted by $\boldsymbol{D}(s)$, representing the transfer function of disturbance effect on the tracking performance. Due to the presence of a feedback loop in the control structure, this transfer function is proportional to the sensitivity transfer function. It is evident that achieving a small tracking error $e_x \approx 0$, in the presence of disturbances still occurs when the controller is high-gain within the required bandwidth. In this case, as the denominator of the disturbance transfer function grows, the magnitude of this transfer function becomes very small.

Although it experiences a change in intensity compared to the sensitivity transfer function alone, owing to the presence of the plant transfer function $\boldsymbol{P}(s)$, achieving effective tracking of the desired path in the presence of input disturbances is still feasible through the use of a high-gain controller. The simultaneous realization of two crucial and distinct goals in control systems, facilitated only by the existence of a feedback loop, is a remarkable accomplishment in the control system.

If achieving the third objective is also feasible using a high-gain controller, the magic of feedback will be even more pronounced. This objective can be examined by mathematically analyzing the impact of model uncertainties on the system output. For this purpose, we use the sensitivity analysis of the input-output transfer function with respect to the model variations. The input-output transfer function of the system, denoted as $\boldsymbol{T}(s)$, can be expressed as follows:

$$\boldsymbol{T}(s) = \frac{\mathscr{X}(s)}{\mathscr{X}_d(s)} = \frac{\boldsymbol{C}(s)\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)}$$

(11.4)

Calculate the normalized variations of this transfer function with respect to the changes in the system model $\boldsymbol{P}(s)$. This is calculated by partially differentiating this transfer function with respect to $\boldsymbol{P}(s)$, and then normalizing it. The expression is as follows:

$$\text{Normalized Sensitivity Function} = \boldsymbol{S}_T^P = \frac{\partial \boldsymbol{T}(s)/\boldsymbol{T}(s)}{\partial \boldsymbol{P}(s)/\boldsymbol{P}(s)}$$

(11.5)

Note that in this expression, by dividing the partial derivative of the numerator and denominator with respect to their respective arguments, the normalized sensitivity function is obtained. With some calculation and

simplification, this transfer function can be written as follows.

$$
\begin{aligned}
\boldsymbol{S}_T^P &= \frac{\partial \boldsymbol{T}(s)}{\partial \boldsymbol{P}(s)} \cdot \frac{\boldsymbol{P}(s)}{\boldsymbol{T}(s)} \\
&= \frac{\boldsymbol{C}(s)(1+\boldsymbol{P}(s)\boldsymbol{C}(s)) - \boldsymbol{C}(s)\boldsymbol{P}(s)\boldsymbol{C}(s)}{(1+\boldsymbol{P}(s)\boldsymbol{C}(s))^2} \cdot \frac{\boldsymbol{P}(s)(1+\boldsymbol{P}}{\boldsymbol{P}(s)\boldsymbol{C}} \\
&= \frac{1+\boldsymbol{P}(s)\boldsymbol{C}(s)}{(1+\boldsymbol{P}(s)\boldsymbol{C}(s))^2} \\
&= \frac{1}{1+\boldsymbol{P}(s)\boldsymbol{C}(s)} \\
&= \boldsymbol{S}(s)
\end{aligned}
$$

It is astonishing to observe that the normalized sensitivity function $\boldsymbol{S}_T^P$ is exactly equivalent to the sensitivity transfer function of the system $\boldsymbol{S}(s)$ defined in Equation 11.2 and introduced in the tracking error analysis. This magic is only witnessed in closed-loop control systems. Thus, by utilizing a high-gain controller, all the desired objectives, including tracking the desired trajectory in the presence of disturbances and uncertainty in the modeling, can be achieved.

The magic of feedback has enabled achieving satisfactory performance even with simple controllers. Among the various linear controllers that can be employed in this structure, a few examples will be studied in the upcoming sections.

### 11.3.2 PD Control

The PD controller is the simplest and most effective linear controller that aims to ensure the tracking of the desired input trajectory. This controller applies the weighted sum on the input error and its derivative as the control effort required to actuate the robot's motion.

$$\mathscr{F}(t) = k_p e_x(t) + k_d \dot{e}_x(t) \tag{11.6}$$

In this relation, $k_p$ is the proportional gain of the controller, and $k_d$ is its derivative gain. This controller directly utilizes the product of the tracking error and the proportional gain to generate the necessary control effort, aiming for a rapid reduction in the tracking error. The use of the derivative of the tracking error serves as a form of prediction of the future state of the tracking error, contributing to the establishment of appropriate transient behavior. This controller, in Laplace space, is expressed as follows with two control gains, $k_p, k_d$:

$$\boldsymbol{C}(s) = k_p + k_d s \tag{11.7}$$

Furthermore, it is typical to consider a second-order model for the robot dynamics in each direction [1].

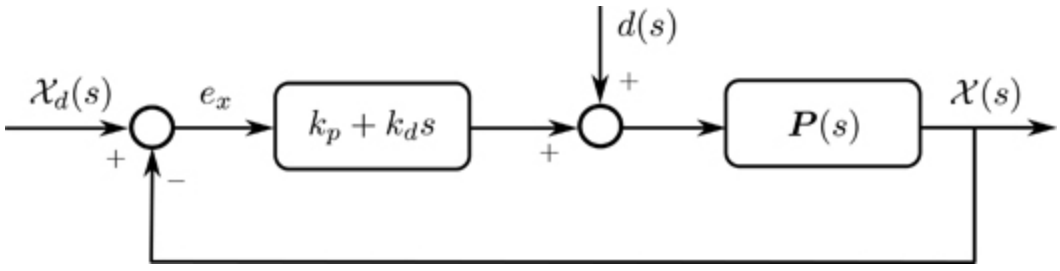
$$\boldsymbol{P}(s) = \frac{1}{s(m_e s + b_e)}$$

(11.8)

In this model, $m_e$ represent the effective mass of the robot in the direction of motion, while $b_e$ represents the effective viscous friction effects acting on the system in this direction. Thus, using the PD controller, the block diagram of the closed-loop system for each joint in the presence of external disturbance is depicted in Figure 11.6. With this block diagram, the transfer function of the motion trajectory for each joint, $\mathscr{X}(s)$, with respect to the desired trajectory input $\mathscr{X}_d(s)$ and the disturbance input $d(s)$, can be obtained.

$$\begin{aligned} \mathscr{X}(s) &= \frac{k_p + k_d s}{\Delta(s)} \mathscr{X}_d(s) + \frac{1}{\Delta(s)} d(s) \\ \Delta(s) &= m_e s^2 + (b_e + k_d)s + k_p \end{aligned}$$

(11.9)




▶ Long Description for Figure 11.6

**FIGURE 11.6**
Block diagram illustrating robot motion in a single direction with disturbance, using a linear model and PD Control. ⏎

The tracking error can also be expressed as follows:

$$\begin{aligned} \mathscr{X}_q(s) &= \mathscr{X}_d(s) - \mathscr{X}(s) \\ &= \frac{m_e s^2 + b_e s}{\Delta(s)} q_d(s) - \frac{1}{\Delta(s)} d(s) \end{aligned} \tag{11.10}$$

As evident in these two equations, the dynamics of the tracking output and tracking error are both second-order systems whose characteristics are determined by the characteristic equation of the closed-loop system $\Delta(s)$. The gains of the PD controller directly appear in this characteristic equation and consequently affect the system's output behavior. To ensure system stability, it is sufficient to choose both controller gains to be positive. Furthermore, if the controller gains are chosen to be sufficiently large, the roots of the characteristic equation will be located farther away from the imaginary axis in the complex plane, resulting in a faster transient response. Now, if the controller gains are chosen in a way that the characteristic equation has complex conjugate roots, the system's output and, consequently, the tracking error of the system will exhibit oscillatory behavior. In this way, by

examining the behavior of second-order systems, the dynamic behavior of the tracking error in the system can be adjusted as desired.

For a standard analysis, rewrite the characteristic equation of the closed-loop system in the following standard form.

$$\Delta(s) = m_e(s^2 + 2\xi\omega_n s + \omega_n^2)$$

(11.11)

where $\omega_n$ is the natural frequency of the closed-loop system in joint $i$, and $\xi$ is the damping ratio. These primary parameters, which shape the dynamics of the second-order system, are adjustable using the control gains. The relationship between the control gains and these primary parameters is obtained as follows.

$$\omega_n^2 = \frac{k_p}{m_e} \quad \longrightarrow \omega_n = \sqrt{k_p/m_e}$$

(11.12)

$$2\xi\omega_n = \frac{b_e + k_d}{m_e} \quad \longrightarrow \xi = \frac{1}{2}\,\frac{b_e + k_d}{\sqrt{k_p m_e}}$$

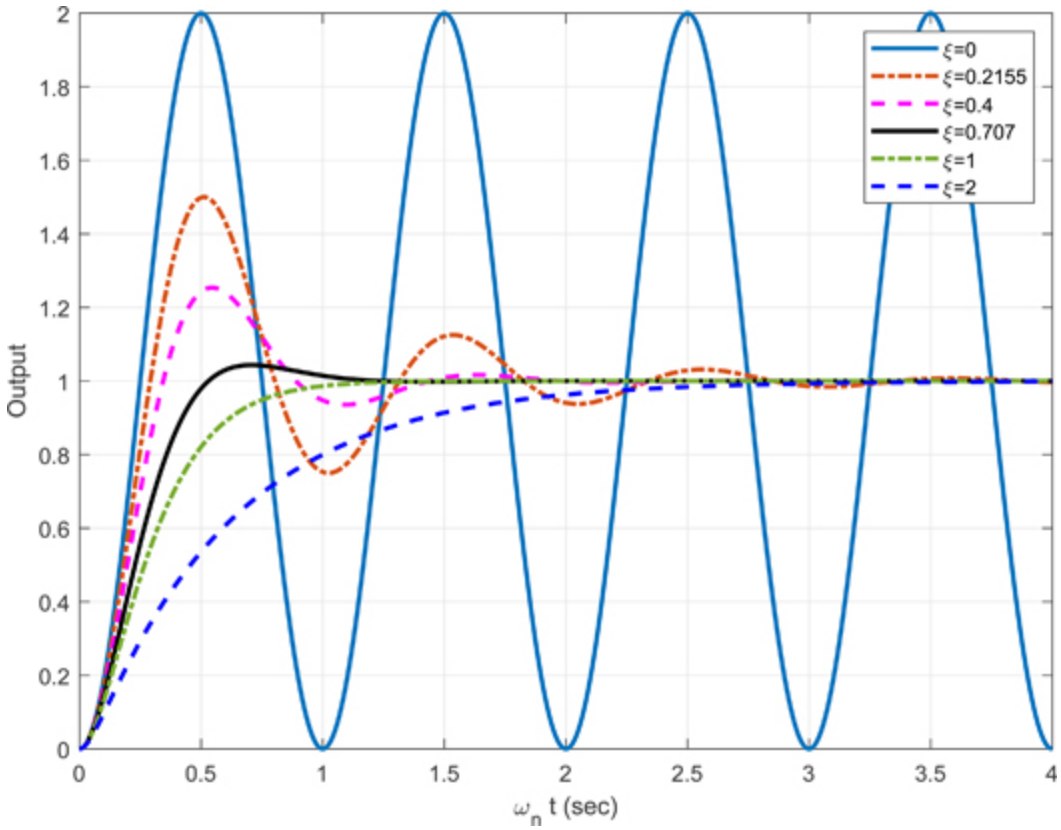
(11.13)

In these equations, it is observed that by increasing the proportional gain $k_p$, the natural frequency of the closed-

loop dynamic system increases. The natural frequency is proportional to the system's response speed, allowing the system to reach its steady-state more quickly. On the other hand, the derivative gain in the controller $k_d$ is proportional to the damping ratio of the system. Assuming the controller gains are positive and the stability of the closed-loop system is maintained, if the ratio $k_d$ to $k_p$ is chosen such that the damping ratio falls within the interval $0 \leq \xi \leq 1$, the characteristic roots of the equation become complex conjugates, leading to an oscillatory behavior in the closed-loop system. In this interval, reducing the derivative gain $k_d$ results in a smaller damping ratio, leading to more pronounced oscillations in the system. However, if the ratio $k_d$ to $k_p$ is selected in a way that the damping ratio exceeds one, the characteristic roots become real, resulting in a slower but oscillation-free behavior of the system. Therefore, by adjusting the proportional and derivative gains in the controller, one can achieve the desired transient behavior.

Figure 11.7 illustrates the closed-loop system's response to a unit step input for various damping ratios in the range of zero to two. As observed from this figure, decreasing the damping ratio increases the level of oscillations, to the extent that with a damping ratio of zero, the system exhibits persistent oscillations and approaches instability. In robotic systems, selecting a damping ratio of $\xi = 0.707$ will result in complex roots of the characteristic equation positioned along the 45 degrees angle on the complex

plane, and the system's time response will converge to the desired value with minimal oscillations and maximum speed. Therefore, tuning control gains to achieve such a response is highly desirable. The desired natural frequency in robotic systems is also adjusted based on the required bandwidth and the saturation limits of the actuator, typically ranging between one to five radians per second.




▶ Long Description for Figure 11.7

**FIGURE 11.7**

Response of a second-order system with different natural frequencies and damping ratios to a unit step input. ⏎

On the other hand, using the “final value Theorem,” the steady-state behavior of the tracking error can also be determined [228].

$$e_{ss} = \lim_{t \to \infty} e_x(t) = \lim_{s \to 0} s\, e_x(s) \tag{11.14}$$

If the input disturbance to the system, denoted as $d(t)$, is considered as a constant value $d$, in the Laplace domain, this disturbance can be treated as a step input with an amplitude of $d$, having the Laplace transform $d(s) = d/s$. In this case, using the tracking error transfer function 11.10 and applying the final value Theorem 11.14, the steady-state error due to the input disturbance can be written as follows:

$$\begin{aligned} e_{ss} &= \lim_{s \to 0} s\, \frac{1}{\Delta(s)} \frac{d}{s} \\ &= \lim_{s \to 0} \frac{d}{m_e s^2 + (b_e + k_d)s + k_p} \\ &= \frac{d}{k_p} \end{aligned} \tag{11.15}$$

This relationship clearly demonstrates that if the disturbance caused by gravitational forces on the robot tends towards zero over time ($d \to 0$), the steady-state

error caused by it also becomes zero. This is in contrast to the scenario where this disturbance does not diminish over time and converges to a constant value; in this case, the steady-state error does not vanish, but using a large proportional gain $k_p \gg 1$ significantly reduces its magnitude. This conclusion aligns with the analysis conducted regarding the magic of feedback and the recommendation to use large gains in the controller.

In robotic systems, the disturbance acting on the robot links due to the gravitational force usually does not converge to zero as time evolves, and for precise tracking at steady-state, special attention must be paid to this issue. One approach to mitigate steady-state error is by incorporating the integral of error into the feedback loop, or by employing a PID controller, which will be discussed in Section 11.3.3.

Although using the integral of error will cause the steady-state error to converge to zero, it will decrease the response speed of the closed-loop system. For this reason, in industrial robots, other compensation methods such as mechanical or hydro-pneumatic weight compensation are often employed in robot design to make more efficient use of actuators with lower power and cost. Furthermore, force/torque compensation in the robot due to gravitational forces can be achieved using a feed-forward loop or predominant inverse dynamics control, which will be discussed in sections 11.3.5 and 12.8.3, respectively.

### 11.3.3 PID Control

The PID controller is the most predominant linear controller used in industrial processes and robotics for the purposes of desirable trajectory tracking. In this controller, in addition to the proportional and derivative terms, the integral of error is also utilized in calculating the control effort.

$$\tau(t) = k_p e_x(t) + k_d \dot{e}_x(t) + k_i \int_0^t e_x(\sigma) d\sigma \tag{11.16}$$

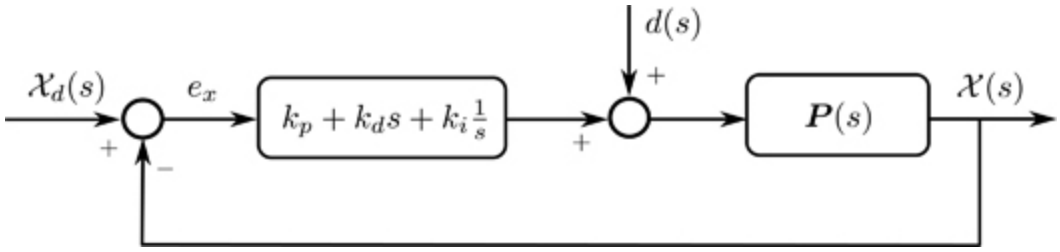
In this equation, $k_p$ represents the proportional gain of the controller, $k_d$ is the derivative gain, and $k_i$ is the integral gain. This controller utilizes the integral of tracking error to reduce the steady-state error of the system toward zero. In Laplace space, this controller is characterized by three control gains as follows:

$$\boldsymbol{C}(s) = k_p + k_d s + \frac{k_i}{s} \tag{11.17}$$

In this way, the block diagram of the closed-loop system for each joint in the presence of external disturbances can be illustrated using the PID controller, as shown in Figure 11.8. Utilizing this block diagram, the dynamics of the

motion for each joint $\mathscr{X}(s)$ can be determined with respect to the desired trajectory input $q_d(s)$, and the disturbance input $d(s)$, as follows.

$$\begin{aligned}\mathscr{X}(s) &= \frac{k_d s^2 + k_p s + k_i}{\Omega(s)} q_d(s) - \frac{s}{\Omega(s)} d(s) \\ \Omega(s) &= m_e s^3 + (b_e + k_d) s^2 + k_p s + k_i\end{aligned} \tag{11.18}$$




▶ Long Description for Figure 11.8

**FIGURE 11.8**
Block diagram illustrating robot motion in a single direction with disturbance, using a linear model and PID Control. ⏎

The tracking error can also be calculated using the following relationship:

$$\begin{aligned}e_x(s) &= q_d(s) - \mathscr{X}(s) \\ &= \frac{m_e s^3 + b_e s^2}{\Omega(s)} q_d(s) + \frac{s}{\Omega(s)} d(s)\end{aligned}$$

(11.19)

As observed, utilizing the PID controller results in the dynamics of the tracking error being of the third order, characterized by the characteristic equation $\Omega(s)$. PID controller gains directly affect this characteristic equation coefficients and, consequently, the behavior of the system's output. By using the Routh-Hurwitz stability criterion, it can be demonstrated that, for the closed-loop system to be stable, the controller gains must all be positive and satisfy the following condition.

$$k_i < \frac{b_e + k_d}{m_e} k_p$$

(11.20)

Moreover, if the controller gains are appropriately selected to satisfy stability conditions, and being chosen sufficiently large, the roots of the characteristic equation will be located farther from the imaginary axis, resulting in a faster transient response in the system. If the controller gains are chosen in a way that the characteristic equation has complex roots, the system output and, consequently, the tracking error will exhibit oscillations.

One of the most important features of the PID controller is the elimination of steady-state error. To investigate this, consider Laplace transforming the tracking error for a unit step reference input $q_d(s) = 1/s$ and a step disturbance with a constant gravitational force $d(s) = d/s$.

$$e_x(s) = \frac{m_e s^2 + b_e s}{\Omega(s)} + \frac{d}{\Omega(s)}$$

(11.21)

It can be easily understood that by using the final value Theorem, in the steady-state, the tracking error will be equal to zero.

$$e_{ss} = \lim_{s \to 0} s\, e_x(s) = 0$$

(11.22)

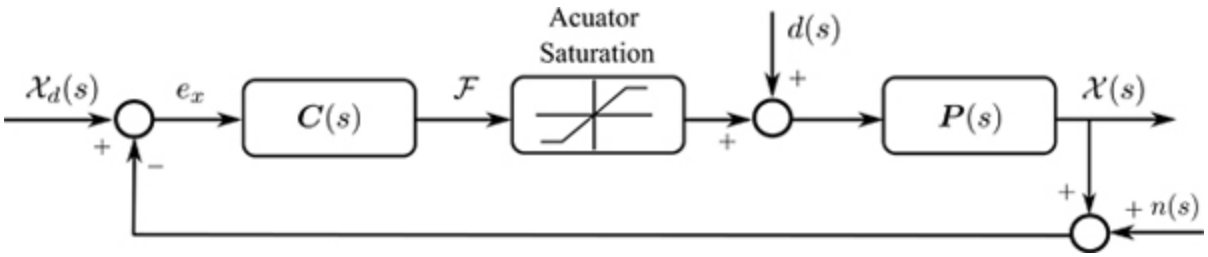
It should be noted that this feature comes with a slower response in tracking performance. Various methods for tuning the parameters of PID controllers have been proposed in the control system literature. However, we refrain from repeating them here and invite interested readers to explore references [229, 230] for further details.

### 11.3.4 Performance Limitations

As mentioned in Section 11.3.1, by using the definition of the sensitivity transfer function $\boldsymbol{S}(s)$ in closed-loop systems, it can be demonstrated that effective trajectory tracking in the presence of simultaneous disturbance and uncertainty in the system modeling is achievable with proper high-gain controller. However, it should be noted that in the presence of limitations in the robot's sensors

and actuators, the performance of the closed-loop system will be affected. Therefore, in the presence of such limitations, measures should be taken in the implementation of control systems, which will be discussed in this section.

Let's begin by theoretically exploring the influence of measurement noise in sensors and actuator saturation on the performance of the closed-loop system. To achieve this, examine the block diagram of the closed-loop system shown in Figure 11.9. As depicted in this figure, the observation and control of the robot are affected by the measurement noise $n$. Furthermore, the control effort commanded by the controller is constrained by the actuator saturation block, as there are limitations on applying high efforts in the actuators.




▶ Long Description for Figure 11.9

**FIGURE 11.9**
Block diagram illustrating control of robot motion in a single direction in the presence of measurement noise and actuator saturation. ⏎

With the introduction of measurement noise $n$ in the output transfer function and the tracking error of the

system, the output transfer function of the system obtained from the relation $\mathscr{X} = \boldsymbol{C}(s)\boldsymbol{P}(s)(\mathscr{X}_d - \mathscr{X} - n)$ can be expressed in terms of three inputs $\mathscr{X}_d$, $d$, and $n$ as follows.

$$\mathscr{X}(s) = \boldsymbol{T}(s)\mathscr{X}_d(s) + \boldsymbol{P}(s)\boldsymbol{S}(s)d(s) - \boldsymbol{T}(s)n(s)$$

(11.23)

In this relation, the sensitivity transfer function $\boldsymbol{S}(s)$ and its complement $\boldsymbol{T}(s)$ in each joint of the robot are defined by the following equations.

$$\boldsymbol{S}(s) = \frac{1}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)}, \;\; \boldsymbol{T}(s) = \frac{\boldsymbol{C}(s)\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)}$$

(11.24)

$$\boldsymbol{S}(s) \; + \; \boldsymbol{T}(s) \; = \; 1$$

(11.25)

Considering that the sum of the sensitivity transfer function $\boldsymbol{S}(s)$ and its complement $\boldsymbol{T}(s)$ is equal to one, $\boldsymbol{T}(s)$ is referred to as the complementary sensitivity transfer function in control literature. Now, by calculating the actual tracking error, which is calculated from the relation $e_x = \mathscr{X}_d - \mathscr{X}$, we have,

$$e_x(s) = \boldsymbol{S}(s)q_d(s) - \boldsymbol{P}(s)\boldsymbol{S}(s)d(s) + \boldsymbol{T}(s)n(s)$$

(11.26)

The third term in the Equations 11.23 and 11.26 indicates that measurement noise appears in the tracking error through the complementary sensitivity transfer function $\boldsymbol{T}(s)$. This means that if a high-gain controller is used in the closed-loop structure, according to the definition, its value approaches one. Thus, the effect of measurement noise will be introduced into the output and tracking error without attenuation.

Although this limitation introduces a significant constraint in the closed-loop system structure, it aligns perfectly with the concept of using feedback in control systems. The fundamental idea in observation-based feedback structures is to control the system output behavior by observing the output and correcting the tracking error based on this observation. Therefore, it is natural that any inaccuracy in observing the output behavior will affect the tracking performance to the same extent.

Fortunately, this apparent limitation can be mitigated to some extent in control systems. This is because measurement noise in sensors often has high-frequency content. If the frequency content of sensor noise falls outside the desirable bandwidth for system tracking, it is possible to keep the sensitivity transfer function very small in this bandwidth by applying large control gains only at the system bandwidth frequencies. Meanwhile, by reducing

these gains outside the system bandwidth, the complementary sensitivity transfer function can be minimized in this range, reducing the impact of sensor noise on tracking error. Designing a controller with these features is feasible in the frequency domain using loop shaping methods [231], while simple PD and PID controllers typically struggle to meet this objective.

Now, let's examine the theoretical effect of actuator saturation on tracking performance. For this purpose, consider the required control effort $\mathscr{F}$ as the output of the closed-loop control system, and obtain its transfer function with respect to the three inputs of the system using the block diagram shown in Figure 11.9. To do this, you can use the relation $\mathscr{F} = \boldsymbol{C}(s)(\mathscr{X}_d - \mathscr{X} - n)$ and simplify it to the following expression.

$$\mathscr{F} = \boldsymbol{P}^{-1}(s)\boldsymbol{T}(s)\left(\mathscr{X}_d - n\right) - \boldsymbol{T}(s)d \tag{11.27}$$

If we examine this relation carefully, it becomes apparent that the inverse of the main system transfer function, $\boldsymbol{P}^{-1}(s)$, enters into the transfer function of the control effort. Considering that in most dynamic systems and linear models of robot arms, the main system transfer function becomes very small at frequencies higher than its bandwidth, the product of $\boldsymbol{P}^{-1}(s)\boldsymbol{T}(s)$ in this frequency range will become very large. Therefore, the control effort required to track input signals with frequency content

outside the system bandwidth becomes notably large. In the controller design, this should be considered, and the bandwidth of the closed-loop system should not exceed the bandwidth of the open-loop system.

Equation 11.27 also reveals another important fact. The second term in this equation shows that the product of disturbance torque and the complementary sensitivity transfer function directly affects the required control effort. Since the sensitivity function of the system is designed to be very small in the system bandwidth to achieve desirable tracking error; therefore, the control effort required to compensate for disturbances caused by gravitational forces in the system will be considerable. This issue has been previously studied in the context of compensating for disturbance torques in systems, emphasizing the preference for using mechanical or hydro-pneumatic weight compensation systems to avoid the need for high-power actuators in the robot structure.

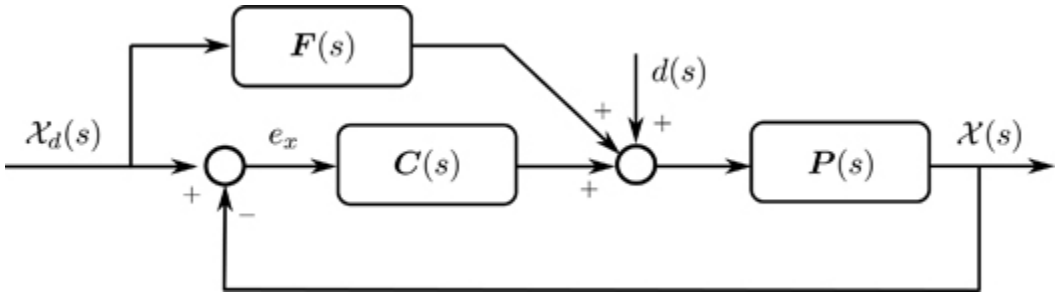
The result of this theoretical analysis of control system design can be summarized as follows. Although increasing the control gain to reduce tracking error in the presence of disturbances and uncertainties in the robot model is important, this control gain should only be increased within the desired bandwidth of the system to reduce the impact of measurement noise and the required control effort. Measurement noise directly affects the tracking error of the system; therefore, the use of appropriate filters to reduce the effect of measurement noise is recommended.

In tuning proportional, derivative, and integral gains of industrial controllers, attention to the required control effort and avoiding it from being in the saturation range is crucial. Thus, it is recommended to determine the proportional and derivative control gains using the desired damping ratio $\xi = 0.707$ and keep this ratio in tuning the bandwidth of the closed loop system. Subsequently, gradually increase the natural frequency of the closed-loop system, and consequently the bandwidth of the system, in such a way that it does not significantly exceed the open-loop system bandwidth and keeps the required control effort in the appropriate range. Furthermore, adjust the integral gain in a way that, while reducing the steady-state error and maintaining the stability of the closed-loop system, it does not have a significant effect on the system's time response speed.

### 11.3.5 Feed-Forward Control

In linear control systems, to enhance performance and reduce tracking error in the presence of disturbances, one can leverage a feed-forward controller in addition to feedback control loops. This control structure, in the presence of disturbances, is illustrated in Figure 11.10. As depicted in this figure, alongside the feedback controller $\boldsymbol{C}(s)$ in the feedback loop, a second feed-forward controller $\boldsymbol{F}(s)$ is employed in the feed-forward path. By utilizing the additional degree of freedom introduced by this feed-forward controller in the closed-loop system structure,

tracking errors can be made smoother, and the impact of disturbances on steady-state error can be completely eliminated.




▶ Long Description for Figure 11.10

**FIGURE 11.10**
Block diagram illustrating robot motion in a single direction with disturbance, using feedback loop and feed-forward controller. ⏎

To analyze the system and design the feed-forward controller, the output transfer function of the system $\mathcal{X}(s)$ in the presence of disturbances and the feed-forward controller $\boldsymbol{F}(s)$ can be expressed as follows.

$$\mathcal{X}(s) = \frac{(\boldsymbol{C}(s) + \boldsymbol{F}(s))\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)}\mathcal{X}_d(s) + \frac{\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)}d(s) \tag{11.28}$$

In this way, the tracking error transfer function is obtained as follows.

$$e_x(s) = \frac{1 - \boldsymbol{F}(s)\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)} \mathscr{X}_d(s) - \frac{\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)} d(s)$$

(11.29)

Compare this equation with the case in which no feed-forward controller is utilized,

$$e_x(s) = \frac{1}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)} \mathscr{X}_d(s) - \frac{\boldsymbol{P}(s)}{1 + \boldsymbol{C}(s)\boldsymbol{P}(s)} d(s)$$

(11.30)

As observed in Equation 11.29, in the presence of a feed-forward controller, the numerator of the tracking error transfer function becomes $1 - \boldsymbol{F}(s)\boldsymbol{P}(s)$. By appropriately selecting the feed-forward controller $\boldsymbol{F}(s)$ such that this expression is set to zero, and considering the stability of the closed-loop system, the tracking error will be reduced to zero for any arbitrary input. This scenario only occurs when the feed-forward controller can be set equal to the inverse of the system transfer function.

$$\boldsymbol{F}(s) = \boldsymbol{P}^{-1}(s)$$

(11.31)

In many systems, including the linear model of individual robot joints, the system transfer function is stable and

minimum-phase, and its inverse can be used as a feed-forward controller.

By reducing the tracking error to zero for any desired motion trajectory in the robot, the effect of disturbances is also reduced. Thus, by considering a feed-forward controller with a large gain in the frequency range where the disturbance due to gravitational force is applied to the robot, the steady-state error caused by the disturbance can be reduced. If the disturbance torque becomes zero over time, the tracking error will also asymptotically converge to zero. Integral gain controllers can also be used to reduce steady-state error and improve tracking performance.

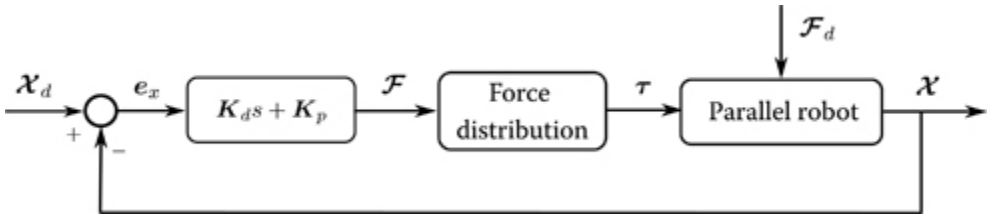
If the disturbance torque can be accurately modeled and calibrated, or if it can be accurately estimated using estimation methods, its effect on the dynamics of the tracking error can be completely compensated. This compensation method will be thoroughly studied in the general case in the context of inverse dynamics control in Section 12.3.

## 11.4 PD Control in Parallel Robots

Thus far, we have explored the fundamental concepts and methodologies for designing linear controllers focused on robot motion in a single direction. This foundational analysis provides a crucial starting point for understanding control systems; however, due to the inherently

multivariable dynamics of parallel robots, implementing these methods directly requires more nuanced consideration.

In this section, we examine in detail the first comprehensive control strategy that can be fully implemented for parallel robots. This strategy is centered around the most basic form of feedback control, tailored specifically for manipulators with complex, interconnected dynamics. As depicted in Figure 11.11, this control structure utilizes a set of independent linear proportional-derivative (PD) controllers, each dedicated to managing a distinct component of the tracking error. The tracking error vector in the task space, denoted by $\boldsymbol{e}_x$, encompasses the discrepancies between the desired and actual positions and orientations. For a typical six-degrees-of-freedom robot like the Stewart-Gough platform, this vector includes six components: $\boldsymbol{e}_x = [e_x, e_y, e_z, e_{\theta_x}, e_{\theta_y}, e_{\theta_z}]^T$, corresponding to translational and rotational errors along different axes.




▶ Long Description for Figure 11.11

**FIGURE 11.11**

PD controller for parallel robots implemented in task space.

⏎

The control strategy adopts a decentralized approach, where each of the six error components is managed by a separate linear PD controller. This setup allows for straightforward tuning and implementation, making it particularly attractive for systems where computational simplicity and clarity are priorities. The PD controller's operation is mathematically represented by the $\boldsymbol{K}_d s + \boldsymbol{K}_p$ block, where $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ are $n \times n$ diagonal matrices that define the derivative and proportional gains, respectively.

Each element on the diagonals of these matrices corresponds to an individual controller gain for a specific error component. This design ensures that each error term is addressed independently, which simplifies the analysis and design process. However, if the controller gain matrices were not restricted to being diagonal, the system would then incorporate off-diagonal elements, leading to a multivariable control structure. In this more complex arrangement, the control action for each joint would depend not only on its corresponding error term but also on the errors in other joints. This interconnected approach could potentially enhance performance by leveraging the full coupling present in parallel robots, but it also complicates the controller gain selection and requires a more detailed understanding of the system dynamics.

Thus, this control structure allows each tracking error component to be addressed individually through separate PD controllers, or the interaction between channels can be accounted for in a multivariable setup. The diagonal

controller design is advantageous because it enables simpler and more focused tuning of each axis or degree of freedom independently, thereby simplifying the complexity of managing the entire system.

The output of the controller, represented by $\boldsymbol{\mathcal{F}}$, defines a comprehensive wrench in the task space. For a general six-degrees-of-freedom robot, such as the Stewart-Gough platform, this wrench includes six components: $\boldsymbol{\mathcal{F}} = [F_x, F_y, F_z, \tau_x, \tau_y, \tau_z]^T$. Each component of the wrench directly manipulates the corresponding tracking error component in task space, ensuring that with appropriately high controller gains, the system can achieve suitable overall tracking performance.

In practical applications, the computed output wrench must be transformed into actuator forces/torques via the force distribution block. For fully parallel manipulators without actuation redundancy, this mapping is typically implemented through the inverse of the manipulator's Jacobian transpose, given by $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$. Although various alternatives to the PD controller can be employed within this framework—such as PID, LQR, or $\mathscr{H}_\infty$ controllers—the PD controller remains a popular choice due to its simplicity and effectiveness in maintaining stability and providing adequate tracking performance.

The proposed decentralized PD controller is notably simple in structure, making it easy to implement and understand. This simplicity extends to the design process, which does not require in-depth knowledge of the

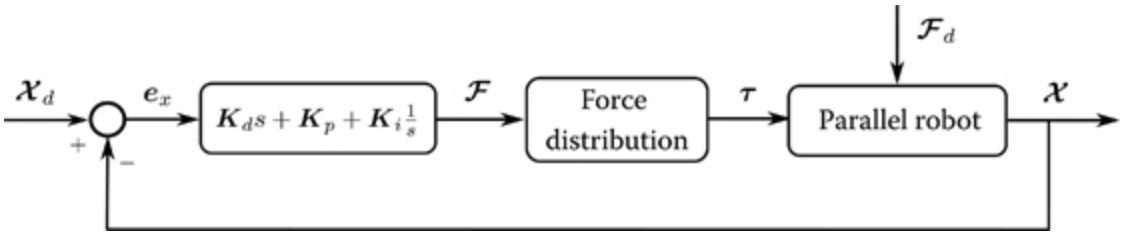
manipulator's dynamic model or calibrating its parameters. The computational demands are minimal, primarily involving straightforward calculations for force distribution. However, this approach's simplicity also means that the tracking performance may not be optimal in all scenarios, particularly in the presence of static tracking errors. Additionally, the necessity for high-gain designs to ensure performance can lead to increased actuation energy requirements, which is a critical consideration in energy-sensitive applications. Moreover, the performance of the closed-loop system can vary significantly depending on the manipulator's configuration, adding another layer of complexity to the system's operational efficiency.

Tuning the controller gains is a critical step that must be performed experimentally, often through trial and error, to achieve the best possible physical realization. This tuning process involves balancing the transient response and steady-state errors, ensuring that the system performs adequately across various configurations. However, as explored before in this book, the dynamic behavior of parallel manipulators is inherently nonlinear and configuration-dependent. This variability poses a significant challenge in identifying a single set of controller gains that deliver consistent performance across all possible configurations. Typically, designers opt for controller gains that optimize tracking accuracy under standard operating conditions, potentially leading to less optimal performance in configurations that were not specifically tuned.

Furthermore, the controller's ability to attenuate measurement noise and external disturbance wrenches applied to the manipulator may be limited, which can impact the system's robustness and reliability.

## 11.5 Multivariable PID Control

Due to the popularity of the PID controller in industry, in this section, we introduce this controller in a multivariable structure for parallel robots. Considering the excellent performance of this type of controller on each joint of the robot, it can also be applied in a multivariable form. In this case, as observed in the block diagram in Figure 11.12, the $n$-dimensional array motion variable $\boldsymbol{\chi}$ is used in the feedback loop, and the multivariable PID controller is constructed using the proportional matrix $\boldsymbol{K}_p$, the derivative matrix $\boldsymbol{K}_d$, and the integral matrix $\boldsymbol{K}_i$. If the gain matrices are assumed to be diagonal, this control structure perfectly aligns with the disjoint PID control for each direction of motion. However, if these matrices are not diagonal, information from the motion of other joints is also used in controlling each joint, making the controller structure multivariable.




▶ Long Description for Figure 11.12

**FIGURE 11.12**

Block diagram of multivariable PID control for a parallel robot. ⏎

The structure of this controller is also very simple and can be easily implemented. The adjustment of its control gains requires minimal robot modeling and calibration. Achieving satisfactory performance in tracking the desired trajectory is possible by using high gains and conducting a tuning process. However, it has been observed by experience that tuning the controller gains can be challenging and may involve significant trial and error, especially when non-diagonal matrices are used in this type of controller.

Given that parameter tuning is conducted for one or a few specific desired motion trajectories, the tracking error for traversing other trajectories in the workspace may not be as favorable. This challenge becomes more pronounced, particularly in the case of steady-state tracking errors. The complete elimination of disturbance torques caused by the gravity vector in different scenarios poses a challenging task, and achieving zero steady-state error for all trajectories poses a significant hurdle.

To enhance the capabilities of this basic control structure and address its limitations, several modifications and improvements can be considered. These enhancements may involve integrating more sophisticated control algorithms, refining the force distribution process, or implementing adaptive mechanisms to adjust to varying conditions. The nonlinear and multivariable control of parallel robots will be examined in detail, in the upcoming chapter. Nonetheless, the controller described in this section serves as a foundational approach and is recommended as an initial attempt in closed-loop experiments due to its straightforward implementation and the ease with which it can be understood and adjusted. This initial setup can provide valuable insights and a baseline performance metric, against which more advanced control strategies can be evaluated and refined.

## 11.6 Case Study I: Planar Cable Robot

In this section, we examine the motion control of the planar 4RPR cable robot, providing a comprehensive exploration of various control strategies and their effectiveness. The planar cable robot, whose architecture is outlined in detail in Section 4.4.1 and depicted in Figure 4.4, features a moving platform supported by four identical RPR kinematic limbs. We conducted a full analysis of the manipulator's Jacobian and singularity properties, as detailed in Section

5.5. Moreover, the dynamic analysis is presented in Section 7.4.

The control strategies we implemented here are the PD and PID controllers in the task space. Throughout this examination, particular attention was paid to the practical implications of each control method. We analyzed how each strategy manages real-world challenges such as noise, model uncertainties, and nonlinearities inherent in the manipulator's dynamics. This detailed analysis not only highlighted the strengths and weaknesses of each control approach but also provided valuable insights into their practical application in similar robotic systems.

### 11.6.1 Decentralized PD Control

The initial control strategy usually considered for parallel robots, employs the most basic form of PD feedback control. As illustrated in Figure 11.11, this control framework utilizes disjoint linear controllers for each error component in the feedback loop.

For planar manipulators, the motion variable in task space is represented by a three-dimensional vector comprising two position coordinates and one orientation angle, $\boldsymbol{\chi} = [x_G, y_G, \phi]^T$. Consequently, the tracking error is defined as $\boldsymbol{e}_x = [e_x, e_y, e_\phi]^T$. This decentralized control scheme incorporates three independent PD controllers, each addressing a specific error component. The PD controller is indicated by the $\boldsymbol{K}_d s + \boldsymbol{K}_p$ block in Figure 11.11, where $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ are $3 \times 3$ diagonal matrices

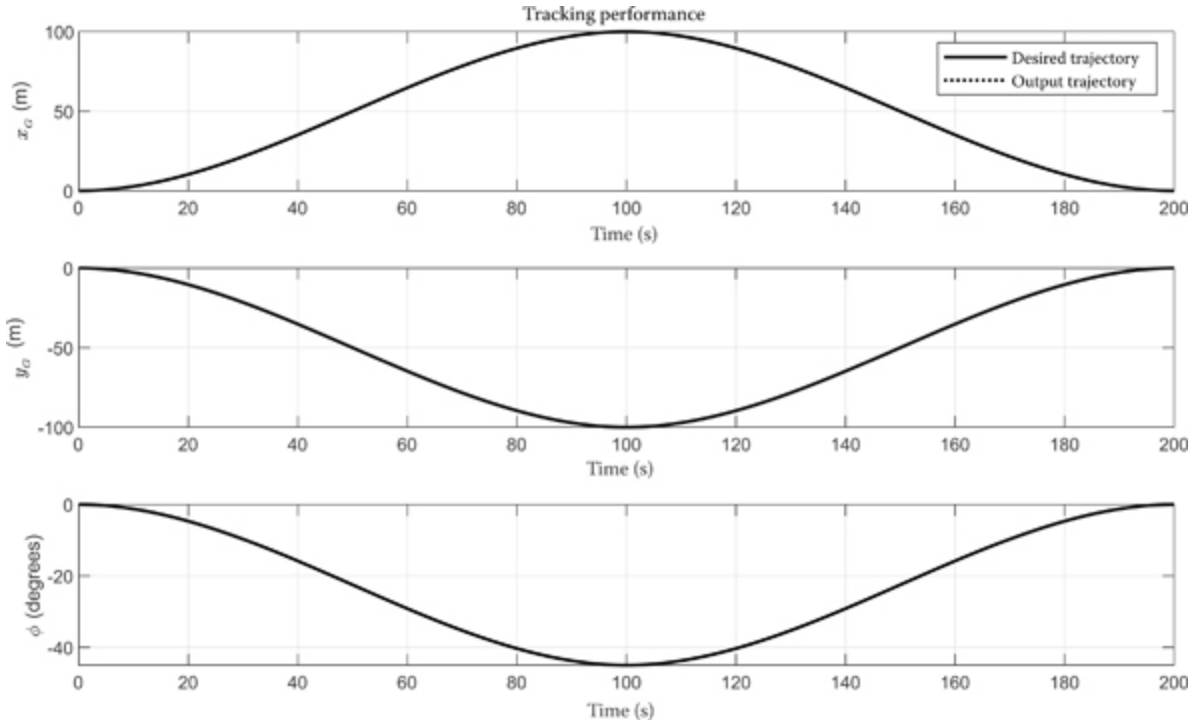
representing the derivative and proportional gains, respectively.

The controller's output, represented by $\boldsymbol{\mathcal{F}}$, is a three-dimensional wrench in the task space. For a planar manipulator with three degrees of freedom, this wrench includes the components $\boldsymbol{\mathcal{F}} = [F_x, F_y, \tau_\phi]^T$. Since the controller's output is specified in the task space, each wrench component directly influences the corresponding tracking error component. Thus, the overall tracking performance of the manipulator can be desirable, only if the controller gains are sufficiently high.

In practical applications, the computed output wrench is converted into actuator forces through the force distribution block. This process involves calculating the necessary actuator forces to generate the desired wrench, which are then applied to the manipulator. As previously mentioned, for a cable-driven redundant manipulator, such as the planar manipulator under discussion, redundancy resolution strategies described in Section 6.6 are used for this conversion.

In the simulations, the geometric and inertial parameters from Table 10.1 are utilized, and the initial conditions for all states are initialized to zero. Additionally, the gravity vector is assumed to be perpendicular to the motion plane, represented as $\boldsymbol{g} = [0, 0, -9.81]^T$. In the initial simulations, no disturbance wrench is applied to the moving platform. The desired trajectory for the moving platform, as shown in Figure 11.13, is defined using cubic polynomial functions

for $x_d$, $y_d$, and $\phi_d$. The simulation duration is set to 200 seconds, with PD gains configured to $\boldsymbol{K}_p = \boldsymbol{K}_d = 10^4 \cdot \text{diag}[1, 1, 100]$, based on extensive trial and error.

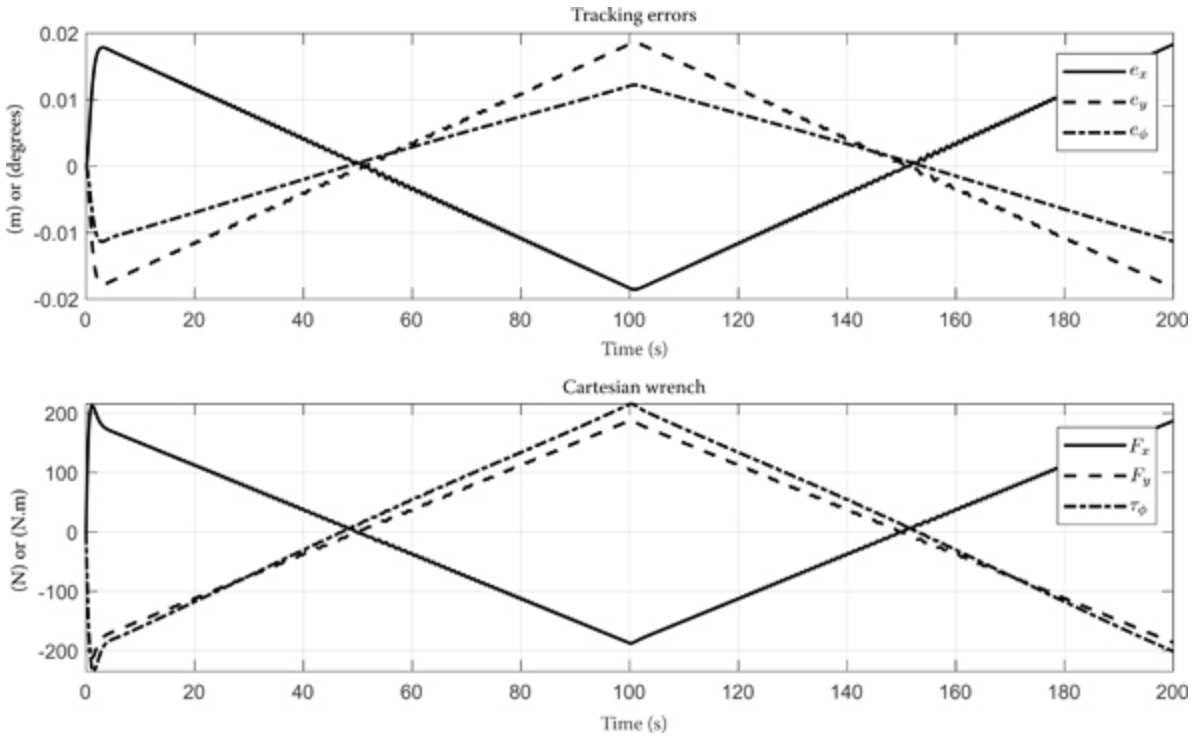



▶ Long Description for Figure 11.13

**FIGURE 11.13**
The desired and output trajectories of the planar 4RPR cable robot under closed loop with decentralized PD controllers. ⏎

From Figure 11.13, it is evident that the desired and achieved closed-loop motions are virtually indistinguishable. For better observation, the tracking error of the moving platform is depicted in the top chart of Figure 11.14. This figure demonstrates that the

decentralized PD controller effectively reduces tracking errors to below 0.02 meters in position and less than 0.012 degrees in orientation. The Cartesian wrench $\boldsymbol{\mathcal{F}} = [F_x, F_y, \tau_\phi]^T$ produced by the PD control action at the moving platform is illustrated in the bottom chart of Figure 11.14. The wrenches exhibit a pattern closely aligned with the desired trajectory accelerations, which are linear for cubic trajectories.

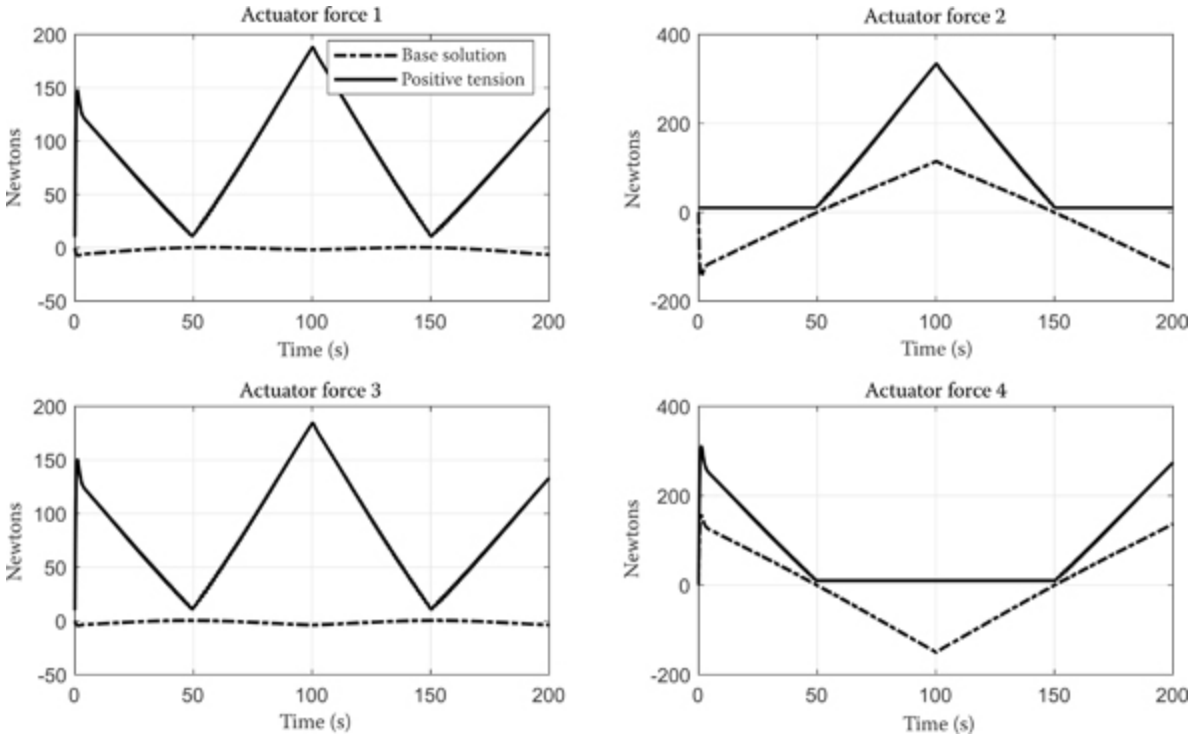



▶ Long Description for Figure 11.14

**FIGURE 11.14**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under decentralized PD control. ⏎

Figure 11.15 presents the actuator forces of the manipulator. Two sets of actuator forces are shown: the dash-dotted lines represent the base solution of the redundancy resolution optimization problem using the pseudo-inverse as given in Equation 6.84, while the solid line shows the final solution obtained through the numerical method for the redundancy resolution problem detailed in Equation 6.87, to ensure positive tension in the cables. The results indicate that as the manipulator moves in the positive $x$ and negative $y$ directions, the forces in the second and fourth limbs are predominant. Additionally and as expected, the tension forces determined by the redundancy resolution scheme remain positive across all configurations.



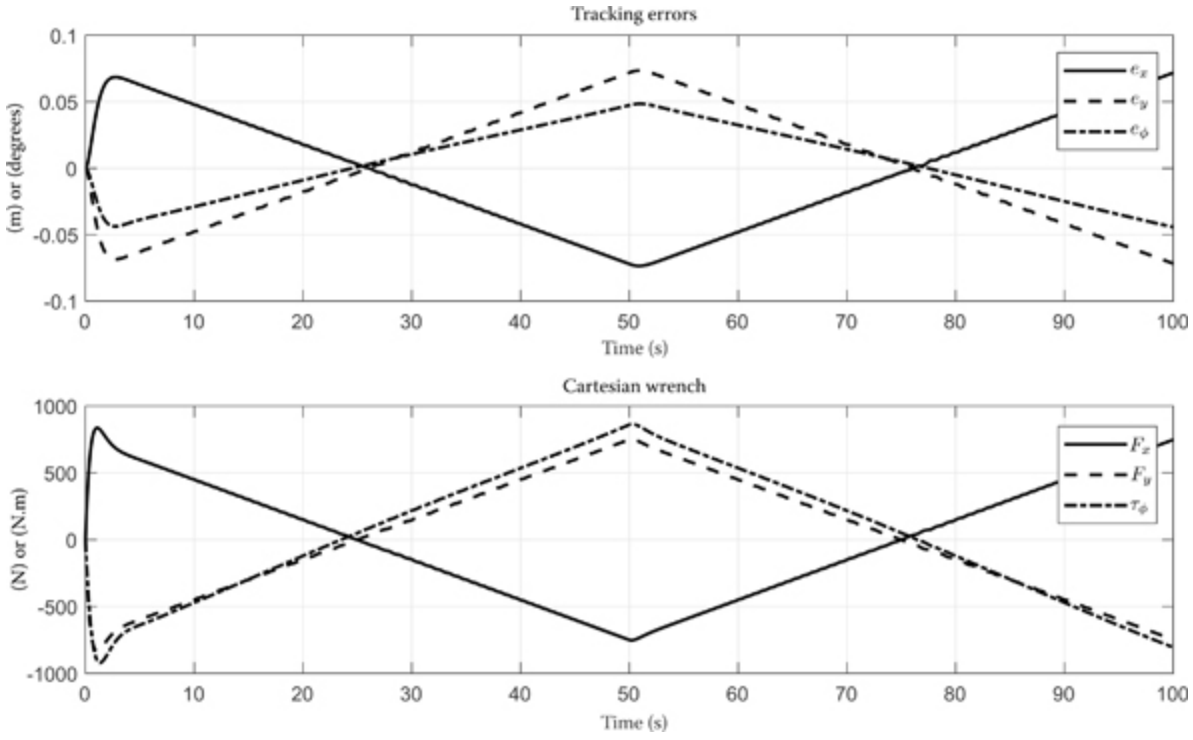

▶ Long Description for Figure 11.15

**FIGURE 11.15**

The actuator forces of the planar 4RPR cable robot generated by decentralized PD controller to track trajectories shown in Figure 11.13. ⏎

## Simulations on Different Trajectories

To assess the effectiveness of the PD controller in real-world applications, a new desired trajectory is examined where the same maneuvers must be completed in just 100 seconds. This is twice as fast as in the previous simulation shown in Figure 11.13. This accelerated timeline challenges the controller's ability to maintain precision and stability. The resulting tracking errors for this faster trajectory and the necessary Cartesian wrench are depicted in Figure 11.16.



▶ Long Description for Figure 11.16

**FIGURE 11.16**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under decentralized PD control, for a more rapid maneuver. ⏎

As observed in the top chart of Figure 11.16, the tracking errors have increased approximately fourfold, indicating more deviation from the desired trajectory. Similarly, the required Cartesian wrenches are also about four times higher than in the prior simulation, highlighting the increased effort needed to follow the trajectory.

This substantial rise in both error and force requirements is primarily due to the higher acceleration and the corresponding inertial forces induced in the closed-loop system. Since cubic trajectories are employed in these simulations, doubling the speed naturally leads to a fourfold increase in acceleration, amplifying the challenges faced by the control system.

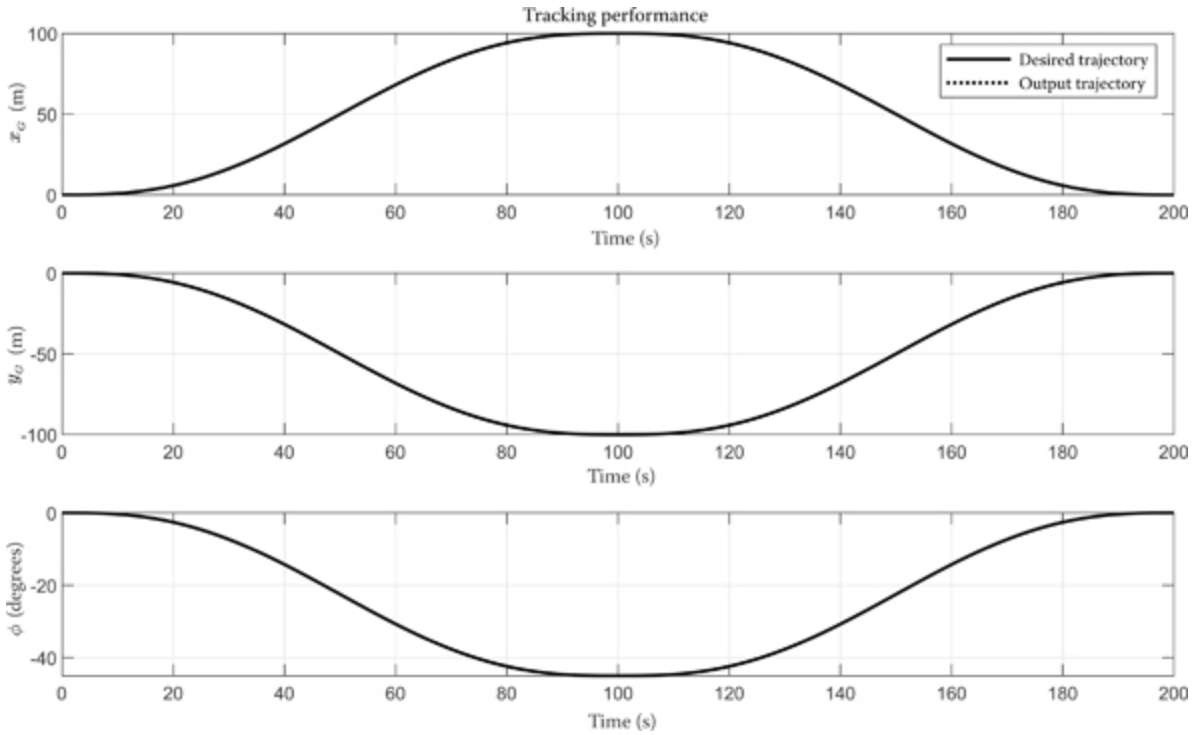
As demonstrated in this simulation, the robot's tracking performance has significantly deteriorated under the new conditions. The sharp increase in tracking errors suggests that the current controller setup is insufficient for maintaining the desired accuracy at this higher speed. To achieve high precision in such demanding scenarios, it is crucial to adjust the controller gains. These gains can be fine-tuned through a process of trial and error to balance transient response and steady-state errors across different

configurations. However, this is not a straightforward task; the manipulator's dynamic behavior varies with its configuration, making it challenging to identify a single set of optimal controller gains that ensure consistent performance across all scenarios.

Typically, the gains are selected to meet the performance requirements for the manipulator's most common operating conditions. For the Large Adaptive Reflector (LAR) application of the planar cable robot under consideration, maneuvers lasting over 200 seconds are more commonly used and therefore, this will be considered in future simulations. Note that, this approach may lead to varied performance in other configurations that were not accounted for during the tuning process, as evidenced by the increased tracking errors in Figure 11.16. This variability underscores the need for a more adaptive control strategy or the consideration of advanced control techniques to maintain consistent performance across all operational scenarios, The topic which is well studied in the upcoming chapter.

To further evaluate the robustness of the PD controller in real-world applications, a new desired trajectory is examined. In this case, the same maneuvers as the first simulation are defined, but using quintic polynomial functions, which offer smoother transitions and more flexibility than cubic polynomials. The simulation duration is extended to 200 seconds as shown in Figure 11.17 to allow for a comparable analysis. The PD gains are kept as

the tuned values of $\boldsymbol{K}_p = \boldsymbol{K}_d = 10^4 \cdot \text{diag}[1, 1, 100]$, consistent with the first simulation to maintain comparability.

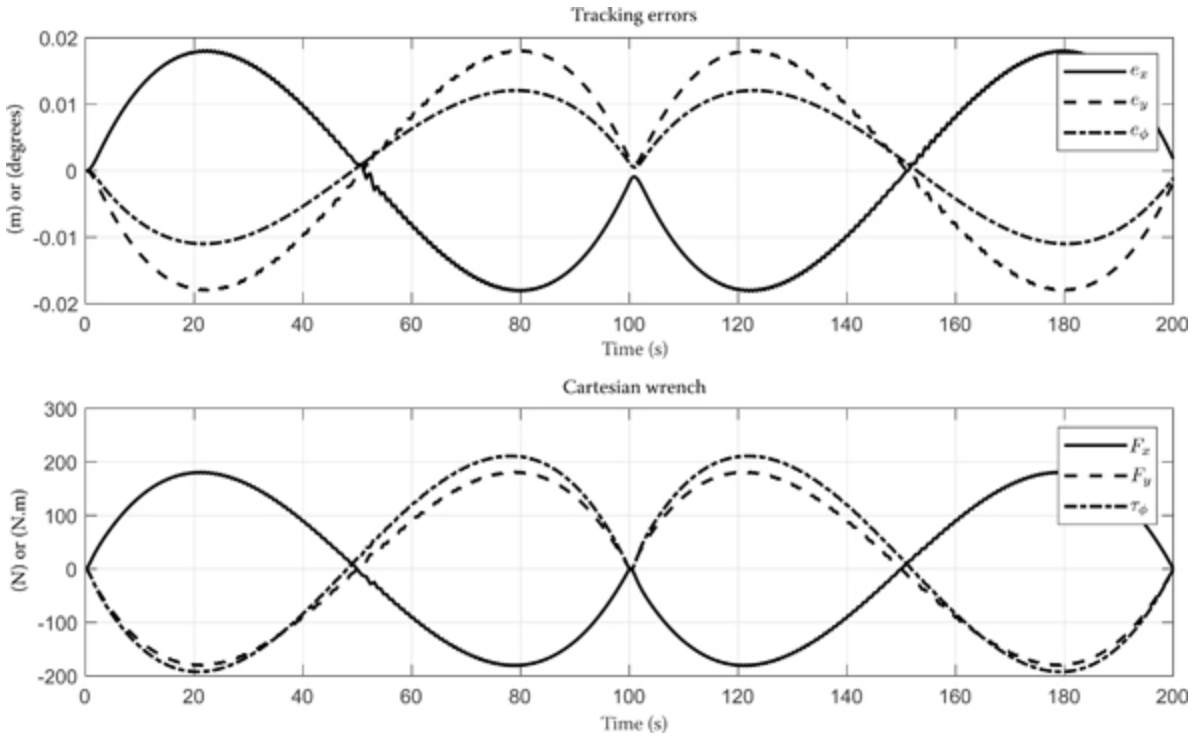



▶ Long Description for Figure 11.17

**FIGURE 11.17**
The desired and output trajectories of the planar 4RPR cable robot designed based on quintic polynomial. ⏎

For this scenario, the tracking error of the moving platform is depicted in the top chart of Figure 11.18. As seen in this figure, although the pattern of the tracking error for this trajectory differs significantly from that of the cubic trajectory, the tracking errors remain below 0.02 meters in position and less than 0.011 degrees in orientation. This performance is comparable to that

observed in the first simulation, indicating that the controller can handle different types of trajectories with similar accuracy.



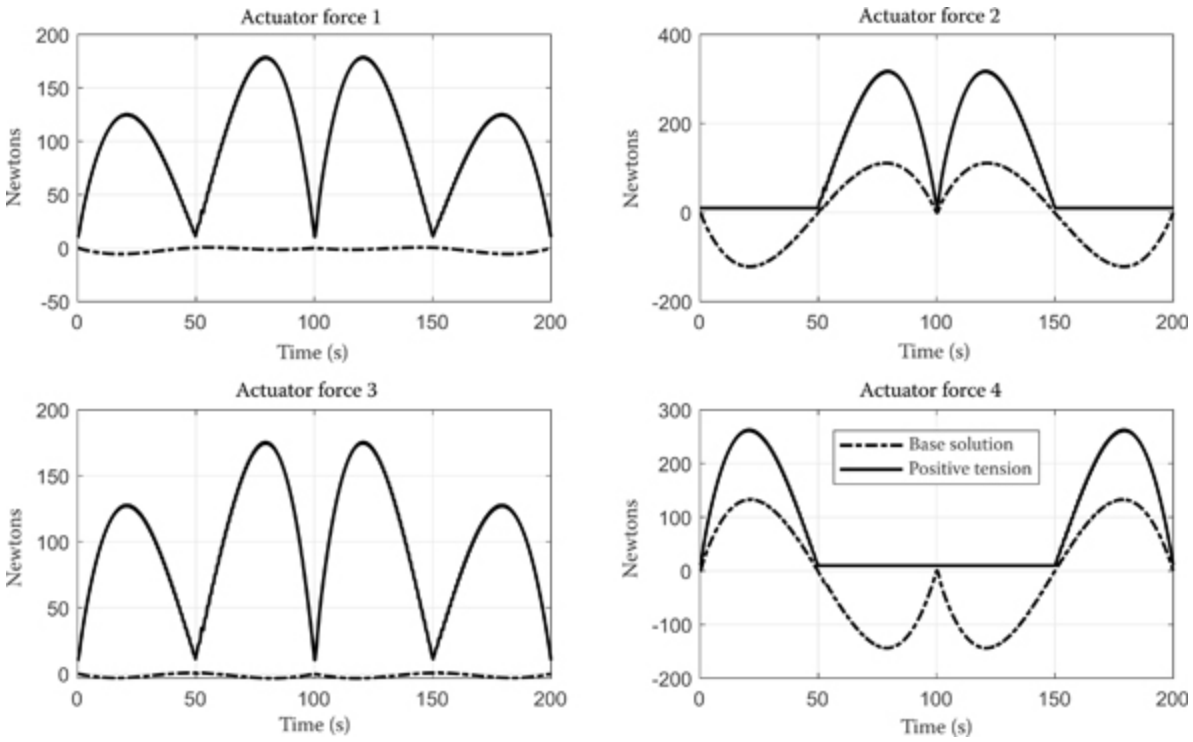

▶ Long Description for Figure 11.18

**FIGURE 11.18**
The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under decentralized PD control for quintic trajectory. ⏎

The Cartesian wrench $\boldsymbol{\mathcal{F}} = [F_x, F_y, \tau_\phi]^T$ produced by the PD control action at the moving platform is illustrated in the bottom chart of Figure 11.14. These wrenches, while similar in magnitude to those required for the cubic trajectory, exhibit a pattern that closely follows the desired trajectory accelerations, which are inherently third-order

polynomials in the case of quintic trajectories. This alignment suggests that the controller effectively manages the dynamic demands of higher-order polynomial paths.

Figure 11.15 presents the actuator forces of the manipulator, providing insight into the mechanical power required on the actuators. As before, two sets of actuator forces are shown: the dash-dotted lines represent the base solution of the redundancy resolution optimization problem, and the solid lines indicate the final solution to ensure positive tension in cables. The distribution of actuator forces in this scenario differs from that observed in the initial simulation, reflecting the changes in trajectory dynamics. However, the peak forces required remain almost the same, demonstrating the controller's capability to maintain consistent force requirements despite different trajectory profiles. Additionally, the tension forces determined by the redundancy resolution scheme remain positive across all configurations, ensuring the physical feasibility of the forces applied to the manipulator. This consistency observed in this comparison is crucial for maintaining suitable performance of the closed-loop system under various operating conditions.




▶ Long Description for Figure 11.19

**FIGURE 11.19**

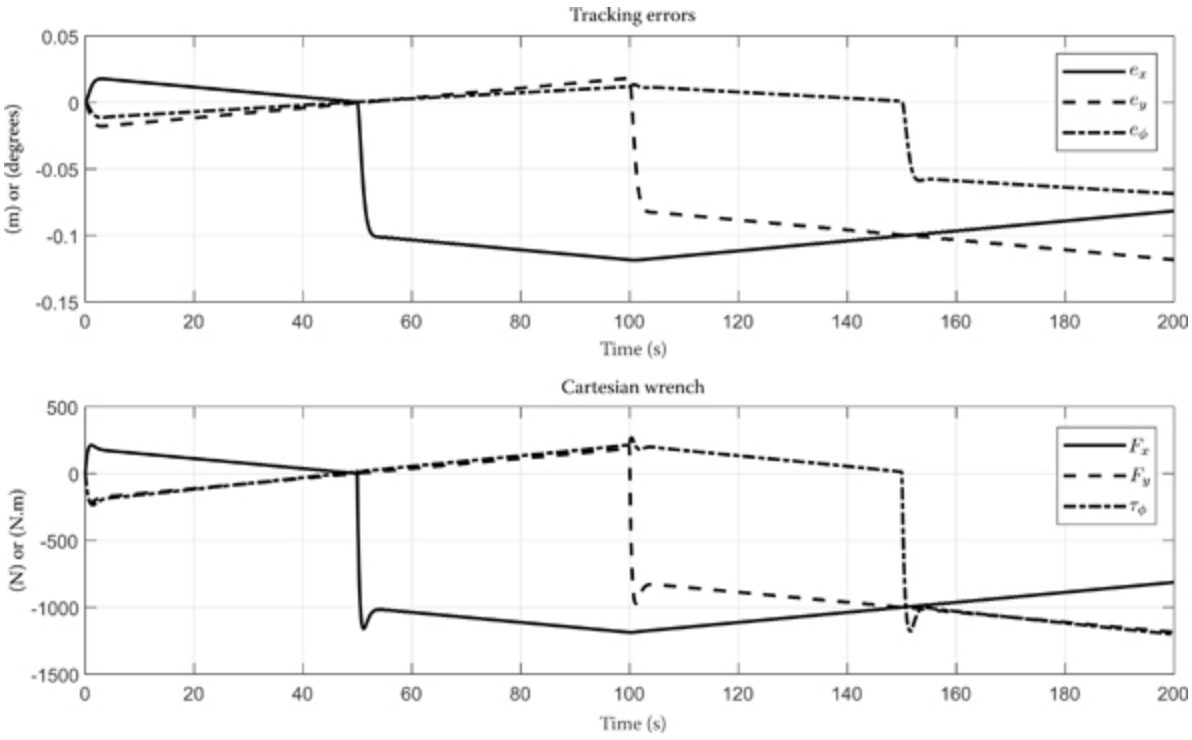
The actuator forces of the planar 4RPR cable robot generated by decentralized PD controller to track trajectories shown in Figure 11.17.

## Effect of Input Disturbance

To assess the PD controller's effectiveness in mitigating external disturbances and measurement noise, two additional simulations are conducted for the closed-loop system. The first simulation involves applying a unit step disturbance force while tracking the cubic trajectories shown in Figure 11.13. To distinguish the effects of various disturbances, a series of specific disturbances are applied at different times: a $1\,kN$ step disturbance force in the $x$

direction ($F_{d_x}$) is introduced at 50 seconds, followed by a $1\,kN$ step disturbance force in the $y$ direction ($F_{d_y}$) at 100 seconds, and a $1\,kNm$ disturbance torque in the $\phi$ direction ($\tau_{d_\phi}$) at 150 seconds. The controller gains are kept at $\boldsymbol{K}_p = \boldsymbol{K}_d = 10^4 \cdot \text{diag}[1, 1, 100]$, consistent with previous tuned settings.

The closed-loop tracking error for this scenario is illustrated in the top chart of Figure 11.20 for the cubic polynomial trajectory depicted in Figure 11.13. When compared to Figure 11.14, it becomes evident that the PD controller does not entirely suppress the influence of external disturbances. The tracking error exhibits distinct spikes corresponding to the moments when the disturbance wrenches are applied. This indicates that while the PD controller can somewhat mitigate disturbances, it cannot fully eliminate their impact, leading to steady-state error in the system's performance. Additionally, the continuous application of disturbances results in the accumulation of static errors within the closed-loop system.




▶ Long Description for Figure 11.20

**FIGURE 11.20**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under decentralized PD control in the presence of disturbance. ⏎

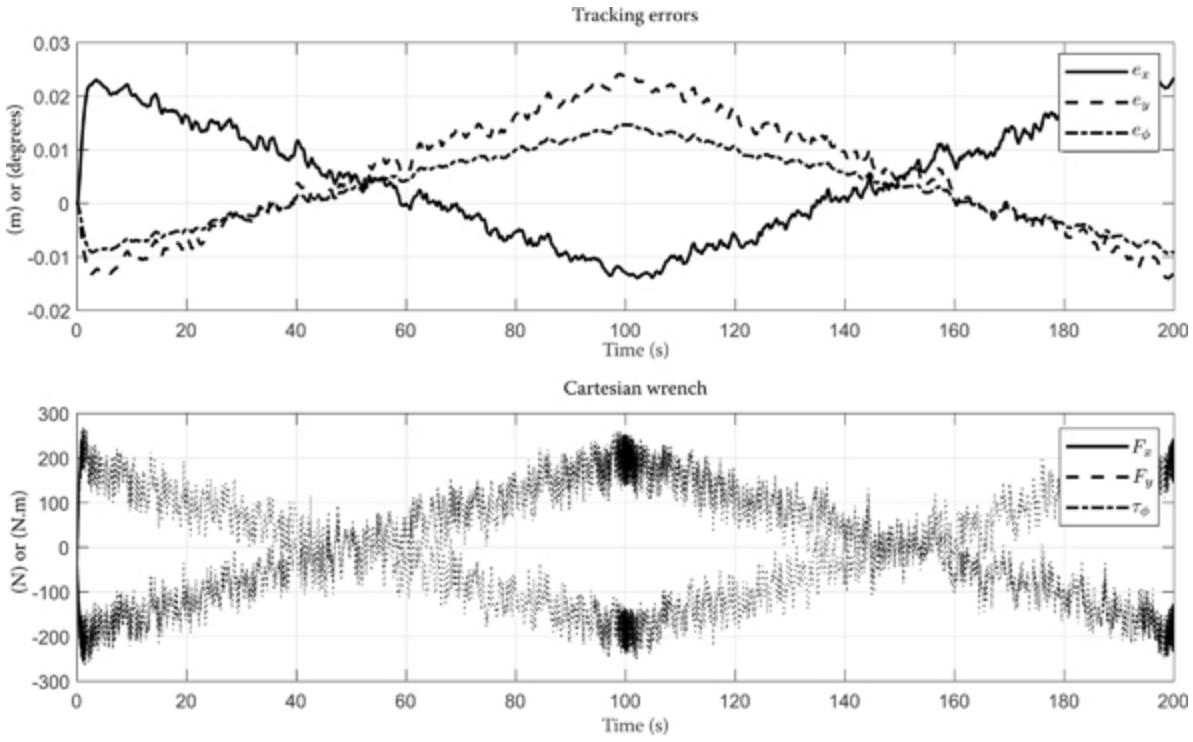
The bottom chart of Figure 11.20 depicts the Cartesian wrench produced by the PD controller while performing the maneuver among disturbances. This figure clearly shows how the disturbances affect the required control forces at 50, 100, and 150 seconds. The wrench components vary significantly at these points, reflecting the increased force demands necessary to counteract the disturbances. This effect underscores the limitations of the PD controller in addressing external forces and the resultant adjustments

needed in the control forces to maintain desired performance.

## Effect of Measurement Noise

Next, we examine the impact of measurement noise by introducing Gaussian noise with an amplitude of 0.1% of the peak values of the original signals into all measured variables $\boldsymbol{\chi}$. Although this noise level is relatively small, its effect on the performance of the decentralized PD controller is significant due to the high-gain settings used in the controller.

Figure 11.21 illustrates that while the overall tracking error only shows a slight increase compared to the noise-free system, the Cartesian wrenches required to maintain the desired trajectory become highly oscillatory. This pronounced oscillation stems from the use of high PD gains to achieve the necessary tracking accuracy. Despite the noise having an amplitude of approximately $10^{-4}$ times the peak values of the measured signals, the large PD gains exacerbate the noise's effect, causing noticeable fluctuations in the actuator forces.




▶ Long Description for Figure 11.21

**FIGURE 11.21**

The closed-loop tracking performance of the planar 4RPR cale robot for decentralized PD control in the presence of measurement noise. ⏎

This oscillatory behavior poses practical challenges. Generating such highly oscillatory forces is often infeasible in real-world applications, leading to potential limitations in achieving consistent tracking performance. Consequently, the desired tracking performance highlighted in Figure 11.21 becomes increasingly difficult to achieve under these noisy conditions. The practical implications of this are significant: while the tracking error might remain relatively controlled, the increased Cartesian forces and

consequently actuator forces and their oscillatory nature can impair the system's overall performance and reliability. Thus, effective noise management and perhaps an adjustment in control strategy are crucial to improve the practical implementation of the controller and enhance performance in noisy environments.

### 11.6.2 Multivariable PID Control

The PID controller is the most widely used controller for trajectory tracking in industrial robots. In this section, we examine the tracking performance of the desired trajectory in the planar 4RPR cable robot using a decentralized implementation of this controller. As illustrated in the block diagram in Figure 11.12, the feedback loop utilizes the motion variable $\boldsymbol{\chi}$, and the decentralized PID controller is constructed with proportional matrix $\boldsymbol{K}_p$, derivative matrix $\boldsymbol{K}_d$, and integral matrix $\boldsymbol{K}_i$. For this simulation, diagonal matrices are used in the design of the PID controller.

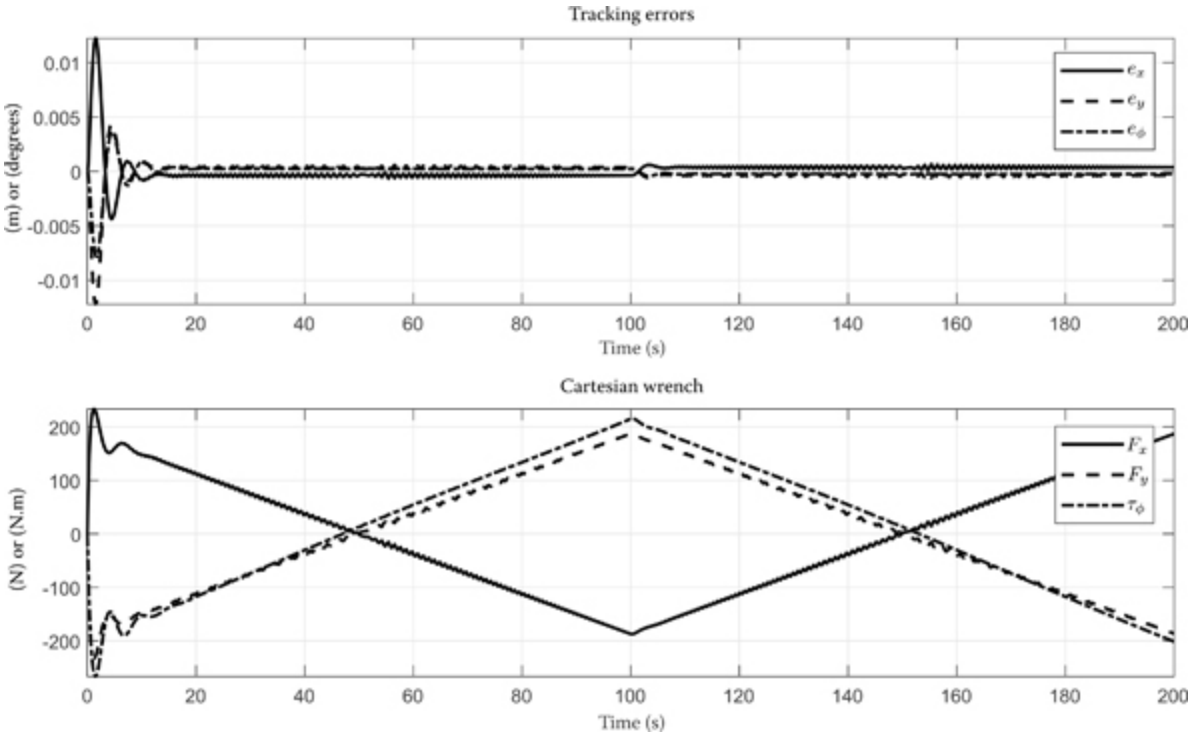
The controller's output, denoted by $\boldsymbol{\mathcal{F}}$, is a three-dimensional wrench in the task space. In practical applications, the computed output wrench is converted into actuator forces through the force distribution block. This process involves determining the necessary actuator forces to produce the desired wrench, which are then applied to the manipulator. As previously mentioned, for a cable-driven redundant manipulator like the planar manipulator under discussion, redundancy resolution strategies described in Section 6.6 can be used for this conversion.

The structure of this controller is straightforward and can be easily implemented. Adjusting its control gains requires minimal robot modeling and calibration. Achieving satisfactory performance in tracking the desired trajectory is possible by using high gains and conducting a tuning process. However, experience has shown that tuning the controller gains can be challenging and may involve significant trial and error, particularly when non-diagonal matrices are employed in this type of controller.

In these simulations, the geometric and inertial parameters from Table 10.1 are used as in the PD controller simulations. In the first simulation, no disturbance wrench or measurement noise is applied to the moving platform, while the desired trajectory for the moving platform is defined using cubic polynomial functions as depicted in Figure 11.13. The simulation duration is set to 200 seconds, with PID gains set to $\boldsymbol{K}_p = \boldsymbol{K}_d = \boldsymbol{K}_i = 10^4 \cdot \text{diag}\,[1,\, 1,\, 100]$, based on the pre-assigned controller gains of the PD controller.

In this first simulation, the tracking error of the moving platform is shown in the top chart of Figure 11.22. This figure demonstrates that the decentralized PID controller effectively reduces tracking errors to about 0.01 meters in position and less than 0.008 degrees in orientation at initial times. Furthermore, the effect of the integral term is clearly evident in the steady-state tracking error, which is significantly reduced to about $5 \times 10^{-3}$ for both translational and rotational motion. This highlights the

advantage of the PID controller in minimizing long-term error accumulation.



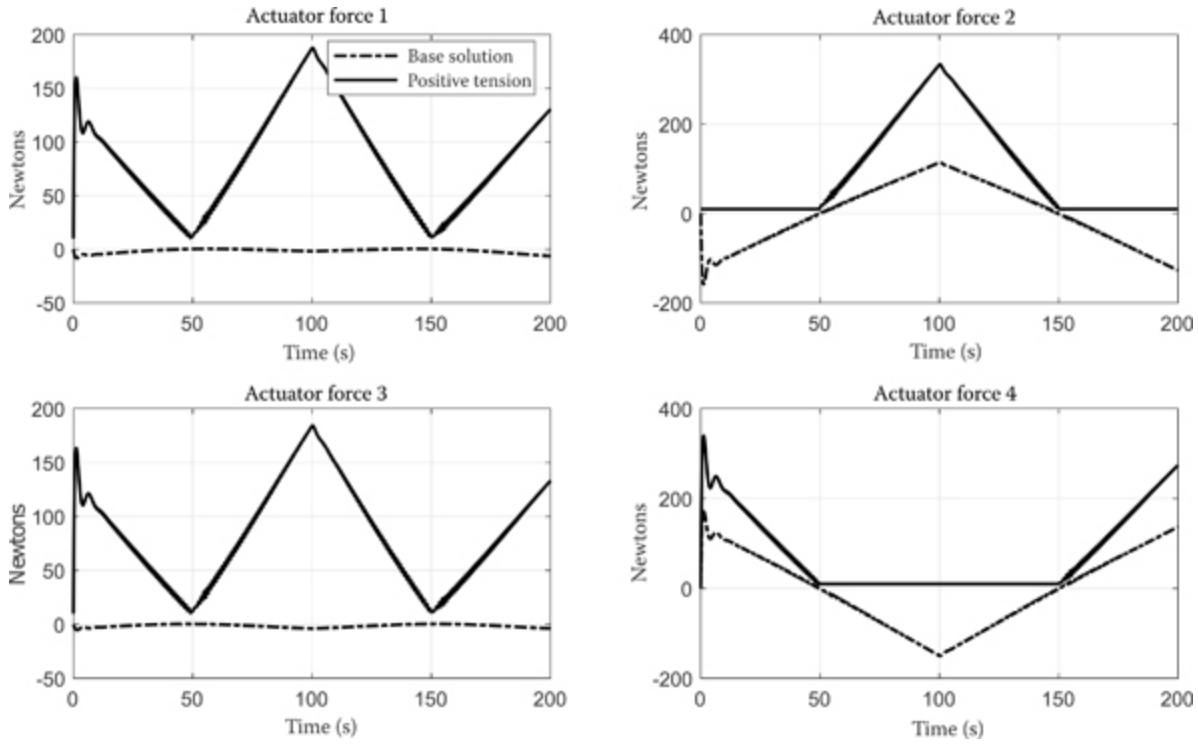

▶ Long Description for Figure 11.22

**FIGURE 11.22**
The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under decentralized PID control. ⏎

The Cartesian wrench $\boldsymbol{f} = [F_x, F_y, \tau_\phi]^T$ produced by the PID controller at the moving platform is illustrated in the bottom chart of Figure 11.22. These wrenches exhibit a pattern closely aligned with the desired trajectory accelerations, reflecting the controller's responsiveness to changes in motion dynamics. The peak values of the wrenches are almost identical to those observed with the

PD controller, indicating that both controllers impose similar maximum demands on the system. However, the PID controller achieves much higher precision, especially in steady-state conditions, due to its integral action. This observation explains why PID controllers are frequently used in industrial robots, as they provide superior tracking accuracy and stability.

Figure 11.23 presents the actuator forces of the manipulator, offering a detailed view of the mechanical outputs required to follow the desired trajectory. As before, two sets of actuator forces are shown: the dash-dotted lines represent the base solution, and the solid lines show the final solution of the optimal redundancy resolution technique. The output actuator forces are very close to those in the case of PD controllers, with only minor changes observed in the transient response. This similarity indicates that the PID controller's adjustments do not significantly alter the overall force distribution, maintaining the system's mechanical consistency. Additionally, the tension forces determined by the redundancy resolution scheme remain positive across all configurations, ensuring the physical feasibility and safety of the applied forces throughout the motion. This consistent positive tension is crucial for preventing cable slack and maintaining precise control of the manipulator.




▶ Long Description for Figure 11.23

**FIGURE 11.23**

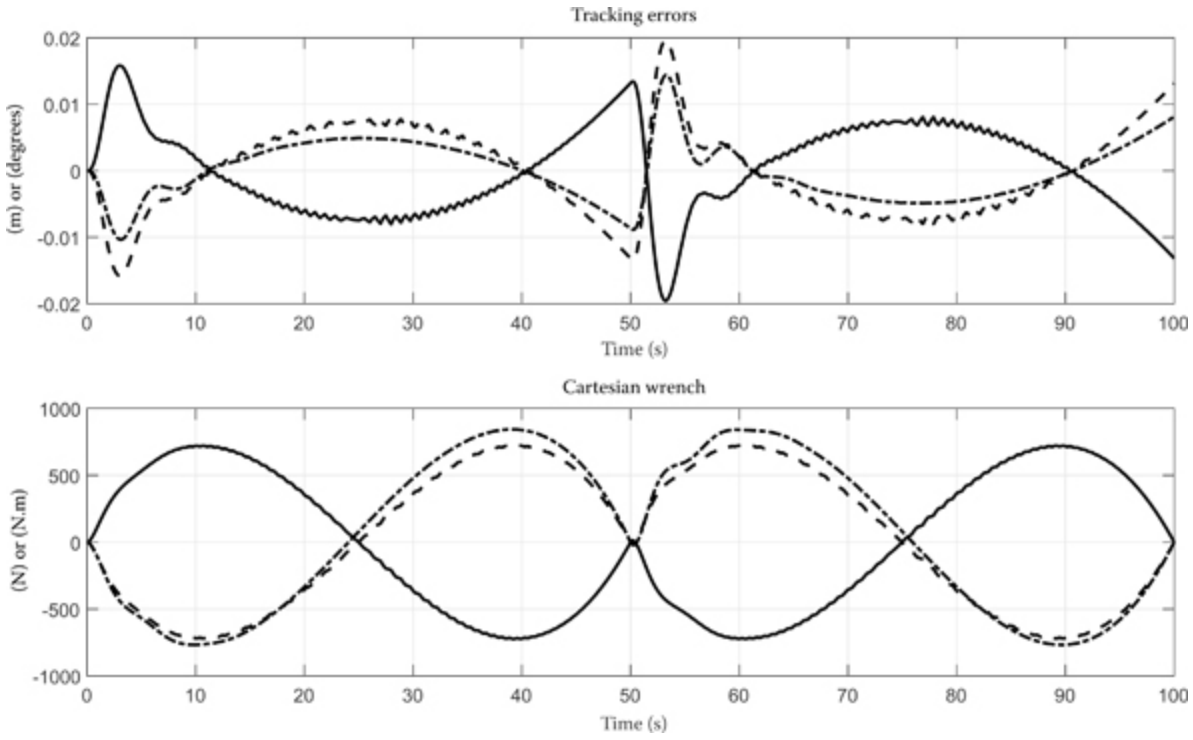
The actuator forces of the planar 4RPR cable robot generated by decentralized PID controller to track the cubic trajectories shown in Figure 11.13. ⏎

## Simulations on Different Trajectories

In the second simulation, a faster maneuver and a different trajectory planning routine are considered. For this purpose, a quintic desired trajectory is examined for the same maneuvers, which must be completed in just 100 seconds. This scenario investigates the effect of using higher-order trajectory planning in a more rapid motion, which is twice as fast as in the previous simulation. All other simulation parameters remain unchanged, and no

disturbance or noise is included in the simulation, while the PID gains are kept as the tuned values of $\boldsymbol{K}_p = \boldsymbol{K}_d = \boldsymbol{K}_i = 10^4 \cdot \text{diag}\,[1, 1, 100]$, consistent with the first simulation to maintain comparability.

In this scenario, the tracking error of the moving platform is depicted in the top chart of Figure 11.24. As seen in this figure, although the pattern of the tracking error for this trajectory differs significantly from that of the cubic trajectory, the tracking errors remain below 0.02 meters in position and less than 0.011 degrees in orientation, even for this more rapid maneuver. This performance is quite comparable to that observed in the first simulation of the PID controller, indicating that the controller can handle different types of trajectories with similar accuracy. However, it is observed that the steady-state tracking error here is larger than that for the slower trajectory.




▶ Long Description for Figure 11.24

**FIGURE 11.24**

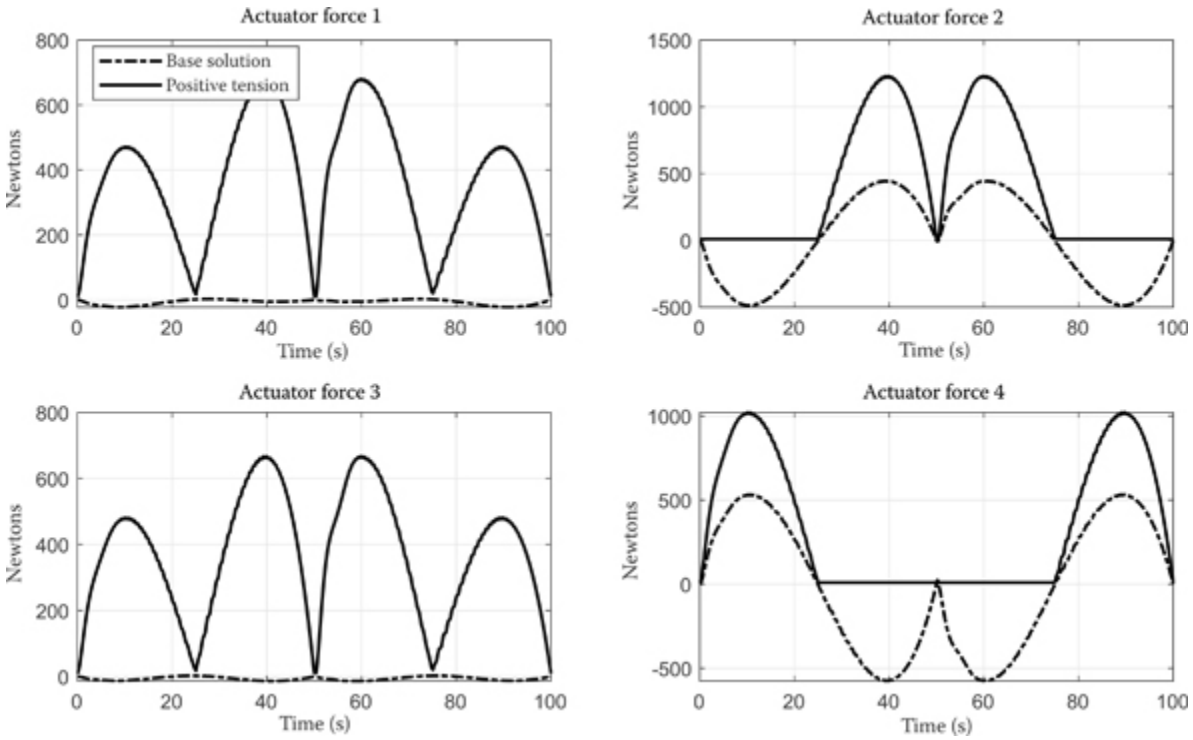
The closed-loop tracking performance and the Cartesian wrench generated by the planar 4RPR cable robot using decentralized PID control for high-speed quintic trajectories. ⏎

The Cartesian wrench $\boldsymbol{\mathcal{f}} = [F_x, F_y, \tau_\phi]^T$ produced by the PID control action at the moving platform is illustrated in the bottom chart of Figure 11.22. These wrenches exhibit a pattern closely aligned with the desired trajectory accelerations, reflecting the controller's responsiveness to changes in motion dynamics. The peak values of the wrenches are almost three times higher than those observed for the slower trajectory, maintaining the overall

precision of the robot for faster maneuvers. This observation highlights the effectiveness of PID controllers in handling a broader range of trajectory demands without requiring further tuning.

## Effect of Input Disturbance

To illustrate the PID controller's significance in mitigating external disturbances and measurement noise, two additional simulations are conducted for the closed-loop system. The first simulation involves applying a unit step disturbance force while tracking the cubic trajectory shown in Figure 11.13. As for the PD controller case, a series of specific disturbances are applied at different instances, namely 50, 100 and 150 seconds. All other simulation parameters including the controller gains are kept consistent with previous settings.



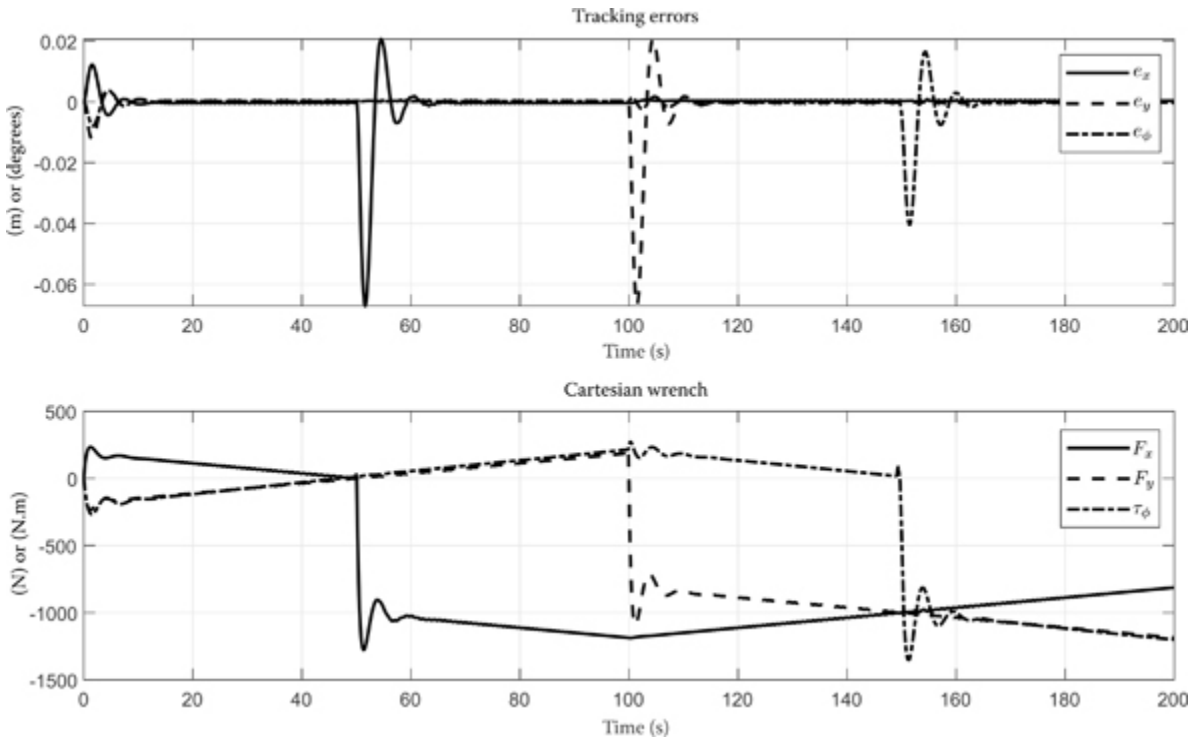

▶ Long Description for Figure 11.25

**FIGURE 11.25**

The actuator forces of the planar 4RPR cable robot, as generated by the decentralized PID controller, for tracking high-speed quintic trajectories.

The closed-loop tracking error for this scenario is illustrated in the top chart of Figure 11.26 for the cubic polynomial trajectory depicted in Figure 11.13. When compared to Figure 11.20, it becomes evident that the PID controller significantly suppresses the influence of external disturbances compared to that of PD controller. The tracking error exhibits distinct spikes corresponding to the moments when the disturbance wrenches are applied, while the effect of disturbance is significantly mitigated as

time evolves. This indicates that the PID controller can significantly attenuate the effect of disturbances on the performance of the robot. Additionally, since step disturbances are used, it is typical to have an accumulation of static errors within the closed-loop system, while the integral term of the controller, has significantly reduced this effect.




▶ Long Description for Figure 11.26

**FIGURE 11.26**

The closed-loop tracking performance and the generated Cartesian wrench of the planar 4RPR cable robot using decentralized PID control for a cubic trajectory in the presence of disturbance. ⏎

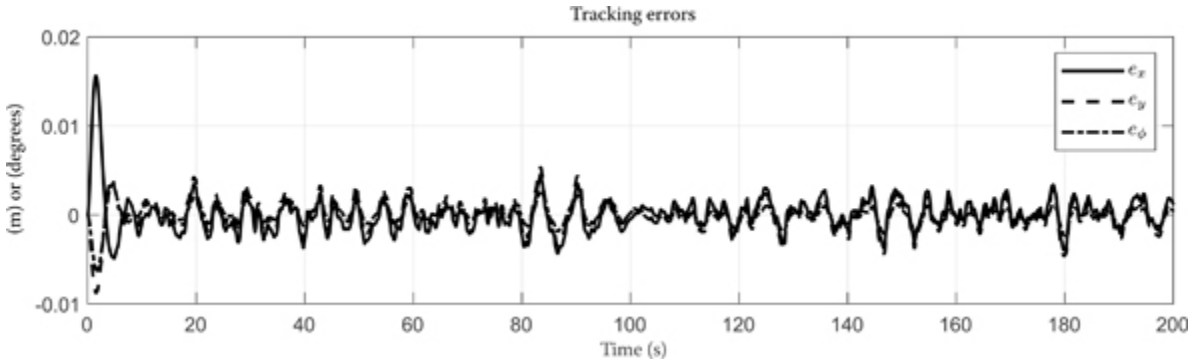
The bottom chart of Figure 11.26 depicts the Cartesian wrench produced by the PID controller while performing the maneuver among disturbances. This figure clearly shows how the disturbances affect the required control wrenches at 50, 100, and 150 seconds. The wrench components vary significantly at these points, reflecting the increased force demands necessary to counteract the disturbances. This effect underscores the limitations of the PID controller in addressing non-vanishing external forces and the resultant adjustments needed in the control forces to maintain the desired performance.

## Effect of Measurement Noise

Next, we investigate the impact of measurement noise by introducing Gaussian noise with an amplitude of 0.1% of the peak values of the original signals into all measured motion variables $\boldsymbol{\chi}$. Although this noise level is relatively low, its influence on the controller's performance is substantial, primarily due to the high-gain settings employed.

Figure 11.21 illustrates the tracking error of the PID controller in the presence of this noise. The results show that the tracking error remains within $5 \times 10^{-3}$, except for the initial moments. This remarkable improvement in tracking accuracy amidst the noise is largely attributed to the integral term of the PID controller, which effectively counteracts the deviations caused by the noise.

The Cartesian wrenches and actuator forces required to achieve this level of precision are very similar to those seen with the PD controller, as depicted in Figure 11.21, which are highly oscillatory. The generation of such oscillatory forces can be challenging in real-world applications due to the limited bandwidth of actuators. This can constrain their ability to produce consistent and smooth control actions. These oscillations may lead to potential difficulties in maintaining reliable and accurate tracking performance in practice. Therefore, to address these challenges, effective noise management strategies and the use of appropriate filters are crucial.




▶ Long Description for Figure 11.27

**FIGURE 11.27**

The performance of closed-loop tracking for the planar 4RPR cable robot using decentralized PID control under conditions of measurement noise.

## 11.7 Case Study II: Stewart Gough Platform

This section explores the motion control of the Stewart–Gough platform, focusing on implementing linear control strategies on the manipulator. The manipulator's structure is detailed in Section 4.6.1 and shown in Figure 4.16. The platform's motion, driven by six piston–cylinder actuators, is thoroughly analyzed in terms of kinematics in Section 4.6 and the Jacobian in Section 5.7.

Dynamic behavior is examined using the Newton–Euler method in Section 7.5 and the virtual work approach in Section 9.6. The closed-form dynamic formulation for simulations is derived in Section 8.5. The PD and PID control strategies are applied in task space, with their closed-loop performances analyzed in subsequent subsections.

### 11.7.1 Decentralized PD Control

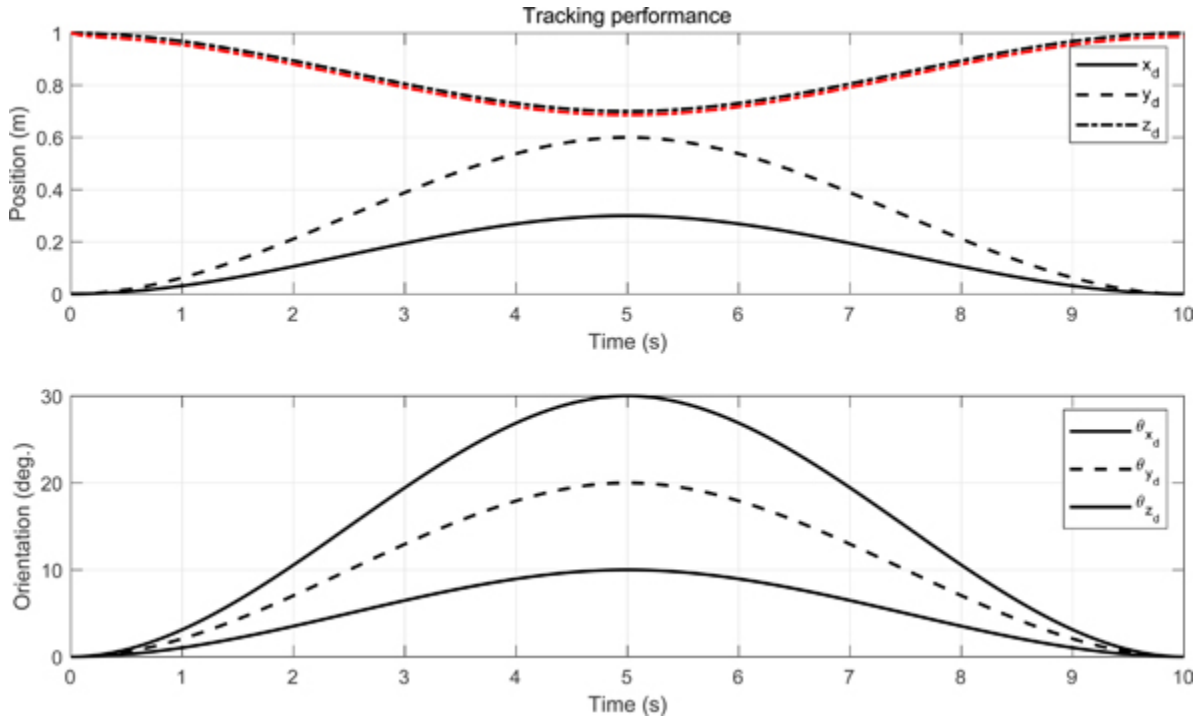
The initial control strategy for parallel robots often employs simple PD feedback control. As shown in Figure 11.11, independent PD controllers manage each error component in the feedback loop. For a Stewart–Gough platform with six degrees of freedom, the motion variable in task space is a six-dimensional array $\boldsymbol{\chi} = [\boldsymbol{x}_p, \boldsymbol{\theta}]^T$, representing the platform's position and orientation.

The tracking error $\boldsymbol{e} = [e_x, e_y, e_z, e_{\theta_x}, e_{\theta_y}, e_{\theta_z}]^T$ is addressed by six independent PD controllers. These controllers are represented by the $\boldsymbol{K}_d \boldsymbol{s} + \boldsymbol{K}_p$ block in Figure 11.11, with $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ as diagonal matrices indicating controller gains. The controller's output, $\boldsymbol{f}$, is a

six-dimensional wrench in task space. For planar manipulators, this wrench includes components like $\boldsymbol{\mathcal{F}} = [F_x, F_y, F_y, \tau_x, \tau_y, \tau_z]^T$. The controller's output is converted into actuator forces via the force distribution block, using the inverse of the Jacobian transpose at nonsingular configurations.

For the simulations, the geometric and inertial parameters are provided in Table 10.2. The initial conditions for all states are set to zero, except for the initial height of the moving platform, which is set at $z_o = 1\,m$ to start the simulations from a non-zero position. In the first set of simulations, the gravity vector is considered in $-z$ direction, $\boldsymbol{g} = [0, 0, -9.81]^T$, while the disturbance wrench applied to the moving platform is considered to be zero.

The desired trajectory of the moving platform is generated using cubic polynomials for position variables $x_d, y_d, z_d$, as well as for rotations $\theta_x, \theta_y, \theta_z$, as illustrated in Figure 11.28. This figure shows a typical trajectory where the moving platform shifts $0.3\,m$ in the $x$ direction, $0.6\,m$ in the $y$ direction, and $-0.3\,m$ in the $z$ direction. Additionally, the orientation of the moving platform involves compound rotations in the $x$, $y$, and $z$ directions with the values shown in Figure 11.28, representing realistic while complex motion scenarios.



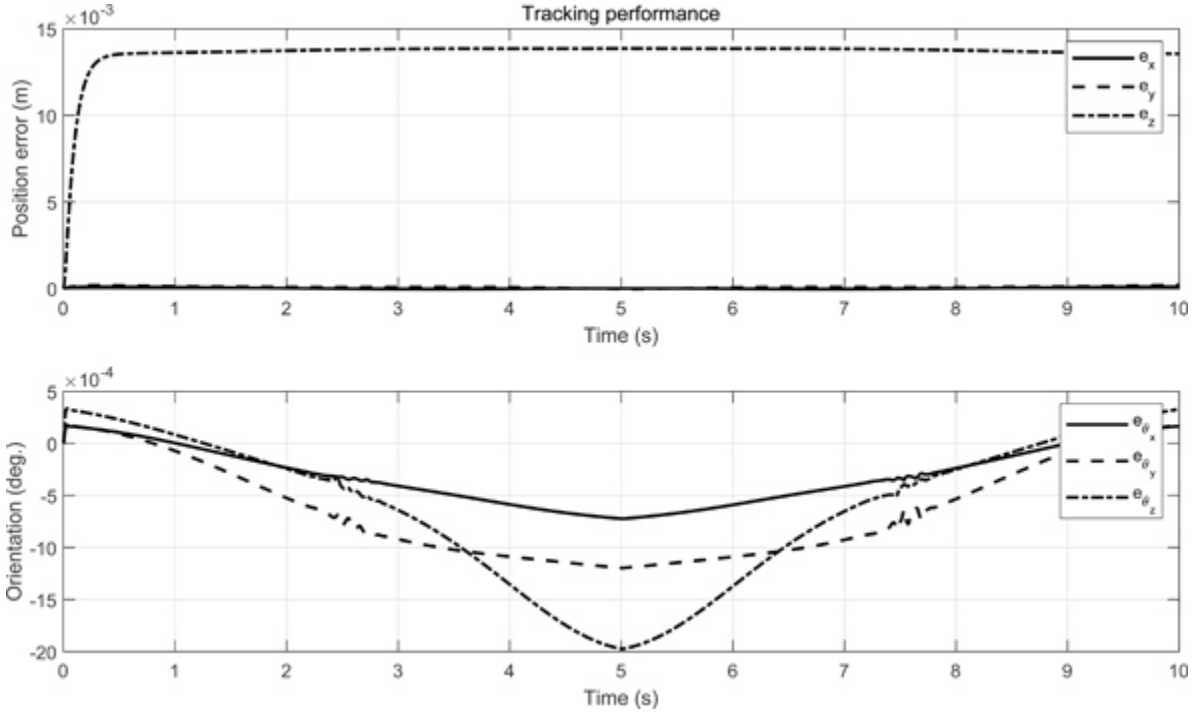

▶ Long Description for Figure 11.28

**FIGURE 11.28**
The closed-loop tracking performance of the Stewart–Gough platform for a cubic trajectory under decentralized PD control with gravity. ⏎

The simulation time is set to $10\,(s)$ to evaluate relatively high accelerative motions for the moving platform. The PD gains used in the simulations are set to $\boldsymbol{K}_p = 10^6 \cdot \mathrm{diag}[1,\, 1,\, 1,\, 10,\, 10,\, 10]$ and $\boldsymbol{K}_d = 10^5 \cdot \boldsymbol{I}_{6\times 6}$ through extensive trial and error to achieve optimal control performance, balancing precision and control effort.

The tracking performance of the moving platform is depicted in Figures 11.28 and 11.29. As clearly seen in Figure 11.28, in the presence of gravity forces, the desired and final closed-loop motions of the system are

indistinguishable in all directions except for $z$, in which a minor error is visible. As illustrated in Figure 11.29, the decentralized PD controller for the manipulator can reduce tracking errors to less than $15\,mm$ in position and less than 0.002° in orientation. These results highlight the controller's precision and robustness, confirming its capability to maintain tight control over the platform's movements even in high-acceleration scenarios. Note that, the precision in $z$ direction is less than that in other directions, due to the effect of gravitational forces.

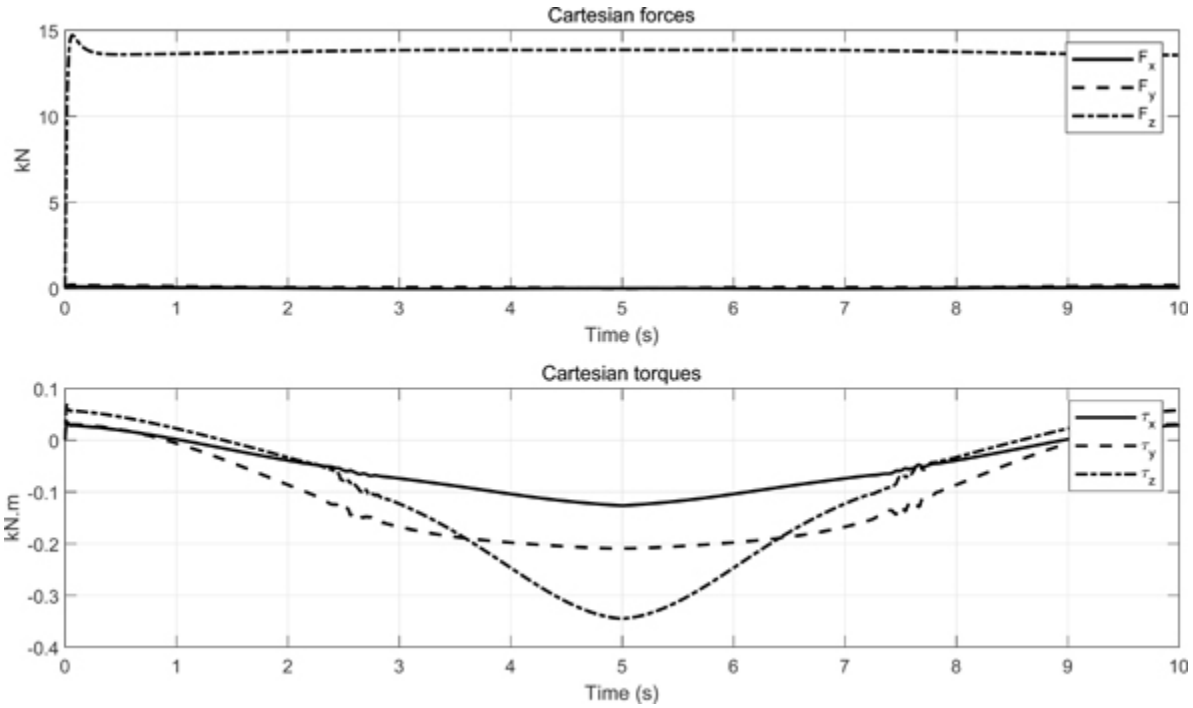



▶ Long Description for Figure 11.29

**FIGURE 11.29**

The closed-loop tracking performance of the Stewart–Gough platform under decentralized PD control with gravitational force. ⏎

The Cartesian wrench $\mathcal{F} = [F_x, F_y, F_z, \tau_x, \tau_y, \tau_z]^T$ generated by the PD control action at the moving platform is depicted in Figure 11.30. This figure highlights how the Cartesian wrench aligns closely with the desired trajectory acceleration, which is linear due to the cubic polynomial trajectory. The force components of the Cartesian wrench reach substantial levels, peaking at approximately $15\,kN$ in $z$ direction while the torque components reach around $0.4\,kNm$. The high value of required force in $z$ direction is to compensate the weight of the moving platform during the maneuver, while in other directions the lower values reflect the PD controller's capability to handle the significant accelerations required by the trajectory.

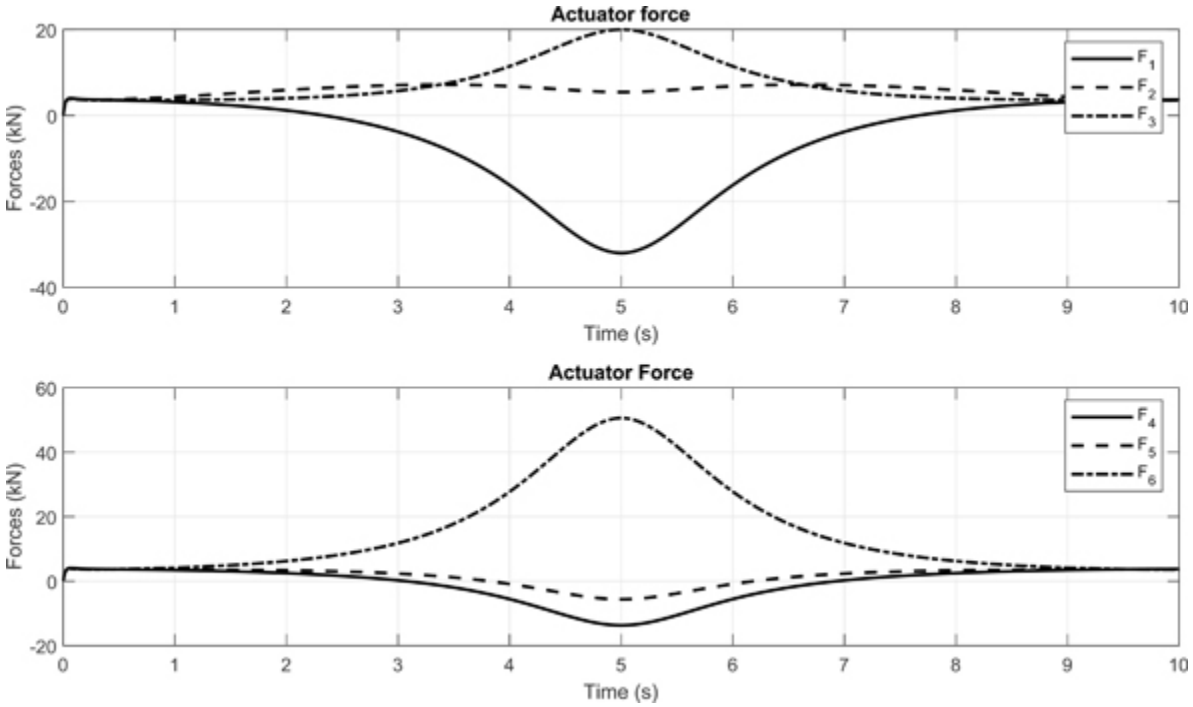



▶ Long Description for Figure 11.30

**FIGURE 11.30**

The Cartesian wrench of the Stewart–Gough platform generated by the PD controller with gravitational force to produce the trajectories shown in Figure 11.28. ⏎

Furthermore, Figure 11.31 shows the actuator forces required to execute the trajectory. It is observed that similarly high actuator forces, in order of a few kN, are necessary to achieve the desired movement. This correlation between high Cartesian wrench values and actuator forces underscores the substantial physical effort and power demand required from the actuators to perform the complex maneuvers dictated by the cubic polynomial trajectory. The results illustrate the effectiveness of the PD controller in managing high accelerations but also highlight the substantial demands placed on the actuators.

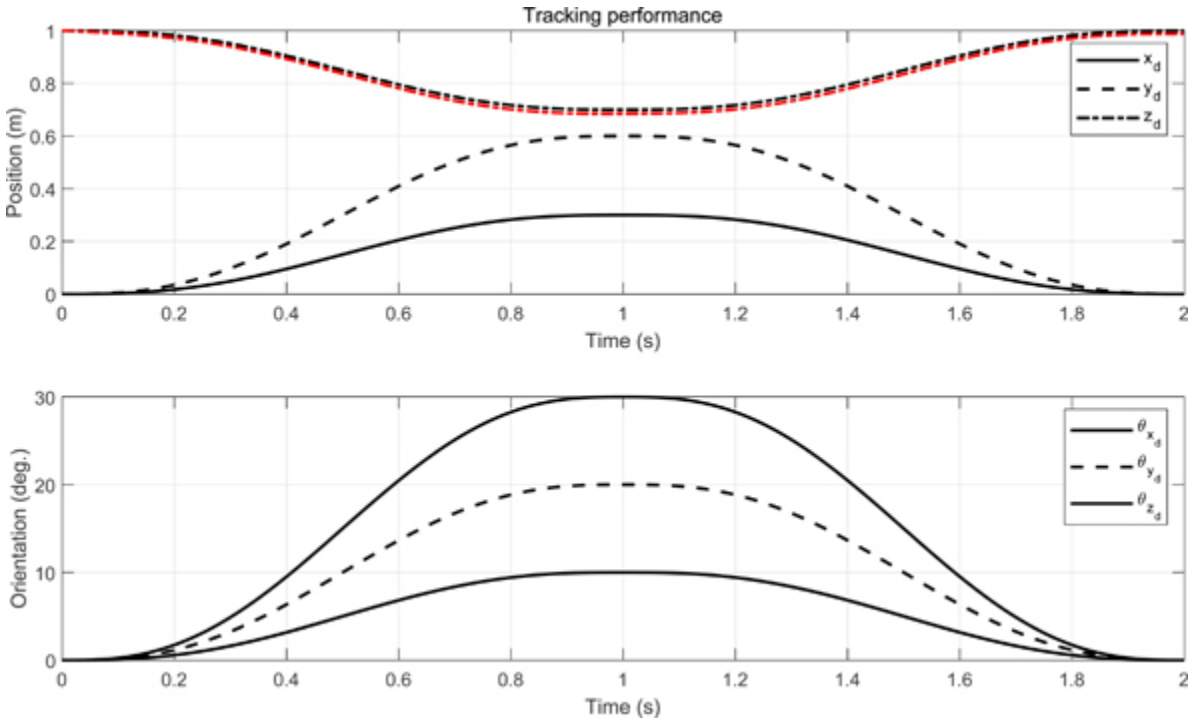



▶ Long Description for Figure 11.31

**FIGURE 11.31**
The actuator forces of the Stewart–Gough platform required to generate the trajectory shown in Figure 11.28 with gravitational force. ⏎

## Simulations on Different Trajectories

In the second simulation, a faster maneuver and a different trajectory planning are considered. For this purpose, a quintic desired trajectory is examined for the same maneuvers as shown in Figure 11.32, which must be completed in just 2 seconds. This scenario investigates the effect of using higher-order trajectory planning in a more rapid motion, which is five times faster than the previous simulation. All other simulation parameters remain unchanged, and no disturbance or noise is included in the simulation, while the PD gains are set to the tuned values of $\boldsymbol{K}_p = 10^6 \cdot \mathrm{diag}[1, 1, 1, 10, 10, 10]$ and $\boldsymbol{K}_d = 10^5 \cdot \boldsymbol{I}_{6\times 6}$ consistent with the first simulation to maintain comparability.

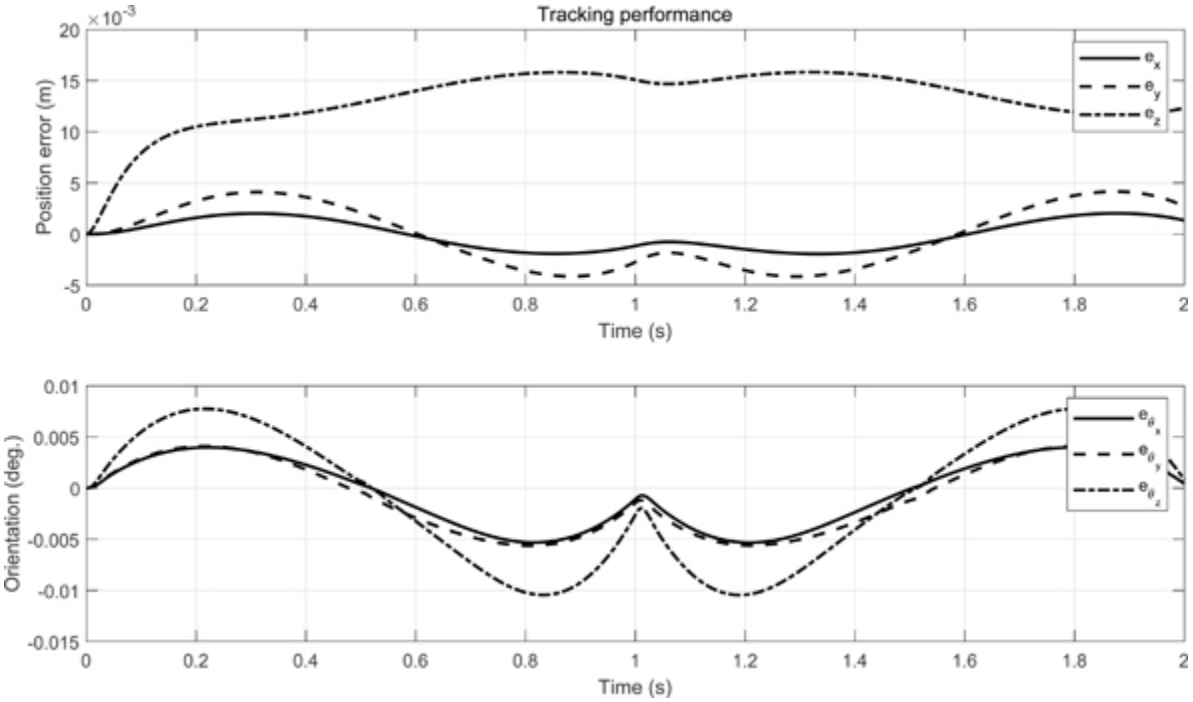



▶ Long Description for Figure 11.32

**FIGURE 11.32**

The closed-loop tracking performance of the Stewart–Gough platform for a high speed quintic trajectory under decentralized PD control with gravity. ⏎

In this scenario, the tracking error of the moving platform is depicted in Figure 11.33. As seen in this figure, although the pattern of the tracking error for this trajectory differs significantly from that of the cubic trajectory, the tracking errors remain about 0.02 meters in position and less than 0.01 degrees in orientation, even for this more rapid maneuver. This performance is quite comparable to that observed in the first simulation of the PD controller. However, it is observed that the tracking error in all the directions is larger than that for the slower trajectory.

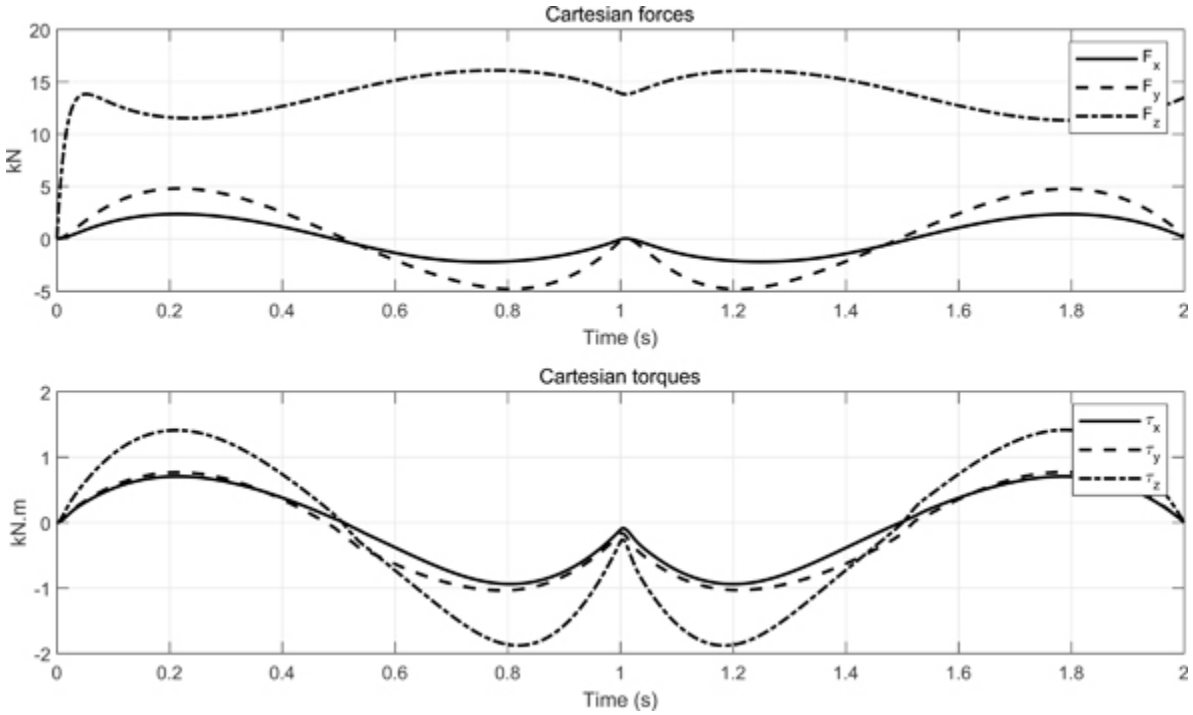



▶ Long Description for Figure 11.33

**FIGURE 11.33**
The closed-loop tracking performance of the Stewart–Gough platform under decentralized PD control for a high-speed quintic trajectory. ⏎

The Cartesian wrench $\boldsymbol{\mathcal{F}} = [F_x, F_y, \tau_\phi]^T$ produced by the PD control action at the moving platform is illustrated in Figure 11.34. These wrenches exhibit a pattern closely aligned with the desired trajectory accelerations, reflecting the controller's responsiveness to changes in motion dynamics. The peak values of the wrenches are almost four times higher than those observed for the slower trajectory, except for $z$ direction. This results in maintaining the overall precision of the robot for faster maneuvers. This observation highlights the effectiveness of PD controllers in

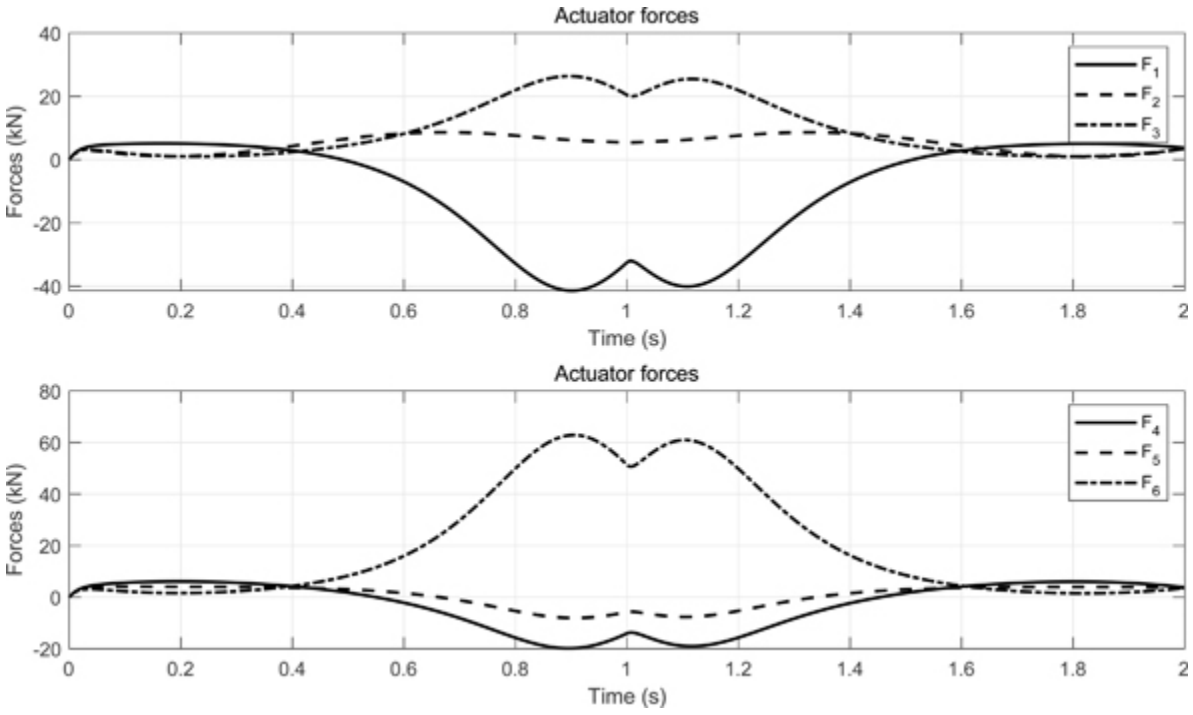
handling a broader range of trajectory demands without requiring further tuning.




▶ Long Description for Figure 11.34

**FIGURE 11.34**

The Cartesian wrench of the Stewart–Gough platform generated by the PD controller with gravitational force to produce high-speed quintic trajectories. ⏎




▶ Long Description for Figure 11.35

**FIGURE 11.35**

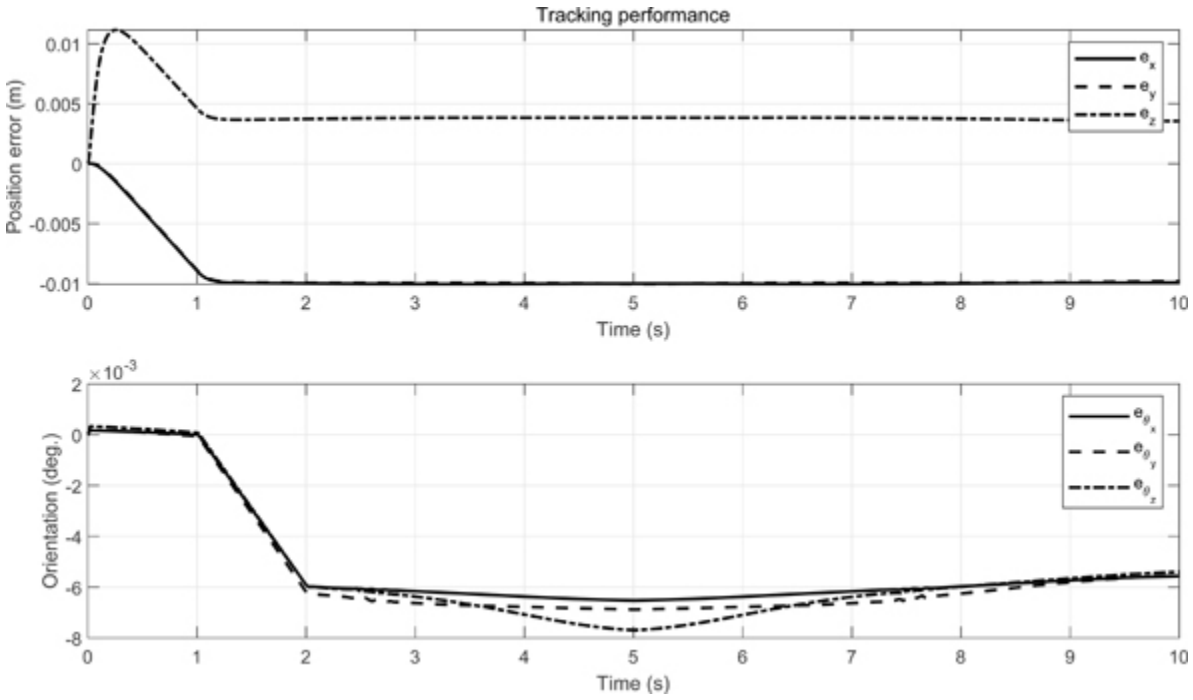
The actuator forces of the the Stewart–Gough platform, as generated by the decentralized PID controller, for tracking high-speed quintic trajectories.

Furthermore, Figure 11.31 shows the actuator forces required to execute the trajectory. It is observed that similarly higher actuator forces are necessary to achieve the desired movement. The results illustrate the effectiveness of the PD controller in managing high accelerations but also highlight the substantial demands placed on both the control system and the actuators.

## Effect of Input Disturbance

To evaluate the PD controller's effectiveness in mitigating the impacts of external disturbances and measurement noise, two additional simulations were performed on the closed-loop system. In the first simulation, a unit step disturbance force is applied while the system follows the original trajectories shown in Figure 11.28. This setup allows for a detailed assessment of how the system responds to disturbances. Specifically, a $10\,kN$ step disturbance force is introduced in the $x$, $y$, and $z$ directions at $1\,s$, followed by a $1\,kN \cdot m$ step disturbance torque in the same directions at 2,$s$. The controller gains are maintained at $\boldsymbol{K}_p = 10^6 \cdot \mathrm{diag}[1,\, 1,\, 1,\, 10,\, 10,\, 10]$ and $\boldsymbol{K}_d = 10^5 \cdot \boldsymbol{I}_{6\times 6}$, as in previous simulations.

The results of the closed-loop tracking error are illustrated in Figure 11.36. Comparing this figure with Figure 11.29 reveals that while the PD controller performs well under ideal conditions, it struggles to completely counteract the effects of external disturbances. The tracking error clearly shows significant spikes at the moments when the disturbance forces and torques are applied. These spikes indicate that the controller cannot fully compensate for sudden changes introduced by disturbances. Additionally, the non-vanishing disturbances cause static errors to accumulate in the system's closed-loop response, which can degrade overall performance.



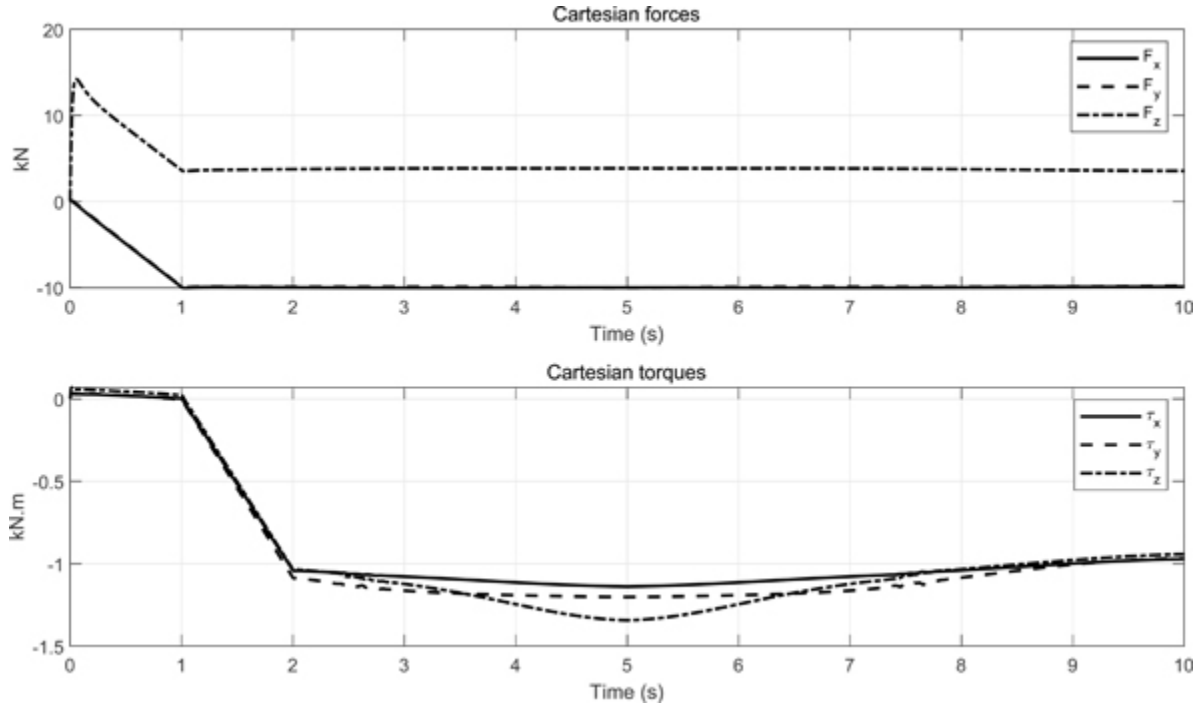

▶ Long Description for Figure 11.36

**FIGURE 11.36**

The closed-loop tracking performance of the Stewart–Gough platform with decentralized PD control under the influence of disturbances. ⏎

Figure 11.37 depicts the Cartesian wrench generated by the PD controller to handle the maneuver amidst disturbances. This figure highlights how the disturbance affects the controller's required forces. Specifically, it shows that the step disturbance wrench at 1 and 2 seconds leads to noticeable adjustments in the forces needed to maintain the trajectory. The illustration underscores the increased demand placed on the controller to maintain accurate motion despite the disruptive forces and torques, demonstrating both the challenges and limitations of the

PD control strategy in real-world conditions where external disturbances are present.



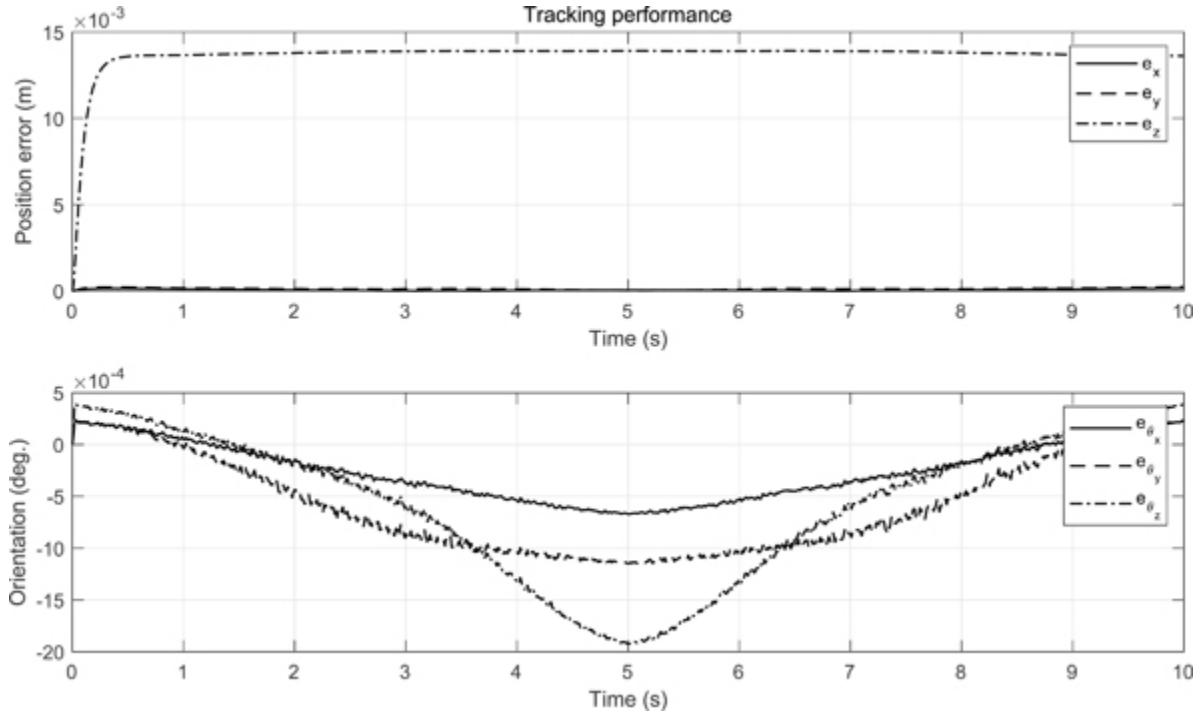

▶ Long Description for Figure 11.37

**FIGURE 11.37**
The Cartesian wrench of the Stewart–Gough platform with decentralized PD control under the influence of disturbances. ⏎

## Effect of Measurement Noise

Next, we explore the impact of measurement noise by introducing Gaussian noise with an amplitude of 0.1% of the peak values of the original signals into all measured variables $\boldsymbol{\chi}$. Although this noise level is quite low, as previously discussed, it has a significant effect on the controller's performance due to the high-gain settings

employed in the system. Figure 11.38 visually depicts the tracking error of the PD controller when this noise is present. The results reveal that the tracking error remains almost identical to the scenario without noise, maintaining overall accuracy, but with some minor oscillations observable in the output, indicating a slight disturbance introduced by the noise.

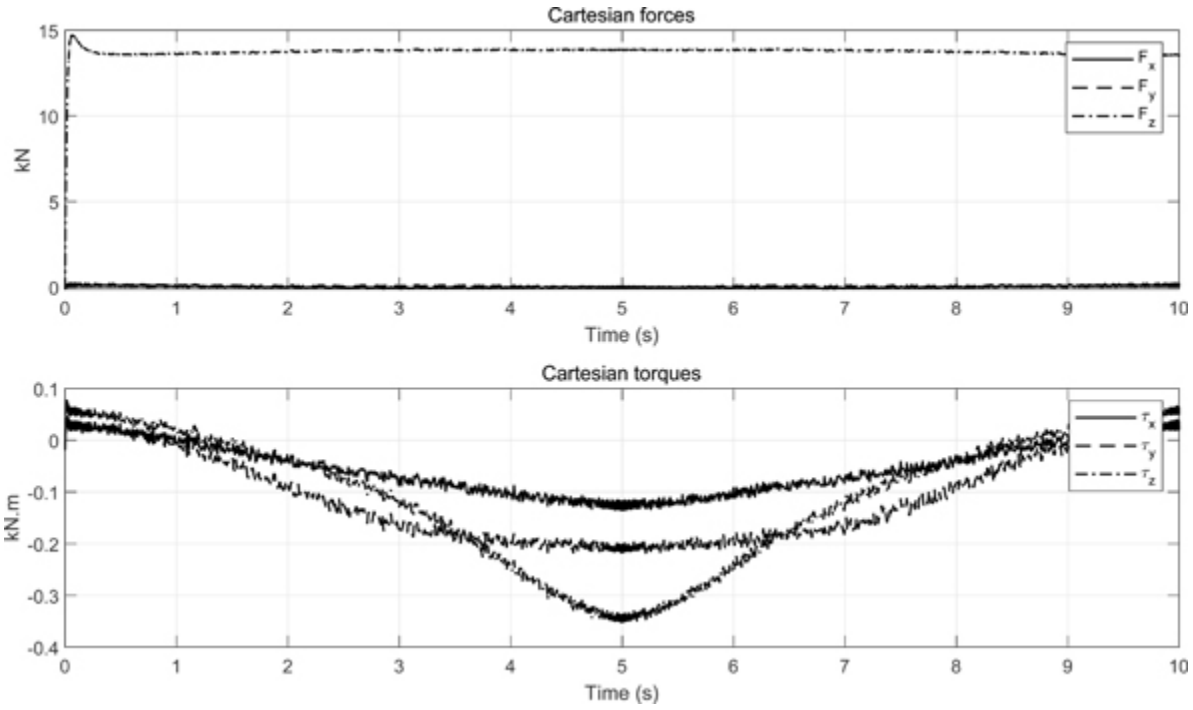



▶ Long Description for Figure 11.38

**FIGURE 11.38**
The closed-loop tracking performance of the Stewart–Gough platform with decentralized PD control under the influence of measurement noise. ⏎

The Cartesian wrenches needed to achieve this level of precision exhibit significant oscillations, as shown in Figure

11.39. Generating these oscillatory forces can be difficult in real-world applications because actuators often have limited bandwidth, restricting their ability to deliver smooth and consistent control actions. These oscillations can pose challenges in maintaining reliable and accurate tracking performance in practical scenarios.




▶ Long Description for Figure 11.39

**FIGURE 11.39**

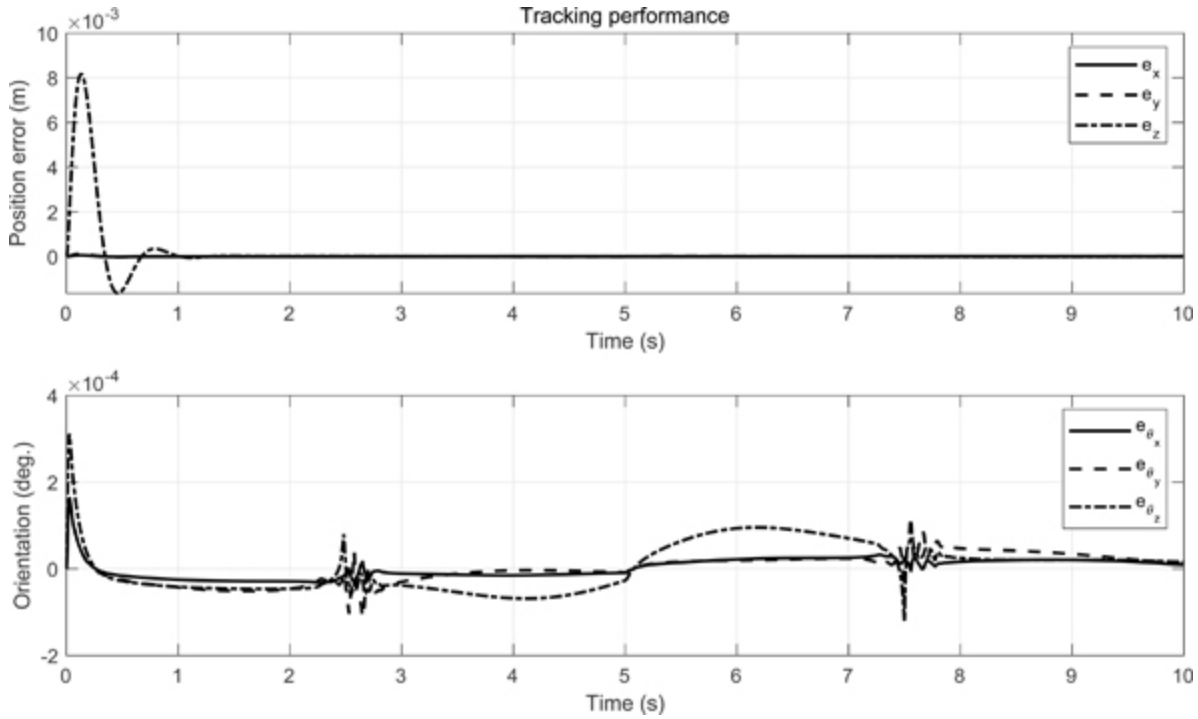
The Cartesian wrench produced by the PD controller for the maneuver depicted in Figure 11.28 while accounting for measurement noise. ⏎

## 11.7.2 Multivariable PID Control

In this section, we examine the tracking performance of the desired trajectory in the Stewart-Gough platform using a decentralized implementation of the PID controller. As illustrated in the block diagram in Figure 11.12, the feedback loop utilizes the motion variable $\boldsymbol{\chi}$, and the decentralized PID controller is constructed with proportional matrix $\boldsymbol{K}_p$, derivative matrix $\boldsymbol{K}_d$, and integral matrix $\boldsymbol{K}_i$. For this simulation, diagonal matrices are used in the design of the PID controller.

In the simulations, all the simulation parameters are used as before. A cubic trajectory is examined first for a period of 10 seconds, while disturbance and measurement noise are neglected. The PID controller gains is set to $\boldsymbol{K}_p = 10^6 \cdot \text{diag}[1, 1, 1, 10, 10, 10], \boldsymbol{K}_d = 10^5 \cdot \boldsymbol{I}_{6\times 6}$, based on the pre-assigned controller gains of the PD controller, while the integral gain is set to $\boldsymbol{K}_i = 10 \cdot \boldsymbol{K}_p$.

In this simulation, the tracking error of the moving platform is illustrated in Figure 11.40. The figure highlights the effectiveness of the decentralized PID controller in minimizing tracking errors, initially reducing the positional error to approximately 8 millimeters and the orientation error to less than $4 \times 10^{-4}$ degrees.




▶ Long Description for Figure 11.40

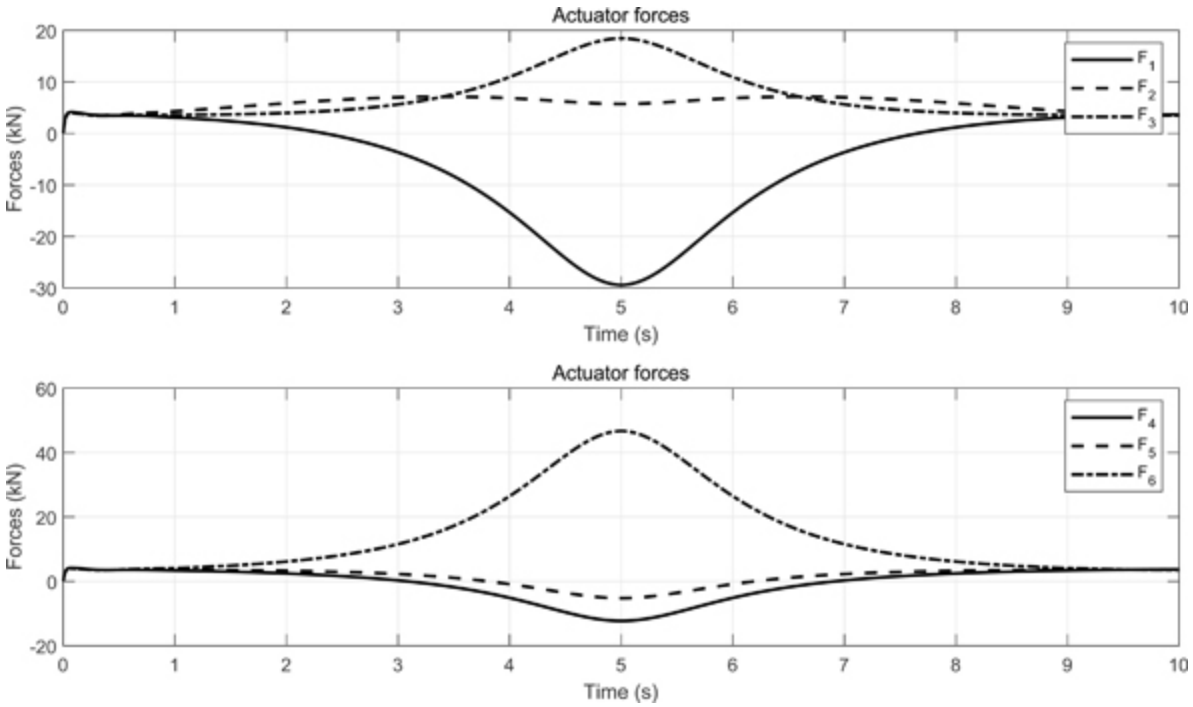
**FIGURE 11.40**

The closed-loop tracking performance of the Stewart-Gough platform under decentralized PID control to track the cubic trajectories shown in Figure 11.28. ⏎

However, the real strength of the PID controller becomes apparent in the steady-state tracking error. Due to the integral term, which compensates for accumulated errors over time, the steady-state error is further minimized to about $4 \times 10^{-5}$ for both translational and rotational motions. This significant reduction underscores the PID controller's advantage in ensuring long-term accuracy and stability by effectively mitigating error accumulation.

Figure 11.41 presents the actuator forces of the manipulator. The output actuator forces are very close to

those in the case of PD controllers. This similarity indicates that the PID controller is capable of providing high precision, without the need of excessive actuator efforts.




▶ Long Description for Figure 11.41

**FIGURE 11.41**

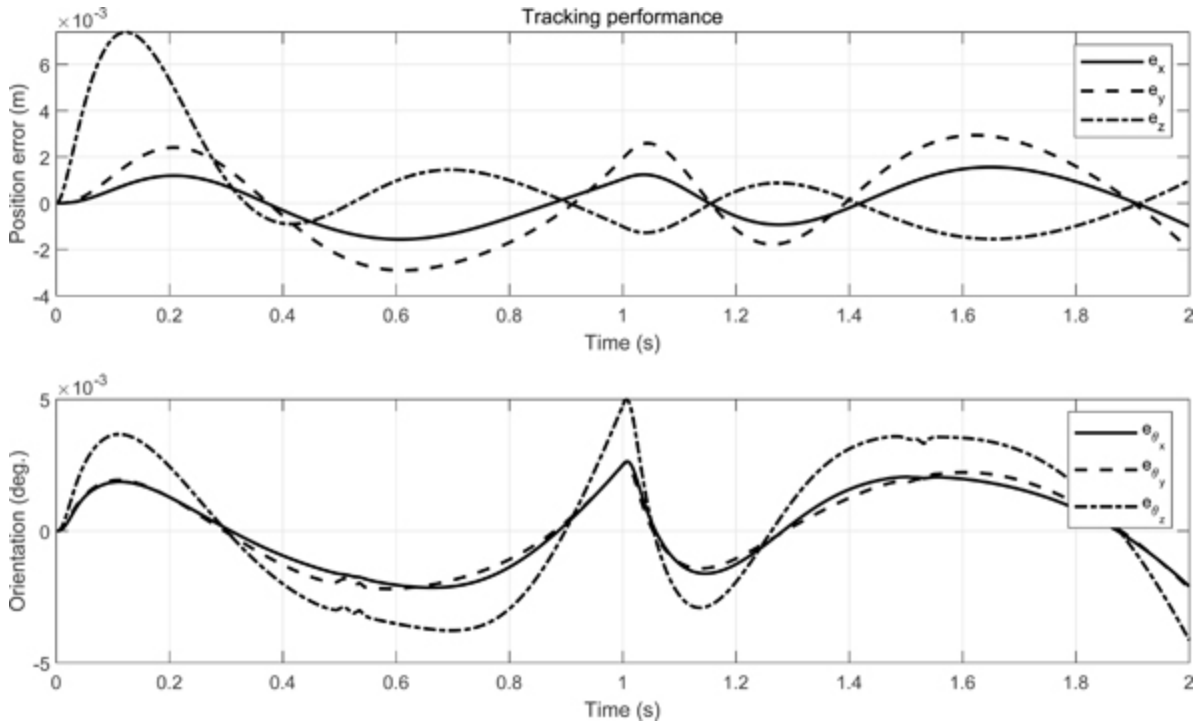
The actuator forces of the Stewart-Gough platform generated by decentralized PID controller to track the cubic trajectories shown in Figure 11.28. ⏎

## Simulations on Different Trajectories

In the second simulation, a high-speed quintic desired trajectory is examined for the same maneuvers, which must be completed in just 2 seconds. All other simulation parameters remain unchanged, and no disturbance or noise is included in the simulation, while the PID gains are set to

the tuned values consistent with the first simulation to maintain comparability.

In this scenario, the tracking error of the moving platform is depicted in the top chart of Figure 11.42. As seen in this figure, although the pattern of the tracking error for this trajectory differs significantly from that of the cubic trajectory, the tracking errors remain below 0.007 meters in position and less than 0.005 degrees in orientation, even for this more rapid maneuver. This performance is quite comparable to that observed in the first simulation of the PID controller, indicating that the controller can handle different types of trajectories with similar accuracy. However, it is observed that the steady-state tracking error here is larger than that for the slower trajectory.

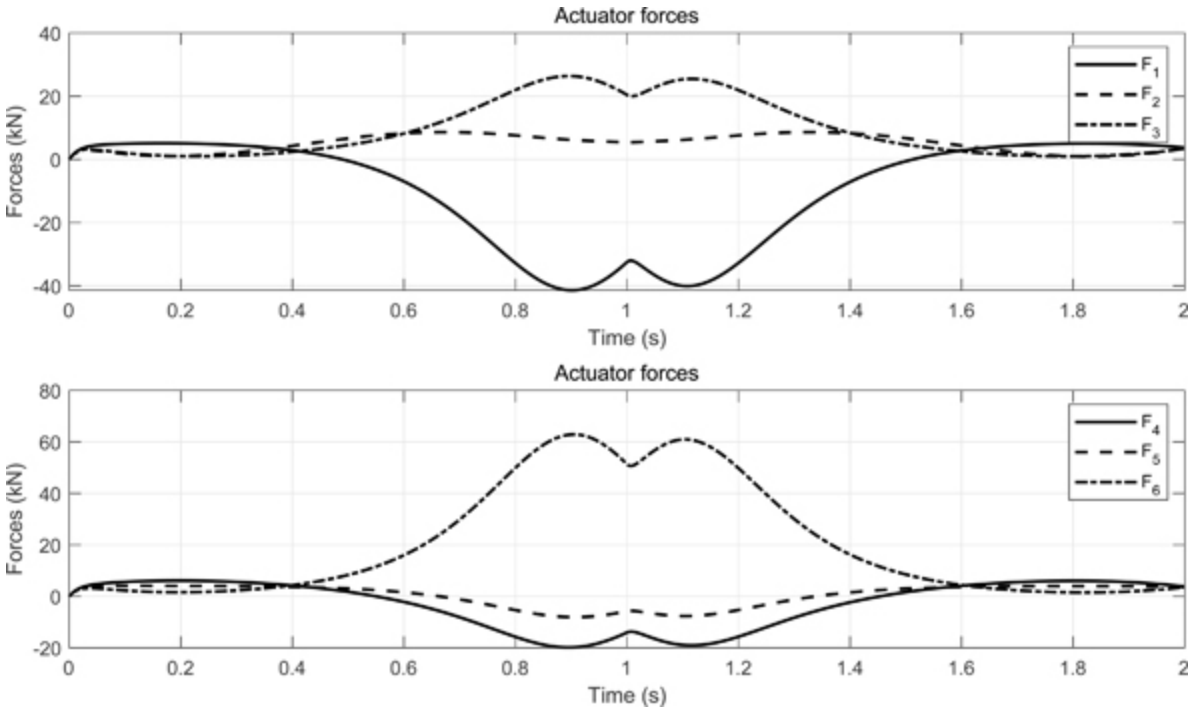


▶ Long Description for Figure 11.42

**FIGURE 11.42**

The closed-loop tracking performance of the Stewart-Gough platform under decentralized PID control to track high-speed quintic trajectories. ⏎

The actuator forces produced by the PID control action is illustrated in Figure 11.43. These wrenches exhibit a pattern closely aligned with the desired trajectory accelerations, reflecting the controller's responsiveness to changes in motion dynamics. The peak values of the actuators are almost the same as those observed for the slower trajectory, maintaining the overall precision of the robot for faster maneuvers. This observation highlights the effectiveness of PID controllers in handling a broader range of trajectory demands without requiring further tuning.




▶ Long Description for Figure 11.43

**FIGURE 11.43**

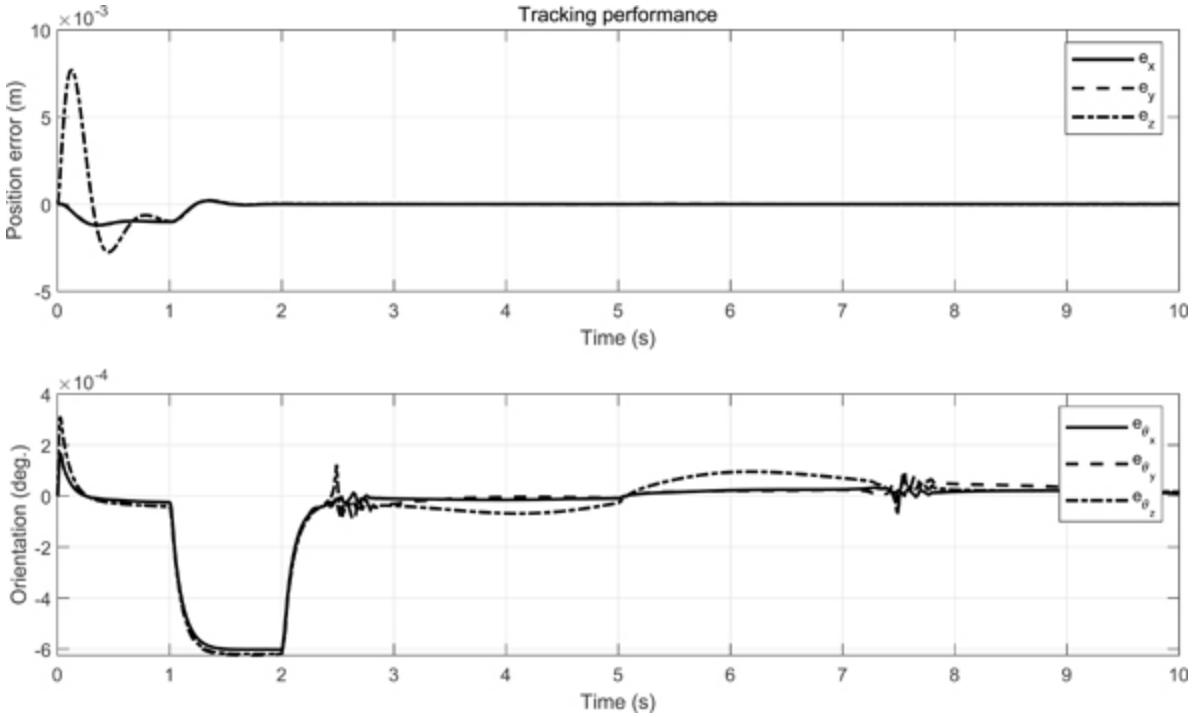
The actuator forces of the Stewart-Gough platform generated by decentralized PID controller, for tracking high-speed quintic trajectories. ⏎

## Effect of Input Disturbance

To illustrate the PID controller's significance in mitigating external disturbances and measurement noise, two additional simulations are conducted for the closed-loop system. The first simulation involves applying a unit step disturbance force while tracking the cubic trajectory shown in Figure 11.28. As for the PD controller case, a series of specific disturbances are applied at two different time instances, namely at one and two seconds. All other

simulation parameters including the controller gains are kept consistent with previous settings.

The closed-loop tracking error for this scenario is illustrated in Figure 11.44 for the cubic polynomial trajectory depicted in Figure 11.28. When compared to the PD controller output shown in Figure 11.36, it becomes evident that the PID controller significantly suppresses the influence of external disturbances. The tracking error exhibits distinct spikes corresponding to the moments when the disturbance wrenches are applied, while the effect of disturbance is significantly mitigated as time evolves. This indicates that the PID controller can significantly reject the effect of disturbances on the performance of the robot. Additionally, since step disturbances are used, it is typical to have an accumulation of static errors within the closed-loop system, while the integral term of the controller, has significantly reduced this effect.



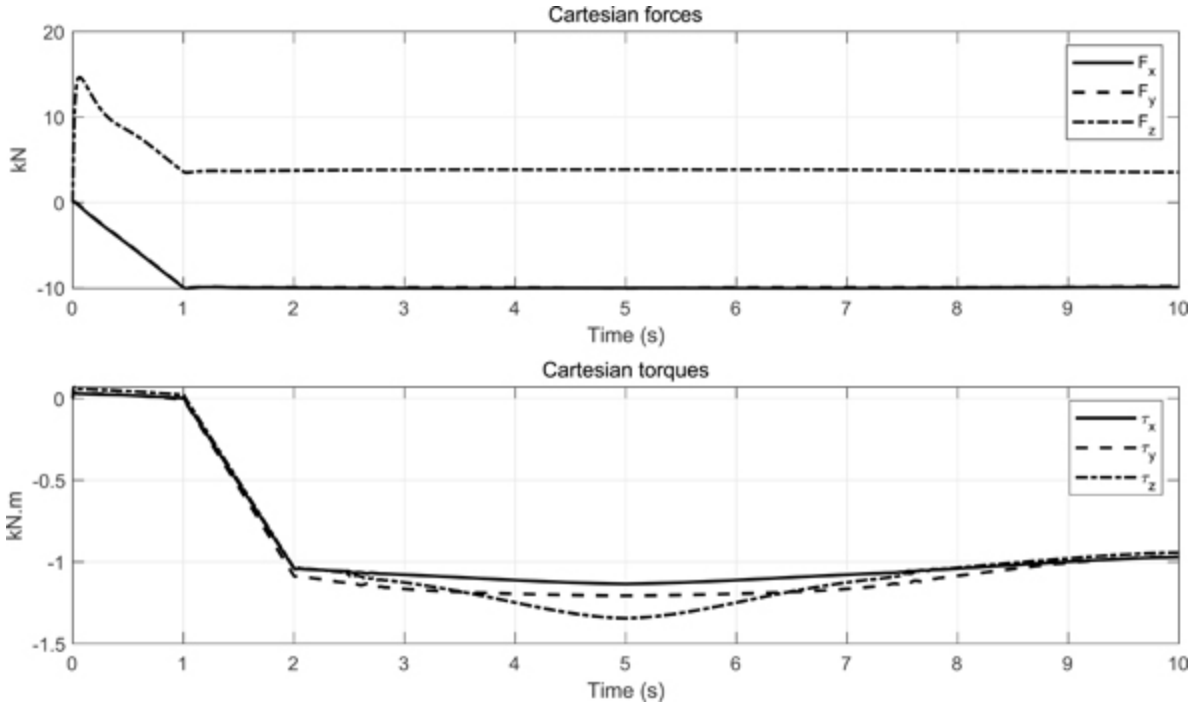

▶ Long Description for Figure 11.44

**FIGURE 11.44**

The closed-loop tracking performance and the generated Cartesian wrenchof the Stewart-Gough platform using decentralized PID control for a cubic trajectory in the presence of disturbance. ⏎

Figure 11.44 depicts the Cartesian wrench produced by the PID controller while performing the maneuver among disturbances. This figure clearly shows how the disturbances affect the required control wrenches at 1, and 2 seconds, when the disturbances are applied. The wrench components vary significantly at these points, reflecting the increased force demands necessary to counteract the disturbances. This effect underscores the limitations of the PID controller in addressing non-vanishing external forces

and the resultant adjustments needed in the control forces to maintain the desired performance.



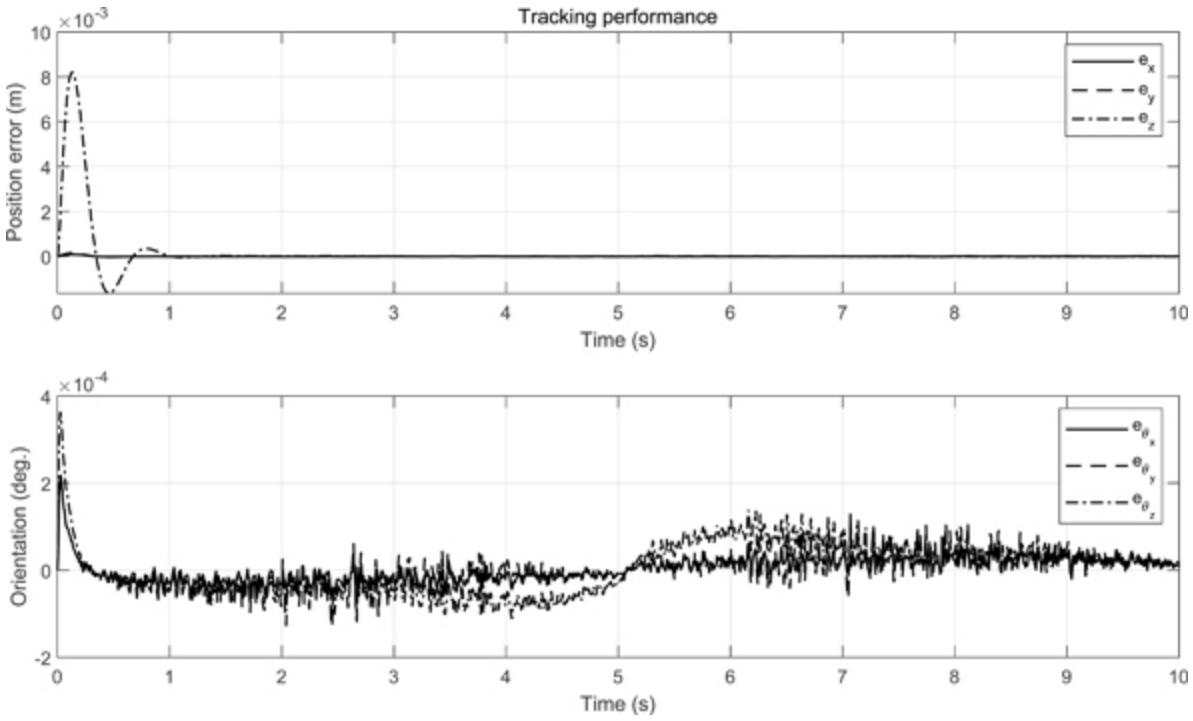

▶ Long Description for Figure 11.45

**FIGURE 11.45**
The Cartesian wrench produced by the PID controller for the maneuver depicted in Figure 11.28 while accounting for disturbances.

## Effect of Measurement Noise

Next, we investigate the impact of measurement noise by introducing Gaussian noise with an amplitude of 0.1% of the peak values of the original signals into all measured variables $\boldsymbol{\chi}$. Figures 11.46 illustrate the tracking error of the PID controller in the presence of this noise. The results show that the tracking error remains within $10^{-5}$ meters,

for positional error, and $2 \times 10^{-4}$ for orientation, except for the initial moments. This remarkable improvement in tracking accuracy amidst the noise is largely attributed to the integral term of the PID controller, which effectively counteracts the deviations caused by the noise.

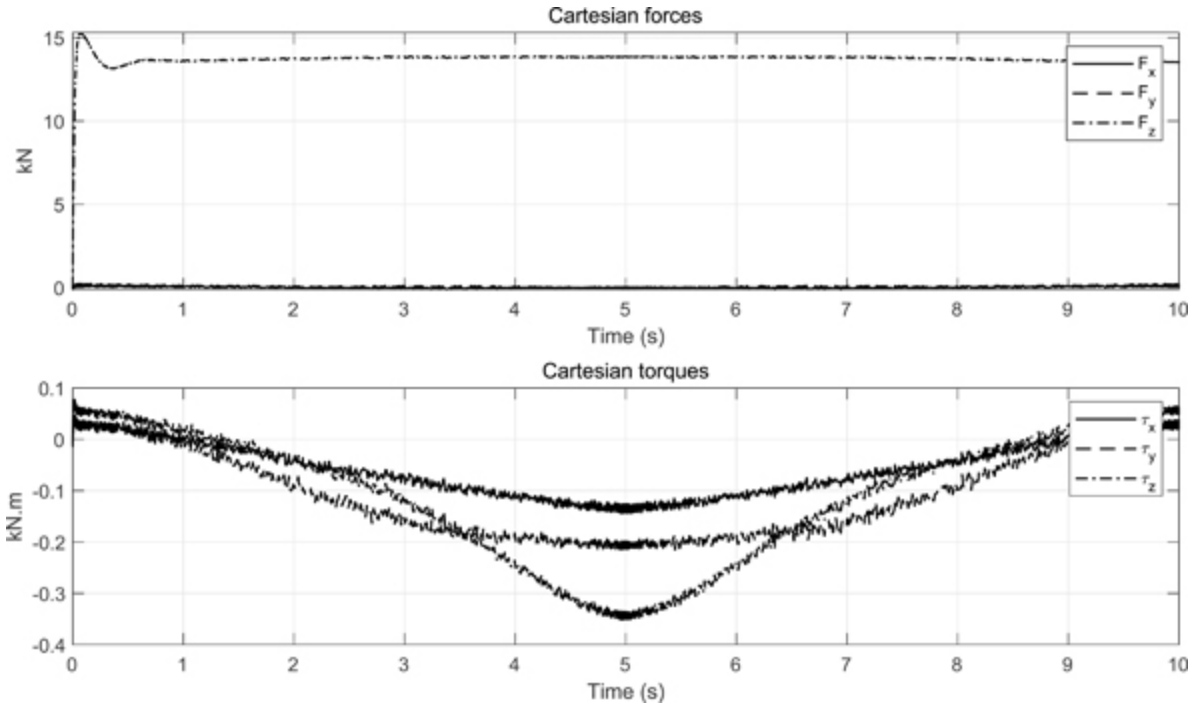



▶ Long Description for Figure 11.46

**FIGURE 11.46**
The closed-loop tracking performance of the Stewart–Gough platform with decentralized PID control under the influence of measurement noise. ⏎

The Cartesian wrenches required to achieve this level of precision, shown in Figure 11.47, are very similar to those seen with the PD controller, as depicted in Figure 11.30, which are highly oscillatory. The generation of such

oscillatory forces can be challenging in real-world applications due to the limited bandwidth of actuators. This can constrain their ability to produce consistent and smooth control actions. These oscillations may lead to potential difficulties in maintaining reliable and accurate tracking performance in practice.




▶ Long Description for Figure 11.47

**FIGURE 11.47**

The Cartesian wrench produced by the PID controller for the maneuver depicted in Figure 11.28 while accounting for measurement noise. ⏎

Therefore, to address these challenges, effective noise management strategies and the use of appropriate filters are crucial. Implementing these measures can help

mitigate the effects of noise and enhance the practical application of the controller, ensuring better performance and stability in environments where measurement noise is a factor.

## Problems

11.1 Consider the five-bar mechanism depicted in Figure 4.27 and discussed in Problems 4.1 and 5.1. Take into account the dynamic formulation along with the geometric and inertial parameters, as well as the typical trajectory outlined in Problem 10.1.

a. For this mechanism, design a decentralized PD controller in the task space and implement the necessary force distribution using the mechanism's Jacobian matrix. Create a program in MATLAB® or Python to simulate the closed-loop tracking performance of the mechanism. Identify appropriate controller gains to achieve effective tracking performance while minimizing control effort.

b. Enhance the performance by implementing a decentralized PID controller in the task space and carry out the required force distribution using the mechanism's Jacobian matrix. Develop a program in MATLAB or Python to simulate the closed-loop tracking performance. Select suitable controller

gains to achieve optimal tracking performance with minimal control effort.

c. Generate and analyze plots of motion trajectories, tracking errors, and the corresponding Cartesian and actuator forces over time for the two controllers. Compare and evaluate the results.

11.2 For the five-bar mechanism depicted in Figure 4.28 and discussed in Problems 4.2 and 5.2, consider the dynamic formulation that includes the geometric and inertial parameters, as well as the typical trajectory from Problem 10.2. Apply the approach outlined in Problem 11.1 to this mechanism.

11.3 Consider the planar pantograph mechanism illustrated in Figure 4.29 and described in Problems 4.3 and 5.3. Utilize the dynamic formulation, including the geometric and inertial parameters, as well as the typical trajectory provided in Problem 10.3. Implement the approach detailed in Problem 11.1 for this mechanism.

11.4 Examine the planar three-degrees-of-freedom 3RR Rparallel manipulator illustrated in Figure 4.30 and discussed in Problems 4.4 and 5.4. Use the dynamic formulation that includes the geometric and inertial parameters, along with the typical trajectory from Problem 10.4. Apply the control methods described in Problem 11.1 to this manipulator.

11.5 Review the planar three-degrees-of-freedom 4RR Rparallel manipulator shown in Figure 4.31 and discussed in Problems 4.5 and 5.4. Apply the dynamic formulation, incorporating the geometric and inertial parameters along with the typical trajectory specified in Problem 10.5. Implement the control techniques described in Problem 11.1 for this manipulator.

11.6 Examine the spatial parallel manipulator depicted in Figure 4.32 and outlined in Problems 4.6 and 5.6. Utilize the dynamic formulation, including the geometric and inertial parameters, as well as the typical trajectory provided in Problem 10.6. Apply the control methods described in Problem 11.1 to this manipulator.

11.7 Analyze the 3UPU parallel manipulator depicted in Figure 4.33 and described in Problems 4.7 and 5.7. Apply the dynamic formulation that includes both the geometric and inertial parameters, along with the typical trajectory outlined in Problem 10.7. Use the control techniques specified in Problem 11.1 to manage this manipulator.

11.8 Review the manipulator detailed in Reference [51] and illustrated in Figure 4.34, as well as in Problems 4.8 and 5.8. Apply the dynamic formulation, including the geometric and inertial parameters and the typical trajectory provided in Problem 10.8. Employ the

control techniques described in Problem 11.1 for this manipulator.

11.9 Examine the three-degrees-of-freedom 3RPS manipulator shown in Figure 4.35 and detailed in Problems 4.9 and 5.9. Apply the dynamic formulation, incorporating the geometric and inertial parameters, along with the typical trajectory from Problem 10.9. Use the control techniques outlined in Problem 11.1 to manage this manipulator.

11.10 Review the three-degrees-of-freedom 3PR Ctranslational parallel manipulator depicted in Figure 4.36 and detailed in Problems 4.10 and 5.10. Utilize the dynamic formulation, incorporating the geometric and inertial parameters, as well as the typical trajectory from Problem 10.10. Apply the control strategies outlined in Problem 11.1 to this manipulator.

11.11 Analyze the Delta robot illustrated in Figure 4.38 and discussed in Problems 4.11 and 5.11. Utilize the dynamic formulation that includes the geometric and inertial parameters, as well as the standard trajectory from Problem 10.11. Implement the control techniques described in Problem 11.1 for this robot.

11.12 Consider the Quadrupteron robot illustrated in Figure 4.39 and outlined in Problems 4.12 and 5.7 [55]. Apply the dynamic formulation, incorporating the geometric and inertial parameters along with the

trajectory specified in Problem 10.12. Implement the control techniques described in Problem 11.1 for this robot.

# 12

# *Multivariable and Nonlinear Control*

DOI: 10.1201/9781003590408-12

This chapter explores a range of advanced control methods for achieving precise motion control in robotic systems. It begins with an introduction while covering multivariable PD control empowered by the feed-forward path. The fundamental controller discussed in this chapter is the inverse dynamics controller (IDC), recognized as a basis for system linearization through feedback and considered a fundamental nonlinear controller in robotics.

Inverse dynamics control calculates and implements a corrective torque on robot actuators based on the system's inverse dynamics, significantly improving linear controllers' performance in the closed-loop system. This approach is fundamental in robot control, forming the basis for many other methods. The discussion progresses to partial feedback linearization and IDC in task space, offering a comprehensive overview of these techniques. A significant

portion of the chapter explores robust and adaptive control, addressing challenges such as model uncertainty in tracking performance. Concepts such as robust inverse dynamics control, chattering mitigation, and adaptive inverse dynamics control are examined in detail.

The chapter concludes with a summary of the various motion control methodologies discussed. To illustrate practical application, two case studies on nonlinear control of a planar cable robot and Stewart-Gough platform are examined, comparing PID control with feed-forward loop, inverse dynamics control, partial feedback linearization, and robust inverse dynamics control. Problems at the end of the chapter encourage readers to apply acquired knowledge, enhancing their understanding and proficiency in implementing these advanced control techniques. Overall, the chapter provides a comprehensive overview of advanced control techniques, serving as a valuable resource for readers interested in nonlinear motion control in robotics.

## 12.1 Introduction

In the previous chapter, the concept of closed-loop control systems for parallel robots and various control topologies was thoroughly explored, laying a strong foundation for understanding the complexities of controlling such systems. This chapter not only revisits these concepts but also dives

deeper into the intricacies of control strategies specifically designed for parallel manipulators.

Although a robotic manipulator with multiple degrees of freedom is typically modeled as a multi-input/multi-output (MIMO) and nonlinear system, many industrial controllers for these robots utilize decentralized linear controllers to manage individual joint movements. This approach simplifies the control problem but does not fully address the complexities of the system's nonlinear dynamics.

However, for parallel manipulators, the closed-kinematic chains within their structure significantly reduce the effectiveness of simple decentralized controllers, leading to performance issues that are not as prevalent in serial robots. As a result, advanced techniques in nonlinear and multivariable control are frequently applied to these manipulators. This chapter elaborates on the foundations of these methods, examining the complexities of designing such controllers through various case studies.

In practice, when decentralized linear controllers on the motion variables do not achieve satisfactory performance in tracking the desired trajectory, nonlinear controllers are employed. In the context of nonlinear controller design, the use of Laplace transforms and linear stability analysis methods, such as Nyquist criteria is not applicable. Instead, control design in the time domain and stability analysis using Lyapunov-based methods become crucial.

The following sections build upon the design of the multivariable PD and PID controllers introduced in the

previous chapter by first incorporating a nonlinear feed-forward term. The primary controller discussed later in this chapter is the inverse dynamics controller, known for forming the foundation for system linearization through feedback and recognized as a key nonlinear controller in robotics [232]. This controller is particularly powerful in its ability to transform the complex, nonlinear dynamics of a manipulator into a set of decoupled, linear systems through feedback, thereby simplifying the control task significantly.

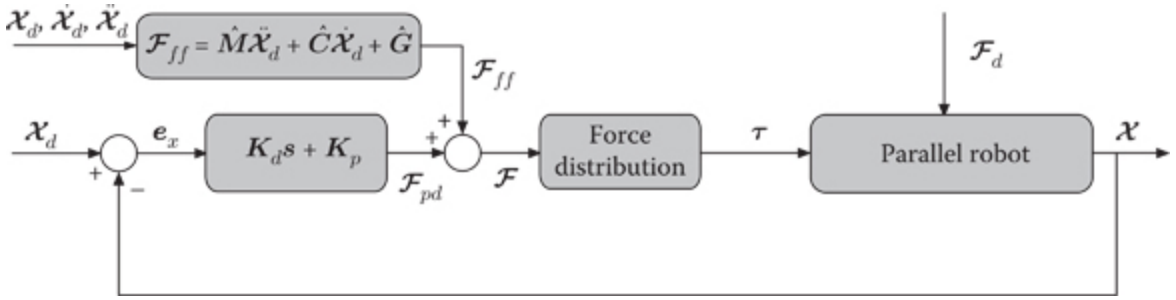
A detailed discussion of the design of robust and adaptive controllers, along with simulation results of these developed methods on two different case studies, will be presented in sections 12.5, 12.7, and 12.8, respectively. These case studies serve to illustrate the practical application of the theoretical concepts discussed, highlighting the challenges and successes encountered in real-world implementations.

## 12.2 Multivariable Control with Feed-Forward

The main approach to enhancing the linear PD and PID controllers involves incorporating a nonlinear feed-forward term specifically designed to minimize steady-state tracking errors. This method requires only limited knowledge of the system's dynamic model. Since the tracking performance of a simple PD controller implemented in the task space can vary across different

configurations, especially when measurement noise and external disturbances affect the manipulator, a feed-forward wrench, denoted as $\mathcal{F}_{ff}$, can be integrated into the controller structure, as illustrated in Figure 12.1.

$$\mathcal{F}_{ff} = \hat{M}(\mathcal{X}_d)\ddot{\mathcal{X}}_d + \hat{C}(\mathcal{X}_d, \dot{\mathcal{X}}_d)\dot{\mathcal{X}}_d + \hat{\mathcal{G}}(\mathcal{X}_d). \tag{12.1}$$




▶ Long Description for Figure 12.1

**FIGURE 12.1**
Feed-forward wrench integrated with the decentralized PD controller in task space. ⏎

As depicted in Figure 12.1, the required inputs for the feed-forward block are the desired task space trajectory $\mathcal{X}_d$ and its derivatives $\dot{\mathcal{X}}_d, \ddot{\mathcal{X}}_d$. This term is referred to as feed-forward because it does not rely on real-time information about the output motion trajectory $\mathcal{X}$ for its generation. Moreover, generating this term necessitates the dynamic formulation of the robot, along with its kinematic and dynamic parameters. In practice, exact knowledge of the

dynamic matrices is typically unavailable, so estimates of these matrices are used instead. Here, $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{\mathcal{C}}}$, and $\hat{\boldsymbol{\mathcal{G}}}$ represent approximate estimates of the manipulator's mass matrix, Christoffel matrix, and gravity vector, respectively.

The information required to compute the feed-forward wrench $\boldsymbol{\mathcal{F}}_{ff}$ is generally available ahead of time. Therefore, the feed-forward term for a specified desired trajectory can be calculated offline, while the decentralized linear controller within the feedback loop is computed in real-time. Although the analysis that follows uses a PD controller, a PID controller could also be employed to further minimize steady-state tracking errors.

In this scenario, with a disturbance wrench $\boldsymbol{\mathcal{F}}_d$ acting on the robot, the closed-loop dynamic equation for the manipulator can be formulated as follows:

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) &= \boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{F}}_d \\ &= \boldsymbol{\mathcal{F}}_{pd} + \boldsymbol{\mathcal{F}}_{ff} - \boldsymbol{\mathcal{F}}_d \\ &= \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x - \boldsymbol{\mathcal{F}}_d + \acute{\boldsymbol{M}} \end{aligned} \tag{12.2}$$

If the knowledge of the system model is accurate enough, and if the values of the matrices and vectors constituting the dynamic model are properly identified through dynamic calibration of the robot, we may assume that the nominal values are equal to the real ones: $\hat{\boldsymbol{M}} = \boldsymbol{M}$, $\hat{\boldsymbol{\mathcal{C}}} = \boldsymbol{\mathcal{C}}$, and

$\hat{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}$. Furthermore, if the performance of the robot controller is satisfactory and the actual motion variables also track the desired motion trajectory with minimal tracking error, it is reasonable to assume that the following approximations are true: $\boldsymbol{\chi}(t) \simeq \boldsymbol{\chi}_d(t)$ and $\dot{\boldsymbol{\chi}}(t) \simeq \dot{\boldsymbol{\chi}}_d(t)$. Under these conditions, the closed-loop dynamic formulation simplifies to:

$$\begin{aligned} &\boldsymbol{M}(\ddot{\boldsymbol{\chi}}_d - \ddot{\boldsymbol{\chi}}) + \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x - \boldsymbol{\mathcal{F}}_d = \mathbf{0}, \\ &\boldsymbol{M}\ddot{\boldsymbol{e}}_x + \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x = \boldsymbol{\mathcal{F}}_d. \end{aligned} \tag{12.3}$$

This equation implies that, provided the previous assumptions are valid, the error dynamics will resemble a set of second-order systems even in the presence of disturbances. If a PID controller is employed, the closed-loop system dynamics will become a third-order system. By appropriately selecting the gains for the PD or PID controller, the transient and steady-state performance of the tracking error can be adjusted to meet specific application needs. This involves utilizing the linear controller techniques discussed in the preceding chapter.

It is crucial to note that if the disturbance wrench on the manipulator is zero or decays to zero over time, the steady-state error in the closed-loop system will also approach zero for all tracking error components. Furthermore, except for the mass matrix, the terms in the error dynamics

are independent of the robot's configuration. This independence simplifies the process of fine-tuning the PD or PID controller gains, facilitating suitable performance throughout the robot's entire workspace.

However, this method encounters several practical limitations. The most significant limitation of this control technique is the stringent assumption that complete knowledge of the dynamic matrices is required. Additionally, notice that the desirable features of this control structure can be achieved only when the desired input tracking is satisfied. As seen before, deriving the dynamic formulations of parallel robots is a very complex and time-consuming task. Dynamic calibration of the robot and determining the model parameters is also a prohibited and time-consuming task.

Under these circumstances, Equation 12.2 demonstrates the dynamic behavior of the tracking error in a more appropriate manner, where nonlinear and multivariable behavior is still present. In this case, considering the error of accurately realizing the dynamic behavior of the system as a significant disturbance vector applied to the system's dynamics, reaching a zero steady-state tracking error by using this structure becomes infeasible. Even if the knowledge of the dynamic model of the system is accurate, achieving precise trajectory tracking through controller design is not easily realized. This introduces another significant error to the dynamics of the tracking error.

Another notable shortcome of this structure is its reliance on the robot's mass matrix, $\boldsymbol{M}(\boldsymbol{\chi})$, resulting in a dependence of the tracking performance on the robot's configuration in the workspace. Consequently, tuning the gains of the PD or PID controller becomes more complicated in practice. To overcome these challenges, the fundamental method of inverse dynamics control has been developed, which will be explored in the following section.

## 12.3 Inverse Dynamics Control

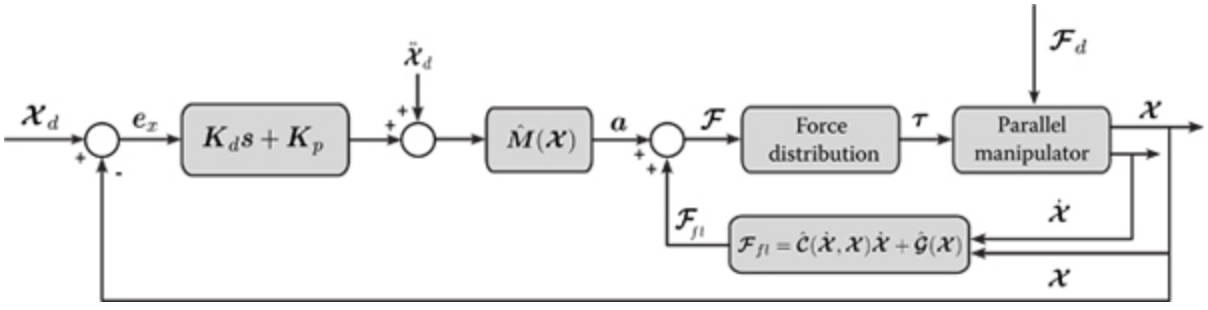
As mentioned earlier, the decentralized PD or PID control structure, even when augmented with a feedforward term, is insufficient for achieving the desired tracking accuracy in the presence of disturbances and uncertainties in the system model. To address these challenges, this section introduces inverse dynamics control, a method specifically designed by researchers to overcome these limitations.

In the robotics literature, this control method is also referred to as the “Computed Torque (CT) method,” especially for serial robot control [232]. On the other hand in control literature, a comprehensive category of approaches, similar to this method, that utilize inverse dynamics to linearize the system dynamics and achieve desirable tracking error in the feedback loop, is termed “Feedback Linearization method” [233, 234]. In this text, we will use the term Inverse Dynamics Control (IDC), a

term frequently used in robotics literature [235, 236, 237], recognizing that both titles accurately describe the method.

IDC works by using the system's inverse dynamics to calculate the wrench required to compensate for the system's dynamics, which is then incorporated as a corrective term into the multivariable PD or PID controller. By applying this computed torque, the multivariable and nonlinear dynamics of the robot are ideally linearized, significantly improving the performance of linear controllers in the closed-loop system. It is important to note that this approach is considered foundational in robot control, with many other techniques having been developed as extensions of IDC.

The general structure of IDC as applied to a parallel manipulator is illustrated in Figure 12.2. In this figure, a corrective wrench $\mathcal{F}_{fl}$ is incorporated into the feedback loop of the closed-loop system. This corrective wrench is calculated using the Christoffel matrix as well as the gravity vector from the manipulator's dynamic formulation. Additionally, the mass matrix is included in the forward path along with the desired trajectory acceleration $\ddot{\mathcal{X}}_d$.




▶ Long Description for Figure 12.2

**FIGURE 12.2**
Overview of the inverse dynamics control configuration applied in task space. ⏎

The outer loop consists of the usual PD or PID controller designed before for the motion control of the robot. Additionally, by imposing the nominal mass matrix of the robot $\hat{\boldsymbol{M}}(\boldsymbol{\chi})$ on the desired motion path, the acceleration vector of the robot's desired motion $\boldsymbol{a}(\boldsymbol{\chi})$ is utilized as another input to the inner loop. It's important to note that in practice, the actual values of the matrices and vectors of the dynamic model of the system are not available, and their nominal values are used in the calculation of the feedback linearization torque $\boldsymbol{\mathcal{F}}_{fl}$. The application of this calculated wrench on the system diminishes the influence of nonlinear dynamic terms of the system, hence this method is renowned as feedback linearization in control literature.

By this means, and using the block diagram shown in Figure 12.2, the wrench applied to the manipulator by the controller can be expressed as follows:

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \boldsymbol{\mathcal{F}}_{fl}, \tag{12.4}$$

where it can be detailed as:

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}),$$

(12.5)

$$\boldsymbol{a} \quad = \ddot{\boldsymbol{\chi}}_d + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x.$$

(12.6)

In the general case of a six-degrees-of-freedom manipulator, $\boldsymbol{a}$ is a $6 \times 1$ vector, representing the motion variable acceleration. We can now formulate the closed-loop dynamics for the manipulator as follows:

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) \quad &= \boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{F}}_d \\ &= \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \\ &= \hat{\boldsymbol{M}}(\boldsymbol{\chi}) \left( \ddot{\boldsymbol{\chi}}_d + \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{K} \right. \\ &\quad + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}) - \boldsymbol{\mathcal{F}} \end{aligned}$$

(12.7)

If the knowledge of the system model is accurate enough and their parameters are properly calibrated, we can assume that $\hat{\boldsymbol{M}} = \boldsymbol{M}, \hat{\boldsymbol{\mathcal{C}}} = \boldsymbol{\mathcal{C}}$, and $\hat{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}$. Under these conditions, the closed-loop dynamic formulation simplifies to:

$$\hat{\boldsymbol{M}}(\boldsymbol{\chi}) \left( \ddot{\boldsymbol{e}}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x \right) = \boldsymbol{\mathcal{F}}_d.$$

(12.8)

This equation indicates that if we use a PD controller in the feedback loop and if there is complete knowledge of the dynamic matrices, the error dynamics behave as a set of second-order systems in the presence of disturbances. Consider a scenario where the manipulator is moving freely in space with no disturbance wrench applied. Furthermore, since the manipulator's mass matrix is positive definite for all configurations, the error dynamics can be simplified to:

$$\ddot{\boldsymbol{e}}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x = \mathbf{0}. \tag{12.9}$$

When a PID controller is used, the closed-loop system's dynamics will become a third-order system. Consequently, by carefully selecting the gains for the PD or PID controller, the tracking error can be minimized to ensure a fast transient response and a low steady-state error for the desired trajectory, by leveraging the linear controller techniques discussed in the previous chapter.

This control method is extensively used in practice because it effectively linearizes and decouples the closed-loop error dynamics. Additionally, since the error dynamics are independent of the robot's configuration, it becomes easier to tune the PD or PID controller gains to maintain consistent performance across the robot's entire workspace.

However, to achieve optimal performance, an accurate system model is required, and the technique is sensitive to

modeling uncertainties. Furthermore, this approach is computationally intensive due to the substantial online calculations needed for closed-loop control. To mitigate these challenges, modifications have been introduced to reduce computational demands while preserving the method's core advantages. These modifications are discussed in Section 12.4 under partial linearization IDC.

Now consider the general scenario where the knowledge of the dynamic matrices is incomplete, meaning $\hat{\boldsymbol{M}} \neq \boldsymbol{M}$, $\hat{\boldsymbol{\mathcal{C}}} \neq \boldsymbol{\mathcal{C}}$, $\hat{\boldsymbol{\mathcal{G}}} \neq \boldsymbol{\mathcal{G}}$, and where an arbitrary disturbance wrench $\boldsymbol{\mathcal{F}}_d \neq \mathbf{0}$ is applied to the moving platform. In this case, the error dynamics can be derived from Equation 12.7 as follows:

$$\ddot{\boldsymbol{e}}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x = \hat{\boldsymbol{M}}^{-1} \Big[ (\boldsymbol{M} - \hat{\boldsymbol{M}}) \ddot{\boldsymbol{\mathcal{X}}} + (\boldsymbol{\mathcal{C}} - \hat{\boldsymbol{\mathcal{C}}}) \dot{\boldsymbol{\mathcal{X}}} + (\boldsymbol{\mathcal{G}} \quad (12.10)$$

The right-hand side of Equation 12.10 is typically non-zero and can be interpreted as an external force influencing the tracking error dynamics. This external force can be effectively countered by using high-gain PD controllers or more advanced control strategies, such as robust or adaptive control methods.

In the robust control approach, modeling uncertainties are systematically analyzed across various configurations to establish an upper bound for the right-hand side of Equation 12.10. Lyapunov-based robust control techniques

are then employed to introduce a corrective term into the PD controller output, ensuring the asymptotic stability of the tracking error even under worst-case scenarios involving modeling uncertainties and external disturbances.

Alternatively, if the estimates of the dynamic formulation matrix can be refined such that their deviation from the true values is minimized, the manipulator's error dynamics can converge to the second-order system described by Equation 12.8. This convergence can be achieved using adaptive control schemes, which are further explained in Section 12.5.

## 12.4 Partial Feedback Linearization

Inverse dynamics control has distinct theoretical advantages that have contributed to its success. However, fully implementing this approach requires determining the robot's dynamic equations and calibrating its parameters through kinematic and dynamic calibration, presenting two significant challenges. First, deriving the robot's dynamic equations in a closed form, particularly the Christoffel matrix, is a complex and time-consuming process, especially for parallel robots with multiple limbs. Additionally, robot calibration necessitates expressing the dynamic equations in a linear regression form and designing suitable experiments to identify these parameters. If the nonlinear dynamics of the robot are

accurately identified, feedback linearization can offer highly efficient system performance. However, the implementation of inverse dynamics control is computationally intensive and time-consuming due to these challenges.

In this section, we introduce a simplified version of inverse dynamics control that significantly reduces computational demands while maintaining key aspects of the closed-loop system, such as stability and zero steady-state tracking error. This method, known in the control community as “partial feedback linearization,” eliminates the need to determine and calibrate the mass and Christoffel matrices. Instead, only the gravity vector is used to linearize a portion of the nonlinear dynamics that directly influence steady-state tracking error, by which this approach can be considered as a weight compensation method using feedback.

Since the gravity vector in the system's dynamic equations is derived solely from the system's potential energy, it is much easier to calculate compared to the mass and Christoffel matrices. Moreover, for calibrating the gravity vector, static experiments and the use of the robot arms’ equilibrium states in various configurations can be employed, thereby reducing the time and cost associated with calibrating gravity-related parameters.

These challenges have led to various extensions of IDC to mitigate the stringent requirements. This section describes the simplest form of partial feedback linearization designed

to reduce computational costs while maintaining the core benefits of the approach. To create the simplest implementable form of Inverse Dynamics Control (IDC), it is helpful to review the complexities involved in deriving the dynamic formulation.

As detailed in Section 8.3, the manipulator's mass matrix $\boldsymbol{M}(\boldsymbol{\chi})$ is obtained from the kinetic energy of the manipulator using Equation 8.19, while the gravity vector $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$ is derived from the potential energy as described in Equation 8.24. Calculating the gravity vector is generally simpler than computing the mass matrix. Additionally, determining the Coriolis and centrifugal matrix $\boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$, defined in Equation 8.46, is significantly more complex than calculating both the gravity vector and the mass matrix.

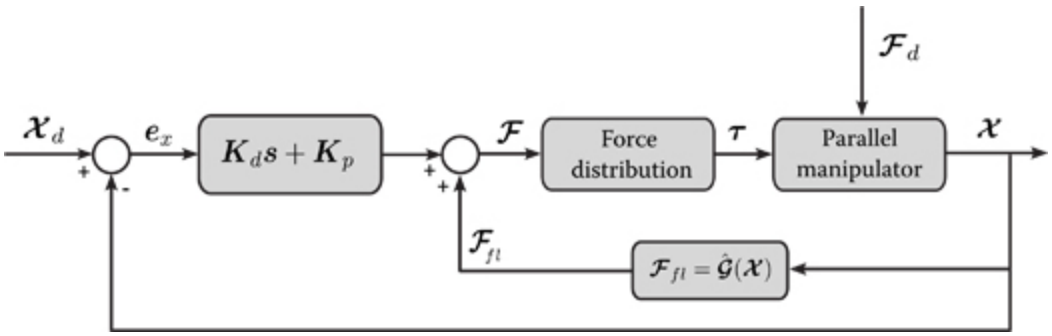
Note that, certain properties of the dynamic matrices $\boldsymbol{M}$, $\boldsymbol{\mathcal{G}}$, and $\boldsymbol{\mathcal{C}}$ can be leveraged in control techniques for parallel manipulators. Notably, the skew-symmetric property of the matrix $\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}$, defined in Equation 8.47, can be utilized effectively. This property can help simplify the IDC approach, making it more feasible for practical implementation.

Consider the dynamic formulation of a parallel robot as given in Equation 11.1, which adheres to the skew-symmetric property of the dynamic matrices. In the simplest version of this method, as shown in Figure 12.3, the input wrench $\boldsymbol{\mathcal{F}}$, is generated solely using the system's

gravity vector in the inner loop and a PD controller in the outer loop.

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}_{pd} + \boldsymbol{\mathcal{F}}_{fl},$$

(12.11)




▶ Long Description for Figure 12.3

**FIGURE 12.3**
Overview of the partial inverse dynamics control configuration applied in task space. ⏎

in which the first term, $\boldsymbol{\mathcal{F}}_{pd}$, is generated by applying a multivariable PD controller to the motion error $\boldsymbol{e}_x$.

$$\boldsymbol{\mathcal{F}}_{pd} = \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x,$$

(12.12)

where $\boldsymbol{e}_x = \boldsymbol{\mathcal{X}}_d - \boldsymbol{\mathcal{X}}$ represents the motion tracking error in task space, and $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are the proportional and derivative gain matrices, respectively. The second term in Equation 12.11 is simplified to include only the nominal

value of the gravity vector, $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$, without incorporating the computationally demanding Christoffel matrix:

$$\boldsymbol{\mathcal{F}}_{fl} = \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}).$$

(12.13)

To eliminate the steady-state tracking error, complete knowledge of the gravity vector, as the only term used in the inner loop linearization with feedback, must be available. In this case, by assuming that:

$$\hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}) = \boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$$

(12.14)

the dynamic behavior of a closed-loop system in the absence of disturbance torque $\boldsymbol{\mathcal{F}}_d$ is expressed as follows.

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} = \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x$$

(12.15)

Observe that the dynamic behavior of the system remains multivariable and nonlinear, with only the effect of the gravity vector removed. Nevertheless, despite the simplicity of this control method, its performance, especially in steady-states, is highly desirable. To validate this claim and analyze the closed-loop stability, we utilize

the Lyapunov stability theorem. A brief review of stability analysis in nonlinear systems is presented in Appendix B.

To examine the tracking error behavior in steady-state, it is necessary for the desired trajectory to converge to a constant value over time; therefore, such analysis can be performed by assuming $\dot{\boldsymbol{\chi}}_d = 0$. For the analysis of the dynamic behavior of the closed-loop system, the total energy of the closed-loop system is considered as the Lyapunov function candidate.

$$V = \frac{1}{2}\dot{\boldsymbol{\chi}}^T \boldsymbol{M}\dot{\boldsymbol{\chi}} + \frac{1}{2}\boldsymbol{e}_x^T \boldsymbol{K}_p \boldsymbol{e}_x. \tag{12.16}$$

What is noticeable in this relation is that the proposed Lyapunov function is composed of two terms: the first term represents the kinetic energy of the system, and the second term represents the potential energy induced by the proportional controller gain, analogous to a mechanical spring storing the system's energy. The proposed Lyapunov function is positive in all points of the workspace, except in points where the traversed trajectory has reached its desired value and has come to a rest, i.e., $\boldsymbol{\chi} = \boldsymbol{\chi}_d \rightarrow \dot{\boldsymbol{\chi}} = 0$. In such case the Lyapunov function is zero, and thus, the necessary positive-definite condition for the Lyapunov function candidate in nonlinear systems are satisfied.

In the Lyapunov theorem, stability analysis and the steady-state behavior of tracking error are carried out by calculating and analyzing the derivative of the Lyapunov function with respect to time. If the derivative of this positive-definite function, is negative-definite, the stability of the closed-loop system is guaranteed, and the tracking error asymptotically converges to zero.

Considering the desired trajectory where $\dot{\boldsymbol{\chi}}_d = 0$, the derivative of the Lyapunov function with respect to time is expressed as follows:

$$\dot{V} = \dot{\boldsymbol{\chi}}^T \boldsymbol{M} \ddot{\boldsymbol{\chi}} + \frac{1}{2} \dot{\boldsymbol{\chi}}^T \dot{\boldsymbol{M}} \dot{\boldsymbol{\chi}} + \dot{\boldsymbol{e}}_x^T \boldsymbol{K}_p \boldsymbol{e}_x. \tag{12.17}$$

At this stage of stability analysis, the dynamic equations of the closed-loop system come into picture. By substituting $\boldsymbol{M}\ddot{\boldsymbol{\chi}}$ from the closed-loop dynamic formulation Equation 11.1 into Equation 12.17, and noting that for a constant desired trajectory $\dot{\boldsymbol{\chi}}_d = \mathbf{0}$, the time derivative of the Lyapunov function simplifies to:

$$\begin{aligned} \dot{V} &= \dot{\boldsymbol{\chi}}^T \left( \boldsymbol{\mathcal{F}}_{pd} + \boldsymbol{\mathcal{F}}_{fl} - \boldsymbol{\mathcal{C}} \dot{\boldsymbol{\chi}} - \boldsymbol{\mathcal{G}} \right) + \frac{1}{2} \dot{\boldsymbol{\chi}}^T \dot{\boldsymbol{M}} \dot{\boldsymbol{\chi}} + \dot{\boldsymbol{e}}_x^T \boldsymbol{K}_p \boldsymbol{e}_x \\ &= \dot{\boldsymbol{\chi}}^T \left( \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x - \boldsymbol{\mathcal{C}} \dot{\boldsymbol{\chi}} \right) + \frac{1}{2} \dot{\boldsymbol{\chi}}^T \dot{\boldsymbol{M}} \dot{\boldsymbol{\chi}} + \dot{\boldsymbol{e}}_x^T \boldsymbol{K}_p \boldsymbol{e}_x. \end{aligned}$$

To simplify this equation, observe that for a constant desired trajectory, $\dot{\boldsymbol{\chi}}_d = \mathbf{0}$, so $\dot{\boldsymbol{e}}_x = \dot{\boldsymbol{\chi}}_d - \dot{\boldsymbol{\chi}} = -\dot{\boldsymbol{\chi}}$. Thus, we have:

$$\begin{aligned}\dot{V} &= \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_p \boldsymbol{e}_x - \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_d \dot{\boldsymbol{\chi}} + \frac{1}{2}\dot{\boldsymbol{\chi}}^T \left(\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}\right)\dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_p \boldsymbol{e} \\ &= -\dot{\boldsymbol{\chi}}^T \boldsymbol{K}_d \dot{\boldsymbol{\chi}} + \frac{1}{2}\dot{\boldsymbol{\chi}}^T \left(\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}\right)\dot{\boldsymbol{\chi}} \\ &= -\dot{\boldsymbol{\chi}}^T \boldsymbol{K}_d \dot{\boldsymbol{\chi}} \leq 0,\end{aligned} \tag{12.18}$$

In the final step, the skew-symmetric property of the dynamic matrices is utilized to simplify the equation. This result shows that the derivative of the Lyapunov function is negative semi-definite. Utilizing the Krasovski-Lasalle's Theorem, the Lyapunov function's negative semi-definite derivative can be used to prove the asymptotic stability of the system and the convergence of the tracking error towards zero.

To prove asymptotic convergence, note that $V$ is always decreasing as long as $\dot{\boldsymbol{\chi}}$ is not zero. Consequently, the manipulator may reach a configuration where $\dot{\boldsymbol{\chi}} = 0$, but $\boldsymbol{\chi} \neq \boldsymbol{\chi}_d$. To demonstrate that this situation cannot occur, observe that at such a configuration $\dot{V} = 0$. Therefore, Equation 12.17 implies that $\dot{\boldsymbol{\chi}} \equiv \mathbf{0}$, which in turn leads to $\ddot{\boldsymbol{\chi}} \equiv \mathbf{0}$. In this case, the closed-loop dynamic formulation:

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} = \boldsymbol{K}_d\dot{\boldsymbol{\chi}} + \boldsymbol{K}_p\boldsymbol{e}_x$$

(12.19)

simplifies to

$$\boldsymbol{K}_p\boldsymbol{e}_x = \mathbf{0},$$

(12.20)

which implies that for any positive proportional controller gain, $\boldsymbol{e}_x = \mathbf{0}$. Therefore, according to Krasovski-Lasalle's Theorem, it is guaranteed that $\boldsymbol{\chi} = \boldsymbol{\chi}_d$ at steady state, ensuring that the closed-loop system achieves asymptotic tracking.

It is important to note that this analysis solely addresses the steady-state behavior of the system, and the transient behavior of tracking error dynamics may not be as optimal as in the case of full inverse dynamics control. Furthermore, it should be emphasized that this control algorithm represents the simplest form of partial linearization through feedback in the absence of complete knowledge of the robot's dynamics. To shape the transient behavior of tracking error dynamics towards desired values, the utilization of available mass or Christoffel matrices, if accessible, or stronger controllers such as PID and $\mathscr{H}_\infty$ controller can provide significant assistance. However, if, for any reason, the cost of determining complete dynamic formulation or conducting calibration

experiments is prohibitive, this simple control structure can ensure the main properties of inverse dynamics control, namely the stability of the robot's motion and convergence of the tracking error towards zero.

## 12.5 Robust and Adaptive Control

Despite its extensive application across various domains, inverse dynamics control faces two key challenges. First, its effectiveness is heavily dependent on having a complete and accurate dynamic model of the system, along with precise parameter calibration. Achieving this in practice is difficult, as model uncertainties are unavoidable, even with significant efforts to refine the system model. Second, the implementation of this approach can be hindered by the necessity to integrate a detailed system model that includes complex nonlinear and multivariable expressions within the closed-loop framework. Using such a complex model for feedback linearization greatly increases computational demands, thereby requiring more advanced and costly hardware for successful operation.

In Section 12.4, a modified inverse dynamics control method is presented, where partial linearization of the dynamic equations is used to substantially reduce computational costs while still ensuring asymptotic convergence of the tracking error for constantly desired trajectories. However, while this modification improves

computational efficiency, it is less effective for the transient performance of the closed-loop system.

To mitigate the impact of model uncertainties on the transient and steady-state behavior of the system, control literature offers two main strategies: robust and adaptive control. In robust control, worst-case uncertainties within the robot's workspace, as expressed on the right side of Equation 12.10, are encapsulated by an upper bound. Lyapunov-based robust control methods are then used to add a corrective term to the inverse dynamics control, compensating for the effects of uncertainties in the worst-case scenario.

Conversely, adaptive control methods begin with a nominal system model within the inverse dynamics control loop. By monitoring the tracking error and incrementally updating the model parameters based on the linear model of the robot, these parameters are refined to achieve the desired tracking performance. The adaptation law for model parameters in this approach is typically derived from a Lyapunov stability analysis of the closed-loop system.

In both robust and adaptive control methods, the robot dynamics is considered as a nonlinear and multivariable system, eliminating the need for complete linearization of the dynamic equations. While both approaches use Lyapunov-based methods to ensure the stability of the closed-loop system in the presence of uncertainties, they differ significantly in how they handle these uncertainties. The robust control approach employs a fixed controller to

compensate for uncertainties at their worst-case level, requiring substantial control effort. In contrast, adaptive control uses a controller with time-varying coefficients that gradually adjusts the model parameters, ensuring convergence in parameter adaptation.

The choice between these two approaches depends on how uncertainties are interpreted. If the entire model structure is assumed to be known and uncertainties are limited to parameter variations, adaptive methods are more suitable. However, if uncertainties arise from the intentional or unintentional omission of certain dynamics in the model, robust control methods are more appropriate. Additionally, hybrid approaches, known as Adaptive Robust Control and Robust Adaptive Control, have been proposed, though the specifics of these methods are beyond the scope of this book [238, 239].

### 12.5.1 Model Uncertainty

The term “model uncertainty” refers to the disparity between the behavior of the nominal model and the actual behavior of the system. As evident in this interpretation, the manner in which the mathematical modeling of a system is performed is crucial for providing a suitable mathematical encapsulation of model uncertainty. If a system is modeled with a linear model using a transfer function, it is natural for the interpretation of uncertainty in the Laplace space, expressible in a frequency domain. On the other hand, if the dynamic model of a robot is

expressed based on the state space and in the form of a nonlinear system, the interpretation of uncertainty in the system can manifest as disturbance torques in the governing differential equation of the system.

In general, uncertainties existing in a system can be classified into two main categories: structured and unstructured uncertainties. In structured uncertainty, it is assumed that the nominal model of the system has the capability to reproduce the dynamic behavior of the system, even though the model parameters may not be accurately calibrated or may change over time and under environmental conditions. Since the structure of the system is known in this type of uncertainty, it is referred to as structured uncertainty. Meanwhile, as this category of uncertainties is based on the lack of precise knowledge about model parameters, it is also known as parametric uncertainty.

Conversely, numerous uncertainties encountered in actual systems stem from an incomplete understanding of the dynamic system behavior, deliberate simplifications, or inaccuracies introduced while modeling certain dynamic behaviors during the modeling procedure. Take, for instance, an electrical resistor: the linearity of the voltage-current relationship holds true if it works under specific conditions, excluding high frequencies, where the resistor shows inductive or capacitive characteristics. It is inherent that the inductive behavior of a resistor cannot be precisely

elucidated by meticulously calibrating the linear relation between voltage-current and resistance value.

Similar to this example, many components and devices working in real systems exhibit such characteristics. They often model these devices with linear models, even though their linear behavior is only valid in a small range of their operating regime. This type of modeling error is called unstructured model uncertainty because in this type of uncertainty, the structure of the model is ambiguous, not its parameters.

Another example of such uncertainty in model structure is observed in the partial feedback linearization control method. As mentioned earlier, assuming that the dynamic model of the system accurately reconstructs the system's behavior, complexities in determining and calibrating the mass and Christoffel matrices have deliberately been avoided by partially linearizing the dynamic behavior of the system. Consequently, in the closed-loop system model, nonlinear expressions persist, and linear control methods alone have to compensate for the lingering effects of this nonlinear dynamic behavior in the system.

In adaptive control methods, considering the need for a reliable nominal model in the system and attempting to adjust model parameters based on tracking errors, structured uncertainties are taken into account. This is while, in robust control methods, the bound on the modeling uncertainty with respect to its main dynamic behavior is encapsulated, and its effect is compensated for

as an unstructured model uncertainty in the worst-case. Consequently, in these methods, unstructured uncertainties are employed in the system modeling.

In the context of robotic systems, when a significant amount of time and energy is invested in precise system modeling and calibration, one can presume that the robot is afflicted solely with parametric uncertainties. Under such circumstances, adaptive control, leveraging linear regression models, can effectively offset the influence of uncertain system parameters. This stands in contrast to scrutinizing the unmodeled nonlinear behavior of the closed-loop system and the perturbing effects it introduces to the system. These terms are delineated on the right side of Equation 12.10, and in the robust control approach, the bounds of these terms are ascertained in the worst-case scenario. Consequently, robust control methods take precedence in mitigating the impact of uncertainties on the system's tracking error. The ensuing sections will explore the robust inverse dynamics control method first, followed by an exploration of the adaptive inverse dynamics control method.

### 12.5.2 Robust Inverse Dynamics Control

The effectiveness of inverse dynamics control hinges on the complete linearization of nonlinear dynamic terms through feedback. However, achieving this ideal in practice is highly challenging and limits the widespread adoption of this technique in industrial robotics. To begin with,

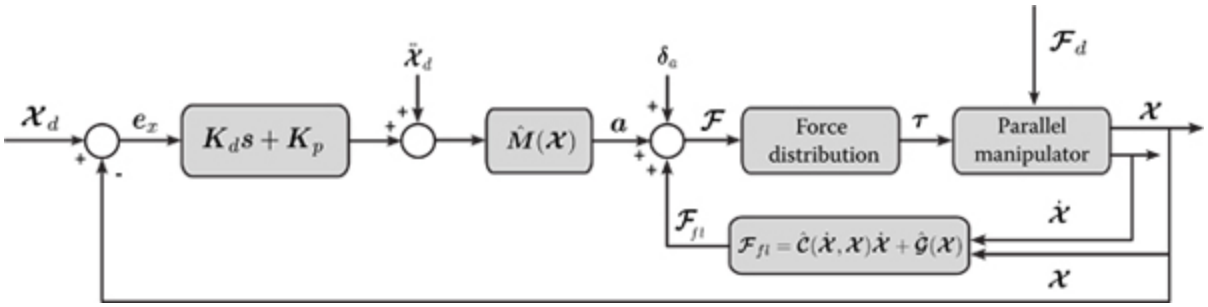
deriving the dynamic equations of a parallel manipulator is a complex task, and rewriting these equations in a closed form poses an additional challenge. For full feedback linearization and the implementation of all dynamic terms within the inverse dynamics control (IDC) framework, not only must the structure and components of these matrices be meticulously derived, but all the kinematic and inertial parameters of the robot manipulator must also be identified and calibrated. This process demands carefully designed experiments and high-precision calibration equipment, which are not always available in every application.

As a result, various uncertainties can arise, including unmodeled dynamics, unknown parameters, calibration errors, unaccounted disturbances, and varying payloads. These factors introduce discrepancies that are not captured in the dynamic model of the manipulator, complicating the implementation of inverse dynamics control in real-world scenarios.

To account for these modeling uncertainties in the closed-loop performance of the manipulator, we begin by revisiting the general closed-form dynamic formulation of the manipulator as provided in Equation 11.1. With this in mind, as shown in Figure 12.4, the inverse dynamics control input $\boldsymbol{\mathcal{F}}$ can be modified as follows:

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\mathcal{X}})\boldsymbol{a}_r + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}}), \tag{12.21}$$

$$\boldsymbol{a}_r \quad = \ddot{\boldsymbol{\mathcal{X}}}_d + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{\delta}_a, \tag{12.22}$$




▶ Long Description for Figure 12.4

**FIGURE 12.4**
General configuration of robust inverse dynamics control implemented in the task space. ⏎

In which the robustifying input $\boldsymbol{a}_r$, which has a dimension of $n$ and acceleration attribute, is used to make the closed loop dynamics robust by incorporating the corrective term $\boldsymbol{\delta}_a$. Figure 12.4 shows the block diagram for implementing robust IDC on a parallel robot. In this diagram, the corrective term is introduced as a wrench in task space, and the actuator torques are determined through a force distribution block.

Note that, as previously defined, the notation $\hat{(\cdot)}$ denotes the estimated value of $(\cdot)$ terms, while $\tilde{(\cdot)}$ represents the error or discrepancy between the estimated

value and the true value of the corresponding term, given by $\tilde{(\cdot)} = \hat{(\cdot)} - (\cdot)$. Therefore,

$$\tilde{\boldsymbol{M}} = \hat{\boldsymbol{M}} - \boldsymbol{M}; \quad \tilde{\boldsymbol{\mathcal{C}}} = \hat{\boldsymbol{\mathcal{C}}} - \boldsymbol{\mathcal{C}}; \quad \tilde{\boldsymbol{\mathcal{G}}} = \hat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{G}}.$$

Similarly, the notion $\tilde{(\cdot)}$ is used to denote the error in the motion variables.

$$\tilde{\boldsymbol{\chi}} = \boldsymbol{\chi} - \boldsymbol{\chi}_d = -\boldsymbol{e}_x, \quad \dot{\tilde{\boldsymbol{\chi}}} = \dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}_d = -\dot{\boldsymbol{e}}_x.$$

Note that this notation is very common in nonlinear control systems, even though with this definition, the tracking error enters subsequent calculations with a negative sign. With this description, the dynamics of the closed-loop system using this control structure can be written as follows.

$$\ddot{\boldsymbol{\chi}} = \boldsymbol{a}_r + \boldsymbol{\eta}\left(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \boldsymbol{a}_r\right)$$

(12.23)

In this equation, $\boldsymbol{\eta}$ is a function that represents the level of uncertainty in the modeling, and can be determined by:

$$\boldsymbol{\eta} = \boldsymbol{M}^{-1}\left(\tilde{\boldsymbol{M}}\boldsymbol{a}_r + \tilde{\boldsymbol{\mathcal{C}}}\dot{\boldsymbol{\chi}} + \tilde{\boldsymbol{\mathcal{G}}}\right)$$

(12.24)

The corrective term in robust control, $\boldsymbol{\delta}_a$, should be designed in a way to reduce the impact of uncertainty $\boldsymbol{\eta}$ on the closed-loop system behavior.

The closed-loop manipulator dynamics described by the uncertain system Equation 12.23 remain nonlinear and coupled due to the modeling uncertainty represented by $\boldsymbol{\eta}$. The robust controller will be designed to ensure that the closed-loop system achieves the desired performance despite the presence of modeling uncertainty $\boldsymbol{\eta}$. The subsequent analysis of the closed-loop tracking performance, considering this modeling uncertainty, will be based on Lyapunov theory.

Numerous approaches have been developed for the robust control of uncertain systems. Here, a nonlinear robust control method based on Lyapunov stability criteria will be introduced. To focus on robustifying the corrective term $\boldsymbol{\delta}_a$, rewrite the robust control effort defined in Equation 12.22 using the $\tilde{(\cdot)}$ notation.

$$\boldsymbol{a}_r = \ddot{\boldsymbol{\chi}}_d - \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_p\tilde{\boldsymbol{\chi}} + \boldsymbol{\delta}_a. \tag{12.25}$$

Substitute the robust control effort provided in Equation 12.25 into the closed-loop dynamic formulation 12.23 and simplify as follows:

$$\ddot{\tilde{\boldsymbol{\chi}}} = -\boldsymbol{K}_d\dot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_p\tilde{\boldsymbol{\chi}} + \boldsymbol{\delta}_a + \boldsymbol{\eta}.$$

(12.26)

If $\boldsymbol{\eta}$ were to be known, compensating for the effects of model uncertainties could be easily accomplished by selecting the corrective term $\boldsymbol{\delta}_a = -\boldsymbol{\eta}$. However, it's crucial to acknowledge that $\boldsymbol{\eta}$ represents all structural and parametric uncertainties in the system and, as a result, cannot be precisely determined. The use of the term “uncertainty” underscores this aspect, since this representation encompasses all unmodeled dynamics, intentional or unintentional simplifications in system modeling, and imprecise knowledge of model parameters during calibration experiments.

Furthermore, as inferred from Equation 12.24, this uncertainty is dependent on the system's state variables $\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}$, and $\boldsymbol{a}_r$. Hence, the upper bound of this term can only be determined through experiments under various operational conditions within the workspace. The corrective term, employing high gains, counteracts the impact of this uncertainty in the worst-case scenario. This process is executed using the Lyapunov redesign method [233]. By defining the state vector as follows,

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \tilde{\boldsymbol{\chi}} \\ \dot{\tilde{\boldsymbol{\chi}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\chi} - \boldsymbol{\chi}_d \\ \dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}_d \end{bmatrix}.$$

(12.27)

Thus, the representation of the closed-loop dynamic of the robot in state space can be written as:

$$\dot{\boldsymbol{\varepsilon}} = \boldsymbol{A}\boldsymbol{\varepsilon} + \boldsymbol{B}(\boldsymbol{\delta}_a + \boldsymbol{\eta}),$$

(12.28)

where

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\boldsymbol{K}_p & -\boldsymbol{K}_d \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix}.$$

(12.29)

The depiction of the system's state space serves to elucidate the modification term $\boldsymbol{\delta}_a$ within the robustness framework. Essential for the stability of the tracking error dynamics is the stability of its linear component, which is characterized by the matrix $\boldsymbol{A}$. To ensure stability, it is imperative to tailor the gains of the PD controller such that the matrix $\boldsymbol{A}$ is Hurwitz. Consequently, all eigenvalues of the matrix $\boldsymbol{A}$ will be strictly positioned to the left of the imaginary axis.

This condition can be ensured using the Lyapunov equation. By appropriately choosing the gains of the PD controller, a symmetric positive-definite matrix $\boldsymbol{P}$ will exist such that, for any symmetric positive-definite matrix $\boldsymbol{Q}$, the Lyapunov equation holds:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} = -\boldsymbol{Q}.$$

(12.30)

Notice that, the primary part of the error equation, incorporating nonlinear and uncertain expressions, still remains. In order to stabilize this component, the corrective term $\boldsymbol{\delta}_a$ must be designed in such a way to satisfy the Lyapunov theorem conditions for the closed-loop system. To achieve this, we define the Lyapunov function candidate for the system using the weighted Euclidean norm of the state error vector of the system.

$$V = \boldsymbol{\varepsilon}^T \boldsymbol{P} \boldsymbol{\varepsilon},$$

(12.31)

In which matrix $\boldsymbol{P}$ is the symmetric positive-definite matrix derived from the Lyapunov equation given in Equation 12.30 for the linear part. To stabilize the error dynamics, $\boldsymbol{\delta}_a$ should be designed so that the derivative of the Lyapunov function along any trajectory is negative definite. Compute this time derivative as follows:

$$\begin{aligned} \dot{V} &= \dot{\boldsymbol{\varepsilon}}^T \boldsymbol{P} \boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}^T \boldsymbol{P} \dot{\boldsymbol{\varepsilon}} \\ &= \boldsymbol{\varepsilon}^T \left( \boldsymbol{A}^T \boldsymbol{P} + \boldsymbol{P} \boldsymbol{A} \right) \boldsymbol{\varepsilon} + 2 \boldsymbol{\varepsilon}^T \boldsymbol{P} \boldsymbol{B} (\boldsymbol{\delta}_a + \boldsymbol{\eta}) \\ &= -\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} + 2 \boldsymbol{\varepsilon}^T \boldsymbol{P} \boldsymbol{B} (\boldsymbol{\delta}_a + \boldsymbol{\eta}). \end{aligned}$$

(12.32)

Note that in the first equation of the tracking error dynamic Equation 12.28 is substituted, and the second

equation is simplified using the Lyapunov Equation 12.30.

The first term in the Equation 12.32 is negative-definite. To ensure the stability of the closed-loop system, $\boldsymbol{\delta}_a$ must be designed in a way that, for the most stringent conditions of uncertainty $\boldsymbol{\eta}$, the second term remains negative-definite. To achieve this, assume that the upper bound of uncertainty $\boldsymbol{\eta}$ has been determined through various experiments on the system and can be represented as a scalar and positive function of state variables and time, denoted as $\rho(\boldsymbol{\varepsilon}, t)$. follows:

$$\| \boldsymbol{\eta} \| \leq \rho(\boldsymbol{\varepsilon}, t).$$

(12.33)

Furthermore, for simplicity, denote $\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{\varepsilon}$ by a vector $\boldsymbol{v}$:

$$\boldsymbol{v} = \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{\varepsilon}.$$

(12.34)

Using this variable, the second term in the derivative of the Lyapunov function 12.32 can be expressed as $\boldsymbol{v}^T(\boldsymbol{\delta}_a + \boldsymbol{\eta})$. This term will be zero if $\boldsymbol{v} = \mathbf{0}$, and may become negative definite if $\boldsymbol{v} \neq \mathbf{0}$, provided the corrective term is chosen as follows:

$$\boldsymbol{\delta}_a = -\rho \frac{\boldsymbol{v}}{\| \boldsymbol{v} \|}.$$

(12.35)

This claim can be proved using the Schwartz inequality as follows:

$$\begin{aligned} \boldsymbol{v}^T(\boldsymbol{\delta}_a + \boldsymbol{\eta}) &= \boldsymbol{v}^T(-\rho\frac{\boldsymbol{v}}{\| \boldsymbol{v} \|} + \boldsymbol{\eta}) \\ &\leq -\rho \| \boldsymbol{v} \| + \| \boldsymbol{v} \| \| \boldsymbol{\eta} \| \\ &= \| \boldsymbol{v} \| (-\rho + \| \boldsymbol{\eta} \|) \\ &\leq 0, \end{aligned}$$

(12.36)

Therefore, as given in Equation 12.33, $\| \boldsymbol{\eta} \| \leq \rho$. Thus,

$$\dot{V} \leq -\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} < 0.$$

(12.37)

These calculations clearly confirm that by designing the corrective term based on the relationship in Equation 12.35, the stability of the closed-loop system is guaranteed under the worst-case uncertainty conditions.

To summarize the robust IDC approach, consider the parallel robot model described in Equation 11.1, where the control effort is computed using Equations 12.21 and 12.22. This control effort includes the standard inverse dynamics control law as well as a corrective term $\boldsymbol{\delta}_a$, given by

$$\boldsymbol{\delta}_a = \begin{cases} -\rho \dfrac{\boldsymbol{v}}{\| \boldsymbol{v} \|} & \text{if } \| \boldsymbol{v} \| \neq 0 \\ 0 & \text{if } \| \boldsymbol{v} \| = 0, \end{cases}$$

(12.38)

where $\boldsymbol{v} = \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{\varepsilon}$ as defined in Equation 12.34, $\boldsymbol{P}$ is the symmetric positive-definite matrix obtained from the solution of the matrix Lyapunov Equation 12.30, $\boldsymbol{\varepsilon}$ represents the augmented motion error defined in Equation 12.27, and $\rho$ is the least upper bound of the modeling uncertainty, with $|\boldsymbol{\eta}| \leq \rho$.

In this design approach, the choice of the corrective term is guided by stability analysis and the application of the Lyapunov theorem. This method is both widely used and effective, as demonstrated by its validation in various other applications [240]. However, there are several possible choices for determining the corrective term, and using the Euclidean norm of the vector $\boldsymbol{v}$ as outlined in Eq. 12.38 is just one approach among many.

Another critical aspect of designing this robust controller is ensuring stability. However, achieving the desired performance in robust trajectory tracking is also crucial. Stability assurance alone does not guarantee the attainment of the desired tracking performance. To address how robust performance can be achieved, it is necessary to revisit the magic of feedback and high-gain utilization, as discussed in Chapter 11. As demonstrated there, in linear systems, feedback combined with high-gain controllers

helps achieve three main goals: accurate tracking, disturbance rejection, and robustness against model inaccuracies. While these principles apply to nonlinear control systems as well, proving their validity is more complex compared to linear systems. The key to achieving the desired performance is the use of high-gain controllers.

Does the corrective term involve high gain? Generally, $\rho$ should account for all uncertainties due to system modeling. When uncertainty is minimal, robust methods may not be necessary, and inverse dynamics control alone is often sufficient and optimal for robot control. The need for large gains usually arises from the practical challenge of precisely determining the upper bound of uncertainty through experimentation. Despite theoretical foundations for defining this bound, a large value is often chosen and adjusted to achieve the desired tracking performance. By using high gains in the controller, the stability of the closed-loop system is ensured, and the desired tracking is achieved despite disturbances and model uncertainties.

Thus, selecting the $\rho$ coefficient, which reflects the level of system uncertainty, becomes a key control challenge alongside tuning the gains of the linear PD controller. Practically, to avoid complexity and ensure closed-loop stability, a large value for this coefficient is selected. Sensor and actuator limitations, as discussed in Section 11.3.4, also apply here and must be considered.

Another notable characteristic of the corrective term is its piecewise and non-smooth nature in both computation

and implementation. For example, in the single-variable case, the expression $\boldsymbol{v}/|\boldsymbol{v}| = v/|v|$ indicates that only the sign of the signal $\mathrm{sgn}(v)$, not its magnitude, is used. With a large negative gain $\rho$, if the signal value $v$ from the tracking error is positive, a large negative corrective term is applied; if the tracking error is negative, a large positive term is applied; and if the tracking error is zero, the corrective term is zero. In the multivariable case, Eq. 12.38 employs weighted signs of the vector elements, preserving the control effort vector's angle.

Controllers with similar characteristics are commonly used in industrial control systems. For instance, the *bang-bang* controller, which features a switching structure between two large positive and negative values, is employed in optimal control [241]. In nonlinear control systems, this type of control is known as “Variable Structure Control (VSC)” or “Sliding Mode Control (SMC),” and it has been applied in various contexts [242, 243]. These controllers share the high-gain switching nature based on the tracking error in the system.

This characteristic is effective and popular for mitigating the impact of modeling uncertainty on tracking error, which is why it is recommended for robust inverse dynamics control in this book. While research into robust control algorithms is ongoing, with numerous books dedicated to this topic [244, 245], a practical limitation of the proposed method is chattering, which leads to

unwanted oscillations. Further discussion on this issue and its mitigation will follow in the next section.

### 12.5.3 Chattering Mitigation

As noted, the corrective term in robust inverse dynamics control, denoted as $\boldsymbol{\delta}_a$ in Equation 12.38, is a piecewise and discontinuous function with respect to time. It only equals zero when the tracking error is zero; otherwise, due to the large gain $\rho$, its magnitude becomes substantial positive or negative values.

In theory, such switching functions are effective for achieving asymptotic tracking stability. However, as shown in Figure 12.5, practical implementation faces challenges due to noise affecting the measurement of joint variables and tracking error calculations. As illustrated, the measured value, contaminated with noise, never exactly reaches zero. Furthermore, high-gain control introduces fast dynamics in the system, causing small but persistent oscillations in the output. This phenomenon, known as “chattering,” not only prevents the tracking error from converging to zero but also introduces noticeable and unwanted oscillations due to high-frequency inputs. This can reduce the lifespan of the robot's components and negatively affect tracking performance. While high gain in the corrective term enhances robustness, it significantly contributes to chattering.

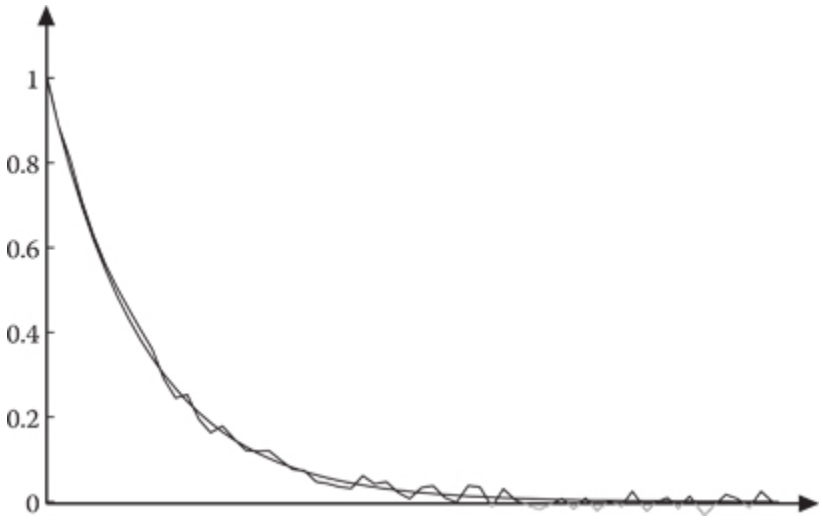



**FIGURE 12.5**
A monotonically decreasing function contaminated with measurement noise may introduce chattering by high-gain switching controller. ⏎

To address chattering, several methods have been proposed in the literature. Continuous approximations of the switching function, such as saturation functions, hyperbolic tangent functions, or fuzzy functions, are suggested [246, 247], as well. Additionally, higher-order sliding mode control methods [248, 249] can reduce chattering under certain conditions.

To mitigate chattering, it is important to smooth the corrective term, particularly because discontinuities causing chattering occur near zero error. The simplest smoothing method is to replace the signum function (*sgn*) with a saturation function (*sat*). This approach replaces the infinite slope of the signum function with a large but bounded slope, converting the discontinuous function into a

continuous one. For a smoother alternative, the hyperbolic tangent function can be used.

Using the saturation function, when the tracking error is within a small range near zero (known as the boundary layer), the control effort transitions smoothly from its positive limit ($\rho$) to its negative limit ($-\rho$). Assuming the use of the saturation function, the corrective term is adjusted as follows:

$$\boldsymbol{\delta}_a = \begin{cases} -\rho \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} & \text{if} \quad \| \boldsymbol{v} \| > \epsilon \\ \\ -\rho \frac{\boldsymbol{v}}{\epsilon} & \text{if} \quad \| \boldsymbol{v} \| \leq \epsilon \end{cases}$$

(12.39)

In this relation, $\epsilon$ represents the boundary layer width, as if the norm of the vector $\boldsymbol{v}$ falls within it ($|\boldsymbol{v}| \leq \epsilon$), the discontinuous switching function is replaced by the linear function $-\rho \frac{\boldsymbol{v}}{\epsilon}$. It is evident that choosing a smaller $\epsilon$ will result in a larger slope for the smoothed linear function, making this approximation closer to the signum function. However, it should be noted that to eliminate the chattering phenomena, the boundary layer width $\epsilon$ must be chosen large enough to cover the range of measurement noise. In practice, by initially selecting a very small value for $\epsilon$, it is gradually increased to observe avoiding undesirable oscillations caused by chattering in the system's behavior.

Although this minor correction of the corrective term is very effective in mitigating the chattering phenomena, the stability analysis performed on the closed-loop system is no longer valid. It is necessary to revisit and rectify the stability analysis of the closed-loop system using the modified corrective term 12.39. For this purpose, consider the previous Lyapunov function candidate 12.31 and differentiate it with respect to time. For $|\boldsymbol{v}| \geq \epsilon$, no change occurs in the control effort compared to before. Therefore, it can be concluded that in this interval, $\dot{V} < 0$.

However, upon entering the boundary layer $|\boldsymbol{v}| \leq \epsilon$, the control effort of the system and, consequently, the derivative of the Lyapunov function with respect to time changes as follows:

$$\begin{aligned} \dot{V} &\leq -\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} + 2\boldsymbol{v}^T(\boldsymbol{\delta}_a + \boldsymbol{\eta}) \\ &\leq -\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} + 2\boldsymbol{v}^T \left( -\rho \frac{\boldsymbol{v}}{\epsilon} + \rho \frac{\boldsymbol{v}}{\| \boldsymbol{v} \|} \right) \\ &\leq -\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} - 2\frac{\rho}{\epsilon} \| \boldsymbol{v} \|^2 + 2\rho \| \boldsymbol{v} \|. \end{aligned} \tag{12.40}$$

By comparing this result with Equation 12.37, it is observed that two new terms have been added, forming a quadratic function with respect to $\| \boldsymbol{v} \|$, where these terms can become positive in the boundary layer. With a simple analysis, it can be concluded that the sum of these two terms will have its maximum value $\frac{1}{2}\epsilon\rho$, when $\| \boldsymbol{v} \| = \epsilon/2$.

Using this upper bound, Equation 12.40 can be simplified as follows.

$$\dot{V} \leq -\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} + \frac{1}{2}\epsilon\rho < 0$$

(12.41)

Thus, $\dot{V}$ will be negative only when:

$$\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon} > \frac{1}{2}\epsilon\rho.$$

(12.42)

On the other hand, we know that $\boldsymbol{\varepsilon}^T \boldsymbol{Q} \boldsymbol{\varepsilon}$ has the following upper and lower bounds:

$$\underline{\lambda}(\boldsymbol{Q}) \parallel \boldsymbol{\varepsilon} \parallel \leq \frac{1}{2}\epsilon\rho \leq \bar{\lambda}(\boldsymbol{Q}) \parallel \boldsymbol{\varepsilon} \parallel$$

(12.43)

where $\underline{\lambda}(\boldsymbol{Q})$ and $\bar{\lambda}(\boldsymbol{Q})$ are the minimum and maximum eigenvalues of the matrix $\boldsymbol{Q}$, respectively. Therefore, with some conservatism, the following condition must be satisfied for $\dot{V}$ to remain negative-definite.

$$\underline{\lambda}(\boldsymbol{Q}) \parallel \boldsymbol{\varepsilon} \parallel > \frac{1}{2}\epsilon\rho$$

(12.44)

This condition can be simplified to the following final form:

$$\| \boldsymbol{\varepsilon} \| > \left( \frac{\epsilon \rho}{2 \underline{\lambda}(\boldsymbol{Q})} \right)^{\frac{1}{2}} \doteq \delta_e.$$

(12.45)

This analysis demonstrates that if the norm of tracking error lies outside the ball $B_\delta$ with radius $\delta_e$, the derivative of the Lyapunov function will remain negative definite. This will cause all state error trajectories in this space to eventually enter the ball $B_\delta$. However, it should be noted that asymptotic convergence of the error towards zero will not occur because the derivative of the Lyapunov function is not negative definite globally, and its negative-definite behavior near the origin is not certain.

This type of stability is referred to in control literature as *Uniformly Ultimately Bounded* (UUB) stability, which in this book, we briefly refer to as UUB stability. For further study on the definition of this stability and its properties, refer to Appendix B. Furthermore, for a deeper analysis of the final steady-state error, we need to study UUB stability using upper and lower bounds of the Lyapunov function. Interested readers are recommended to refer to the reference [233, Section 4.8] for more details.

Nevertheless, what is directly inferred from Equation 12.45 is that the final steady-state error in the system state

variables is proportional to the product of the uncertainty level $\rho$ and the width of the boundary layer $\epsilon$. This implies that to reduce the steady-state error, a better understanding of the system is required, and as much as possible, the boundary layer width should be minimized. This reveals the trade-off between mitigating the chattering phenomena and reducing the ultimate steady-state error in the system.

### 12.5.4 Adaptive Inverse Dynamics Control

An advanced method within inverse dynamics control is its adaptive version. In adaptive inverse dynamics control (AIDC), the full dynamic formulation of the system is employed in the inverse dynamics calculations. Initially, nominal values for matrices and vectors are used. Subsequently, the numerical values of the model's dynamic parameters are adaptively adjusted to minimize the tracking error in the closed-loop system. Ideally, this adaptation process seeks to asymptotically reduce system errors toward zero.

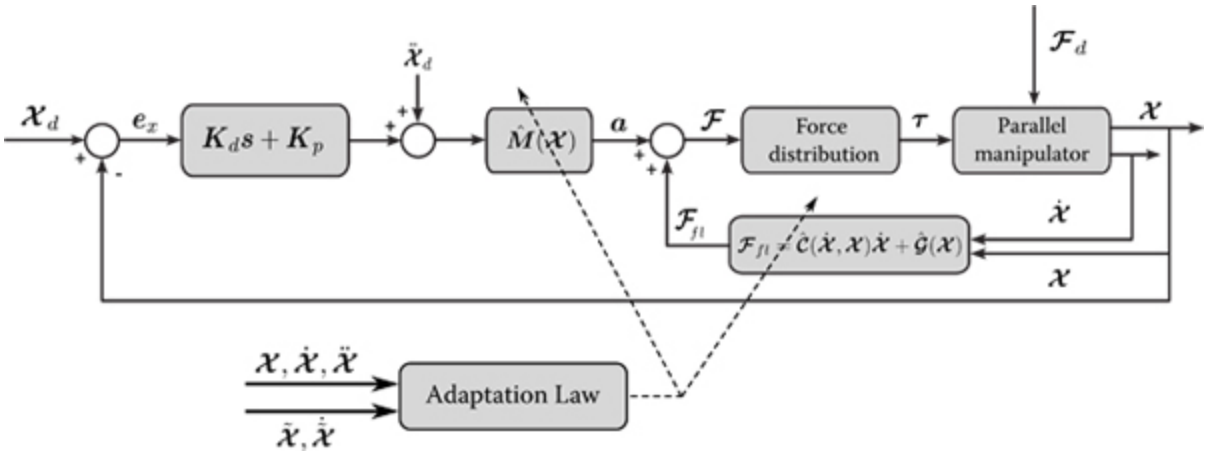
It is important to note that while both robust and adaptive control methods aim to achieve the same objective —eliminating the impact of model uncertainty on the closed-loop system's performance—their approaches differ. Robust control uses a fixed controller with large gains to handle model uncertainty in the worst-case scenario. In contrast, adaptive control involves adjusting the controller based on real-time tracking errors and updating model

parameters over time. Thus, in AIDC, ensuring the stability of the closed-loop system is crucial, but so is the convergence of parameter estimation.

In robust control, precise knowledge of the model structure is not necessary; only the bounds of structural uncertainty need to be defined. Conversely, adaptive control requires a detailed understanding of the model structure in the form of linear regression, focusing on parameter estimation within the model. Given the potential for divergence in parameter estimation, robust control methods are often preferred in practical applications.

In AIDC, as shown in Figure 12.6, the inverse dynamics control structure is used under the assumption that the robot's inverse dynamic equations are known in the nominal state. The adaptation law is then applied to identify real-time parameters of the model in a linear regression format, which are used in inverse dynamics calculations to address the effects of uncertain model structures on tracking performance. As demonstrated in Section 8.3.3, by determining the dimension of the base inertial parameter space, the dynamic equations can be reformulated in a linear regression form based on these parameters.

$$\boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \boldsymbol{C}(\boldsymbol{x}, \dot{\boldsymbol{x}})\dot{\boldsymbol{x}} + \boldsymbol{G}(\boldsymbol{x}) = \boldsymbol{Y}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}})\boldsymbol{\theta}. \tag{12.46}$$




▶ Long Description for Figure 12.6

**FIGURE 12.6**

General configuration of adaptive inverse dynamics control implemented in task space. ⏎

where the matrix $\boldsymbol{Y}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}}, \ddot{\boldsymbol{\mathcal{X}}})$ serves as the regressor, and $\boldsymbol{\theta}$ is the vector of robot base inertial parameters.

To develop the adaptive IDC, recall the general closed-form dynamic formulation of the manipulator given in Equation 11.1, and the inverse dynamics control input $\boldsymbol{\mathcal{F}}$ as

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\mathcal{X}})\boldsymbol{a} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}}), \tag{12.47}$$

$$\boldsymbol{a} = \ddot{\boldsymbol{\mathcal{X}}}_d - \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\mathcal{X}}}} - \boldsymbol{K}_p\tilde{\boldsymbol{\mathcal{X}}}, \tag{12.48}$$

In this relation, nominal matrices in the inverse dynamics are utilized, and their values are subsequently adaptively adjusted based on the tracking error of the desired trajectory using the adaptation law. Similar to before, if we use the notation $\hat{(\cdot)}$ to represent the nominal value, and the notation $\tilde{(\cdot)} = \hat{(\cdot)} - (\cdot)$, to denote the error between the nominal and actual values we can write:

$$\tilde{\boldsymbol{M}} = \hat{\boldsymbol{M}} - \boldsymbol{M}; \quad \tilde{\boldsymbol{\mathcal{C}}} = \hat{\boldsymbol{\mathcal{C}}} - \boldsymbol{\mathcal{C}}; \quad \tilde{\boldsymbol{\mathcal{G}}} = \hat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{G}},$$

Furthermore, we will use the notation $\tilde{(\cdot)}$ to denote the error in the joint variables.

$$\tilde{\boldsymbol{\chi}} = \boldsymbol{\chi} - \boldsymbol{\chi}_d, \qquad \dot{\tilde{\boldsymbol{\chi}}} = \dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}_d.$$

Let's analyze the closed-loop dynamic formulation of the manipulator provided in Equation 11.1 and substitute the AIDC control effort $\boldsymbol{\mathcal{F}}$ as defined in Equation 12.47.

$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}} &= \boldsymbol{\mathcal{F}} \\ &= \hat{\boldsymbol{M}}\left(\ddot{\boldsymbol{\chi}}_d - \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_p\tilde{\boldsymbol{\chi}}\right) + \hat{\boldsymbol{\mathcal{C}}}\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}. \end{aligned}$$

By adding and subtracting $\ddot{\boldsymbol{\chi}}$ on the right side of this equation and simplifying, we can write:

$$\tilde{\boldsymbol{M}}\ddot{\boldsymbol{\chi}} + \tilde{\boldsymbol{C}}\dot{\boldsymbol{\chi}} + \tilde{\boldsymbol{\mathcal{G}}} = \hat{\boldsymbol{M}}\left(\ddot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_p\tilde{\boldsymbol{\chi}}\right).$$

Accordingly, the dynamic tracking error equation can be written based on the linear regression form of system dynamics.

$$\begin{aligned}\ddot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\chi}}} - \boldsymbol{K}_p\tilde{\boldsymbol{\chi}} &= \hat{\boldsymbol{M}}^{-1}\left(\tilde{\boldsymbol{M}}\ddot{\boldsymbol{\chi}} + \tilde{\boldsymbol{C}}\dot{\boldsymbol{\chi}} + \tilde{\boldsymbol{\mathcal{G}}}\right) \\ &= \hat{\boldsymbol{M}}^{-1}\boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}})\tilde{\boldsymbol{\theta}}.\end{aligned}$$

(12.49)

Note that the linear regression model $\boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}})$, as described in Equation 12.46, is applied to simplify the preceding equation. The closed-loop dynamics can then be expressed in state-space form with the state vector defined earlier:

$$\boldsymbol{\varepsilon} = \begin{bmatrix}\tilde{\boldsymbol{\chi}} \\ \dot{\tilde{\boldsymbol{\chi}}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\chi} - \boldsymbol{\chi}_d \\ \dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}_d\end{bmatrix}.$$

(12.50)

Thus, the closed-form dynamic formulation provided in Equation 12.49 can be expressed in state-space form as follows:

$$\dot{\boldsymbol{\varepsilon}} = \boldsymbol{A}\boldsymbol{\varepsilon} + \boldsymbol{B}\boldsymbol{\Phi}\tilde{\boldsymbol{\theta}},$$

(12.51)

where

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\boldsymbol{K}_p & -\boldsymbol{K}_d \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{\Phi} = \hat{\boldsymbol{M}}^{-1}\boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}}).$$

(12.52)

The representation of the system's state space demonstrates the adaptation mechanism effectively. Similar to what was expressed in the analysis of robust inverse dynamics control, the necessary condition for the stability of the tracking error dynamics is that its linear part be stable. The linear part of this dynamics is represented by the matrix $\boldsymbol{A}$, and for stability, it is necessary to choose the control gains of the PD controller in such a way that all the eigenvalues of the matrix $\boldsymbol{A}$ will lie strictly to the left of the imaginary axis.

This condition can be ensured using the Lyapunov equation. With the appropriate selection of the gains of the PD controller, there will exist a symmetric positive-definite matrix $\boldsymbol{P}$ such that, for any arbitrary symmetric positive-definite matrix $\boldsymbol{Q}$, the following Lyapunov equation holds true:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} = -\boldsymbol{Q}.$$

(12.53)

However, the main part of the error equation, which incorporates nonlinear and uncertain terms, $bB\boldsymbol{\Phi}\tilde{\boldsymbol{\theta}}$, is still present. To ensure the convergence of the model parameter estimation and the stability of the closed-loop system, the adaptive law must be designed in a way that the effect of this nonlinear term does not lead the system to instability. Here, we also employ the Lyapunov stability theorem. For this purpose, we propose a Lyapunov function candidate for the system using the Euclidean norm of the weighted state vector in addition to the weighted estimation error vector of the parameters:

$$V = \boldsymbol{\varepsilon}^T \boldsymbol{P} \boldsymbol{\varepsilon} + \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma} \tilde{\boldsymbol{\theta}},$$

(12.54)

In this equation, the matrix $\boldsymbol{P}$ is determined by solving the Lyapunov equation given in 12.53. Additionally, the weighting matrix $\boldsymbol{\Gamma}$ is another symmetric positive-definite matrix used in the adaptive law. The inclusion of the system state variables $\boldsymbol{\varepsilon}$ along with the parameter estimation variables $\boldsymbol{\theta}$ in the Lyapunov function ensures the comprehensive examination of the closed-loop system's stability, incorporating both the desired controller and the adaptation law simultaneously.

Considering the positive-definiteness of the Lyapunov candidate function, if its derivative is negative-definite, the

stability of the closed-loop system is guaranteed. Therefore, to achieve this, we will differentiate this function with respect to time.

$$\begin{aligned}\dot{V} &= \boldsymbol{\varepsilon}^T \left(\boldsymbol{A}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}\right)\boldsymbol{\varepsilon} + 2\tilde{\boldsymbol{\theta}}^T \boldsymbol{\Phi}^T \boldsymbol{B}^T \boldsymbol{P}\boldsymbol{\varepsilon} + 2\tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}\dot{\tilde{\boldsymbol{\theta}}} \\ &= -\boldsymbol{\varepsilon}^T \boldsymbol{Q}\boldsymbol{\varepsilon} + 2\tilde{\boldsymbol{\theta}}^T \left(\boldsymbol{\Phi}^T \boldsymbol{B}^T \boldsymbol{P}\boldsymbol{\varepsilon} + \boldsymbol{\Gamma}\dot{\tilde{\boldsymbol{\theta}}}\right).\end{aligned}$$

(12.55)

Note that in the first equation of the tracking error dynamics 12.49 is substituted, and the second equation is simplified using the Lyapunov Equation 12.53. The first term in Equation 12.55 is negative-definite. To guarantee the stability of the closed-loop system, the adaptive law should be designed such that the second term equals zero. Given that $\dot{\tilde{\boldsymbol{\theta}}} = \dot{\hat{\boldsymbol{\theta}}}$, the adaptive law can be formulated as follows:

$$\dot{\hat{\boldsymbol{\theta}}} = -\boldsymbol{\Gamma}^{-1} \boldsymbol{\Phi}^T \boldsymbol{B}^T \boldsymbol{P}\boldsymbol{\varepsilon}.$$

(12.56)

By utilizing this adaptive law, the second term in the Equation 12.55 is eliminated, and considering the derivative of the Lyapunov function, it will become negative-definite with respect to the state variables.

$$\dot{V} = -\boldsymbol{\varepsilon}^T \boldsymbol{Q}\boldsymbol{\varepsilon} < 0.$$

(12.57)

Given that the derivative of the Lyapunov function is negative-definite, the stability of the closed-loop system is guaranteed, while the parameter estimation error remains bounded.

Figure 12.6 depicts the structure of adaptive inverse control. As shown, this control method mirrors inverse dynamics control but differs in that the robot's inverse dynamics equations are adapted according to the adaptive law described in Equation 12.56. In this setup, in addition to tuning the PD controller matrices, the positive-definite matrix $\boldsymbol{\Gamma}$ must also be adjusted. A larger norm of $\boldsymbol{\Gamma}$ reduces the adaptation intensity of model parameters, enhancing stability and decreasing oscillations in state variables, though at the expense of slower response and a longer time to address parameter uncertainties during transient behavior.

Implementing the adaptive law requires the second derivative of the robot joint motions $\ddot{\boldsymbol{q}}$ and the inverse of the estimated effective mass matrix $\hat{\boldsymbol{M}}$ to compute $\boldsymbol{\Phi}$. The need for joint accelerations poses a significant challenge, complicating the implementation of this approach. However, if the linear controller gains are well-tuned, the desired joint accelerations $\ddot{\boldsymbol{q}}_d$ can be used instead of actual joint motions, which helps mitigate this constraint. Although this adjustment practically alleviates the issue, it adds complexity to stability analysis and convergence guarantees for the estimated parameters. The robotics

control literature includes other methods to address these challenges, such as passivity-based approaches that can overcome both of these constraints [2, Section 9.4].

---

## 12.6 Summary of Motion Control Techniques

This chapter introduces various control techniques designed for parallel robots. Leveraging the dynamic formulation provided for the parallel robot, numerous model-based control strategies have been developed for implementation in both task space and joint space. These techniques range from basic decentralized PD control to more sophisticated robust and adaptive inverse dynamics control. A summary of the techniques is provided below for a comprehensive overview.

### Dynamic Formulations

The dynamic formulation of a parallel robot can be directly expressed as a function of the motion variable $\boldsymbol{\mathcal{X}}$ in the task space, as follows:

$$\boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{F}} + \boldsymbol{\mathcal{F}}_d.$$

### Decentralized PD Control

The simplest control strategy for a parallel robot is a decentralized PD controller, where each error component is controlled independently. When implemented in the task space, the control effort is computed as:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x,$$

where the actuator effort is typically determined using a force distribution scheme. A detailed discussion on such force distribution can be found in Section 6.6. For a fully parallel manipulator, the actuator forces can be calculated as $\boldsymbol{\tau} = \boldsymbol{J}^{-T} \boldsymbol{\mathcal{F}}$ in nonsingular configurations.

## Feed-Forward Control

To overcome the performance limitations of simple PD control, the control effort can be augmented with a feed-forward wrench, expressed as:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}_{pd} + \boldsymbol{\mathcal{F}}_{ff},$$

where

$$\boldsymbol{\mathcal{F}}_{ff} = \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x + \hat{\boldsymbol{M}}(\boldsymbol{\mathcal{X}}_d) \ddot{\boldsymbol{\mathcal{X}}}_d + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{X}}_d, \dot{\boldsymbol{\mathcal{X}}}_d) \dot{\boldsymbol{\mathcal{X}}}_d + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}}_d)$$

## Inverse Dynamics Control

In inverse dynamics control, the nonlinear dynamics of the robot model is utilized to introduce a corrective term to the decentralized PD controller. When this approach is applied in the task space, the control effort is given by:

$$\begin{aligned} \boldsymbol{\mathcal{F}} &= \hat{\boldsymbol{M}}(\boldsymbol{\mathcal{X}})\boldsymbol{a} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}}) \\ \boldsymbol{a} &= \ddot{\boldsymbol{\mathcal{X}}}_d + \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x. \end{aligned}$$

In general, the tracking error dynamics can be expressed as:

$$\ddot{\boldsymbol{e}}_x + \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x + \hat{\boldsymbol{M}}^{-1}\left[\tilde{\boldsymbol{M}}\ddot{\boldsymbol{\mathcal{X}}} + \tilde{\boldsymbol{\mathcal{C}}}\dot{\boldsymbol{\mathcal{X}}} + \tilde{\boldsymbol{\mathcal{G}}} + \boldsymbol{\mathcal{F}}_d\right] = \boldsymbol{0}.$$

## Partial Feedback Linearization

To lower the computational cost of inverse dynamics control, partial linearization of the dynamic formulation can be employed by compensating only for gravity, while maintaining the asymptotic tracking stability of the closed-loop system. In this case, the control input wrench in the task space is simplified to:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}}).$$

The stability of the tracking dynamics in the closed-loop system can be analyzed using the following Lyapunov

function:

$$\dot{V} = \dot{\boldsymbol{\chi}}^T \boldsymbol{M} \ddot{\boldsymbol{\chi}} + \frac{1}{2} \dot{\boldsymbol{\chi}}^T \dot{\boldsymbol{M}} \dot{\boldsymbol{\chi}} + \boldsymbol{e}_x^T \boldsymbol{K}_p \boldsymbol{e}_x.$$

Stability analysis of the closed-loop system, in this case, demonstrates that the simplified version of inverse dynamics control can achieve asymptotic tracking for constantly desired trajectories.

## Robust Inverse Dynamics Control

To address modeling uncertainties in inverse dynamics control, the following robust control scheme is proposed in the task space:

$$\begin{aligned} \boldsymbol{\mathcal{F}} &= \hat{\boldsymbol{M}}(\boldsymbol{\chi}) \boldsymbol{a}_r + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}) \\ \boldsymbol{a}_r &= \ddot{\boldsymbol{\chi}}_d + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x + \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{\delta}_a, \end{aligned}$$

In this scheme, the robustifying corrective term $\boldsymbol{\delta}_a$ is determined through a Lyapunov stability analysis of the tracking error dynamics. The tracking error dynamics can be decomposed into the following linear and nonlinear components:

$$\dot{\boldsymbol{\varepsilon}} = \boldsymbol{A}\boldsymbol{\varepsilon} + \boldsymbol{B}(\boldsymbol{\delta}_a + \boldsymbol{\eta}),$$

where $\boldsymbol{\eta}$ encapsulates the norm-bounded modeling uncertainties and

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\boldsymbol{K}_p & -\boldsymbol{K}_d \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix}.$$

The corrective term $\boldsymbol{\delta}_a$ can be determined as:

$$\boldsymbol{\delta}_a = \begin{cases} -\rho \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} & \text{if } \| \boldsymbol{v} \| > \epsilon \\ \\ -\rho \frac{\boldsymbol{v}}{\epsilon} & \text{if } \| \boldsymbol{v} \| \leq \epsilon \end{cases}$$

where $\boldsymbol{v}$ is given by $\boldsymbol{v} = \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{\varepsilon}$, with $\boldsymbol{P}$ being the solution to the matrix Lyapunov equation 12.30, and $\epsilon$ representing a smoothing threshold. It is demonstrated that incorporating this corrective term into the standard inverse dynamics control results in uniform ultimate bounded tracking errors for the closed-loop system.

## Adaptive Inverse Dynamics Control

In the adaptive version of inverse dynamics control, complete feedback linearization is achieved by adaptively updating the dynamic formulation matrices. The error dynamics in this case are

$$\dot{\boldsymbol{\varepsilon}} = \boldsymbol{A} \boldsymbol{\varepsilon} + \boldsymbol{B} \boldsymbol{\Phi} \tilde{\boldsymbol{\theta}},$$

in which

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\boldsymbol{K}_p & -\boldsymbol{K}_d \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{\Phi} = \hat{\boldsymbol{M}}^{-1}\boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}}).$$

Using Lyapunov stability analysis, the following Lyapunov function is employed:

$$V = \boldsymbol{\varepsilon}^T \boldsymbol{P} \boldsymbol{\varepsilon} + \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma} \boldsymbol{\theta},$$

The following parameter adaptation law is derived for updating:

$$\dot{\hat{\boldsymbol{\theta}}} = -\boldsymbol{\Gamma}^{-1} \boldsymbol{\Phi}^T \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{\varepsilon}.$$

By this means, the closed-loop system attains asymptotic tracking performance, while the parameter estimation errors stay bounded.

## 12.7 Case Study I: Planar Cable Robot

In this section, the motion control of a planar 4RPR cable robot is thoroughly examined, with a focus on applying various control topologies discussed in this book, allowing for an in-depth analysis of their efficacy. An extensive comparison of the closed-loop performances of these

topologies is conducted, shedding light on the strengths and weaknesses of each approach.

The architecture of the planar manipulator under investigation is detailed in Section 4.4.1 and visually represented in Figure 4.4. As previously mentioned, this particular cable robot features a moving platform that is supported by four cables, each possessing an identical RPR kinematic structure. A comprehensive Jacobian and singularity analysis of this manipulator is provided in Section 5.5, while the dynamic analysis is discussed in Section 7.4.

The control strategies are presented in a step-by-step manner, starting with basic linear approaches and progressing to more advanced techniques. Initially, the closed-loop performances and the associated control efforts are thoroughly examined, offering a detailed insight into how each strategy impacts the system's overall performance. Following this, a comparative study is conducted, showcasing the most effective control methods and serving as a reference for future advancements and applications in related areas.

The previous chapter offers a detailed analysis of linear controller performance, exploring various simulation alternatives. Initially, PD and PID controllers are evaluated in a closed-loop system for both simple and cubic trajectories, followed by an assessment of a quintic trajectory under input disturbances and measurement noise. The simulation results indicate that the outcomes for

different trajectories are quite similar, provided suitable controller gains are selected.

Moreover, it was observed that the PD controller's tracking performance was not as effective as that of the PID controller, particularly across different configurations and when the manipulator faced external disturbances. The PID controller showed better results in reducing tracking errors, especially in steady-state conditions and under disturbance inputs. However, it did not completely mitigate the effects of measurement noise, and high-bandwidth actuators were still required to achieve the necessary control efforts.

To make the comparisons manageable, a typical yet multifaceted scenario is considered in the simulations conducted in this section. This approach enables a fair comparison between different control methods. In these simulations, the geometric and inertial parameters from Table 10.1 are utilized, and the results for quintic trajectories over a 200-second time span are reported, with similar outcomes observed for cubic trajectories.

Additionally, a series of unit step disturbances are applied at various intervals in all simulations. Specifically, a $100\,N$ step disturbance force in the $x$ direction ($F_{d_x}$) is introduced at 50 seconds, followed by a $100\,N$ step disturbance force in the $y$ direction ($F_{d_y}$) at 100 seconds, and a $100\,Nm$ disturbance torque in the $\phi$ direction ($\tau_{d_\phi}$) at 150 seconds. No measurement noise is included in these simulations.

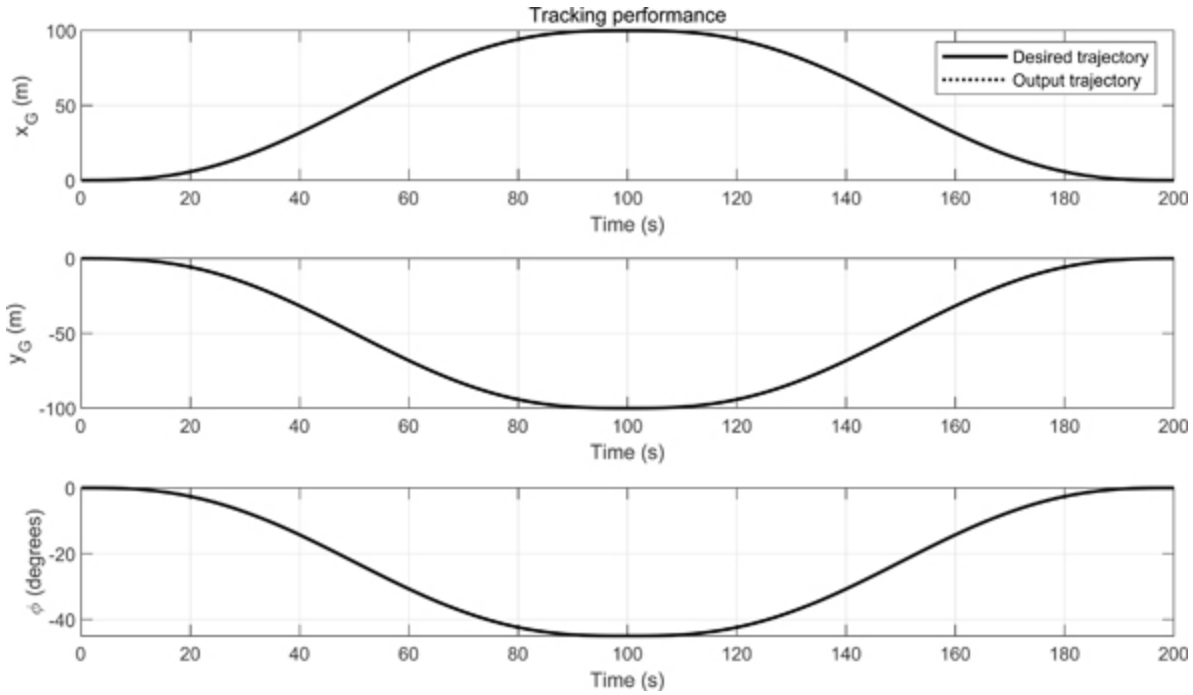
These simulations also consider significant parameter uncertainties, set at 50%, to assess the closed-loop system's performance under challenging conditions with all the model-based controllers. The controller's output, represented by $\boldsymbol{\mathcal{F}}$, is then translated into actuator forces via the force distribution block. As previously mentioned, redundancy resolution methods for cable-driven redundant manipulators, as detailed in Section 6.6, are applied for this projection.

### 12.7.1 Basic PID Control

The PID controller is the most commonly used controller for trajectory tracking in industrial robots. In this section, we begin by evaluating the tracking performance of the desired trajectory in the planar 4RPR cable robot using a decentralized implementation of this controller in task space, as shown in Figure 11.12. For the sake of comparison, the PID gains are set to $\boldsymbol{K}_p = \boldsymbol{K}_d = \boldsymbol{K}_i = 10^4 \cdot \text{diag}\,[1, 1, 100]$, consistent with the pre-tuned controller gains of the PID controller discussed in the previous chapter.

The simulation results are illustrated in Figures 12.7 to 12.9. The overall tracking performance of the closed-loop system is shown in Figure 12.7. As depicted, the trajectory traversed by the moving platform closely matches the desired trajectory, with negligible differences. To better visualize the tracking error, refer to the top chart of Figure 12.8. The tracking error is very suitable and generally less

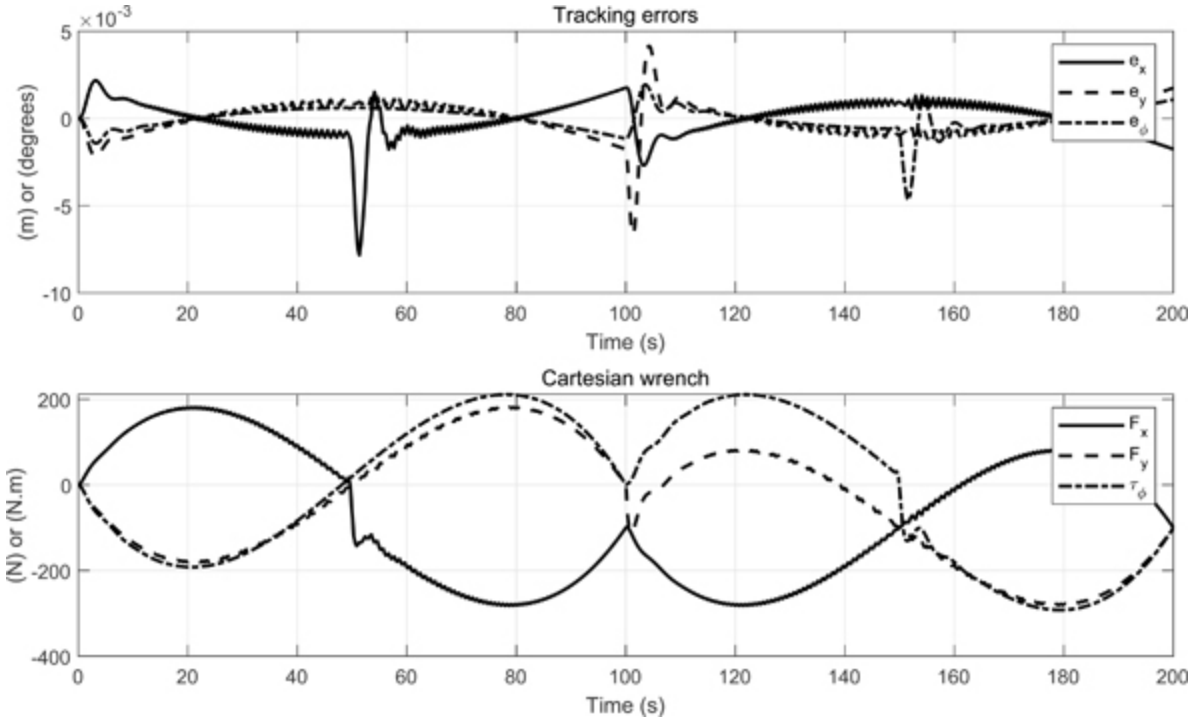
than five millimeters in the $x$ and $y$ directions and 0.002 degrees in the $\phi$ direction. Additionally, the tracking error shows distinct spikes corresponding to the times when disturbance wrenches are applied, though the impact of these disturbances diminishes over time. This demonstrates that the PID controller effectively reduces the impact of disturbances on the robot's performance, particularly at steady-state.




▶ Long Description for Figure 12.7

**FIGURE 12.7**

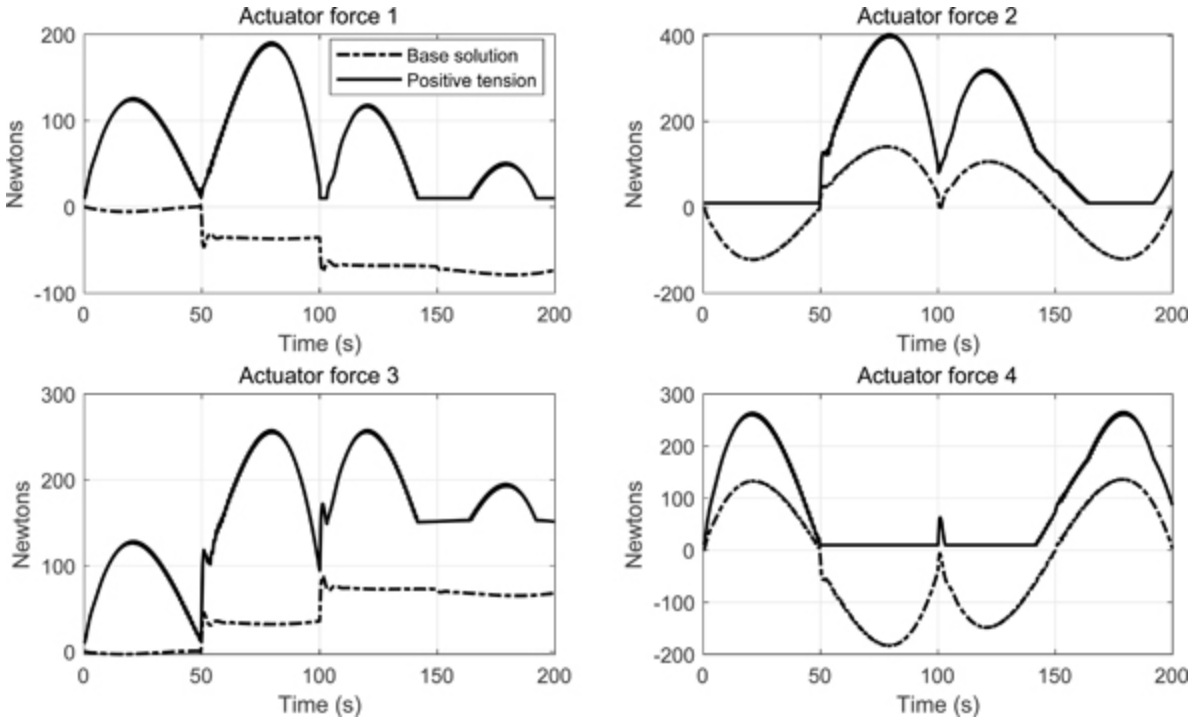
The closed-loop tracking performance of the planar 4RPR cable robot with PID control for a quintic trajectory. ⏎




▶ Long Description for Figure 12.8

**FIGURE 12.8**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under PID control. ⏎




▶ Long Description for Figure 12.9

**FIGURE 12.9**

The actuator forces of the planar 4RPR cable robot generated by PID controller to track trajectories shown in Figure 12.7. ⏎

The bottom chart of Figure 12.8 illustrates the Cartesian wrench generated by the PID controller during maneuvers with disturbances. This chart clearly indicates how disturbances affect the required control wrenches at 50, 100, and 150 seconds. The variation in wrench components at these times reflects the increased force demands needed to counteract the disturbances. This highlights the PID controller's capability to counteract the effects of disturbances and manage the control effort required when disturbances are introduced to the system. The amount of

Cartesian wrench required is about $\pm 250\, N$ and $N.m$, which is comparable to the amplitude of the disturbance force/torques applied to the system.

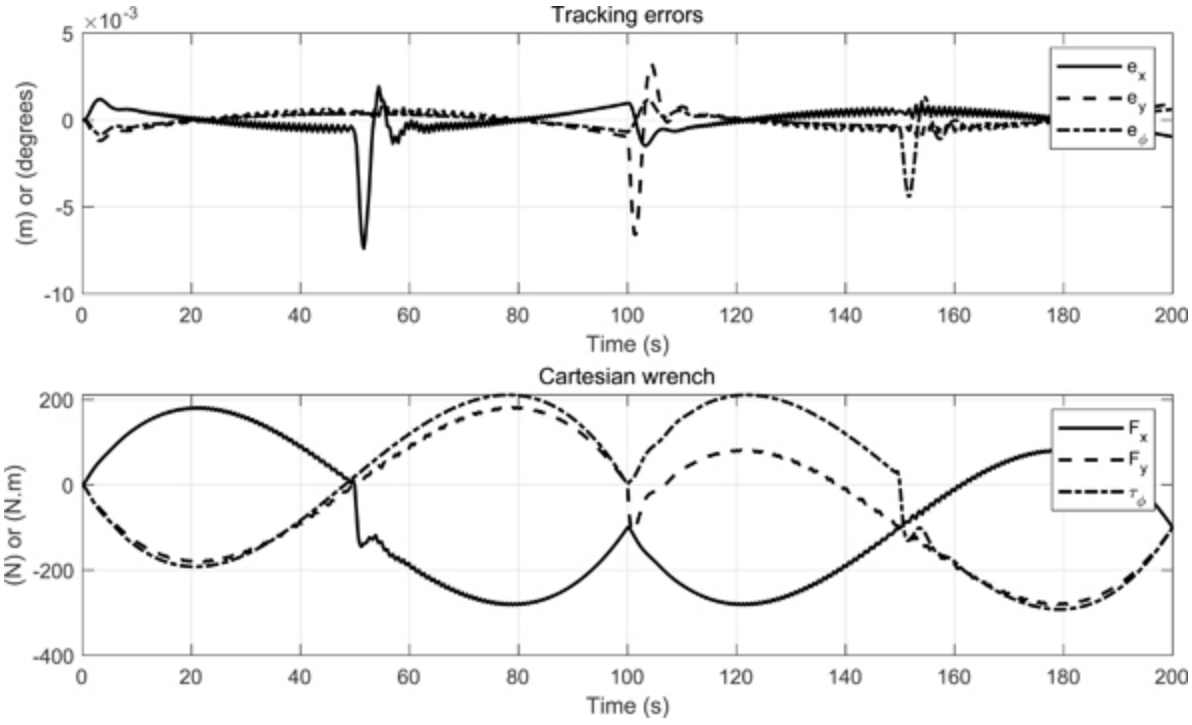
Figure 12.9 displays the actuator forces of the manipulator, with two sets of forces shown: the dash-dotted lines represent the base solution of the redundancy resolution optimization problem, and the solid lines denote the final optimized solution. The results indicate that the actuator forces needed for this maneuver are all less than 400 Newtons, which is quite manageable. Additionally, the tension forces determined by the redundancy resolution scheme remain positive across all configurations, ensuring that the forces applied to the manipulator are physically feasible.

### 12.7.2 PID Control with Feed-Forward

In this section, we will investigate the impact of adding a feed-forward term based on the dynamic model of the system to the previously tuned PID controller. As discussed, this method requires complete knowledge of the dynamic model of the system and calibrated robot parameters. However, since precise identification of kinematic and dynamic parameters involves various calibration experiments and comes with a significant computational cost, in this study, we assume these parameters are not accurately known. In the conducted simulations, we consider parameter uncertainties at a level of 50% to

examine the response of the closed-loop system with the feed-forward controller under these severe conditions.

The closed-loop tracking error for this scenario is illustrated in the top chart of Figure 12.10 for the quintic polynomial trajectory depicted in Figure 12.7. When compared to Figure 12.8, very similar responses are achieved in PID control with or without the feed-forward term. This results roots in the fact that the integral term in the per-tuned PID controller can effectively reduce the tracking error and the effect of disturbances, while a feed-forward term with 50% uncertainty in the modeling parameter could slightly contribute in the performance.

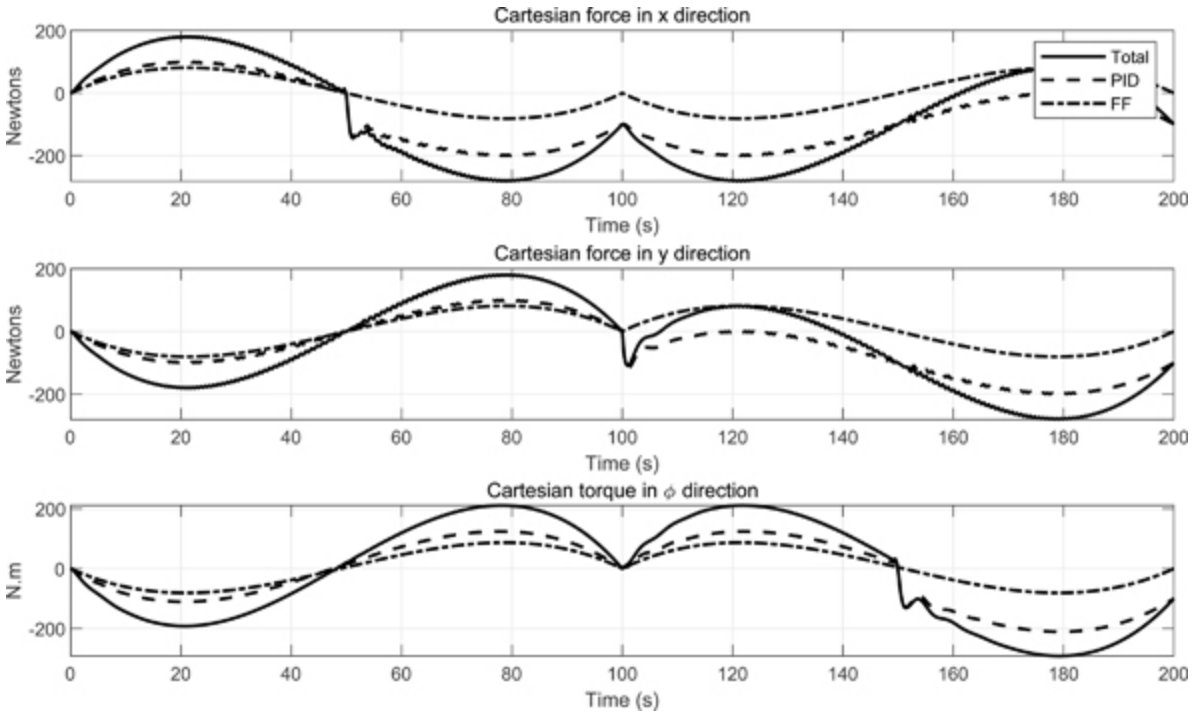



▶ Long Description for Figure 12.10

**FIGURE 12.10**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under PID control and feed-forward term. ⏎

To better illustrate the contributions of the PID control and the feed-forward term, Figure 12.11 presents the Cartesian wrench generated by this controller. In the figure, the solid line is the total cartesian wrench, the dashed line represents the contribution of the PID term, while the dash-dotted line shows the feed-forward term. Initially, due to the poor knowledge of model parameters in this simulation, the contributions of the PID term and the feed-forward term are nearly identical during the first 50 seconds, when no disturbance wrench is applied. This indicates that, under these conditions, both terms are equally involved in maintaining system performance.



▶ Long Description for Figure 12.11

**FIGURE 12.11**
Components of the Cartesian wrench of the planar 4RPR manipulator generated by PID controller with feed-forward term. ⏎

However, when a disturbance wrench is introduced in a specific direction, the contribution of the PID term becomes more prominent along that direction, effectively addressing the disturbance and stabilizing the system. In contrast, the feed-forward term responds primarily to the inertial wrenches and follows the acceleration trend of the moving platform, providing a complementary role. This behavior clearly demonstrates the critical role of the linear controller structure, especially the integral term, in mitigating the effects of disturbances and ensuring robust performance.

The importance of this dynamic becomes even more apparent when comparing it to another simulation involving a PD controller with a feed-forward term. In that case, the feed-forward term predominates, yet the overall tracking performance of the closed-loop system is significantly reduced. This comparison underscores the effectiveness of the PID controller, particularly in its ability to manage and reduce the impact of disturbances through its integral action, which enhances the system's stability and accuracy in tracking the desired trajectory.

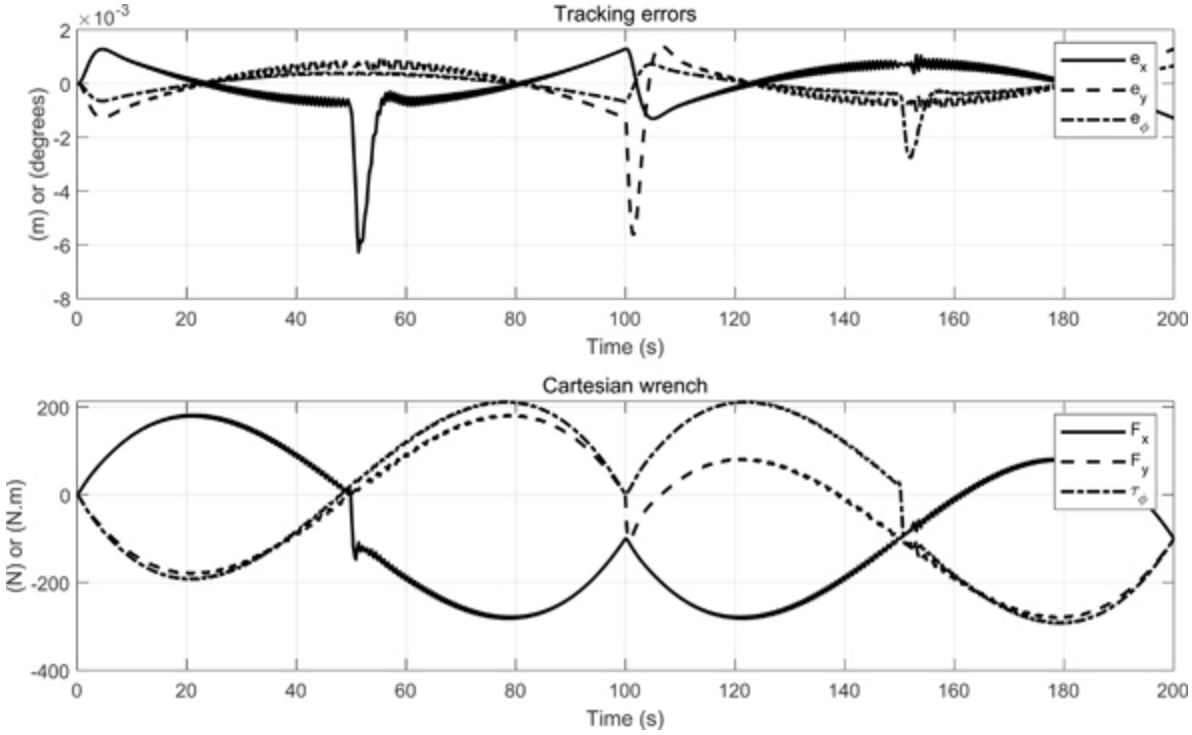
### 12.7.3 Inverse Dynamics Control

The overall configuration of the inverse dynamics control is outlined in the block diagram depicted in Figure 12.2. This configuration employs both internal and external feedback loops. Within the internal loop, the wrench $\boldsymbol{\mathcal{F}}_{fl}$ is computed at each instant based on feedback from the motion variables in task space and is then employed as a compensatory term on the robot. Considering that, in this control method, the PID controller gain matrices are multiplied by the robot's inertia matrix $\hat{\boldsymbol{M}}(\boldsymbol{q})$, the values of these gains should be adjusted differently compared to the previous control methods to enable a fair comparison. In simulating this control approach, by employing a less trial and error process compared to the previous scenario, the PID controller gain matrices can be tuned more effectively as follows.

$$\boldsymbol{K}_p = \boldsymbol{K}_d = 2\boldsymbol{K}_i = 10 \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12.58}$$

By utilizing these gain matrices, the necessary matching for tracking error and control effort, such as the PID controller with feed-forward, can be achieved. By this means, comparing the results of these methods will be more meaningful.

The closed-loop tracking error for this scenario is illustrated in the top chart of Figure 12.12 for the quintic polynomial trajectory depicted in Figure 12.7. When compared to Figure 12.10, it is observed that a twice better performance is obtained and the tracking error is reduced to two millimeters in the $x$ and $y$ directions and 0.001 degrees in the $\phi$ direction. However, as before, distinct spikes corresponding to the times when disturbance wrenches are applied are seen, though the impact of these disturbances diminishes over time.



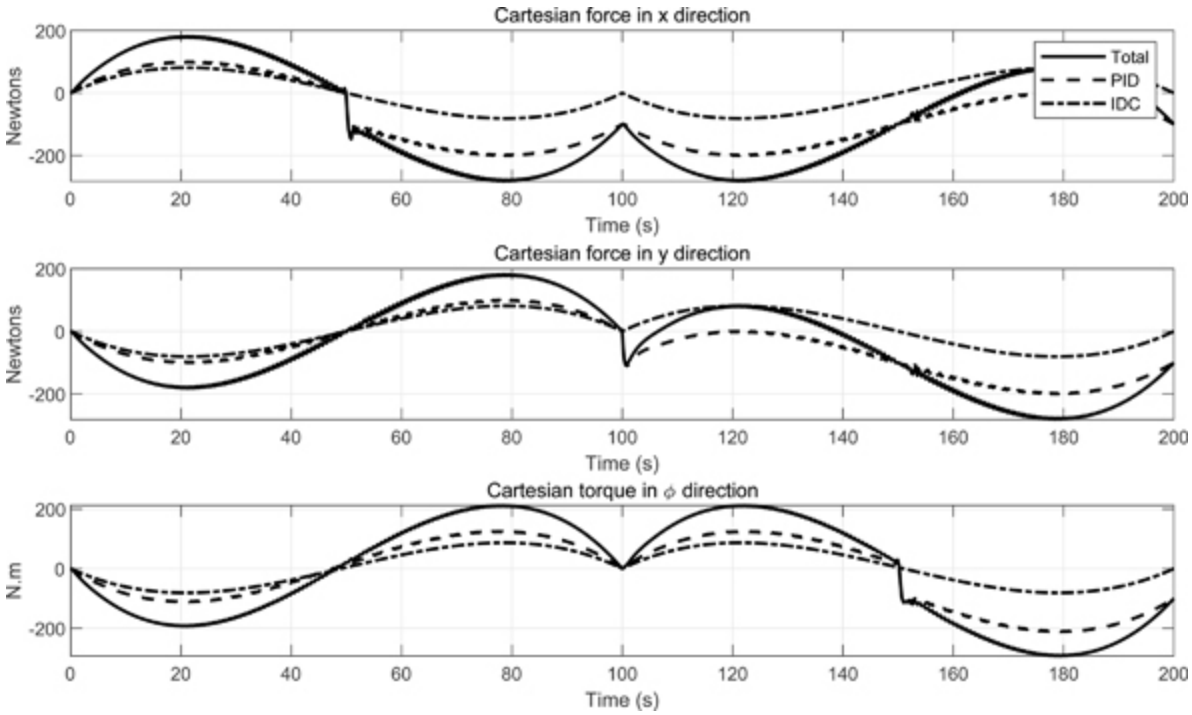

▶ Long Description for Figure 12.12

**FIGURE 12.12**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under

inverse dynamics with PID control. ⏎

To illustrate the roles of inverse dynamics and PID control, Figure 12.13 displays the Cartesian wrench produced by this controller. The solid line represents the total wrench, the dashed line shows the contribution from the PID term, and the dash-dotted line indicates the contribution of inverse dynamics control. The overall control effort required is similar to that with and without the feed-forward term in the PID controller, as seen in Figure 12.13. During the first 50 seconds, when no disturbance is applied, the contributions from both inverse dynamics and PID controls are nearly identical. However, once a disturbance wrench is introduced in a specific direction, the PID term's contribution becomes more pronounced, effectively addressing the disturbance and stabilizing the system. In contrast, inverse dynamics control primarily deals with the inertial wrenches and tracks the acceleration trend of the platform, providing a complementary role. This highlights the crucial role of the linear controller structure, especially the integral term, in mitigating disturbances and ensuring robust performance. In the simulation results, actuator forces are not shown since the Cartesian wrenches required are identical to those in pure PID control. Consequently, the actuator forces are very similar to those depicted in Figure 12.9.




▶ Long Description for Figure 12.13

**FIGURE 12.13**

Components of the Cartesian wrench of the planar 4RPR manipulator generated by inverse dynamics with PID controller. ⏎

## 12.7.4 Partial Feedback Linearization

Inverse dynamics control offers several advantages that make it appealing for practical use. However, as discussed in Section 12.3, applying this method requires complete knowledge of the dynamic formulation matrices. To design the simplest feasible version of IDC, let's revisit the partial feedback linearization IDC introduced in Section 12.4. In its simplest form, the control input $\mathcal{F}$ is composed of two components, as shown in Equation 12.11. The first

component, $\boldsymbol{\mathcal{F}}_{pd}$, is produced by the linear PD od PID controller acting on the tracking error $\boldsymbol{e}_x$ in task space, while the second component is merely the manipulator's gravity vector at any given configuration $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$, as given in Equation 12.13. In this approach, the computationally demanding Coriolis and centrifugal terms are excluded from the feedback.

In this scenario, since the planar motion of the cable robot is perpendicular to the gravity vector, gravitational forces do not impact the motion dynamics, meaning $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) = \mathbf{0}$ in the three task variable directions, $\boldsymbol{\mathcal{X}}$. With this setup, the partial feedback linearization simplifies to a purely linear controller using PD or PID in the loop, and as a result, no additional simulations need to be reported here.

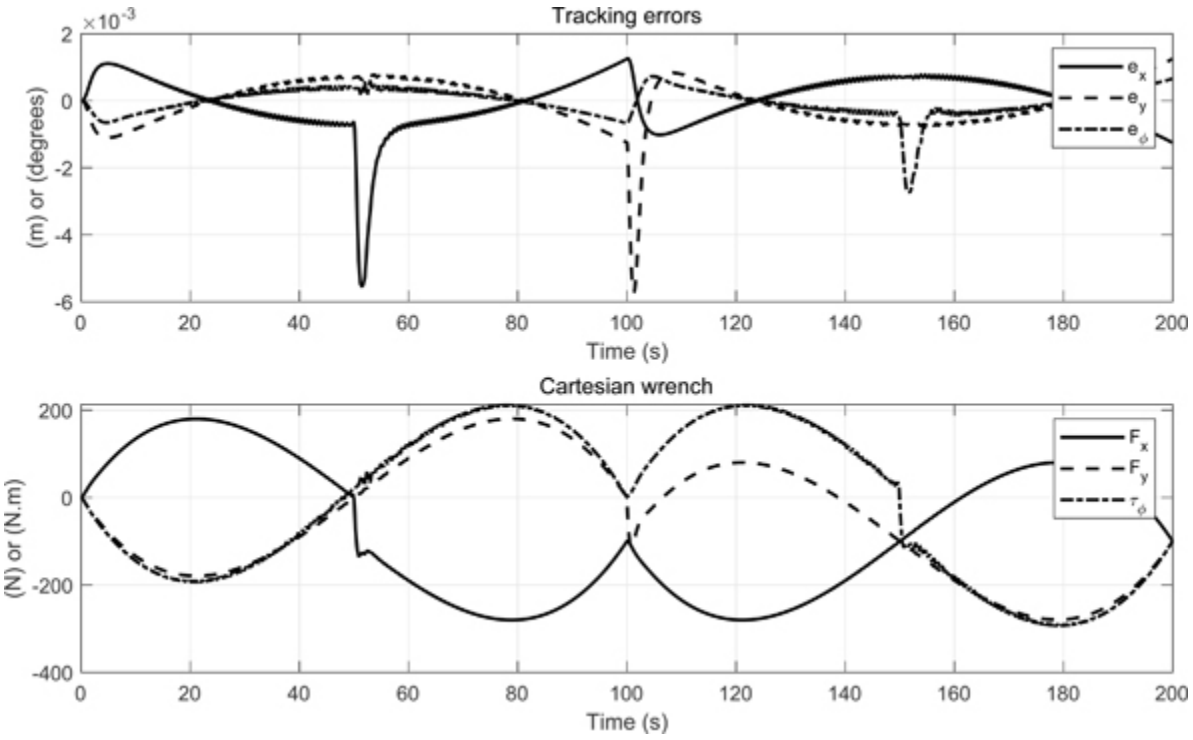
### 12.7.5 Robust Inverse Dynamics Control

The success of inverse dynamics control depends on accurately identifying the system's dynamic model and parameters, which is impossible to achieve with full precision in real-world systems. To overcome this limitation, robust inverse dynamics control introduces a robust term, $\boldsymbol{\delta}_a$, into the control framework, as defined by the relationship in 12.38. To reduce chattering effects in this control approach, the robust term is smoothly integrated within the boundary layer, as described by the Equation 12.39.

In robust inverse dynamics control, one of the adjustable parameters is the positive-definite and symmetric matrix $\boldsymbol{Q}$ from the Lyapunov Equation 12.30, which defines the matrix $\boldsymbol{P}$. By tuning the diagonal elements of this matrix, the control effort distribution across the robot joints can be modified. In the simulations for this section, setting $\boldsymbol{Q}$ as a diagonal matrix with all elements equal to one results in an even distribution of control effort across all position and velocity variables.

Other parameters of the robust inverse dynamics control are determined through trial and error, with $\rho = 2 \times 10^3$ and $\epsilon = 0.3$ chosen to handle a 50% uncertainty in model parameters while minimizing chattering in the control effort of the closed-loop system. Additionally, the PID controller gain matrices are set to the pre-optimized values used in inverse dynamics control, as given in Equation 12.58.

The closed-loop tracking error for this scenario is shown in the top chart of Figure 12.14, based on the quintic polynomial trajectory illustrated in Figure 12.7, despite the presence of input disturbances. When compared to Figure 12.12, the results demonstrate similar performance, with the tracking error being held to two millimeters in the $x$ and $y$ directions and to 0.001 degrees in the $\phi$ direction. However, as with previous results, noticeable spikes occur at the moments when disturbance wrenches are applied, while gradually have less impact on the system over time.



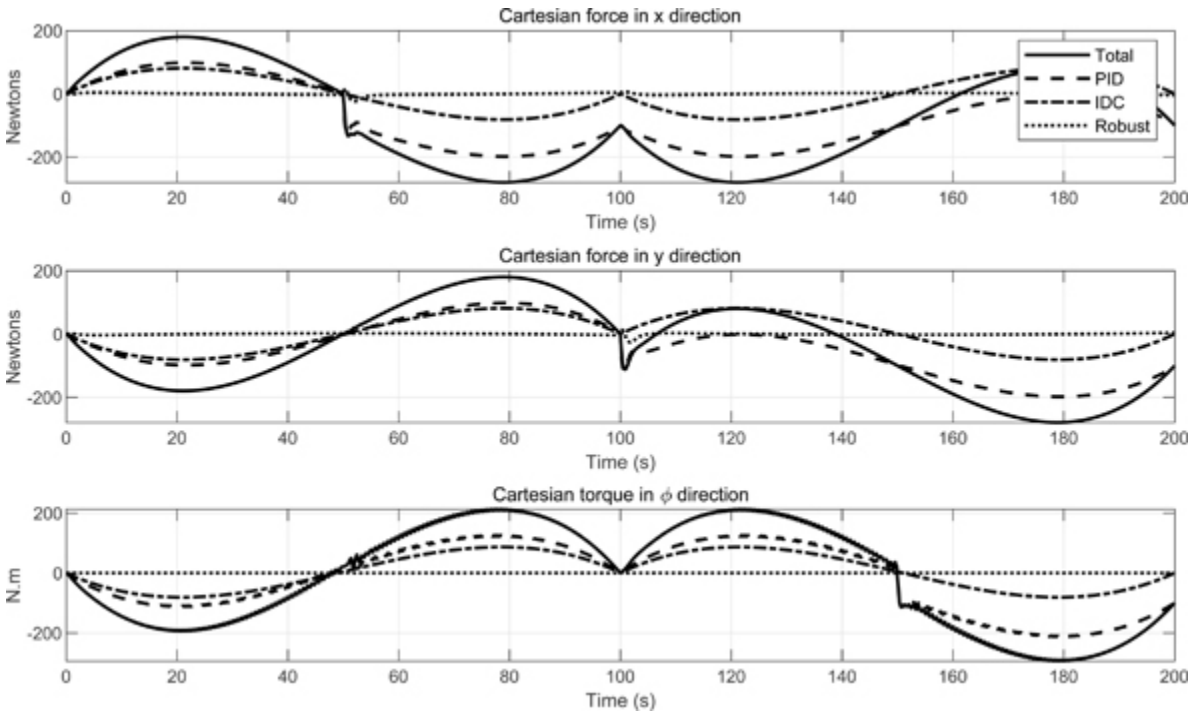

▶ Long Description for Figure 12.14

**FIGURE 12.14**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under robust inverse dynamics with PID control. ⏎

To highlight the role of the robust term in comparison to inverse dynamics and PID control, Figure 12.15 illustrates the details of the Cartesian wrench generated by this controller. The solid line represents the total wrench, the dashed line corresponds to the PID term's contribution, the dash-dotted line shows the contribution from inverse dynamics control, and the dotted line represents the effect of the robust term $\boldsymbol{\delta}_a$. The overall control effort remains comparable to that of inverse dynamics control, as depicted

in Figure 12.13. As shown, during the initial 50 seconds when no disturbance is present, the contributions from both inverse dynamics and PID controls are nearly identical, with the robust term having little effect. However, when a disturbance wrench is applied in a specific direction, the PID term's contribution increases significantly, effectively countering the disturbance and stabilizing the system. Meanwhile, the inverse dynamics control primarily manages the inertial wrenches and tracks the platform's acceleration, complementing the overall control effort.




▶ Long Description for Figure 12.15

**FIGURE 12.15**

Components of the Cartesian wrench of the planar 4RPR manipulator generated by robust inverse dynamics with

PID controller. ⏎

The effect of the robust term becomes noticeable only during the transient periods following the application of disturbances. Although the magnitude of this term is significantly smaller than that of the PID controller, its influence is evident in producing smoother transitions during these times. To further investigate the role of the robust term, simulations were conducted using a PD controller with 80% uncertainty in the system parameters. In this scenario, the magnitude and distribution of the robust term increased substantially. However, the overall performance of the closed-loop system did not match the results reported here. This underscores the vital importance of the linear controller structure, particularly the integral term, in handling disturbances and maintaining robust system performance.

In the simulation results, actuator forces are not presented because the total Cartesian wrenches required are nearly identical to those in pure PID control. As a result, the actuator forces closely resemble those shown in Figure 12.9.

### 12.7.6 Adaptive Inverse Dynamics Control

As previously outlined, adaptive inverse dynamics control initially employs a rough estimate of the dynamic parameters. This estimate is then refined so that the discrepancy between the actual dynamic matrix values and

their estimates gradually approaches zero. Therefore, in an adaptive controller, the dynamic model's structure is assumed to be known, with only the parameters being updated in real-time to adjust the controller settings.

To develop the adaptive inverse dynamics control, the dynamic formulation must be expressed as a linear regression model, represented in Equation 12.46 by $\boldsymbol{Y}(\boldsymbol{X}, \dot{\boldsymbol{X}}, \ddot{\boldsymbol{X}})$ in terms of the adaptive parameters $\boldsymbol{\theta}$. For the planar manipulator in question, the adaptive parameters include the moving platform mass $M$, the moving platform moment of inertia $I_m$, and the cable density $\rho_m$. Therefore, the adaptive parameter vector is defined as:

$$\boldsymbol{\theta} = \begin{bmatrix} M \\ I_m \\ \rho_m \end{bmatrix}. \tag{12.59}$$

Upon closely examining the dynamic formulation of the planar manipulator as described in Equation 8.99, it is evident that the left-hand side of this equation can be expressed in a linear regression form with respect to $\boldsymbol{\theta}$. The linear regression matrix $\boldsymbol{Y}$ consists of three columns: the first two columns correspond to dynamic terms related to the moving platform, while the third column pertains to cable dynamics. The first two columns are derived from the equation $\boldsymbol{M}_p\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}_p$, where the mass matrix $\boldsymbol{M}_p$ and the

gravity vector $\boldsymbol{\mathcal{G}}_p$ are specified in Equations 8.87 and 8.90, respectively. Thus, the first two columns of the linear regression matrix are given by

$$\boldsymbol{Y} = [\boldsymbol{Y}_1 \quad \boldsymbol{Y}_2 \quad \boldsymbol{Y}_3]; \quad \boldsymbol{Y}_1 = \begin{bmatrix} \ddot{x}_G - \boldsymbol{\mathcal{G}} \cdot \hat{\boldsymbol{x}} \\ \ddot{y}_G - \boldsymbol{\mathcal{G}} \cdot \hat{\boldsymbol{y}} \\ 0 \end{bmatrix}; \quad \boldsymbol{Y}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\phi} \end{bmatrix}. \tag{12.60}$$

The last column of the linear regression matrix, which corresponds to cable dynamic terms, can be expressed as:

$$\boldsymbol{Y}_3 = \boldsymbol{M}_{li_\rho} \ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}_{li_\rho} \dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}_{li_\rho}. \tag{12.61}$$

In this derivation, the terms $\boldsymbol{M}_{i_\rho}$, $\boldsymbol{\mathcal{G}}_{i_\rho}$, and $\boldsymbol{\mathcal{C}}_{i_\rho}$ are obtained by excluding the $\rho_m$ term from their corresponding terms $\boldsymbol{M}_i$, $\boldsymbol{\mathcal{G}}_i$, and $\boldsymbol{\mathcal{C}}_i$ as detailed in Equations 8.64, 8.75, and 8.84, respectively. These modified terms are then substituted into Equation 8.98 to generate $\boldsymbol{M}_{li\rho}$, $\boldsymbol{\mathcal{C}}_{li\rho}$, and $\boldsymbol{\mathcal{G}}_{li_\rho}$. This process effectively factors out the parameter $\rho_m$ from the equation, allowing it to be expressed in linear regression form. Consequently, the dynamic equation of the planar manipulator can be written as:

$$\boldsymbol{M}\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}} = \boldsymbol{Y}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, \ddot{\boldsymbol{\chi}}) \cdot \boldsymbol{\theta},$$

(12.62)

where $\boldsymbol{\theta}$ is defined in Equation 12.59 and $\boldsymbol{Y}$ is given in Equation 12.60.

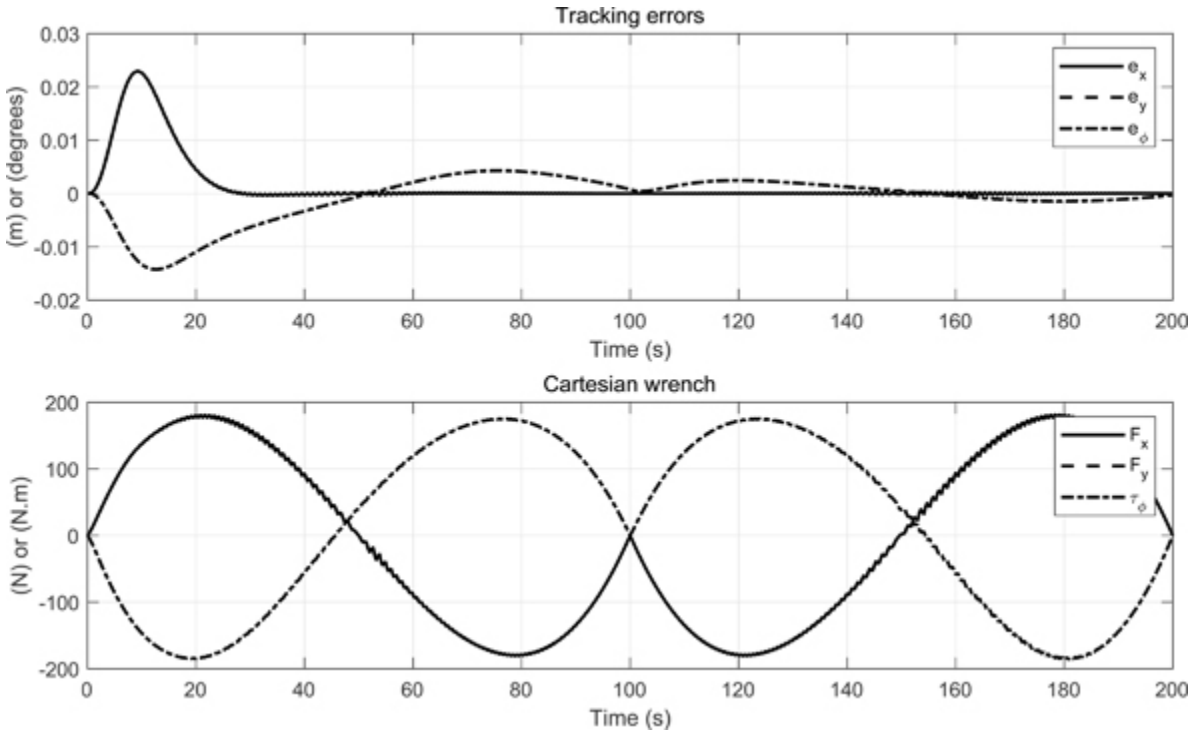
It is important to note that implementing this equation in the control law requires knowledge of the moving platform acceleration vector $\ddot{\boldsymbol{\chi}}$. Since acceleration sensors are typically noisy and costly, and numerical differentiation of velocity signals is not practical due to noise amplification, alternative filtering methods like the Kalman filter can be used to estimate the acceleration vector from position or velocity signals. In the simulations conducted for the planar manipulator, however, $\ddot{\boldsymbol{\chi}}_d$ has been used as a substitute for $\ddot{\boldsymbol{\chi}}$, since, with a well-tuned controller, the difference between these two variables is negligible in practice.

The implementation of adaptive inverse dynamics control is based on the block diagram shown in Figure 12.6. This approach closely resembles regular inverse dynamics control, but with the key difference that the values of the dynamic formulation matrices are updated using the adapted inertial parameters, which are determined online through the adaptation law outlined in Equation 12.56.

In the simulations, the adaptation matrix $\boldsymbol{\Gamma}^{-1}$ is set to an appropriate positive definite matrix, and Equation 12.56 is incorporated into the main routine by augmenting $\hat{\boldsymbol{\theta}}$ with the robot's state variables. To verify the system's robustness against significant uncertainties, a severe 50% perturbation in the dynamic parameters $\boldsymbol{\theta}$ is considered.

To analyze the core of the adaptation law and verify the achievable performance, a PD controller with $\boldsymbol{K}_P = \boldsymbol{I}_{3\times3}$ and $\boldsymbol{K}_d = 2\sqrt{2}\boldsymbol{I}_{3\times3}$ gains is considered, where $\boldsymbol{I}_{3\times3}$ represents a $3 \times 3$ identity matrix. Furthermore, the adaptive controller parameters are set as $\boldsymbol{\Gamma}^{-1} = \text{diag}(40 \cdot M_o^2, 2 \times 10^5 \cdot I_{m_o}^2, 150 \cdot \rho_{m_o}^2)$. This configuration aims to normalize the effects of the three adaptive variables through a thorough examination of various simulations for typical trajectory maneuvers. In this setup, the nominal values of the moving platform mass squared $M_o = 2500$ kg, the moving platform moment of inertia $I_{m_o} = 3.5 \times 10^5$, and the cable density $\rho_{m_o} = 0.215$ are firstly used as the diagonal elements of the adaptation matrix $\boldsymbol{\Gamma}^{-1}$. Then, these values are then adjusted with appropriate gains to ensure a suitable adaptation rate for all parameters. It is important to note that proper tuning of this gain is crucial to achieve the desired parameter adaptation and tracking performance, which is a challenging task in adaptive control.

The closed-loop tracking error for this scenario is depicted in the top chart of Figure 12.16, based on the quintic polynomial trajectory shown in Figure 12.7, while neglecting the effect of input disturbances. The results clearly indicate that adaptive control effectively reduces the steady-state tracking errors after around 20 seconds, once the adaptation transient has settled. The steady-state error is maintained at two millimeters in the $x$ and $y$ directions and at 0.001 degrees in the $\phi$ direction.



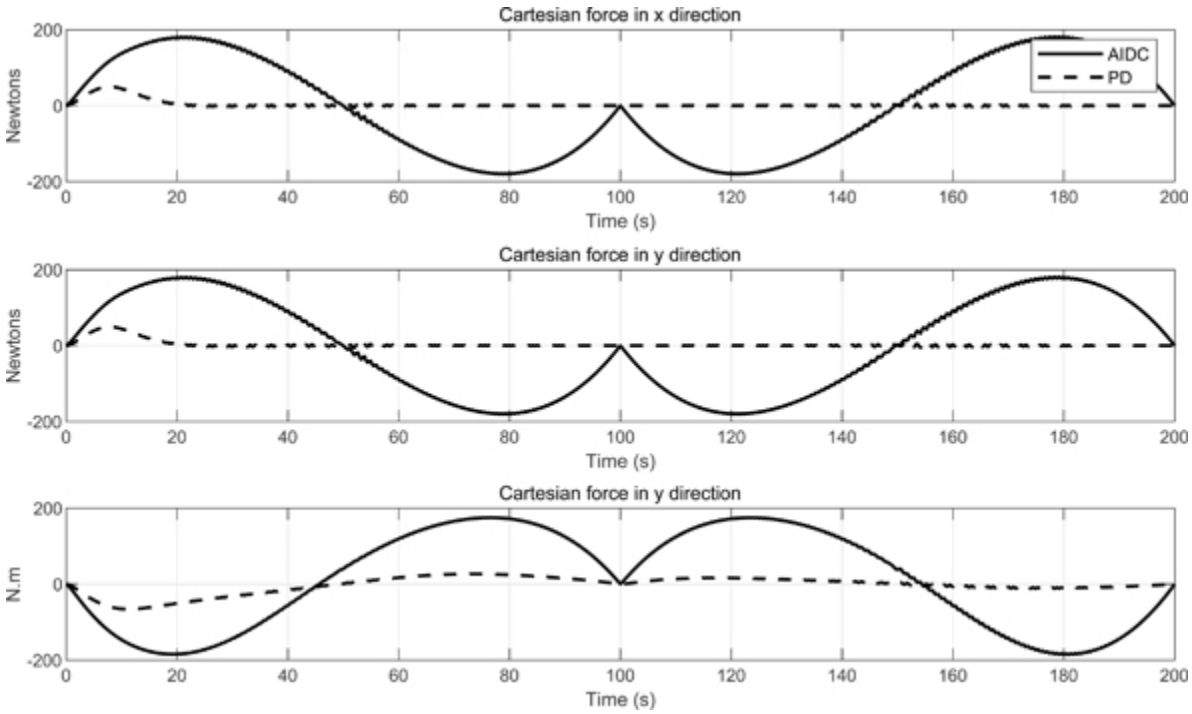

▶ Long Description for Figure 12.16

**FIGURE 12.16**

The closed-loop tracking performance, and the generated Cartesian wrench of the planar 4RPR cable robot under adaptive inverse dynamics with PD control. ⏎

To emphasize the role of the adaptive term compared to PD control, Figure 12.17 shows the details of the Cartesian wrench generated by the controller. The solid line represents the adaptive inverse dynamics wrench, while the dashed line represents the contribution of the PD term. As shown in the figure, during the first twenty seconds of the simulations, when parameter adaptation is reaching steady-state values, the PD control's contribution is comparable to that of the adaptive inverse dynamics.

However, the overall control effort remains similar to that of inverse dynamics control, with the adaptive inverse dynamics being the primary contributor.

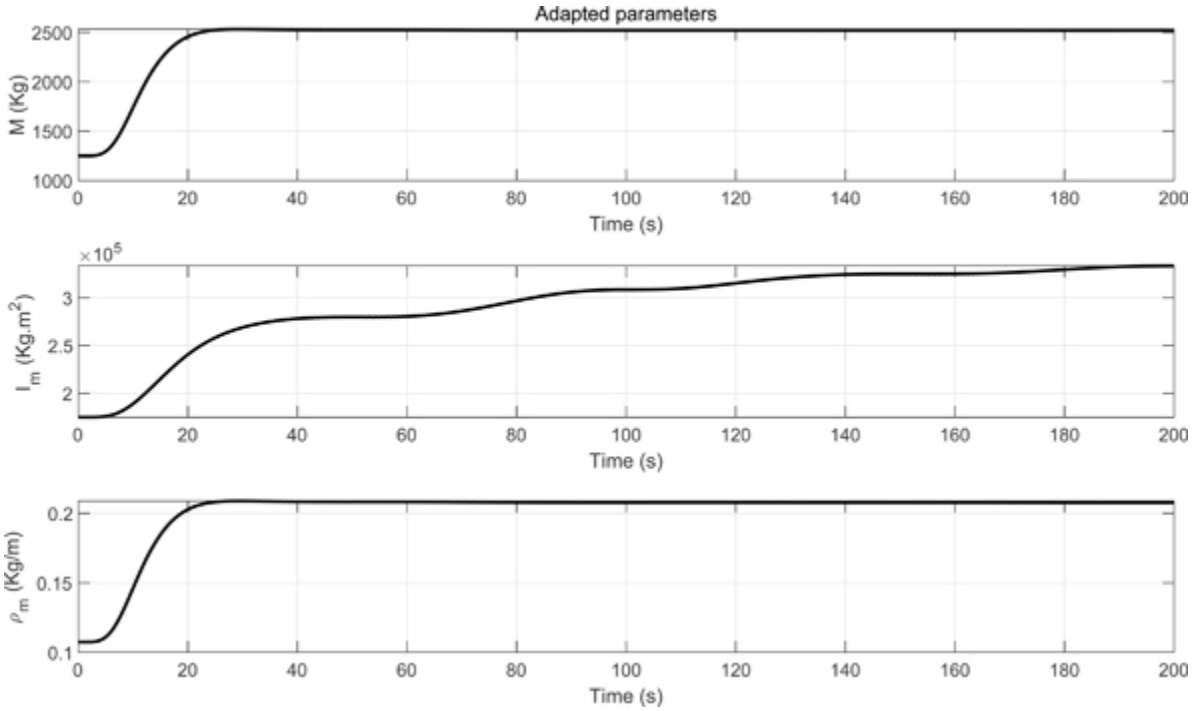



▶ Long Description for Figure 12.17

**FIGURE 12.17**
Components of the Cartesian wrench of the planar 4RPR manipulator generated by adaptive inverse dynamics with PD controller. ⏎

This notable performance is achieved through the rapid convergence of the dynamic parameters $\hat{\boldsymbol{\theta}}$ to their nominal values, as illustrated in Figure 12.18. The adaptation gains are carefully tuned to ensure that the estimates of the inertial parameters, which are initially set to 50% of their nominal values, converge to the true values quickly. The

convergence of parameters $M$ and $\rho_m$ to their nominal values is highly accurate, while the convergence for the $I_m$ parameter is less precise. Although increasing the gain associated with the $I_m$ parameter in $\mathbf{\Gamma}^{-1}$ could enhance performance, it is not done in this simulation due to the potential negative impact on the robust stability of the closed-loop performance.




▶ Long Description for Figure 12.18

**FIGURE 12.18**

Adaptation of the planar 4RPR cable robot parameters under adaptive inverse dynamics with PD control. ⏎

## 12.8 Case Study II: Stewart–Gough Platform

In this section, the motion control of the Stewart–Gough platform is examined in detail, and the control topologies presented in this chapter are implemented on this manipulator. The architecture of this manipulator is described in detail in Section 4.6.1 and depicted in Figure 4.16. As previously mentioned, the spatial motion of the moving platform in this manipulator is generated by six piston–cylinder actuators. A complete kinematic analysis of this manipulator is provided in Section 4.6, and its Jacobian analysis is presented in Section 5.7.

The dynamic analysis of this manipulator, utilizing both the Newton–Euler and virtual work methods, is provided in Section 7.5 and Section 9.6, respectively. The closed-form dynamic formulation employed for the simulations in this section is derived in Section 8.5. In the subsequent sections, various control strategies are applied to this manipulator. These control approaches are introduced progressively, starting from simpler ones and moving toward more comprehensive methods, with their closed-loop performances analyzed and compared.

### 12.8.1 Basic PID Control

The PID controller is the most commonly used controller for trajectory tracking in industrial robots. In this section, as the initial control strategy, we evaluate the tracking performance of the desired trajectory on the Stewart-Gough platform using a decentralized implementation of the PID controller. In the control structure depicted in

Figure 11.12, a set of independent PID controllers is applied in a feedback loop for each error component. For a fully parallel manipulator with six degrees of freedom, like the Stewart-Gough platform, the motion variable in task space is represented by the six-dimensional twist vector:

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{\theta} \end{bmatrix} \tag{12.63}$$

This vector consists of the linear motion, represented by the position vector of the moving platform's center of mass $\boldsymbol{x}_p = \begin{bmatrix} x_p & y_p & z_p \end{bmatrix}^T$, and the orientation of the moving platform, represented by screw coordinates:

$$\boldsymbol{\theta} = \theta \begin{bmatrix} \boldsymbol{s}_x & \boldsymbol{s}_y & \boldsymbol{s}_z \end{bmatrix}^T = \begin{bmatrix} \theta_x & \theta_y & \theta_z \end{bmatrix}^T \tag{12.64}$$

The tracking error is defined as $\boldsymbol{e} = \begin{bmatrix} e_x & e_y & e_z & e_{\theta_x} & e_{\theta_y} & e_{\theta_z} \end{bmatrix}^T$, with the decentralized controller comprising six independent PID controllers, each operating on a distinct error component. The PID controller is represented by the $\boldsymbol{K}_d \boldsymbol{s} + \boldsymbol{K}_p + \boldsymbol{K}_i \frac{1}{s}$ block in Figure 11.12, where $\boldsymbol{K}_d$, $\boldsymbol{K}_p$, and $\boldsymbol{K}_i$ are $6 \times 6$ diagonal matrices representing the gains for the derivative, proportional, and integral controllers, respectively.

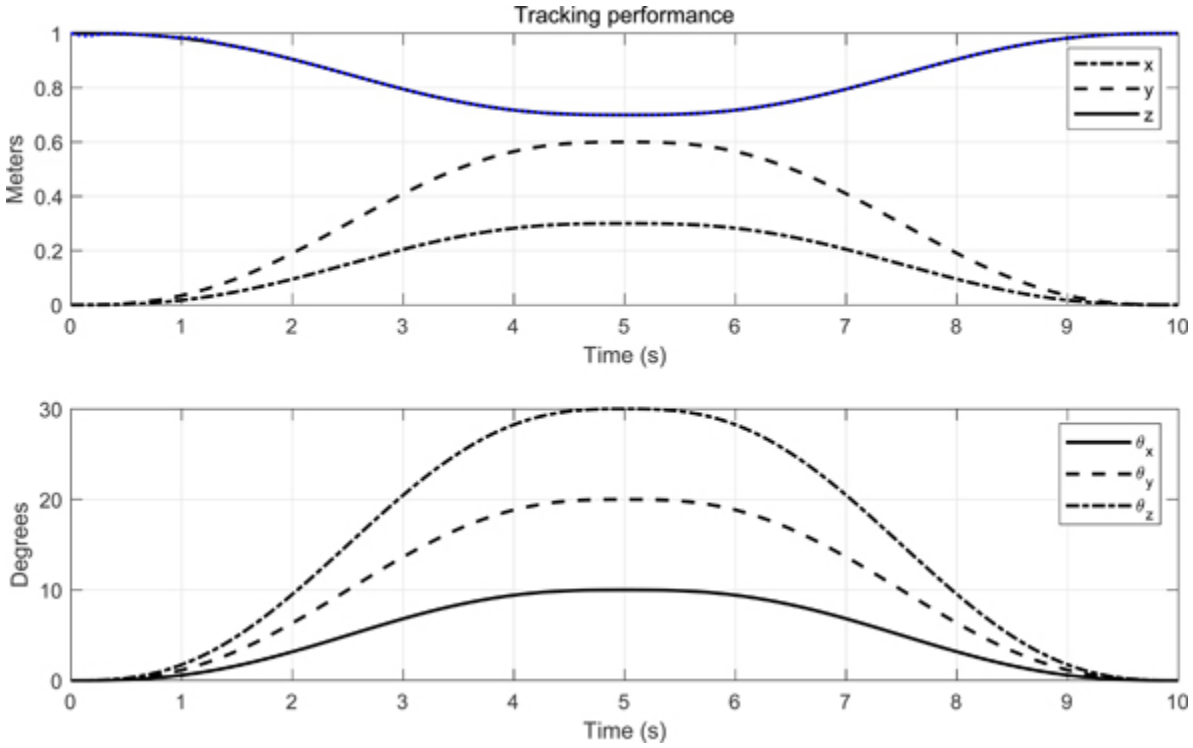
The controller's output is denoted by $\boldsymbol{\mathcal{F}}$, which represents a full-scale wrench in the task space: $\boldsymbol{\mathcal{F}} = \begin{bmatrix} F_x & F_y & F_z & \tau_x & \tau_y & \tau_z \end{bmatrix}^T$. Since the output is defined in the task space, each wrench component directly influences the corresponding tracking error component. The manipulator's overall tracking performance is optimized when high controller gains are used.

In practice, the computed output wrench is translated into actuator forces via the force distribution block. This mapping determines the actuator forces required to generate the specified wrench, which is then applied to the manipulator. As mentioned earlier, for fully parallel manipulators such as the Stewart-Gough platform, the inverse of the Jacobian transpose can be used for this projection in nonsingular configurations.

For the simulations, the geometric and inertial parameters are given in Table 10.2, and all initial state conditions are set to zero, except for the initial height of the moving platform, which is set to $z_o = 1$ m. The gravity vector is assumed to act in the $-z$ direction, $\boldsymbol{\mathcal{G}} = \begin{bmatrix} 0, & 0, & -9.81 \end{bmatrix}^T$, and the disturbance wrench applied to the moving platform consists of unit step wrenches. A $10\,kN$ step disturbance force is applied in the $x$, $y$, and $z$ directions at the first second, followed by a $1\,kN.m$ step disturbance torque in the $x$, $y$, and $z$ directions at the second second.

The desired trajectory for the moving platform is generated using quintic polynomials for $x_d$, $y_d$, $z_d$, as well as

for $\theta_x$, $\theta_y$, and $\theta_z$, as shown in Figure 12.19. This figure illustrates a typical trajectory where the platform moves $0.3\,m$ in the $x$ direction, $0.6\,m$ in the $y$ direction, and $-0.3\,m$ in the $z$ direction. Additionally, the orientation of the moving platform undergoes rotations of 10, 20, and 30 degrees in the $x$, $y$, and $z$ directions, respectively, as depicted in the bottom graph of Figure 12.19. The simulation duration is set to 10 seconds to assess moderate acceleration of the moving platform. The controller gains are set to $\boldsymbol{K}_p = 10^6 \cdot \mathrm{diag}[1,1,1,10,10,10]$, $\boldsymbol{K}_d = 10^5 \cdot \mathrm{diag}[1,1,1,1,1,1]$, and $\boldsymbol{K}_i = 10^7 \cdot \boldsymbol{I}_{6\times 6}$, determined through a series of trial-and-error adjustments.




▶ Long Description for Figure 12.19

**FIGURE 12.19**

The desired and output trajectories of the Stewart–Gough platform in closed loop for decentralized PID in the presence of input disturbance. ⏎

The tracking performance of the moving platform is illustrated in Figures 12.19 and 12.20. As indicated in Figure 12.19, the desired trajectory and the final closed-loop motion of the system are nearly indistinguishable. Additionally, as shown in Figure 12.20, the PID controller for the manipulator could successfully reduce the transient tracking errors to less than $8\,mm$ in position and less than $0.006°$ in orientation. The transient error in translational motion exhibits an overshoot at the start of the motion and when disturbance forces are applied to the moving platform at the first second. This is also reflected in the orientation error at the second second, where the disturbance torque is applied, causing the error to display a jumpy behavior. However, this error is significantly minimized in the steady state, reducing to less than $0.03\,mm$ in position and $5 \times 10^{-4}$ degrees in orientation, thanks to the effective influence of the integral term of the PID controller.

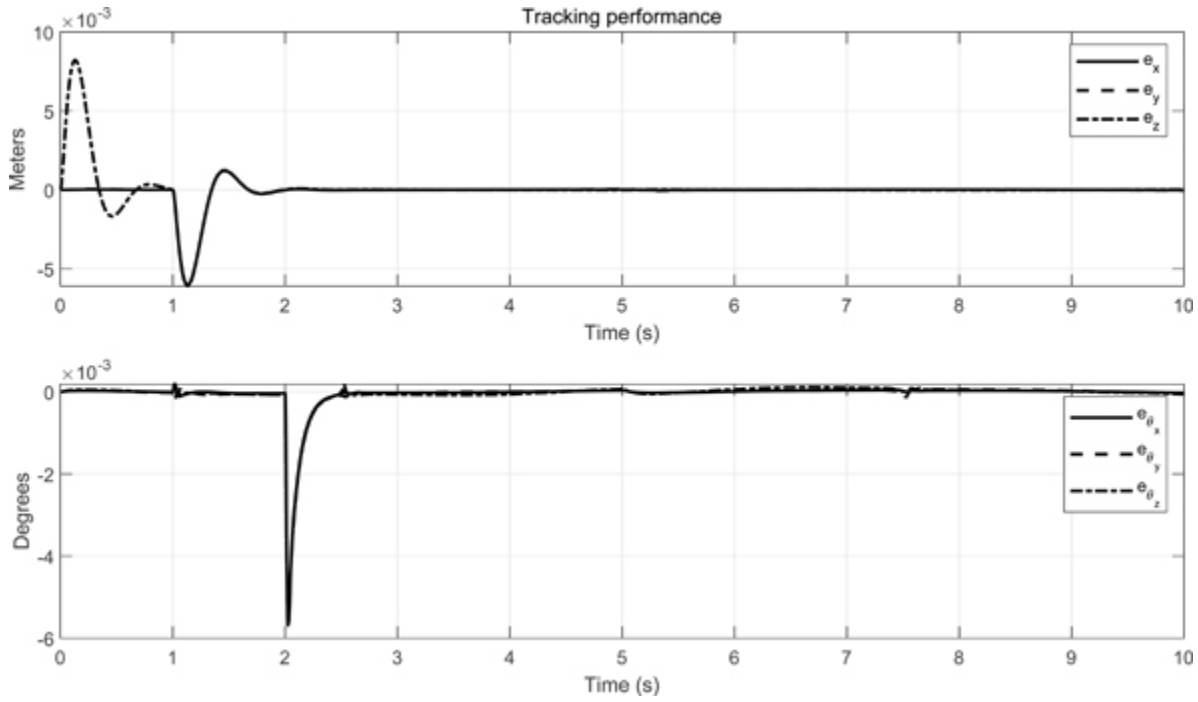
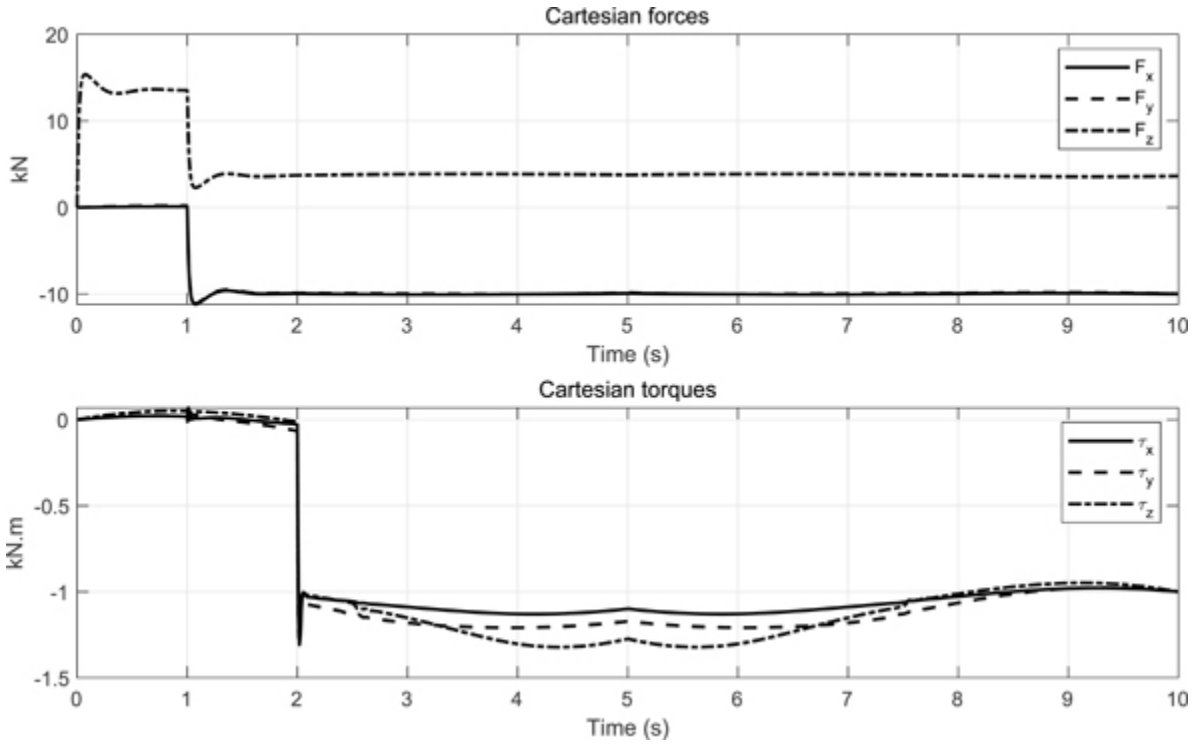



**FIGURE 12.20**
The closed-loop tracking performance of the Stewart–Gough platform using decentralized PID control in the presence of disturbances. ⏎

The Cartesian wrench necessary to achieve the desired tracking performance is depicted in Figure 12.21. The top chart of this figure presents the controller forces in $kN$ across three directions, while the bottom chart displays the controller torques in $kN.m$. The maximum and minimum forces required to generate such motion are limited to 15 and $-12$ Newtons, occurring at the start of the motion and when disturbance forces are applied to the moving platform at the first second. Similarly, the applied torque exhibits a spike of approximately $-1\,kN.m$ at the second second to counteract the disturbances. Since non-vanishing

disturbance forces and torques are applied to the moving platform, the steady-state values of the required wrench components do not approach zero.

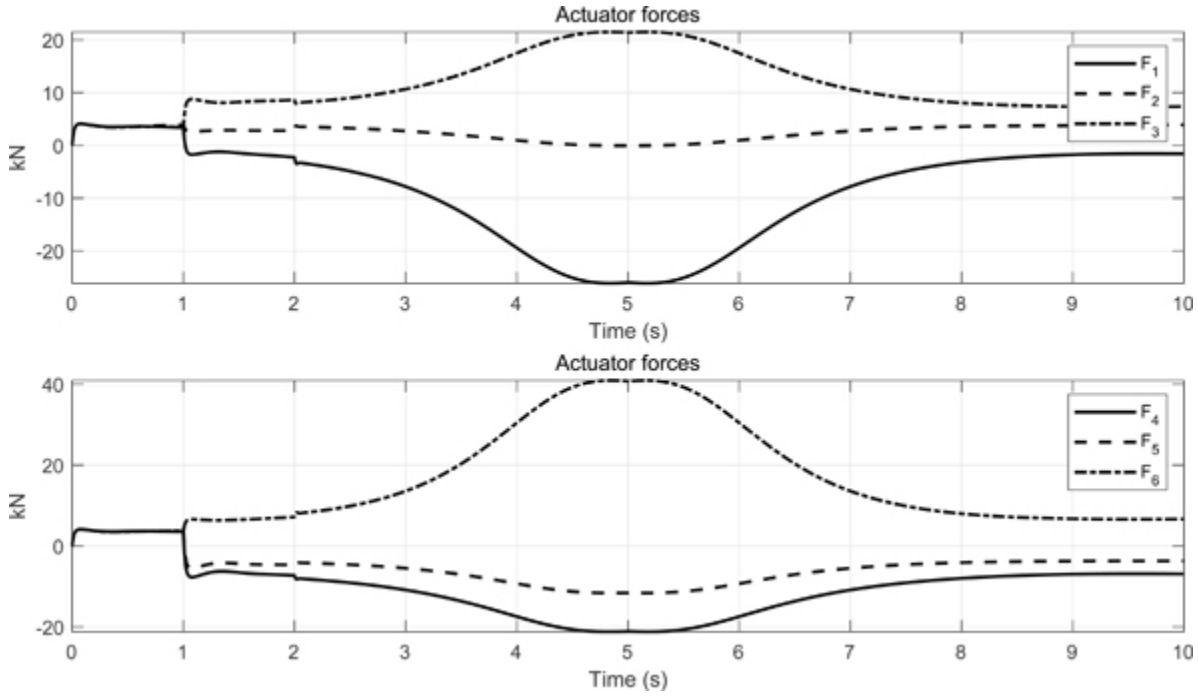



▶ Long Description for Figure 12.21

**FIGURE 12.21**
The Cartesian wrench of the Stewart–Gough platform generated by the PID controller in the presence of disturbances. ⏎

The actuator forces resulting from the projection of the PID control action into joint space are shown in Figure 12.22. As depicted in this figure, the actuator forces required to complete this task are relatively high, ranging from $-20$ to $40\,kN$. A noticeable change in the required actuator efforts occurs at the first second when the

disturbance force is applied to the moving platform, while the maximum control effort is observed at the fifth second when the highest acceleration is needed for the trajectory. The distribution of the actuator forces aligns with the positive translational motion in the $x$ and $y$ directions, and negative motion in the $z$ direction. The PID controller generates high forces to accommodate the significant accelerations necessary for following the desired trajectory.




▶ Long Description for Figure 12.22

**FIGURE 12.22**

The actuator forces generated by the PID control of the Stewart–Gough platform in the presence of disturbances. ⏎

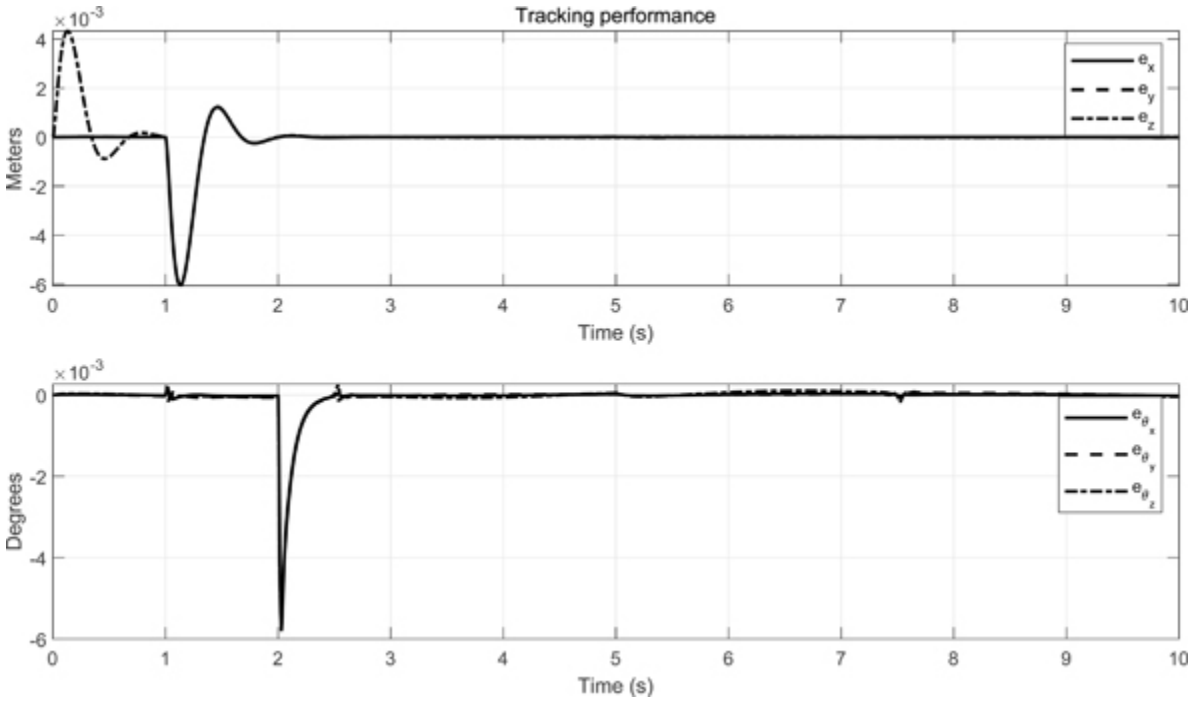
## 12.8.2 Feed-Forward Control

As discussed in the previous section, the tracking performance of the PID controller implemented in task space is highly effective in the presence of external disturbances, particularly at a steady state. However, the transient response of the tracking error was not as favorable when using the PID controller alone. To evaluate the impact of incorporating a feed-forward wrench, denoted by $\mathcal{F}_{ff}$, into the PID controller structure, the simulation on the Stewart–Gough platform is repeated with the same initial conditions and disturbances. The feed-forward wrench is generated from the dynamic model of the manipulator in the task space, as represented by Equation 12.1. To compute this term, the robot's dynamic formulation, along with its kinematic and dynamic parameters, must be known.

In practice, exact information on the dynamic matrices is often unavailable. Therefore, in simulations, estimates of these matrices are used, incorporating a significant 50% perturbation in all kinematic and inertial parameters. For comparison purposes, the PID controller gains are set to carefully adjusted values of $\boldsymbol{K}_p = 10^6 \cdot \mathrm{diag}[1,1,1,10,10,10]$, $\boldsymbol{K}_d = 10^5 \cdot \boldsymbol{I}_{6\times 6}$, and $\boldsymbol{K}_i = 10^7 \cdot \mathrm{diag}[1,1,1,10,10,10]$, consistent with the decentralized PID control case.

Figure 12.23 illustrates the closed-loop tracking performance of the Stewart–Gough platform with feed-forward control for the typical trajectory shown in Figure 12.19. Comparing this figure to the PID control results in

Figure 12.20, it is clear that the feed-forward action improves the transient tracking errors twofold, reducing them to approximately 4,*mm* in position while maintaining the orientation error at 0.006°, as before.



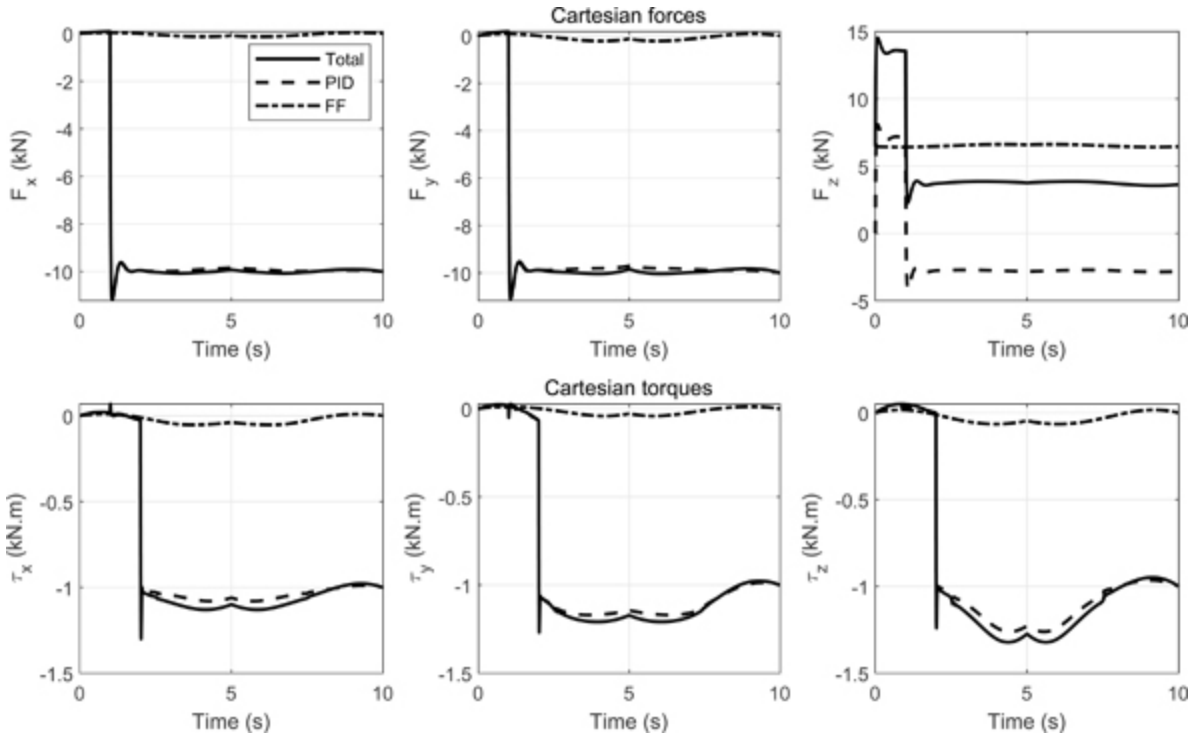

▶ Long Description for Figure 12.23

**FIGURE 12.23**

The closed-loop tracking performance of the Stewart–Gough platform with a feed-forward controller in the presence of disturbances. ⏎

To assess the real contribution of the feed-forward term in improving the tracking error, observe the Cartesian wrench produced by the controller, where the contributions of both the PID and feed-forward terms are highlighted, as shown in Figure 12.24. In this figure, the Cartesian forces

in the three positional directions are displayed in the top charts, while the Cartesian torques are shown in the bottom charts. Additionally, in each figure, the total Cartesian wrench component is represented by a solid line, the PID contribution by a dashed line, and the feed-forward term by a dash-dotted line.




▶ Long Description for Figure 12.24

**FIGURE 12.24**

The Cartesian wrench of the Stewart–Gough platform generated by feed-forward controller in the presence of disturbance. ⏎

In most cases, the PID term is dominant, except for the force component in the $z$ direction, where the feed-forward term has a significant impact. This is primarily due to the

gravitational force acting on the moving platform, which is largely compensated by the feed-forward term. If we had full knowledge of the controller, and the 50% modeling uncertainty was not introduced in the simulations, the feed-forward term would be the sole contributor in this scenario. However, since we deliberately assumed incomplete knowledge of the kinematic and dynamic parameters, both the PID controller and the feed-forward term contributed almost equally to the tracking performance. This is particularly evident in the twofold improvement of the positional $z$ component tracking performance, as shown in Figure 12.23.

The actuator forces are also derived using the inverse Jacobian transpose for this simulation. Since the total Cartesian wrenches required to perform the task are very similar to those in the PID controller case, the resulting actuator forces are likewise quite similar, as shown in Figure 12.22, and are therefore not presented separately here.

### 12.8.3 Inverse Dynamics Control

The overall configuration of the inverse dynamics control is outlined in the block diagram depicted in Figure 12.2. This configuration employs both internal and external feedback loops. Within the internal loop, the wrench $\mathbf{f}_{fl}$ is computed at each instant based on feedback from task variables and is then employed as a compensatory term on the robot. In the external loop, we consider using a PID controller, to

compare the result to the previous simulations. Consequently, the exerted control effort on the system is equivalent to,

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \boldsymbol{\mathcal{F}}_{fl},$$

(12.65)

where it can be detailed as:

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}),$$

(12.66)

$$\boldsymbol{a} = \ddot{\boldsymbol{\chi}}_d + \boldsymbol{K}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_p\boldsymbol{e}_x + \boldsymbol{K}_i \int_0^t \boldsymbol{e}_x(\tau)d\tau.$$

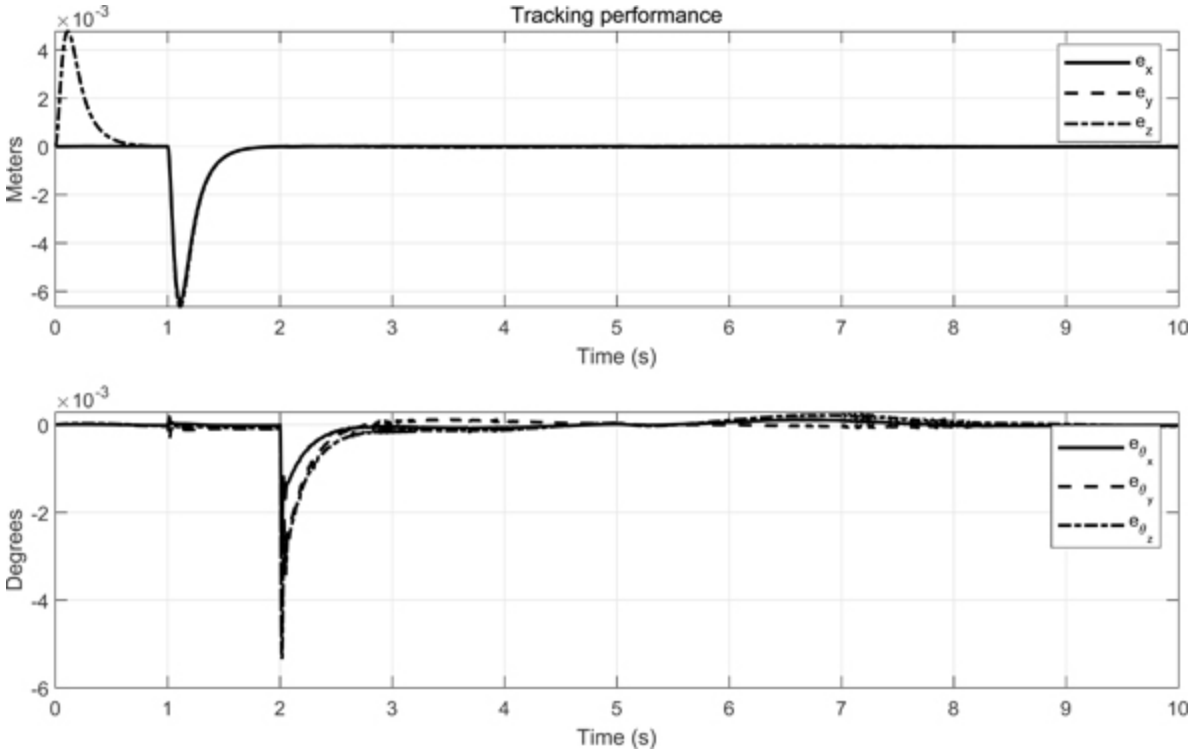
(12.67)

Note that to generate the inverse dynamics control terms, the dynamic formulation of the robot, along with its kinematic and dynamic parameters, is required. In practice, exact knowledge of the dynamic matrices is often unavailable, and as before, a significant 50% perturbation in all kinematic and inertial parameters is applied in the simulations.

Considering that, in this control method, the PID controller gain matrices are multiplied by the robot's inertia matrix $\hat{\boldsymbol{M}}(\boldsymbol{\chi})$, the values of these gains should be

adjusted differently compared to the previous control methods to enable a fair comparison. In simulating this control approach, by employing a less trial and error process compared to the previous scenario, the PID controller gain matrices can be tuned more effectively as follows: $\boldsymbol{K}_p = 2 \times 10^3 \cdot \mathrm{diag}[1,1,1,50,50,50]$, $\boldsymbol{K}_d = 10^2 \cdot \mathrm{diag}[1,1,1,2,2,2]$, and $\boldsymbol{K}_i = 10^4 \cdot \mathrm{diag}[1,1,1,50,50,50]$. By utilizing these gain matrices, the necessary matching for tracking error and control effort, such as the PID controller with feed-forward, can be achieved. By this means, comparing the results of these methods will be more meaningful.

The outcomes of the simulation regarding the tracking error in the closed-loop system are shown in Figure 12.25. This figure illustrates the tracking performance of the Stewart–Gough platform with an inverse dynamics controller for the typical trajectory depicted in Figure 12.19. As clearly observed, the inverse dynamics controller behaves similarly to the feed-forward controller and improves the transient tracking errors of the pure PID controller by a factor of two. Consequently, the tracking errors are reduced to approximately $4\,mm$ in position and $0.006°$ in orientation. The transient error in translational motion shows an overshoot at the start of the motion and when disturbance forces are applied to the moving platform in the first second. However, in the steady state, this error is greatly minimized, reducing to less than $0.02\,mm$ in position and $2 \times 10^{-4}$ degrees in orientation.

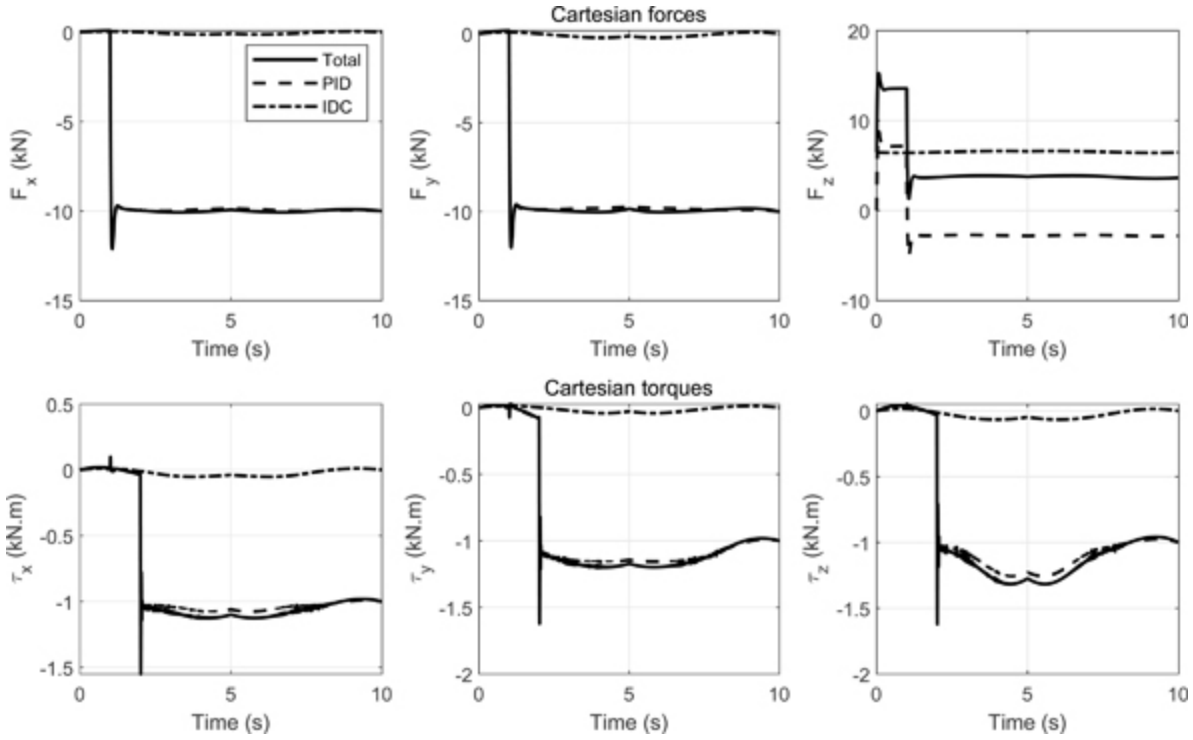



▶ Long Description for Figure 12.25

**FIGURE 12.25**
The closed-loop tracking performance of the Stewart–Gough platform for inverse dynamics controller in the presence of disturbance. ⏎

To gain a deeper understanding of the significance of inverse dynamics control, it is helpful to analyze the wrench components produced by the controller and applied to the moving platform, as well as those exerted on the robot's actuators, illustrated in Figure 12.26. In this figure, the contributions of both the PID and inverse dynamics components are highlighted. The total Cartesian wrench is represented by a solid line, the PID contribution by a

dashed line, and the inverse dynamics component by a dash-dotted line.




▶ Long Description for Figure 12.26

**FIGURE 12.26**

The Cartesian wrench of the Stewart–Gough platform generated by inverse dynamics controller in the presence of disturbance. ⏎

Similar to the case of feed-forward control, the PID term is dominant in most instances due to the disturbances applied to the system, with the PID controller effectively mitigating their effects. However, inverse dynamics control significantly influences the force component in the $z$ direction, primarily due to the gravitational force acting on the moving platform, which is largely compensated by the

inverse dynamics control. Since the knowledge of the kinematic and dynamic parameters was incomplete, with a 50% modeling uncertainty introduced in the simulations, both the PID controller and the inverse dynamics control contributed almost equally to the overall tracking performance.

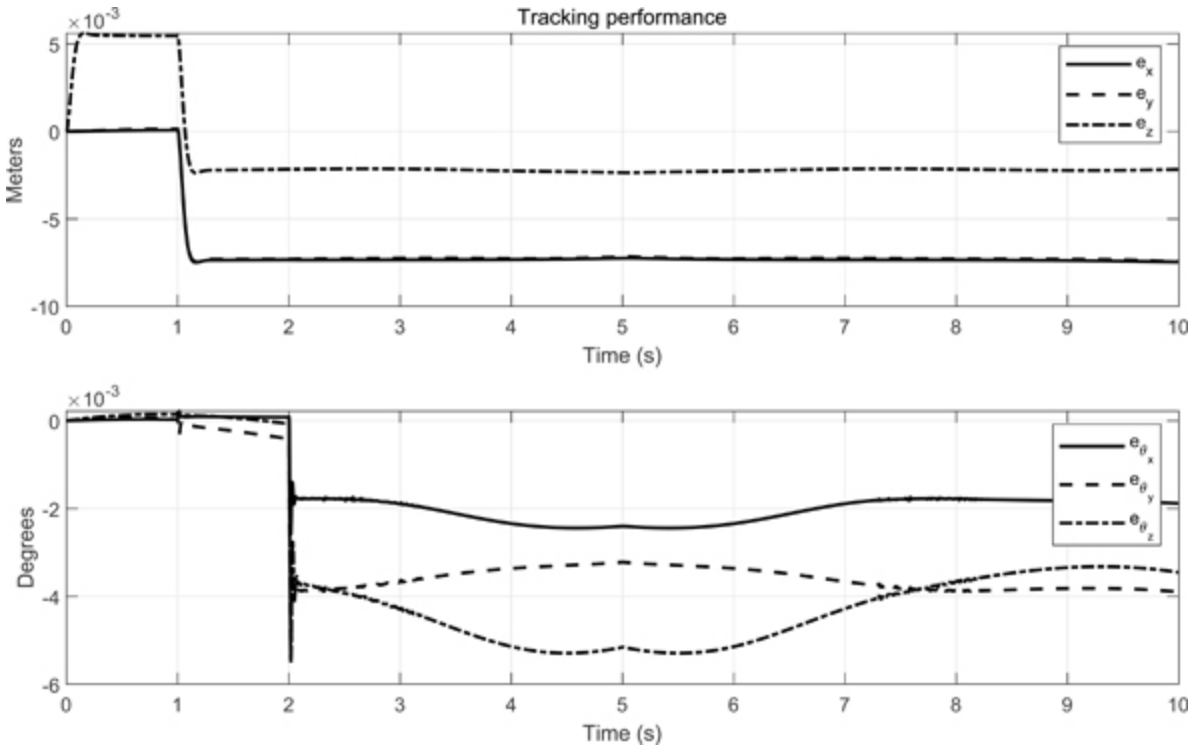
The actuator forces are also calculated using the inverse Jacobian transpose for this simulation. Since the total Cartesian wrenches required to perform the task are very similar to those in the PID controller case, the resulting actuator forces are similarly close, as depicted in Figure 12.22, and are thus not presented separately here.

### 12.8.4 Partial Feedback Linearization

Inverse dynamics control offers several practical advantages. However, as discussed before, applying this method requires complete knowledge of the dynamic formulation matrices. To create the simplest feasible IDC, we revisit the partial linearization IDC introduced in Section 12.4. This basic form of the IDC control wrench $\mathcal{F}$ consists of two components, as described in Equation 12.11. The first term, $\mathcal{F}_{pd}$, is generated by a PD control based on the motion error $e_x$, outlined in Equation 12.12, while the second term represents only the gravity vector of the manipulator, $\mathcal{G}(\mathcal{X})$, as described in Equation 12.13. In this implementation, the computationally demanding Coriolis and centrifugal terms are omitted from the feedback.

To facilitate comparison with previously evaluated controller structures, a 50% perturbation has been applied to all kinematic and inertial parameters. The PD controller gains are set to $\boldsymbol{K}_p = 2 \times 10^3 \cdot \text{diag}[1, 1, 1, 50, 50, 50]$ and $\boldsymbol{K}_d = 10^2 \cdot \text{diag}[1, 1, 1, 2, 2, 2]$, similar to the settings in the inverse dynamics control framework, while no integral term is used in this controller.

The tracking performance of the closed-loop system with the partially linearized inverse dynamics controller for this configuration is illustrated in Figure 12.27. As illustrated in this figure, the partial feedback linearization controller effectively reduces tracking errors to less than $7.5\,mm$ in position and under $0.006°$ in orientation. However, since no integral term is included in the linear control section, the steady-state errors persist over time. It is noteworthy that if a PID controller had been implemented, the steady-state behavior would likely mirror that of the previous cases.



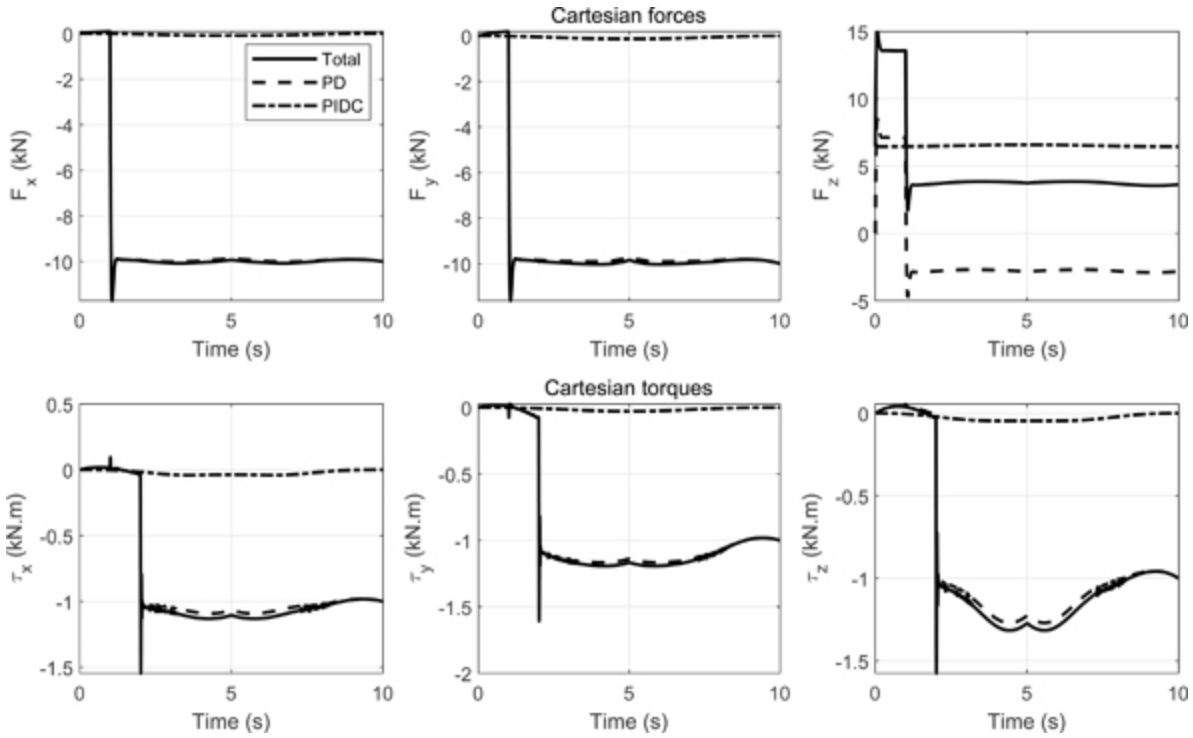

▶ Long Description for Figure 12.27

**FIGURE 12.27**

The closed-loop tracking performance of the Stewart–Gough platform using a partial feedback linearization controller with PD control and gravity compensation. ⏎

The Cartesian wrench required to achieve the desired tracking performance is shown in Figure 12.28. Similar to earlier cases, the maximum and minimum forces required to generate this motion are constrained to 15 and $-12$ Newtons, respectively, peaking at the start of the motion and when the disturbance forces are applied to the moving platform at the first second. Likewise, the applied torque spikes to around $-2\,kN.m$ at the second second to counteract disturbances. Due to the continuous presence of

non-vanishing disturbance forces and torques on the platform, the steady-state values of the wrench components do not reach zero.




▶ Long Description for Figure 12.28

**FIGURE 12.28**

The Cartesian wrench of the Stewart–Gough platform generated by the partial feedback linearization controller using PD control and gravity compensation. ⏎

To better understand the role of inverse dynamics control, it is useful to examine the wrench components produced by the controller. Both the PD and partial feedback linearization contributions are shown, with the PD term represented by a dashed line and the other by a dash-dotted line. As with full inverse dynamics control, the PD

term dominates most cases due to its ability to attenuate the disturbances. However, the gravity compensation term significantly impacts the $z$-direction force component, which is primarily compensated through inverse dynamics control. Given the incomplete knowledge of kinematic and dynamic parameters, with a 50% modeling uncertainty factored into the simulations, both the PID controller and the inverse dynamics control contributed almost equally to the tracking performance.

Although the tracking performance differs notably in this case compared to the previous ones, particularly in steady-state, the total Cartesian wrenches needed to execute the task are quite similar, and thus the resulting actuator forces are likewise comparable. For this reason, they are not shown separately here but can be seen in Figure 12.22.

This simulation confirms the analysis provided in Section 12.4, demonstrating that even without incorporating the Coriolis and centrifugal matrix in the feedback loop, the PD control structure with gravity compensation still yields satisfactory performance. Moreover, it verifies that the skew-symmetric property of the dynamic formulation matrices offers enough assurance that partial feedback linearization does not cause tracking divergence in practice. This shows that even when the closed-loop dynamics are not fully linearized, the controller remains robust and effective for the Stewart–Gough platform.

### 12.8.5 Robust Inverse Dynamics Control

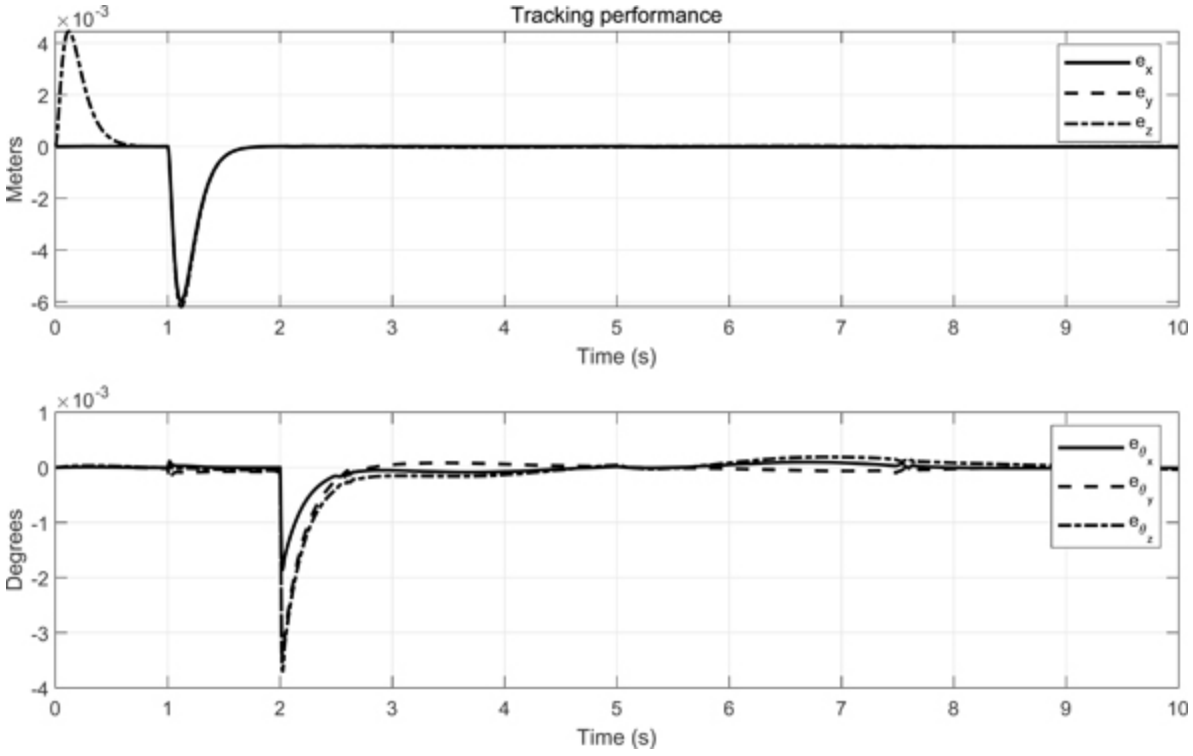
The performance of inverse dynamics control hinges on the complete linearization of nonlinear dynamic terms through feedback. However, this requirement is rarely met in practice due to inevitable modeling uncertainties, which can limit the practical effectiveness of IDC in parallel robots. To mitigate such uncertainties in the closed-loop performance, a robust term, denoted as $\boldsymbol{\delta}_a$, is introduced, as discussed in Section 12.5.2. This term is added to the standard IDC control structure to enhance robustness.

Figure 12.4 presents the robust IDC implementation block diagram for a parallel robot. In this diagram, the corrective term $\boldsymbol{\delta}_a$ is incorporated as a wrench in task space. The formulation of this corrective term is given in Equation 12.39, where the vector $\boldsymbol{v}$ is defined in Equation 12.34 to adjust the control for better compensation of the uncertainties.

In the following simulations, the robust inverse dynamics approach is applied to the Stewart–Gough platform. To evaluate the controller's performance under significant uncertainties, a 50% perturbation is introduced in all kinematic and inertial parameters. For comparison with previously tested controller structures, the PID controller gains are set to $\boldsymbol{K}_p = 2 \times 10^3 \cdot \mathrm{diag}[1,1,1,50,50,50]$, $\boldsymbol{K}_d = 10^2 \cdot \mathrm{diag}[1,1,1,2,2,2]$, and $\boldsymbol{K}_i = 10^4 \cdot \mathrm{diag}[1,1,1,50,50,50]$, consistent with the IDC simulations. Additionally, the robust controller parameters are chosen as $\rho = 10^3$ and $\epsilon = 500$, after careful evaluation

of different simulations for the typical trajectory shown in Figure 12.19.

The tracking performance of the closed-loop system using the robust inverse dynamics controller for this configuration is shown in Figure 12.29. As depicted in the figure, the positional tracking errors of the robust inverse controller closely resemble those of the standard inverse dynamics controller. However, a slight improvement is noticeable in the orientation tracking errors. This controller effectively reduces positional tracking errors to less than $4\,mm$ and orientation errors from 0.006° to below 0.004°.

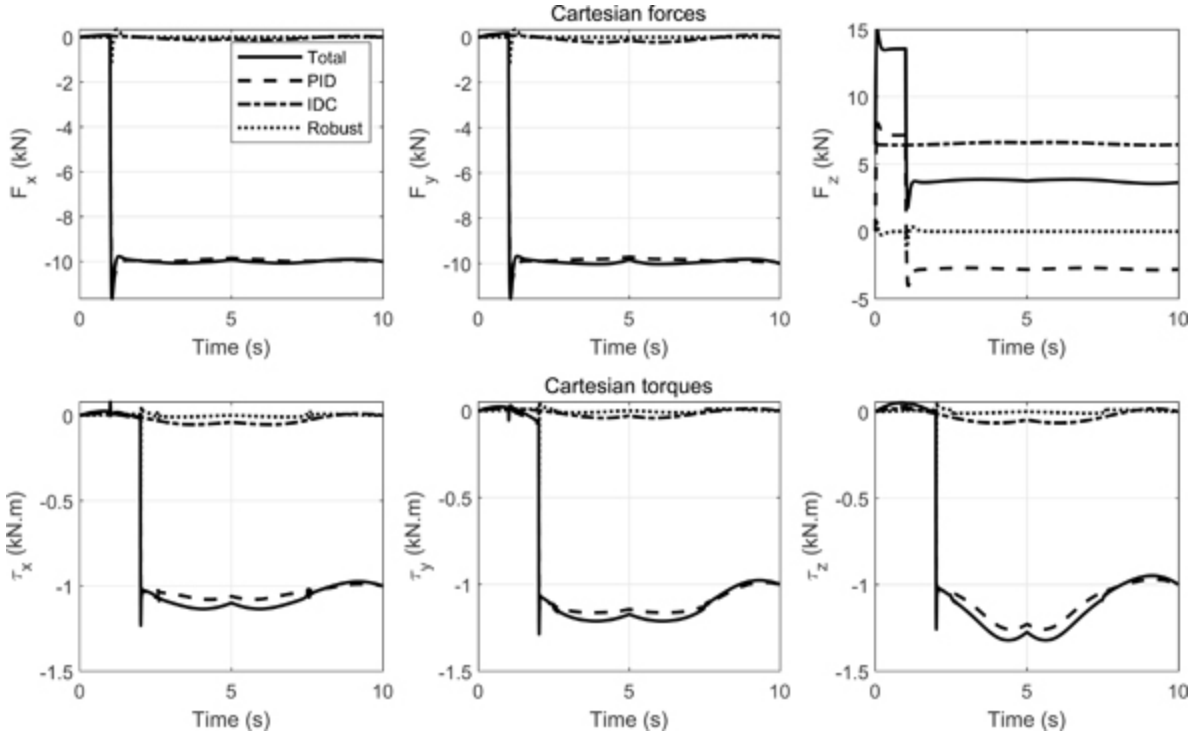



▶ Long Description for Figure 12.29

**FIGURE 12.29**

The closed-loop tracking performance of the Stewart–Gough platform using a robust inverse dynamics controller in the presence of disturbances. ⏎

Similar to previous observations, the transient errors in translational motion exhibit an overshoot at the beginning of the motion and when external disturbance forces are applied to the moving platform, particularly at the first second. Nonetheless, once the system reaches steady-state, these errors are significantly reduced, with positional errors falling below $0.02\,mm$ and orientation errors dropping from $2 \times 10^{-4}$ degrees to $5 \times 10^{-5}$ degrees.




▶ Long Description for Figure 12.30

**FIGURE 12.30**

The Cartesian wrench generated by the robust inverse dynamics controller for the Stewart–Gough platform in the presence of disturbances.

To better appreciate the impact of inverse dynamics control, it is insightful to examine the wrench components generated by the controller and applied to the moving platform, as well as those acting on the robot's actuators, as illustrated in Figure 12.26. This figure distinguishes the contributions of the PID controller, inverse dynamics, and robust control components. The overall Cartesian wrench is shown as a solid line, with the PID contribution as a dashed line, the inverse dynamics term as a dash-dotted line, and the robust component as a dotted line.

As with standard inverse dynamics control, the PID term predominates in most cases, largely due to disturbances affecting the system. The PID controller is highly effective in counteracting these disturbances. However, the influence of the robust inverse dynamics control becomes apparent during the transient phase, particularly when disturbances are applied to the moving platform. As shown in the figure, at the first and second seconds, the robust term is activated, visibly impacting the transient tracking errors in both translational and rotational components.

This relatively small control effort significantly enhances the tracking performance, particularly in reducing orientation errors. It smooths the transient response and decreases the maximum orientation error from 0.006° to 0.004°. Moreover, in the steady-state phase, the orientation

tracking error improves from $2 \times 10^{-4}$ degrees to $5 \times 10^{-5}$ degrees.

In this simulation, the actuator forces are calculated using the inverse Jacobian transpose. Since the total Cartesian wrenches required for the task closely resemble those observed in the PID controller case, the resulting actuator forces are also very similar. As shown in Figure 12.22, the forces are nearly identical, so a separate presentation of the actuator forces is not provided here.

## Problems

12.1 Consider the five-bar mechanism depicted in Figure 4.27 and discussed in Problems 4.1 and 5.1. In this problem, we will take into account the dynamic formulation, geometric and inertial parameters, and the typical trajectory outlined in Problem 10.1. The PD and PID controllers for this robot are outlined in Problem 11.1.

a. Improve the performance of the PID controller by implementing a feed-forward control scheme. Simulate the closed-loop behavior while considering a 50% perturbation in all geometric and inertial parameters. Tune the controller gains to achieve suitable tracking performance with limited control effort.

b. Design an inverse dynamics controller for the mechanism while accounting for a 50% perturbation in all geometric and inertial parameters. Simulate the closed-loop performance and tune the controller gains to ensure suitable tracking performance with limited control effort.

c. Implement a partial feedback linearization approach in the inverse dynamics controller, considering a 50% perturbation in all geometric and inertial parameters. Simulate the closed-loop performance and use the controller gains established in the full inverse dynamics controller.

d. Plot and analyze the motion trajectories, tracking errors, and the Cartesian and actuator forces versus time for the different controllers. Compare the results to evaluate the effectiveness of each control strategy.

12.2 For the five-bar mechanism depicted in Figure 4.28 and discussed in Problems 4.2 and 5.2, consider the dynamic formulation that includes the geometric and inertial parameters, as well as the typical trajectory from Problem 10.2. The PD and PID controllers for this robot are outlined in Problem 11.2. Apply the approach outlined in Problem 12.1 to this mechanism.

12.3 Consider the planar pantograph mechanism illustrated in Figure 4.29 and described in Problems 4.3 and 5.3. Utilize the dynamic formulation,

including the geometric and inertial parameters, as well as the typical trajectory provided in Problem 10.3. The PD and PID controllers for this robot are detailed in Problem 11.3. Implement the approach detailed in Problem 12.1 for this mechanism.

12.4 Examine the planar three-degrees-of-freedom 3RR Rparallel manipulator illustrated in Figure 4.30 and discussed in Problems 4.4 and 5.4. Use the dynamic formulation that includes the geometric and inertial parameters, along with the typical trajectory from Problem 10.4. The PD and PID controllers for this robot are detailed in Problem 11.4. Apply the control methods described in Problem 12.1 to this manipulator.

12.5 Review the planar three-degrees-of-freedom 4RR Rparallel manipulator shown in Figure 4.31 and discussed in Problems 4.5 and 5.4. Apply the dynamic formulation, incorporating the geometric and inertial parameters along with the typical trajectory specified in Problem 10.5. The PD and PID controllers for this robot are outlined in Problem 11.5. Implement the control techniques described in Problem 12.1 for this manipulator.

12.6 Examine the spatial parallel manipulator depicted in Figure 4.32 and outlined in Problems 4.6 and 5.6. Utilize the dynamic formulation, including the geometric and inertial parameters, as well as the

typical trajectory provided in Problem 10.6. The PD and PID controllers for this robot are detailed in Problem 11.6. Apply the control methods described in Problem 12.1 to this manipulator.

12.7 Analyze the 3UPU parallel manipulator depicted in Figure 4.33 and described in Problems 4.7 and 5.7. Apply the dynamic formulation that includes both the geometric and inertial parameters, along with the typical trajectory outlined in Problem 10.7. The PD and PID controllers for this robot are outlined in Problem 11.7. Use the control techniques specified in Problem 12.1 to control this manipulator.

12.8 Review the manipulator detailed in Reference [51] and illustrated in Figure 4.34, as well as in Problems 4.8 and 5.8. Apply the dynamic formulation, including the geometric and inertial parameters and the typical trajectory provided in Problem 10.8. The PD and PID controllers for this robot are detailed in Problem 11.8. Employ the control techniques described in Problem 12.1 for this manipulator.

12.9 Examine the three-degrees-of-freedom 3RPS manipulator shown in Figure 4.35 and detailed in Problems 4.9 and 5.9. Apply the dynamic formulation, incorporating the geometric and inertial parameters, along with the typical trajectory from Problem 10.9. The PD and PID controllers for this robot are detailed

in Problem 11.9. Use the control techniques outlined in Problem 12.1 to manage this manipulator.

12.10 Review the three-degrees-of-freedom 3PR Ctranslational parallel manipulator depicted in Figure 4.35 and detailed in Problems 4.10 and 5.10. Utilize the dynamic formulation, incorporating the geometric and inertial parameters, as well as the typical trajectory from Problem 10.10. The PD and PID controllers for this robot are detailed in Problem 11.10. Apply the control strategies outlined in Problem 12.1 to this manipulator.

12.11 Analyze the Delta robot illustrated in Figure 4.38 and discussed in Problems 4.11 and 5.11. Utilize the dynamic formulation that includes the geometric and inertial parameters, as well as the standard trajectory from Problem 10.11. The PD and PID controllers for this robot are detailed in Problem 11.11. Implement the control techniques described in Problem 12.1 for this robot.

12.12 Consider the Quadrupteron robot illustrated in Figure 4.39 and outlined in Problems 4.12 and 5.7. Apply the dynamic formulation, incorporating the geometric and inertial parameters along with the trajectory specified in Problem 10.12. The PD and PID controllers for this robot are detailed in Problem 11.12. Implement the control techniques described in Problem 12.1 for this robot.

12.13 Revisit the five-bar mechanism as described in Problem 12.1.

a. For this mechanism, design a robust inverse dynamics controller in the task space, and implement the required force distribution in terms of the mechanism's Jacobian matrix. Develop a program in MATLAB® to simulate the closed-loop tracking performance of the mechanism. Consider a 50% perturbation in all geometrical and inertial parameters, and tune the controller gains to achieve suitable tracking performance with a limited control effort.

b. Plot and analyze motion trajectories, tracking errors, and the Cartesian and actuator forces versus time for this controller. Compare the results to those obtained in Problem 12.1.

12.14 Revisit the five-bar mechanism as described in Problem 12.2, and reiterate Problem 12.13 for this mechanism.

12.15 Reexamine the planar pantograph mechanism as outlined in Problem 12.3, and revisit Problem 12.13 in the context of this mechanism.

12.16 Reconsider the planar three-degrees-of-freedom 3RR Rparallel robot as described in Problem 12.4, and reconsider Problem 12.13 in the context of this parallel robot.

12.17 Reexamine the planar three-degrees-of-freedom 4RR Rparallel robot as described in Problem 12.5, and reevaluate Problem 12.13 within the context of this parallel robot.

12.18 Revisit the spatial parallel manipulator as detailed in Problem 12.6, and reconsider Problem 12.13 within the context of this parallel manipulator.

12.19 Reexamine the 3UPU parallel manipulator as outlined in Problem 12.7, and reassess Problem 12.13 in the context of this parallel manipulator.

12.20 Reevaluate the 3UPS parallel manipulator as described in Problem 12.8, and reconsider Problem 12.13 within the framework of this parallel manipulator.

12.21 Reassess the 3RPS parallel manipulator as detailed in Problem 12.9, and revisit Problem 12.13 in the context of this parallel manipulator.

12.22 Revisit the 3PR Cparallel manipulator as presented in Problem 12.10, and review Problem 12.13 within the scope of this parallel manipulator.

12.23 Reassess the Delta robot as detailed in Problem 12.11, and reconsider Problem 12.13 in relation to this parallel manipulator.

12.24 Reevaluate the Quadrupteron robot as described in Problem 12.12, and revisit Problem 12.13 concerning

this parallel robot.

# 13

# *Force Control*

DOI: 10.1201/9781003590408-13

This chapter examines various approaches and techniques used in the force control of robotic systems. It starts with an introduction that establishes a foundation for comprehending force control principles. Next, the chapter investigates different force control configurations, such as cascade control and force feedback in both the outer and inner loops. The concept of compliance control is then examined, highlighting the similarities and distinctions to position control. The process of implementing compliance control is then explored, beginning with single-degree-of-freedom and later expanding into a more complex multivariable control strategy.

The study advances with the direct force control method, detailing the design of cascade control, while the force feedback loop is considered in the outer loop. An in-depth performance analysis of closed-loop systems is performed for a multivariable system. Additionally, hybrid position-

force control techniques are discussed, demonstrating their efficiency in managing force-related tasks.

To illustrate the real-world application of these methods, the chapter presents two case studies on the force control methodologies discussed, applied to a planar cable robot and the Stewart-Gough platform. Both compliance and direct force control methods are explored within this context, showcasing their utility and effectiveness. Moreover, the inclusion of problems at the chapter's conclusion encourages readers to apply what they have learned, thereby deepening their understanding and skill in implementing these advanced control techniques in practical scenarios. Overall, the chapter provides a detailed overview of force control principles, methods, and their real-world applications in robotic systems.

## 13.1 Introduction

In the preceding two chapters, an in-depth exploration of various control topologies and schemes for the motion control of parallel robots was conducted. These controllers are particularly effective when the robot's moving platform accurately tracks a desired motion trajectory, ensuring smooth and precise movements in tasks where no external forces interfere. However, in many real-world applications, the moving platform is not operating in isolation. It frequently interacts with its surrounding environment,

necessitating a carefully controlled force, or interaction wrench, to manage the contact forces during these interactions.

Applications such as precision machining, grinding, or micro-assembly—where the robot's platform may be required to exert or resist forces while maintaining accuracy—highlight the need for this type of control. In such scenarios, the contact wrench serves as a more reliable indicator of the interaction state than simply monitoring the position and orientation of the moving platform. Accurately controlling these forces is critical for ensuring successful task execution without causing damage or losing precision.

The force control problem, therefore, revolves around calculating the actuator forces required to generate a prescribed wrench (a combination of force and torque) on the manipulator's moving platform while maintaining its desired motion. This becomes crucial when the robot is in contact with rigid or unyielding surfaces, where the interaction forces can significantly influence the robot's performance. In such cases, controlling these interacting forces is essential for tasks like grinding, where applying the right amount of force could increase the quality of the work. The force control algorithms introduced in this chapter, along with those in the next, provide solutions to these complex challenges. These algorithms modify position trajectories dynamically based on the sensed wrench, ensuring that the robot responds appropriately to

the forces exerted by or against it, thereby achieving an optimal balance between motion and force.

When a robot is equipped with high-gain controllers and a rigid structure, as is often the case with parallel manipulators, the motion control scheme generates significant actuator forces to minimize tracking errors. This means that if a purely motion-based control system is used and the manipulator comes into contact with an external environment, the robot does not detect the presence of the obstacle. Instead, it continues applying increasingly high forces in an attempt to reduce tracking errors. In such situations, if the robot's environment is less rigid than the manipulator, it risks damaging or breaking the object it is interacting with. On the other hand, if the environment is more rigid than the robot, the excessive forces could cause damage to the robot itself, compromising its internal structure. Either scenario is highly undesirable, especially when the robot is interacting with a human, as it could result in serious injury.

In these circumstances, motion control schemes alone are inadequate. The manipulator must be equipped with additional sensors beyond the standard motion detectors to detect the interaction and regulate the forces being applied. Various types of wrench sensors have been developed for this purpose, with six-degrees-of-freedom sensors being the most commonly used. These sensors provide comprehensive information about the interacting forces and torques, enabling precise control during

interaction. Additionally, joint torque or link force measurement units can be employed to determine how the interacting forces are projected into the joint space, providing further insight into how to manage the robot's response.

The integration of wrench sensors, whether in the task space or joint space, expands the possibilities for implementing various force control topologies in manipulators. However, incorporating these sensors does not mean that traditional motion sensors, commonly used in motion control schemes, become unnecessary. Motion sensors and their associated control topologies remain essential because controlling the manipulator's motion is still one of the main goals. Yet, when force control schemes are required, motion measurements alone are typically insufficient. The type and configuration of the wrench sensors used in the manipulator allow for the development of different force control topologies, which are discussed in detail in Section 13.2. In many of these topologies, both motion and wrench measurements are employed simultaneously, with a hierarchical structure that prioritizes the main control objective.

While force control schemes are generally essential only when a parallel manipulator engages with a rigid environment, in the case of cable-driven parallel manipulators, this requirement becomes even more critical, extending to pure motion control schemes as well. As mentioned earlier, when cables are used for force

transmission, there is an inherent limitation that the control system must address: maintaining positive tension in all cables at all times. Cables, unlike rigid links, can only pull, not push, which means they must always remain in tension to function correctly. If any cable loses tension, the manipulator loses control of that part of the structure, potentially leading to instability or imprecise movements. This creates a unique challenge in designing effective control strategies for cable-driven manipulators.

Theoretical approaches have been developed to address this issue by utilizing the manipulator's redundant degree(s) of freedom to ensure that all cables are in tension throughout the operation. These methods are elaborated in Section 6.6, where the principles behind using redundancy in the manipulator's configuration to maintain cable tension are discussed in detail. However, implementing these techniques in real-world applications is more complex. In practice, it is usually necessary to measure the actual forces being transmitted through the cables and incorporate these force measurements into the control system. This enables real-time adjustments to ensure that tension is maintained and that the manipulator performs accurately without risking cable slack.

## 13.2 Controller Topologies

In the force control of a parallel manipulator, it is assumed that the controller calculates the necessary actuator forces and torques to ensure that the robot's motion adheres to a specified trajectory when in contact with a rigid environment. Additionally, the controller adjusts the motion trajectories based on the measured wrench or the relationships between force and motion. We can represent the motion variable as the generalized coordinate of the moving platform, defined earlier $\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x}_p & \boldsymbol{\theta} \end{bmatrix}^T$, where the linear motion is denoted by $\boldsymbol{x}_p = \begin{bmatrix} x_p & y_p & z_p \end{bmatrix}^T$, and the orientation of the moving platform is expressed using screw coordinates as $\boldsymbol{\theta} = \theta \begin{bmatrix} \boldsymbol{s}_x & \boldsymbol{s}_y & \boldsymbol{s}_z \end{bmatrix}^T = \begin{bmatrix} \theta_x & \theta_y & \theta_z \end{bmatrix}^T$.

Furthermore, let us examine the force/torque variable in the previously defined task space, represented as the wrench $\boldsymbol{\mathcal{F}} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{n} \end{bmatrix}^T$. Here, the force is expressed as $\boldsymbol{F} = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$, while the torque exerted on the moving platform is denoted by $\boldsymbol{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$. The motion variable in the joint space can be represented as $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_m \end{bmatrix}^T$, and the actuator forces in the joint space are indicated by $\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \dots & \tau_m \end{bmatrix}^T$, where $m$ signifies the number of actuators. It is important to note that the robot Jacobian matrix facilitates the necessary mapping between the rate of the motion twist and the joint velocities through $\dot{\boldsymbol{q}} = \boldsymbol{J} \dot{\boldsymbol{\chi}}$, and it also relates the projection of the moving platform twists to the actuator forces vector via $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$.

Next, let's examine the general closed-form dynamic formulation of a parallel robot as outlined in Equation 11.1, which has been adapted to account for the scenario where the robot interacts with the environment and experiences disturbance wrenches on the moving platform:

$$\boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) + \boldsymbol{\mathcal{F}}_e = \boldsymbol{\mathcal{F}} = \boldsymbol{J}^T\boldsymbol{\tau}. \tag{13.1}$$

In this equation, $\boldsymbol{M}(\boldsymbol{\mathcal{X}})$ represents the mass matrix, which can be obtained from the manipulator's kinetic energy as described in Equation 8.19. The term $\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})$ refers to the Christoffel matrix provided in Equation 8.46, while $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$ denotes the gravity vector as outlined in Equation 8.24. Additionally, $\boldsymbol{\mathcal{F}}_e$ signifies the interacting wrench between the moving platform and the environment at the center of mass, $\boldsymbol{\tau}$ indicates the actuator force vector, and $\boldsymbol{\mathcal{F}}$ represents the projection of the actuator force vector into the task space.

To develop a force controller for parallel manipulators, it is essential to first present various controller topologies designed for this purpose, followed by different techniques for controller design based on these topologies. The upcoming sections will introduce the controller topologies, after which several controller design methods will be outlined.

Control topology refers to the arrangement of the control system used to calculate the actuator forces and torques

from the measurements. In the context of force control for parallel manipulators, the controller determines the required actuator forces and torques to follow a desired position and orientation trajectory when the robot interacts with a stiff environment. Furthermore, the controller adjusts the motion trajectories according to the sensed wrench or the relationship between force and motion.

In a general scenario, the desired motion of the moving platform can be represented by the desired generalized coordinate of the manipulator, referred to as $\boldsymbol{\mathcal{X}}_d$. This variable shares the same dimensions and structure as the manipulator's motion variable $\boldsymbol{\mathcal{X}}$. To achieve this motion within a closed-loop framework, it is essential to measure the output motion of the manipulator using an instrumentation system. Since the final motion of the moving platform is the primary focus in the motion control of parallel manipulators, it is preferable to utilize instrumentation systems that can directly provide the motion variable $\boldsymbol{\mathcal{X}}$ with the necessary accuracy from the outset. Such systems would measure the final position and orientation of the moving platform. However, common sensors for motion measurement typically include linear or angular position sensors installed on the actuated joints, which allow for the direct measurement of the joint motion variable $\boldsymbol{q}$.

In a force control scheme, the desired interacting wrench between the moving platform and the environment may be significant. This quantity can be represented as $\boldsymbol{\mathcal{F}}_d$, which

shares the same dimensions and structure as the manipulator wrench $\mathcal{F}$. To execute this control task within a closed-loop framework, it is essential to measure the output wrench of the manipulator using an instrumentation system. Given that the final interacting force of the moving platform is crucial in the force control of parallel manipulators, it is advantageous to employ instrumentation systems that can provide the interacting wrench $\mathcal{F}$ with the necessary accuracy right from the start. Commercial measurement devices designed to measure a six-degrees-of-freedom wrench are available and are known as six-degrees-of-freedom wrench sensors. These instruments are commonly used in the wrists of serial manipulators for applications involving force interaction with the environment. The same devices can also be utilized on the moving platform of a parallel robot where the final end effector of the manipulator is mounted.

While numerous commercial six-degrees-of-freedom wrench sensors are available on the market, they are typically more costly than individual joint force or torque measurement units. A six-degrees-of-freedom wrench sensor can be employed in situations where precise measurement and control of the interaction force are essential. Another option for force measurement is the direct assessment of actuator forces or torques. Many commercial linear actuators are designed with embedded force measurement capabilities, and numerous servo motors include torque measurement units in their

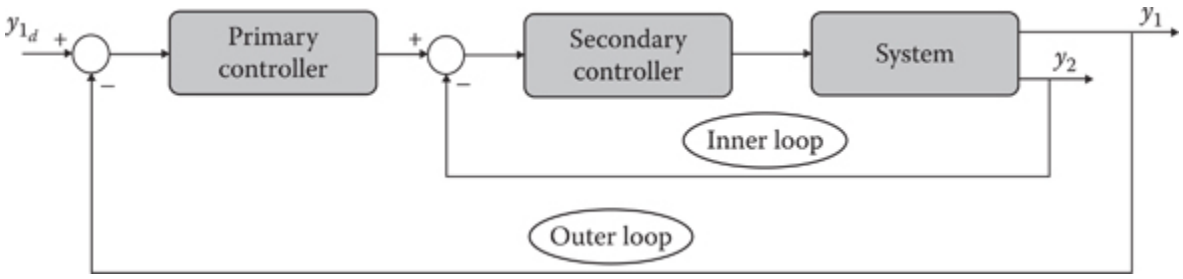
assembly. Consequently, in some applications, it may be advantageous to utilize direct measurements of actuator forces and torques for feedback control. In such instances, let $\boldsymbol{\tau}_m$ represent the measured variable. Depending on the measurement devices integrated into the robot's structure, various control topologies are recommended for force control schemes, which will be addressed in the following sections.

### 13.2.1 Cascade Control

In a typical force control scheme, the primary goal is to track the interacting wrench between the moving platform and the environment. However, it is important to recognize that motion control of the robot during its interaction with the environment is also a secondary objective. Once the contact between the robot's moving platform and the environment is broken, motion control then becomes the primary focus. To achieve these two objectives with varying priorities within a single control system, a hierarchy of two feedback loops is often employed in practice [229]. This control topology is referred to as cascade control, which is applied when multiple measurements are available alongside a single primary control variable. This approach effectively addresses the main objectives while allowing for the consideration of secondary objectives through careful design of the system controller(s).

Cascade control is implemented by nesting the control loops, as depicted in Figure 13.1. The control system

illustrated in this figure consists of two control loops, where the outer loop is designated as the primary loop, and the inner loop is termed the secondary loop, which serves to achieve a secondary objective within the closed-loop system. It is also possible to have a cascade control system with additional nested loops.




▶ Long Description for Figure 13.1

**FIGURE 13.1**

Block diagram of a closed-loop system with cascade control. ⏎

The system's performance can be enhanced by incorporating additional measurement variables into its structure. In a state-feedback configuration, all state variables that characterize the system's dynamics are utilized for feedback. In this scenario, the control system can be regarded as a cascade control system with $n$ nested loops, where $n$ represents the number of system states. However, in industrial applications, it is common to implement only two nested loops, particularly in the force control schemes for parallel manipulators. In these cascade control schemes, the measured variables include both

motion and the interacting wrench. As mentioned previously, these quantities can be measured in either the task space or the joint space, which leads to the recommendation of different control topologies depending on the specific set of measurement variables used.

To enhance the performance of the control system for a specific objective, it is crucial to select appropriate variables for the internal and external feedback loops and to design suitable controllers for each feedback system. Although the specifics may vary across the different topologies outlined in the following sections, some general principles apply to creating an effective cascade control system. A fundamental concept in cascade control design is the ideal scenario where the inner loop is configured so tightly that the secondary loop functions like a perfect servo, responding swiftly to internal control commands. This approach is widely employed in various applications, where an almost-perfect actuator is designed to respond to the requested commands using inner control feedback.

The design criteria for the inner loop emphasize achieving a high gain, ensuring that the time response of the secondary variable is at least five times faster than that of the primary variable. This configuration allows the inner loop to effectively counteract disturbances and unmodeled dynamics within the internal feedback structure [229]. Furthermore, it is essential to establish a well-defined relationship between the primary and secondary variables

to ensure alignment in the objectives pursued in both the primary and secondary loops.

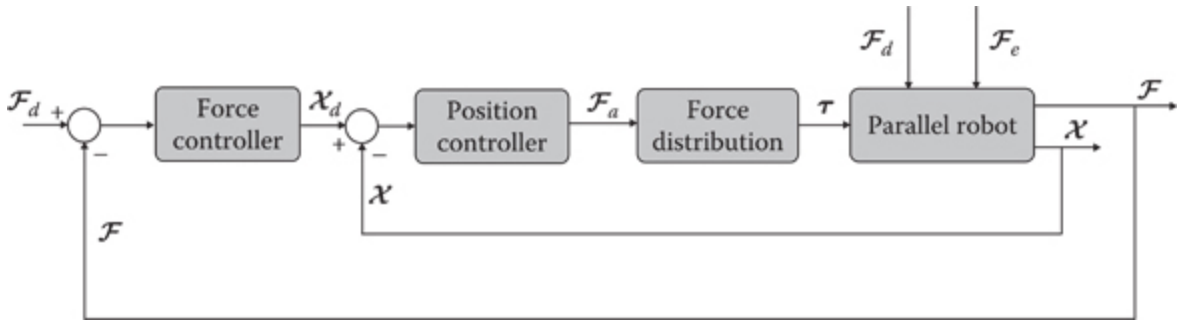
### 13.2.2 Force Feedback in the Outer Loop

Consider the force control schemes where force tracking is the primary objective. In this scenario, it is recommended that the outer loop of the cascade control structure is designed using wrench feedback, while the inner loop relies on position feedback. Given that various types of measurement units can be employed in parallel robots, different control topologies can be developed to implement this cascade structure.

For instance, when both the wrench and position variables are measured in the task space configuration, the moving platform wrench (applied at the center of mass) $\mathcal{F}$ can be measured using a six-degrees-of-freedom wrench sensor, while the motion variable $\mathcal{X}$ may be captured by a measurement unit. Although these measurements can be relatively costly, when both are implemented in the task space, the closed-loop system can achieve accurate force and positioning performance for the manipulator.

Figure 13.2 illustrates the initial cascade control topology for the system in which both measured variables are located in the task space. These variables include the resulting wrench measured at the center of mass of the moving platform, denoted as $\mathcal{F}$, and the motion variable of the moving platform, represented by $\mathcal{X}$. The inner loop is designed with position feedback, while the outer loop is

based on force feedback. This configuration means that the primary controller serves as a force controller to ensure the achievement of force-tracking objectives, whereas the secondary objective focuses on position tracking.




▶ Long Description for Figure 13.2

**FIGURE 13.2**
Initial cascade topology of force feedback control: position is managed in the inner loop while force is controlled in the outer loop. The moving platform wrench $\mathcal{F}$ and the motion variable $\mathcal{X}$ are measured in the task space. ⏎

As depicted in Figure 13.2, the controller operates in the task space, with the output from the force controller block being fed into the motion controller. This output can be interpreted as the desired motion trajectory generated for the inner loop.

The output of the motion controller is also designed in the task space. To convert this output into implementable actuator forces $\tau$, a force distribution block is included in this topology. As discussed in the motion control topologies, this force distribution block is responsible for mapping the

generated wrench in the task space, denoted as $\boldsymbol{\mathcal{F}}_a$, to its corresponding actuator forces and torques $\boldsymbol{\tau}$.

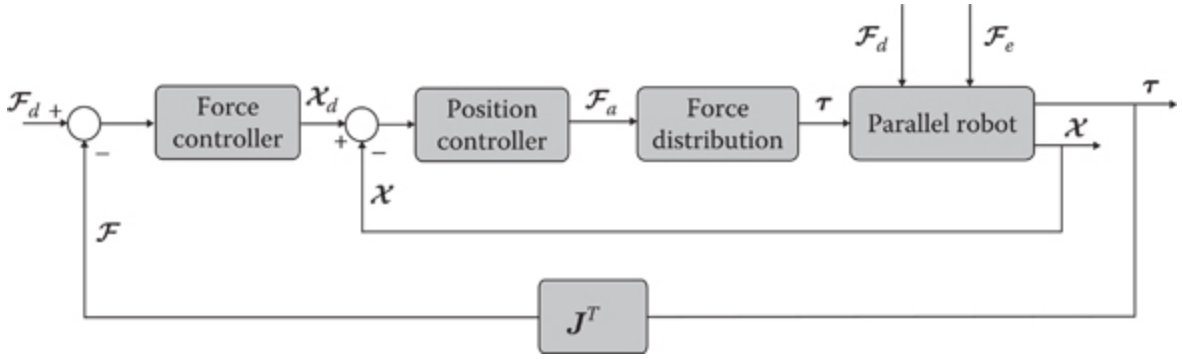
In a fully parallel manipulator, such as the Stewart–Gough platform, where the number of actuators equals the number of degrees of freedom, this mapping can be achieved using the Jacobian transpose of the manipulator, $\boldsymbol{J}^T$. Given that this mapping for parallel robots is described by the equation $\boldsymbol{\mathcal{F}}_a = \boldsymbol{J}^T\boldsymbol{\tau}$ for a fully parallel manipulator, the actuator forces can be derived using $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}_a$ at nonsingular configurations.

In cases where the parallel manipulator is not fully parallel, the force distribution block may require additional computations. For such situations, redundancy resolution techniques discussed in Section 6.6 can be employed as part of the force distribution routine.

Alternative force control topologies can be proposed based on the variations in position and force measurements. Consider a general parallel manipulator comprising several limbs with both passive and active joints, as illustrated in Figure 1.11. As previously mentioned, it is typically easier to either measure the actuator forces and torques $\boldsymbol{\tau}$ directly or derive them from joint space measurements.

Assuming that the motion variable in the task space, $\boldsymbol{\mathcal{X}}$, is still measured, the topology shown in Figure 13.3 is recommended for the control system. In this configuration, the measured actuator force and torque vector $\boldsymbol{\tau}$ is mapped to its corresponding wrench in the task space using the

Jacobian transpose mapping, given by the relationship $\mathcal{F} = J^T \tau$. This mapping allows the controller structure to remain in the task space, and the force and position controllers can be designed using the same techniques developed for the initial topology.

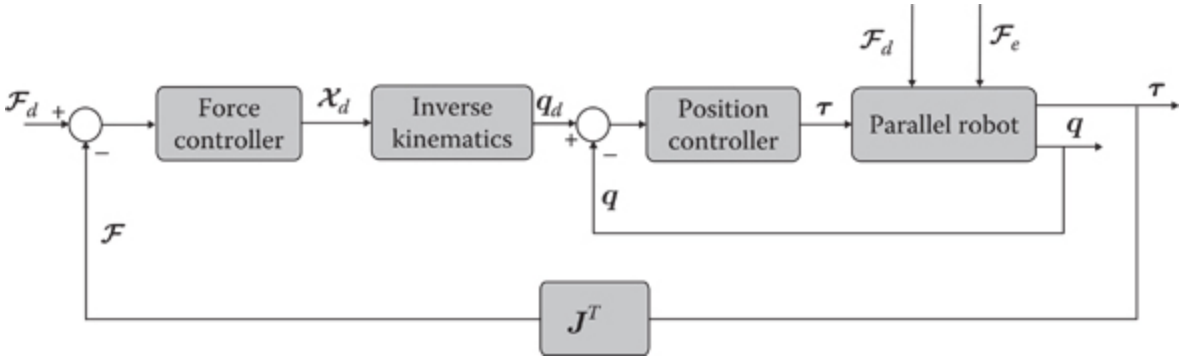



▶ Long Description for Figure 13.3

**FIGURE 13.3**
Second cascade topology of force feedback control: position is managed in the inner loop while force is controlled in the outer loop. The actuator forces and torques $\tau$ and the motion variable $\mathcal{X}$ are measured. ⏎

Consider the scenario where both force and motion variables are measured in the joint space. Figure 13.4 presents the force control topology operating in the joint space, where the inner loop is based on the measured motion variable in the joint space, $q$, and the outer loop utilizes the measured actuator force and torque vector. In this configuration, it is advisable to design the force controller in the task space, employing the Jacobian transpose mapping to project the measured actuator force

and torque vector into its corresponding wrench in the task space.




▶ Long Description for Figure 13.4

**FIGURE 13.4**

Third cascade topology of force feedback control: position is managed in the inner loop while force is controlled in the outer loop. The actuator forces and torques $\boldsymbol{\tau}$ and the active joint motion variable $\boldsymbol{q}$ are measured in the joint space. ⏎

However, the inner loop is constructed in the joint space since mapping the motion variable in the joint space to its equivalent in the task space necessitates a forward kinematic solution, which can be relatively complex for parallel manipulators. Therefore, the desired motion variable $\boldsymbol{\chi}_d$ is converted into joint space using the inverse kinematic solution shown in Figure 13.4. As a result, the controller is designed based on the joint space error $\boldsymbol{e}_q$ and is fully implemented in the joint space.

Consequently, the structure and characteristics of the position controller in this topology differ significantly from

those in the first two topologies. In this control topology, the input to the controller is the motion error in the joint space, denoted as $\boldsymbol{e}_q$, while the output is the actuator force and torque $\boldsymbol{\tau}$, which is also represented in the joint space. In contrast, the previous topologies have the motion controller receiving the motion error in the task space, $\boldsymbol{e}_x$, with the output in the joint space.

As a result, employing independent controllers for each joint may be more appropriate for the third topology illustrated in Figure 13.4. While various other control topologies can be found in the literature, not all are discussed in this section to maintain a manageable level of analytical complexity. Additional topologies may be developed based on the concepts presented in this section.

The topologies discussed in this section primarily focus on pure force control schemes, which are detailed in Section 13.4. Pure force control refers to scenarios where the main objective is force tracking while the manipulator is in contact with the environment. The secondary goal of these control algorithms is to maintain the stability of motion and, to some extent, achieve motion tracking when the robot's end effector is released from the environment.

Various control schemes have been developed to meet the necessary force-tracking requirements during force interactions between the manipulator and the environment. These control strategies are presented in a progression, ranging from simple foundational ideas to more

comprehensive and effective structures, as outlined in Section 13.4.

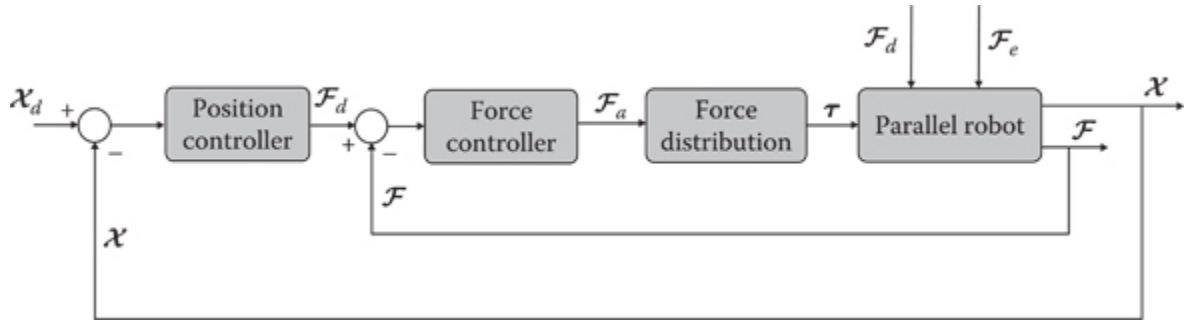
### 13.2.3 Force Feedback in the Inner Loop

Consider the force control schemes for a parallel manipulator where the motion-force relationship is the primary objective. In this scenario, force tracking is not the main focus; instead, it is recommended that the outer loop of the cascade control structure incorporates motion control feedback while the inner loop relies on force feedback. Given the various types of measurement units that can be utilized in parallel robots, different control topologies can be developed to implement these cascade controllers.

For instance, if both wrench and twist variables are measured in the task space, the wrench applied to the center of mass of the moving platform, denoted as $\mathcal{F}$, can be captured using a six-degrees-of-freedom wrench sensor. Meanwhile, the motion variable $\mathcal{X}$ can be obtained from another measurement unit. Although these measurements can be relatively costly, if both are implemented in the task space, the closed-loop system is capable of achieving an accurate motion-force relationship for the manipulator.

Figure 13.5 illustrates the cascade control topology for a system where both measured variables are in the task space. The measured quantities include the resulting wrench at the center of mass of the moving platform, denoted as $\mathcal{F}$, and the motion variable of the moving

platform, represented as $\mathcal{X}$. In this configuration, the inner loop is designed around force feedback, while the outer loop is based on position feedback.




▶ Long Description for Figure 13.5

**FIGURE 13.5**

Fourth cascade topology of force feedback control: the inner loop focuses on force control while the outer loop addresses position control. The moving platform wrench $\mathcal{F}$ and the motion variable $\mathcal{X}$ are measured in the task space. ⏎

The primary controller functions as a motion controller, with the secondary controller acting as a force-tracking controller. This arrangement ensures that when the manipulator is not in contact with a stiff environment, the primary controller maintains position tracking. However, when an interacting wrench $\mathcal{F}_e$ is applied to the moving platform, this structure effectively manages the force-motion relationship. Additionally, the controller operates in the task space, and the output from the motion controller feeds into the force controller. This setup can be

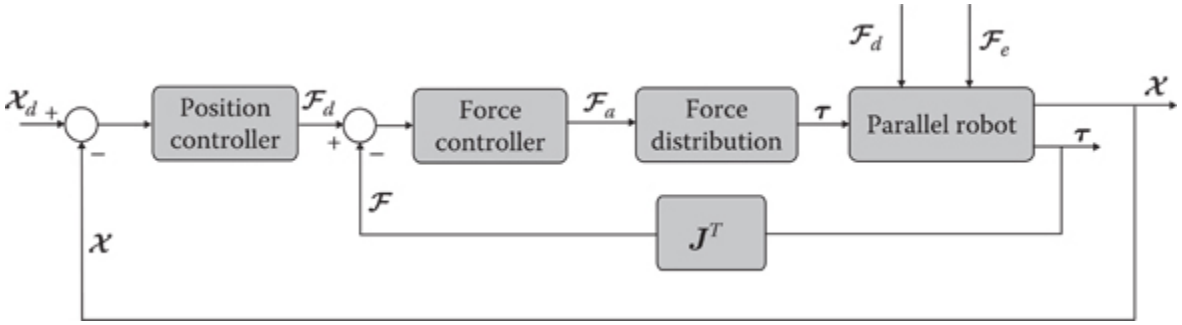
interpreted as the outer loop generating a desired force trajectory for the inner loop.

The output of the force controller is also structured in the task space, and to convert it into the actuator force $\boldsymbol{\tau}$, a force distribution block is utilized within this framework. As explained in detail earlier, the force distribution block translates the produced wrench in the task space, represented by $\boldsymbol{\mathcal{F}}_a$, into the corresponding actuator forces and torques $\boldsymbol{\tau}$. In a fully parallel manipulator like the Stewart–Gough platform, where the number of actuators matches the number of degrees of freedom, this mapping can be formulated using the Jacobian transpose of the manipulator, $\boldsymbol{J}^T$.

Given that this mapping for parallel robots is defined by $\boldsymbol{\mathcal{F}}_a = \boldsymbol{J}^T\boldsymbol{\tau}$, for a fully parallel manipulator, the actuator forces can be obtained using $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}_a$ at nonsingular configurations. This relationship allows for precise control of the manipulator's motion by ensuring that the desired forces in the task space are effectively translated into the necessary actuator outputs.

Other options for force control topology may be proposed based on variations in position and force measurements. Consider a general parallel manipulator composed of several limbs featuring both passive and active joints, as illustrated in Figure 1.11. As mentioned previously, it is typically much simpler to measure the actuator forces and torques $\boldsymbol{\tau}$ directly. If the motion variable in the task space, denoted as $\boldsymbol{\chi}$, is still being measured, the topology shown

in Figure 13.6 is recommended for the control system. In this setup, the measured actuator force/torque vector $\boldsymbol{\tau}$ is mapped to its corresponding wrench in the task space via the Jacobian transpose mapping, as it is known that $\boldsymbol{\mathcal{F}} = \boldsymbol{J}^T \boldsymbol{\tau}$.

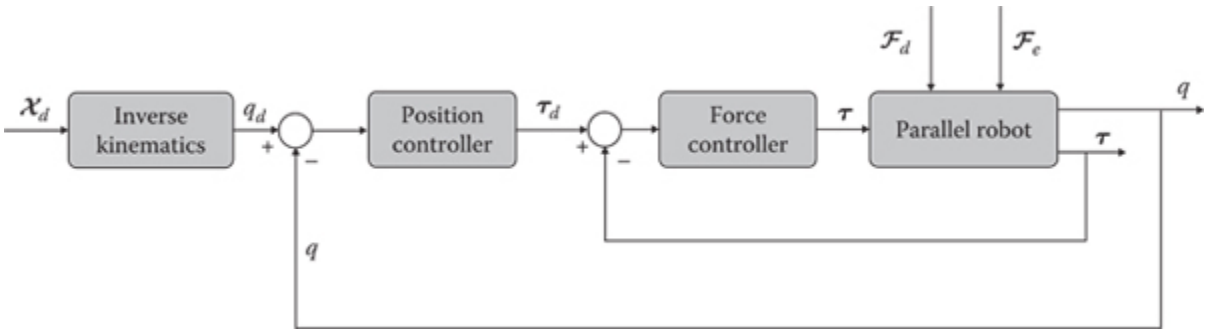



▶ Long Description for Figure 13.6

**FIGURE 13.6**
Fifth cascade topology of force feedback control: The inner loop employs force control, while the outer loop focuses on position control. Actuator forces/torques $\boldsymbol{\tau}$ and the motion variable $\boldsymbol{\mathcal{X}}$ are measured. ⏎

This mapping not only streamlines the calculation process but also allows the controller structure to remain in the task space, enabling the design of both force and position controllers in a manner similar to that used in the previous topology. Additionally, the unified structure of the control system supports the integration of various control strategies, such as adaptive and robust control, to further enhance performance under uncertain conditions.

Consider the situation where both force and motion variables are measured within the joint space. The force control topology illustrated in Figure 13.7 operates in the joint space, where the inner loop relies on the measured actuator force/torque vector $\boldsymbol{\tau}$, while the outer loop utilizes the measured actuated joint position vector $\boldsymbol{q}$. In this configuration, the desired motion in the task space is converted to the joint space using the inverse kinematic solution shown in Figure 13.4, with both position and force feedback designed in the joint space. As a result, the position controller is constructed based on the joint space error $\boldsymbol{e}_q$ and is entirely implemented within the joint space. Consequently, the structure and characteristics of the position controller in this topology are fundamentally different from those of the two previously proposed topologies.




▶ Long Description for Figure 13.7

**FIGURE 13.7**

Sixth cascade topology of force feedback control: The inner loop utilizes force control while the outer loop manages

position control. Actuator forces/torques $\boldsymbol{\tau}$ and the active joint motion variable $\boldsymbol{q}$ are measured in the joint space. ⏎

In this control topology, the input to the controller is the motion error in the joint space, denoted as $\boldsymbol{e}_q$, while the output of the controller is the actuator force/torque vector $\boldsymbol{\tau}$, which is also expressed in the joint space. In contrast, the previous topologies have the motion controller's input as the motion error in the task space, represented by $\boldsymbol{e}_x$, with the output also situated in the task space. Consequently, using independent controllers for each joint may be appropriate for this topology.

If model-based controllers, such as inverse dynamics control, are to be implemented within this framework, the computational burden would significantly increase, as it would require dynamic formulation for parallel manipulators in the joint space. While numerous other control topologies exist in the literature, they are not discussed in detail in this section to maintain a manageable level of complexity in the analysis. Nevertheless, alternative control topologies can be developed based on the concepts presented in this section.

The topologies discussed in this section are primarily employed in impedance control schemes, which will be elaborated upon in the upcoming chapter. Impedance control refers to the approach where neither force nor position requirements are exclusively prioritized; instead, the dynamic relationship between the interacting force and the robot's motion is managed through the system's

structure. Various control strategies have been devised to meet the impedance requirements when the manipulator interacts with its environment through force. These control methods are presented sequentially, progressing from simple foundational concepts to more comprehensive and effective structures in Chapter 14.

In the subsequent sections, the concept of force control techniques will be introduced, starting with a basic one-loop control scheme utilizing a stiffness control algorithm. Following this, pure force control techniques will be explored.

## 13.3 Stiffness Control

The increasing demand for precise positioning in industrial robots has compelled mechanical designers to develop much stiffer robotic manipulators. Parallel robots are particularly well-suited for this purpose, as their closed-kinematic structure inherently enhances the robot's stiffness. When high-gain controllers are applied to these rigid robot structures, the resulting manipulators are capable of delivering high-precision motion from the start. This level of precision is ideal when the robot operates freely in space without any contact with objects throughout its entire workspace. To maintain this condition, these robots are typically operated in isolated areas enclosed by safety barriers.

However, for a robotic manipulator equipped with a high-gain motion controller and a very stiff structure, interacting with a rigid environment can pose significant challenges. Furthermore, such a setup can be dangerous if the robot comes into contact with a human operator. Consider a scenario in which an obstacle suddenly appears in front of a robot's preplanned trajectory during its motion. When the robot collides with the obstacle, its design to track the predetermined trajectory with high accuracy may result in damage to the obstacle if it is more compliant than the robot. Conversely, the robot itself may suffer damage to its components. This situation becomes especially hazardous if the obstacle happens to be a human operator who inadvertently crosses the robot's path.

To mitigate these risks, it is beneficial to introduce some level of compliance into the robot's structure. This compliance can be incorporated either into the mechanical design of the robot or into its control system. By doing so, the robot can better absorb impacts and reduce the likelihood of causing harm to either the environment or itself, ultimately enhancing safety during operation.

Incorporating compliance into the structure of robots is a newly developed approach that designers are exploring for industrial applications. This compliance can be achieved by utilizing lightweight and relatively long links in robots, such as those employed in space missions [250] and minimally invasive surgery applications [251]. Additionally, advanced joint transmission systems [252, 253] and

tendon-type components in the robot's structure [254] can also contribute to this compliance.

While the induced compliance in the robot's structure facilitates smoother interactions with a rigid environment, developing appropriate control algorithms to maintain high tracking precision for such robots presents a significant challenge. Balancing the benefits of compliance with the need for accuracy requires innovative control strategies that can adapt to varying operational conditions while ensuring that performance standards are met.

Another approach to incorporating compliance into a robot is through its control structure. By utilizing an appropriate control topology and designing suitable controller gains, a robot that is mechanically rigid can exhibit compliant responses. Implementing compliance within the control structure may be particularly advantageous for industrial robots, as the compliance characteristics can be easily adjusted through software. In this scenario, the mechanical design of the robot remains similar to those used in high-precision motion-tracking applications, while its controller gains can be tuned to provide the necessary compliance for situations where the robot interacts with a rigid environment. This type of control algorithm is referred to as stiffness control [184, 255] or compliance control [256, 257], with the term “stiffness control” being used in this book.

The control topology utilized in stiffness control closely resembles that of the motion control topologies discussed

in Section 11.2, as this control structure primarily measures and utilizes motion variables for feedback. The interacting force is subsequently estimated using compliant models of the environment, which is then incorporated into the robot's dynamic formulation and the closed-loop characteristics analysis of the system.

The main goal of this control algorithm is to maintain the closed-loop stiffness of the manipulator at a desired level, facilitating suitable interactions with a rigid environment. To develop this concept and conduct the analysis at a manageable level, we will first present the fundamentals of stiffness control for a single-degree-of-freedom robot in the following section. Subsequently, we will expand on the general case in Section 13.3.2.

### 13.3.1 Single-Degree-of-Freedom Stiffness Control

Consider a parallel robot with one-degree-of-freedom motion that interacts with a stiff environment. As depicted in Figure 13.8, the motion of the robot is along the $x$ direction, where it has made contact with the environment at the position $x_e$. It is assumed that the robot's stiffness is greater than that of the environment. Consequently, due to this interaction, the environment experiences a deformation in the $x$ direction, quantified by the amount $x - x_e$. If the compliance of the environment is modeled

using a linear relationship, the interacting force in the $x$ direction can be expressed as:

$$f_e = k_e \, \Delta x = k_e (x - x_e), \tag{13.2}$$

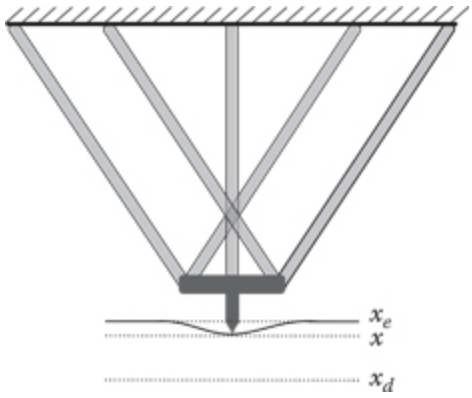



**FIGURE 13.8**
Compliant environment in a stiffness control scheme; analysis for a one-degree-of-freedom system. ⏎

where $\Delta x$ represents the compliant deformation of the body, $k_e$ is the linear stiffness coefficient of the environment in the $x$ direction, and $f_e$ denotes the interacting force between the robot and the environment. The general dynamic formulation of parallel robots, as given in Equation 13.1, can be simplified to a scalar equation expressed as follows, assuming there are no disturbance forces:

$$m(x)\ddot{x} + c(x, \dot{x})\dot{x} + g(x) + f_e = f,$$

(13.3)

where $m(x)$ represents the effective mass of the robot in the $x$ direction, $c(x, \dot{x})\dot{x}$ accounts for the effective Coriolis and centrifugal terms in the $x$ direction, $g(x)$ indicates the gravity term in the $x$ direction, $f_e$ represents the interaction force with the stiff environment in the $x$ direction, and $f$ is the projection of the actuator forces in the $x$ direction.

It can be demonstrated that various control algorithms designed for pure motion control of robots can be adapted for stiffness control with some modifications. To illustrate this concept, let us consider the general inverse dynamics control strategy outlined in Section 12.3 for this analysis. This control structure serves as the foundation for developing more advanced robust and adaptive control systems, which are comprehensively discussed in Section 12.5. Thus, it is employed as the base controller for the stiffness analysis of parallel robots in this section.

To implement such a controller, only the motion variable needs to be measured. This measurement can be conducted either in the task space or in the joint space, as previously described. However, in the case of a single-degree-of-freedom robot, we assume that the motion variable $x$ is measured in the task space and utilized in the feedback. Consequently, the topology of the stiffness control scheme employed for this scenario is the one illustrated in Figure 11.2.

Now, let's consider full information feedback linearization along with a proportional-derivative (PD) control approach

for inverse dynamics control (IDC) in the task space. Thus, the projection of actuator forces in the task space can be expressed as:

$$f = c(x, \dot{x})\dot{x} + g(x) + k_p e_x + k_d \dot{e}_x,$$

(13.4)

where $k_p$ and $k_d$ are the scalar PD gains, and the tracking error $e_x$ is defined as $e_x = x_d - x$, with $x_d$ representing the desired motion trajectory. As previously mentioned, implementing such a controller requires only the feedback of the motion variable. Additionally, the control command is generated based on the robot's desired trajectory. For effective interaction with the environment, careful consideration must be given to the desired trajectory $x_d$, which will be further discussed later in this section through subsequent analysis.

To analyze the characteristics of the control law, we can investigate the behavior of the closed-loop system by substituting the control law presented in Equation 13.4 into the manipulator dynamics described in Equation 13.3. By assuming complete knowledge of the centrifugal and gravity terms, these terms will cancel out in the closed-loop dynamic formulation, resulting in the following expression:

$$m(x)\ddot{x} + k_e(x - x_e) = k_p(x_d - x) + k_d(\dot{x}_d - \dot{x}).$$

(13.5)

The closed-loop dynamics of the robot can be further simplified to the following equation by factorizing the motion variable $x$:

$$m(x)\ddot{x} + k_d\dot{x} + (k_p + k_e)x = k_e x_e + k_p x_d + k_d \dot{x}_d. \tag{13.6}$$

It is important to note that the inverse dynamics controller can linearize the closed-loop dynamics if complete information about the Coriolis and gravity terms is available. In this scenario, the transient and steady-state behaviors of the closed-loop system can be analyzed using the Laplace transform.

Applying the Laplace transform to both sides of Equation 13.6, we can express the motion variable $x(s)$ in the s-domain as follows:

$$x(s) = \frac{(k_p + k_d s)x_d(s) + k_e x_e(s)}{ms^2 + k_d s + (k_p + k_e)}. \tag{13.7}$$

To examine the steady-state characteristics of the motion variable, we take into account constant input variables applied to the system. For a fixed desired trajectory with a steady-state value of $\bar{x}$, we can express the input using a step function in the s-domain, given by $x(s) = \bar{x}/s$. Similarly, let's consider a smooth, flat obstacle in the environment, characterized by a constant $x_e$ with a steady-

state value of $\bar{x}_e$ in the time domain, and represented in the s-domain as a step input $x_e(s) = \bar{x}_e/s$. The steady-state behavior of the motion variable in the time domain can be described using the final value theorem in the s-domain as [258].

$$\bar{x} = \lim_{t \to \infty} x(t) = \lim_{s \to 0} s\, x(s).$$

(13.8)

Consequently, the steady-state value for the motion variable $\bar{x}$ can be determined using the following equation:

$$\begin{aligned} \bar{x} &= \lim_{s \to 0} \frac{(k_p + k_d s)\bar{x}_d + k_e \bar{x}_e}{ms^2 + k_d s + (k_p + k_e)} \\ &= \frac{k_p \bar{x}_d + k_e \bar{x}_e}{k_p + k_e}. \end{aligned}$$

(13.9)

Let's investigate the steady-state behavior of the interaction force represented by $\bar{f}_e$. The steady-state value of the interaction force with the environment is calculated according to Equation 13.2 as

$$
\begin{aligned}
\bar{f}_e &= k_e(\bar{x} - \bar{x}_e) \\
&= \frac{k_e(k_p\bar{x}_d + k_e\bar{x}_e)}{k_p + k_e} - \frac{k_e\bar{x}_e(k_p + k_e)}{k_p + k_e} \\
&= \frac{k_e k_p(\bar{x}_d - \bar{x}_e)}{k_p + k_e}.
\end{aligned}
$$

(13.10)

However, it is important to note that the environment stiffness coefficient $k_e$ is several orders of magnitude greater than the controller gain $k_p$, specifically $k_e \gg k_p$. Therefore, the steady-state value of the interaction force can be approximated using the following equation:

$$
\begin{aligned}
\bar{f}_e &\simeq \frac{k_e k_p}{k_e}(\bar{x}_d - \bar{x}_e) \\
&\simeq k_p(\bar{x}_d - \bar{x}_e).
\end{aligned}
$$

(13.11)

Equation 13.11 highlights an important aspect of the stiffness control scheme: although the environment stiffness coefficient $k_e$ is typically very high, the proposed controller structure allows the interaction force at steady-state, $\bar{f}_e$, to be connected to the corresponding environment deflection at steady-state, $\bar{x}_d - \bar{x}_e$, through the controller gain $k_p$. As a result, the interacting stiffness of the manipulator can be effectively managed, even in the

presence of very stiff environments, by selecting a moderate controller gain $k_p$. This enables the interaction force of the robot with a rigid environment to be adjusted, incorporating a degree of flexibility into the robot's structure, even in designs that are inherently very rigid. This characteristic is the primary reason this controller structure is referred to as “stiffness control.”

Additionally, it is essential to recognize the significance and impact of defining the desired trajectory within this structure by analyzing Equation 13.11. To achieve a moderate stiffness for the manipulator from the beginning, it is sufficient to position the desired trajectory slightly inside the environment's surface. This concept is demonstrated in Figure 13.8, where the desired trajectory $x_d$ is situated within the body surface when the robot is in contact with a stiff environment.

To illustrate this idea, a graphical representation of Equation 13.11 is provided in Figure 13.9. As shown in Figure 13.9, a virtual spring-damper with controller gains $k_p$ and $k_d$ is connected to the tip of the end effector, functioning between $x_e$ and $x_d$. When the robot makes contact with the environment, the force generated by the virtual spring-damper maintains the end effector's contact with the surface, ensuring that the force-motion relationship complies with the required stiffness. Conversely, when the robot moves freely and is not in contact with any surface, $x = x_e$, and this virtual spring-damper compels the end effector to follow the desired

trajectory. In this way, the interaction forces can be finely adjusted to achieve the desired manipulator stiffness by carefully selecting the controller gain $k_p$ and thoughtfully designing the robot's desired motion trajectory.

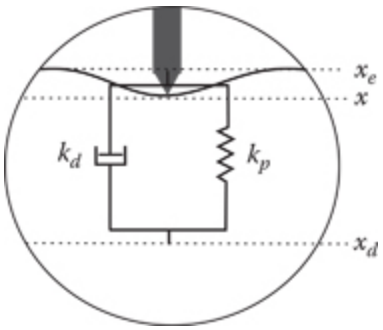



**FIGURE 13.9**
Compliant environment in stiffness control scheme; virtual spring–damper model. ⏎

While the structure of the stiffness control scheme described in this section resembles that of the position control scheme outlined in Section 12.3, there are notable differences in how these schemes are implemented in practice. In a pure motion control scenario, the robot is assumed to move freely throughout its entire maneuver without making contact with any surfaces. Consequently, high PD gains are recommended to ensure the manipulator achieves the desired trajectory-tracking performance.

In contrast, the stiffness control routine acknowledges that the robot interacts with a stiff environment, meaning that interaction forces have a significant impact on motion. When the robot is in contact with the environment, the desired trajectory is modified based on the characteristics

of the contact surface. Moreover, during this contact, the primary goal of the controller is to maintain the end effector's contact with the environment while adjusting the interaction force to align with a desired stiffness relationship. This approach allows the robot to perform moderate trajectory tracking when it is not in contact with the environment, as the controller gains are set lower than those used in a pure motion control scheme. However, when the robot does come into contact with the environment, controlling the robot's stiffness at the desired level facilitates a smooth and safe interaction between the robot and its surroundings.

This control scheme is particularly advantageous when the robot is in contact with a predetermined object during its motion. The interaction forces can be effectively managed through the stiffness control scheme without the necessity for any force measurement. However, it is essential that the accuracy of motion measurements exceeds that of a pure motion control scheme. This requirement is crucial for detecting desired motion trajectories that are deliberately positioned slightly inside the contact surface.

The desired trajectory is designed to be of a greater magnitude than the elastic deformation of the object during contact, but still significantly smaller than the overall motion of the robot. Consequently, a motion measurement unit with high resolution across a broad operational range

is necessary to implement stiffness control routines successfully in practice.

### 13.3.2 General Stiffness Control

The concept of stiffness control has been developed for a single-degree-of-freedom manipulator, and the steady-state behavior of the control system is analyzed using the Laplace transform. To extend this concept to a manipulator with multiple degrees of freedom, we consider a parallel robot capable of multiple degrees of freedom interacting with a stiff environment. In this context, the motion of the robot's end effector is represented by the motion vector $\boldsymbol{\chi}$. For a general six-degrees-of-freedom manipulator, this motion vector comprises the linear motion vector $\boldsymbol{x}$ as well as the orientation variable $\boldsymbol{\theta}$, preferably expressed using screw coordinates. Thus, we have $\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{\theta} \end{bmatrix}^T$.

During the robot's interaction with the environment, an interaction wrench $\boldsymbol{\mathcal{F}}_e$ must be included in the dynamics formulation. For a general six-degrees-of-freedom manipulator, the interaction wrench is composed of the contact force $\boldsymbol{F}_e$ and the torque $\boldsymbol{n}_e$, expressed as $\boldsymbol{\mathcal{F}}_e = \begin{bmatrix} \boldsymbol{F}_e & \boldsymbol{n}_e \end{bmatrix}^T$. It is assumed that the robot's stiffness exceeds that of the environment; as a result of this interaction, the environment undergoes both positional and orientational deformation, represented by $\boldsymbol{\chi} - \boldsymbol{\chi}_e$. If the linear and angular compliance of the environment is modeled using a linear relationship, the interacting force can be formulated as

$$\boldsymbol{\mathcal{F}}_e = \boldsymbol{K}_e \Delta(\boldsymbol{\mathcal{X}}) = \boldsymbol{K}_e(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_e),$$

(13.12)

where $\Delta\boldsymbol{\mathcal{X}}$ signifies the compliant deformation of the body, and $\boldsymbol{K}_e$ denotes the stiffness matrix of the environment, which includes the linear stiffness coefficients related to $\boldsymbol{x}$ and the angular stiffness coefficients associated with $\boldsymbol{\theta}$. Additionally, $\boldsymbol{\mathcal{F}}_e$ represents the interaction wrench between the robot and the environment. The general dynamic formulation for parallel robots, as given in Equation 13.1, can be expressed by the following equation in the absence of any disturbance wrench:

$$\boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) + \boldsymbol{K}_e(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_e) = \boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{\mathcal{F}},$$

(13.13)

where $\boldsymbol{M}(\boldsymbol{\mathcal{X}})$ represents the robot mass matrix, $\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})$ denotes the Christoffel matrix, $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$ indicates the gravity vector, $\boldsymbol{\tau}$ is the actuator force vector, $\boldsymbol{\mathcal{F}}$ is the projection of the actuator force vector into the task space, and finally, $\boldsymbol{\mathcal{F}}_e$ represents the interaction wrench.

As discussed in Section 13.3.1, various motion control schemes can be employed for stiffness control, with careful consideration given to the controller gain and the adjustment of the desired trajectory. To analyze the characteristics of stiffness control for a multiple-degrees-of-freedom robot, we can consider the partial feedback

linearization IDC developed in Section 12.4. It is important to note that, similar to the single-degree-of-freedom case, only the motion variable is measured for implementing such a controller, and this measurement is assumed to be carried out in the task space. Therefore, the topology of the stiffness control scheme used for this measurement is illustrated in Figure 11.2.

By employing partial feedback linearization along with PD control in the task space, the projection of the actuator forces in the task space can be defined by the equation

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) + \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x, \tag{13.14}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ represent the PD gain matrices, and the tracking error $\boldsymbol{e}_x$ is defined as $\boldsymbol{e}_x = \boldsymbol{\chi}_d - \boldsymbol{\chi}$, with $\boldsymbol{\chi}_d$ indicating the desired motion trajectory in the task space. As mentioned earlier, only the motion variable feedback is necessary for implementing this controller, and the control command is generated based on the robot's desired trajectory.

To analyze the characteristics of the control law, we will investigate the behavior of the closed-loop system by substituting the control law presented in Equation 13.14 into the manipulator dynamics given in 13.13. By assuming partial linearization in the control structure, the Coriolis and centrifugal terms persist in the closed-loop system

dynamics, while the gravity term is entirely eliminated through feedback linearization.

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) = \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x. \tag{13.15}$$

It is important to note that while the gravity term vanishes from the closed-loop dynamics when complete information about the gravity vector is available for partial feedback linearization, Equation 13.15 still describes a nonlinear and coupled dynamic system. In this scenario, the steady-state behavior of the closed-loop system can be analyzed using Lyapunov analysis.[1]

[1]The reader is encouraged to refer to the discussion in Appendix B regarding Lyapunov stability. ⏎

To analyze the steady-state behavior of the motion variable, we consider constant input variables applied to the system. For this analysis, we assume that the desired motion trajectory of the robot, $\boldsymbol{\chi}_d$, is constant, and the contact surface in the robot's interaction, $\boldsymbol{\chi}_e$, is also constant. For the asymptotic stability analysis of the error dynamics, we will consider the following Lyapunov function candidate for the system:

$$V(\boldsymbol{\chi}) = \frac{1}{2}\dot{\boldsymbol{\chi}}^T \boldsymbol{M}\dot{\boldsymbol{\chi}} + \frac{1}{2}\boldsymbol{e}_x^T \boldsymbol{K}_p \boldsymbol{e}_x + \frac{1}{2}(\boldsymbol{\chi} - \boldsymbol{\chi}_e)^T \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e$$

(13.16)

The Lyapunov function candidate comprises the kinetic energy of the manipulator and the potential energy associated with the proportional feedback effort $\boldsymbol{K}_p \boldsymbol{e}_x$, as well as the elastic deformation term $\boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e)$. Consequently, $V(\boldsymbol{\chi})$ is a positive definite scalar function. The Lyapunov stability method is based on demonstrating that, along any motion trajectory of the robot, the time derivative of the Lyapunov function is negative definite. For the specific case of a constant desired trajectory and a constant contact surface considered in this analysis, the time derivative of the Lyapunov function can be derived as follows:

$$\dot{V}(\boldsymbol{\chi}) = \dot{\boldsymbol{\chi}}^T \boldsymbol{M} \ddot{\boldsymbol{\chi}} + \frac{1}{2} \dot{\boldsymbol{\chi}}^T \dot{\boldsymbol{M}} \dot{\boldsymbol{\chi}} + \dot{\boldsymbol{e}}_x^T \boldsymbol{K}_p \boldsymbol{e}_x + \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_e (\boldsymbol{\chi} - \boldsymbol{\chi}_e)$$

(13.17)

Substituting $\boldsymbol{M}\ddot{\boldsymbol{\chi}}$ from the closed-loop dynamic formulation 13.15 into Equation 13.17, and noting that for constant desired trajectory motion, $\dot{\boldsymbol{\chi}}_d = \boldsymbol{0}$, leads to a simplification of the time derivative of the Lyapunov function as follows:

$$\begin{aligned}\dot{V} &= \dot{\boldsymbol{\chi}}^T \left(-\boldsymbol{\mathcal{C}}\dot{\boldsymbol{\chi}} - \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) + \boldsymbol{K}_p \boldsymbol{e}_x + \boldsymbol{K}_d \dot{\boldsymbol{e}}_x\right) \\ &\quad + \frac{1}{2}\dot{\boldsymbol{\chi}}^T \dot{\boldsymbol{M}} \dot{\boldsymbol{\chi}} + \dot{\boldsymbol{e}}_x^T \boldsymbol{K}_p \boldsymbol{e}_x + \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e).\end{aligned}$$

To simplify this equation, we observe that for a constant desired trajectory, $\dot{\boldsymbol{\chi}}_d = \boldsymbol{0}$. Consequently, the tracking error rate of change can be expressed as $\dot{\boldsymbol{e}}_x = (\dot{\boldsymbol{\chi}}_d - \dot{\boldsymbol{\chi}}) = -\dot{\boldsymbol{\chi}}$. Hence, we have:

$$\begin{aligned}\dot{V} &= \frac{1}{2}\dot{\boldsymbol{\chi}}^T \left(\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}\right)\dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) + \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_p \boldsymbol{e}_x \\ &\quad -\dot{\boldsymbol{\chi}}^T \boldsymbol{K}_d \dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_p \boldsymbol{e}_x + \dot{\boldsymbol{\chi}}^T \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) \\ &= -\dot{\boldsymbol{\chi}}^T \boldsymbol{K}_d \dot{\boldsymbol{\chi}} + \frac{1}{2}\dot{\boldsymbol{\chi}}^T \left(\dot{\boldsymbol{M}} - 2\boldsymbol{\mathcal{C}}\right)\dot{\boldsymbol{\chi}} \\ &= -\dot{\boldsymbol{\chi}}^T \boldsymbol{K}_d \dot{\boldsymbol{\chi}} \leq 0\end{aligned} \tag{13.18}$$

In the last equality, the skew-symmetric property of the dynamic formulation matrices is utilized to further simplify the equation. This result implies that the derivative of the Lyapunov function is negative semidefinite. While this finding confirms stability in the sense of Lyapunov, it does not ensure that asymptotic tracking is achieved.

To analyze the asymptotic behavior, we apply Krasovskii-LaSalle's Theorem.[1] It is important to note that $V$ is always decreasing, provided that $\dot{\boldsymbol{\chi}}$ is nonzero. Therefore, the

steady-state behavior can be examined through the asymptotic analysis of Equation 13.15, where both $\dot{\boldsymbol{\chi}}$ and $\ddot{\boldsymbol{\chi}}$ are zero at steady state. In this scenario, from Equation 13.15, we can conclude that

$$\lim_{t\to\infty} \left[\boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) - \boldsymbol{K}_p(\boldsymbol{\chi}_d - \boldsymbol{\chi})\right] = 0 \tag{13.19}$$

[1]Refer to Appendix C. ⏎

or

$$\lim_{t\to\infty} \left[(\boldsymbol{K}_e + \boldsymbol{K}_p)\boldsymbol{\chi} - \boldsymbol{K}_e\boldsymbol{\chi}_e - \boldsymbol{K}_p\boldsymbol{\chi}_d\right] = 0. \tag{13.20}$$

Let us denote the steady-state value of the motion variable as $\bar{\boldsymbol{\chi}}$. Additionally, under the assumption of a constant desired trajectory, we denote the steady-state value of the desired trajectory by $\bar{\boldsymbol{\chi}}_d$ and the steady-state value for the contact surface by $\bar{\boldsymbol{\chi}}_e$. Therefore, Equation 13.20 can be utilized to derive the steady-state value of the motion variable as follows:

$$\bar{\boldsymbol{\chi}} = \left[\boldsymbol{K}_e + \boldsymbol{K}_p\right]^{-1}\left[\boldsymbol{K}_e\bar{\boldsymbol{\chi}}_e + \boldsymbol{K}_p\bar{\boldsymbol{\chi}}_d\right]. \tag{13.21}$$

Let us investigate the steady-state behavior of the interaction wrench, denoted as $\bar{\boldsymbol{\mathcal{F}}}_e$. The steady-state value of the interaction wrench with the environment can be calculated using Equation 13.12 as follows:

$$\bar{\boldsymbol{\mathcal{F}}}_e = \boldsymbol{K}_e(\bar{\boldsymbol{\chi}} - \bar{\boldsymbol{\chi}}_e).$$

(13.22)

Substituting Equation 13.21 into Equation 13.22 and simplifying yields:

$$\bar{\boldsymbol{\mathcal{F}}}_e = \boldsymbol{K}_e[\boldsymbol{K}_e + \boldsymbol{K}_p]^{-1}\boldsymbol{K}_p(\bar{\boldsymbol{\chi}}_d - \bar{\boldsymbol{\chi}}_e).$$

(13.23)

However, it is important to note that the environment stiffness coefficient $\boldsymbol{K}_e$ is several orders of magnitude greater than the controller matrix gain $\boldsymbol{K}_p$, that is, $\boldsymbol{K}_e \gg \boldsymbol{K}_p$. As a result, the matrix $[\boldsymbol{K}_e + \boldsymbol{K}_p]^{-1}$ can be approximated by $\boldsymbol{K}_e^{-1}$. Consequently, the steady-state value of the interaction force can be approximated by the following equation:

$$\bar{\boldsymbol{\mathcal{F}}}_e \simeq \boldsymbol{K}_p(\bar{\boldsymbol{\chi}}_d - \bar{\boldsymbol{\chi}}_e).$$

(13.24)

Similar to the single-degree-of-freedom case, Equation 13.24 highlights a key aspect of the stiffness control

scheme: although the environment stiffness coefficient $\boldsymbol{K}_e$ is typically very large, the proposed controller structure allows the steady-state interaction force $\bar{\boldsymbol{f}}_e$ to be related to the corresponding steady-state environment deflection $\bar{\boldsymbol{\chi}}_d - \bar{\boldsymbol{\chi}}_e$ through the controller gain $\boldsymbol{K}_p$. Consequently, the interacting stiffness of the manipulator can be effectively controlled even in very stiff environments by selecting a moderate controller matrix gain $\boldsymbol{K}_p$. This means that the interaction of the robot with a stiff environment can be adjusted to incorporate embedded flexibility within the structure of the robot, even for very rigid mechanical designs.

This analysis uncovers another significant point: even when the complete information regarding the mass matrix and the Coriolis and centrifugal matrix is not utilized in the IDC structure, and the closed-loop dynamics is not entirely linearized, the PD control structure with gravity compensation can still achieve the desired stiffness control strategy at steady state. In this context, the skew-symmetric property of the dynamic formulation matrices ensures that the Lyapunov function candidate will monotonically converge to zero along all motion trajectories.

Although the stiffness control scheme presented in this section closely resembles the position control scheme discussed in Section 12.4, there are distinct differences in how these schemes are implemented in practice. In pure motion control, the assumption is that the robot operates

freely in space without making contact with any surfaces throughout its maneuver. As a result, high PD gains are recommended to ensure the required trajectory tracking performance for the manipulator.

In contrast, the stiffness control routine assumes that the robot comes into contact with a stiff environment, which significantly influences its motion due to interaction forces. Consequently, during the segments of motion where the robot is in contact with the environment, the desired trajectory is adjusted based on the contact surface. Moreover, in these phases of motion, the primary objective of the controller is to maintain contact between the moving platform and the environment while allowing the interaction force to align with the desired stiffness. This approach enables the robot to perform moderate trajectory tracking when not in contact with the environment, as the controller gains are set lower than those in pure motion control.

However, the ability to smoothly adjust the desired trajectory in response to contact dynamics also necessitates a more complex sensing and feedback mechanism. The robot must be equipped with accurate sensors that can detect contact with the environment and provide real-time feedback on the interaction forces. This is crucial for ensuring that the adjustments made to the desired trajectory do not compromise the overall motion performance of the robot.

Additionally, by implementing this stiffness control strategy, the robot can adapt to varying stiffness in the environment. This adaptability is especially useful in dynamic environments where the properties of surfaces may change unexpectedly, requiring the robot to adjust its interaction forces on the fly. As a result, the stiffness control scheme not only enhances the robot's operational versatility but also contributes to a more resilient and robust performance in diverse application contexts.

### 13.3.3 Case Study I: Planar Cable Robot

This section provides a comprehensive examination of stiffness control for a planar 4-RPR manipulator. The structure of the planar manipulator is detailed in Section 4.4.1 and illustrated in Figure 4.4. As previously mentioned, the moving platform of this manipulator is supported by four limbs, each featuring an identical RPR kinematic configuration.

A thorough Jacobian and singularity analysis of the manipulator can be found in Section 5.5, while the dynamic analysis is presented in Section 7.4. Various motion control strategies have been implemented on this manipulator, with their effectiveness analyzed in detail in Section 12.7. In the following sections, we will implement the stiffness control strategy tailored for this manipulator.

The initial stiffness control strategy implemented on the system utilizes a combination of partial feedback linearization and PD linear control. The control effort

exerted on the system is expressed in Equation 13.14. It is important to note that the motion variable in the task space is represented as $\boldsymbol{\chi} = [x_G \quad y_G \quad \phi]^T$, while the tracking error is defined as $\boldsymbol{e}_x = [e_x \quad e_y \quad e_\phi]$.

The PD controller is indicated by $(\boldsymbol{K}_d s + \boldsymbol{K}_p)\boldsymbol{e}_x$, where $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ are $3 \times 3$ diagonal matrices representing the gains for the derivative and proportional stiffness controllers, respectively. The controller's output is denoted by $\boldsymbol{\mathcal{F}}$, which comprises $\boldsymbol{\mathcal{F}} = [F_x \quad F_y \quad \tau_\phi]^T$. In practical terms, the computed output wrench is converted into actuator forces through redundancy resolution methods outlined in Section 6.6.

As illustrated in Figure 13.10, a flat surface is positioned in the robot's workspace, allowing the center of the robot's moving platform to come into contact with a rigid environment during part of its maneuver. In the previously analyzed maneuvers for the planar manipulator, the desired trajectory of the moving platform was modeled as a smooth cubic polynomial function for $x_d$ and $y_d$, progressing from zero initial conditions up to a maximum of $100\,m$, as depicted in Figure 11.13. The simulation duration was set to $200\,s$, while the desired position for $\phi_d$ also followed a smooth cubic polynomial function, transitioning from 0° to 45°. This baseline trajectory is altered in the simulations conducted for stiffness control to assess the interaction with the flat object shown in Figure 13.11 within the workspace.

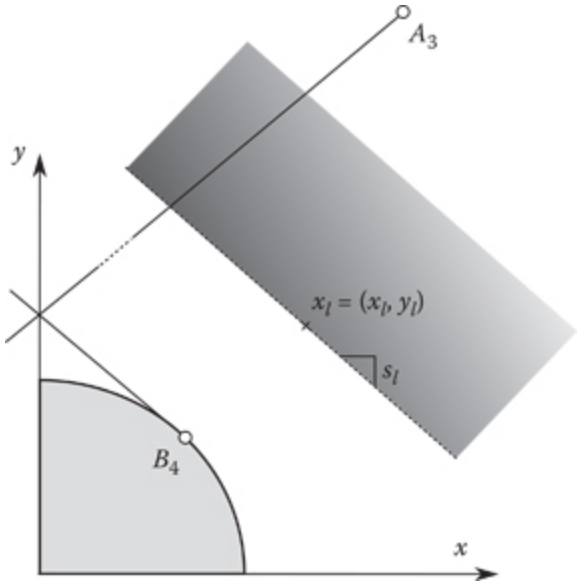
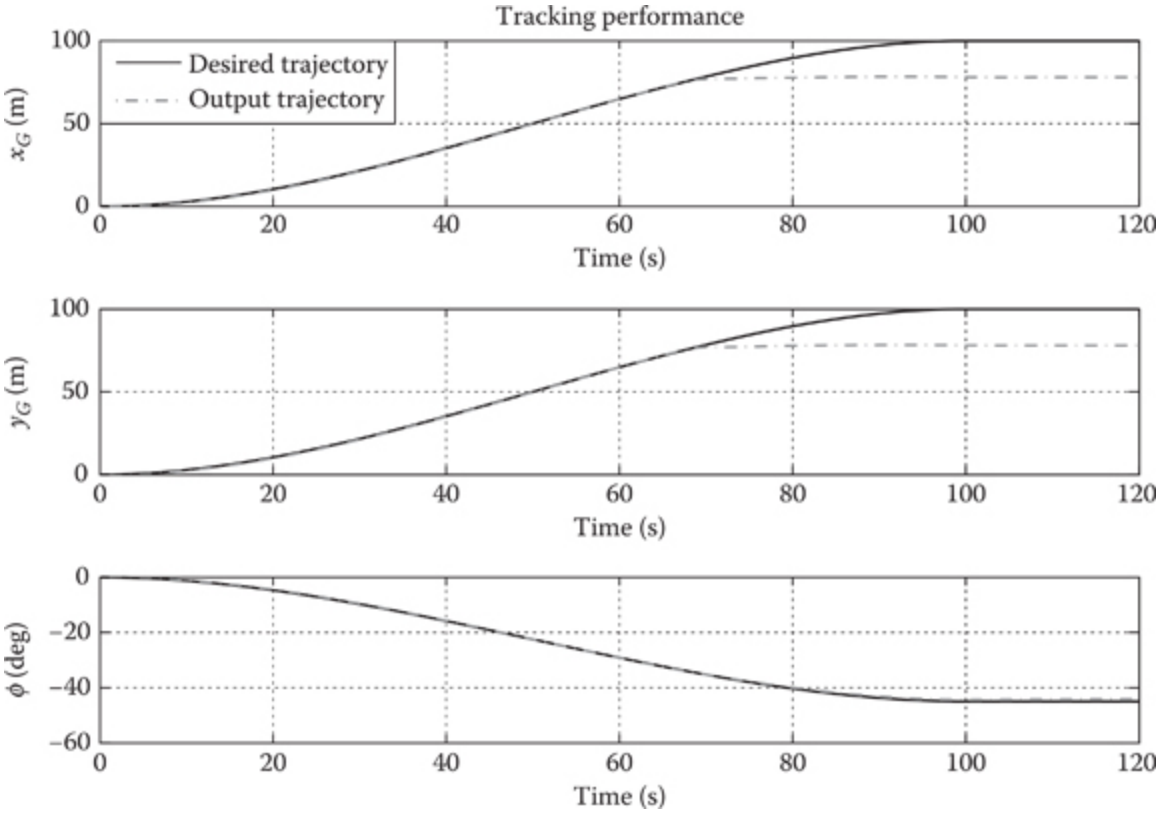



**FIGURE 13.10**

Schematic representation of a flat object located within the workspace of the planar 4RPR manipulator. ⏎



▶ Long Description for Figure 13.11

**FIGURE 13.11**
The desired and output trajectories of the planar 4RPR manipulator during a collision with a flat object; implementing partial IDC stiffness control. ⏎

As depicted in Figure 13.11, the desired trajectory in the subsequent simulations begins from zero initial conditions, reaching a final position of $100\,m$ over a duration of $100\,s$, and then remains at this position until $120\,s$. The object is positioned within the robot's workspace, located $75\,m$ from the initial zero position. As illustrated in Figure 13.10, the flat object is assumed to have a linear contact surface, represented by a line that passes through a point labeled as $\boldsymbol{x}_l = (x_l, y_l)$, with a slope of $s_l$ in the $x$–$y$ plane. Consequently, the surface of the object is defined by the following line equation:

$$y = y_l + s_l(x - x_l). \tag{13.25}$$

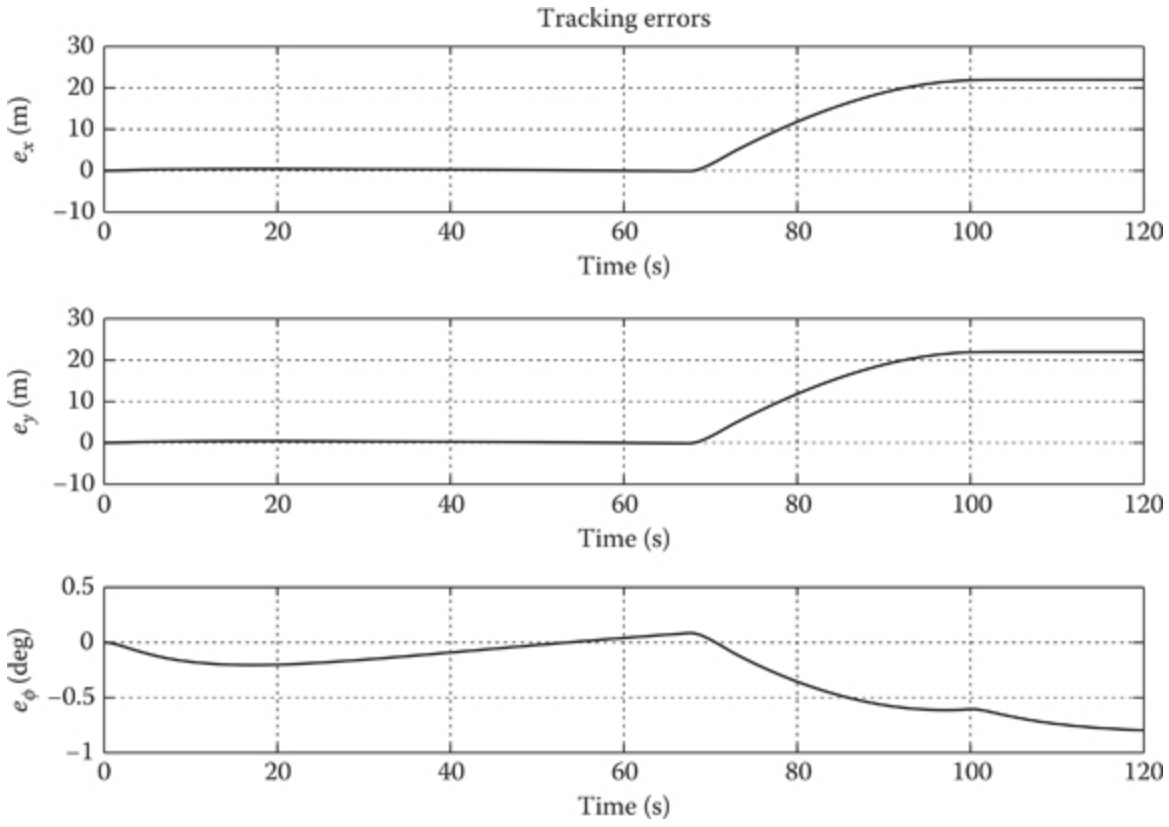
For the simulations, it is assumed that $\boldsymbol{x}_l = (75, 75)$ and that the slope of the contact surface is $s_l = -1$. Consequently, in the designated trajectory illustrated in Figure 13.11, the center of the moving platform makes contact with the flat object at approximately $67$ s, maintaining this contact until the end of the simulation

period. The final 20 s of the simulation are considered, during which all desired trajectories remain constant to analyze the steady-state behavior of the closed-loop force and position trajectories. The stiffness of the object is modeled using a linear stiffness gain of $\boldsymbol{K}_e = 10^3 \cdot \boldsymbol{I}_{3\times3}$ (N/m) and a linear structural damping of $\boldsymbol{C}_e = 10^3 \cdot \boldsymbol{I}_{3\times3}$ $(\mathrm{N}\cdot\mathrm{m/s})$. Therefore, the contact force is represented by

$$\boldsymbol{\mathcal{F}}_e = \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) + \boldsymbol{C}_e(\dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}_e). \tag{13.26}$$

The geometric and inertial parameters utilized in the simulations are presented in Table 10.1, and all initial state conditions are set to zero. In the subsequent simulations, the disturbance wrench applied to the moving platform is assumed to be zero, while the gravity vector is represented as $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$. The control effort is derived from Equation 13.14, and since the gravity term does not affect the $x$ and $y$ directions, the gravity compensation term is zero in these directions, meaning that $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) = \boldsymbol{0}$. To compare the results with a full IDC implementation, the PD controller gains are set to $\boldsymbol{K}_d = 0.1 \cdot \boldsymbol{M}(\boldsymbol{\chi}) \cdot \boldsymbol{I}_{3\times3}$ and $\boldsymbol{K}_p = \boldsymbol{M}(\boldsymbol{\chi}) \cdot \boldsymbol{I}_{3\times3}$. The controller gains employed here are lower than those used for pure position control, as the goal is to manage the force-motion relationship when the moving platform is in contact with a rigid environment. Therefore, with these controllers, achieving high positioning accuracy is not the primary focus.

The final motion of the moving platform and the tracking errors are depicted in Figures 13.11 and 13.12. As shown in Figure 13.11, the desired and final closed-loop motions of the system are nearly identical before the moving platform collides with the object. At $67\,s$, the collision occurs, causing the motion of the moving platform to remain almost constant on the surface of the object. More details regarding the motion of the moving platform at this moment can be seen in Figure 13.12, where it is observed that following the collision, the moving platform experiences a smooth and nondestructive motion on the object's surface, resulting in only minor deflection of the object. Additionally, it is important to note that no interaction torque is included in the simulations, which is reflected in the relatively consistent tracking error in orientation before and after the collision. The tracking performance of this controller is inferior to that of a pure position controller, as the PD gains are set to lower values to maintain a smooth and nondestructive force-motion interaction during contact.



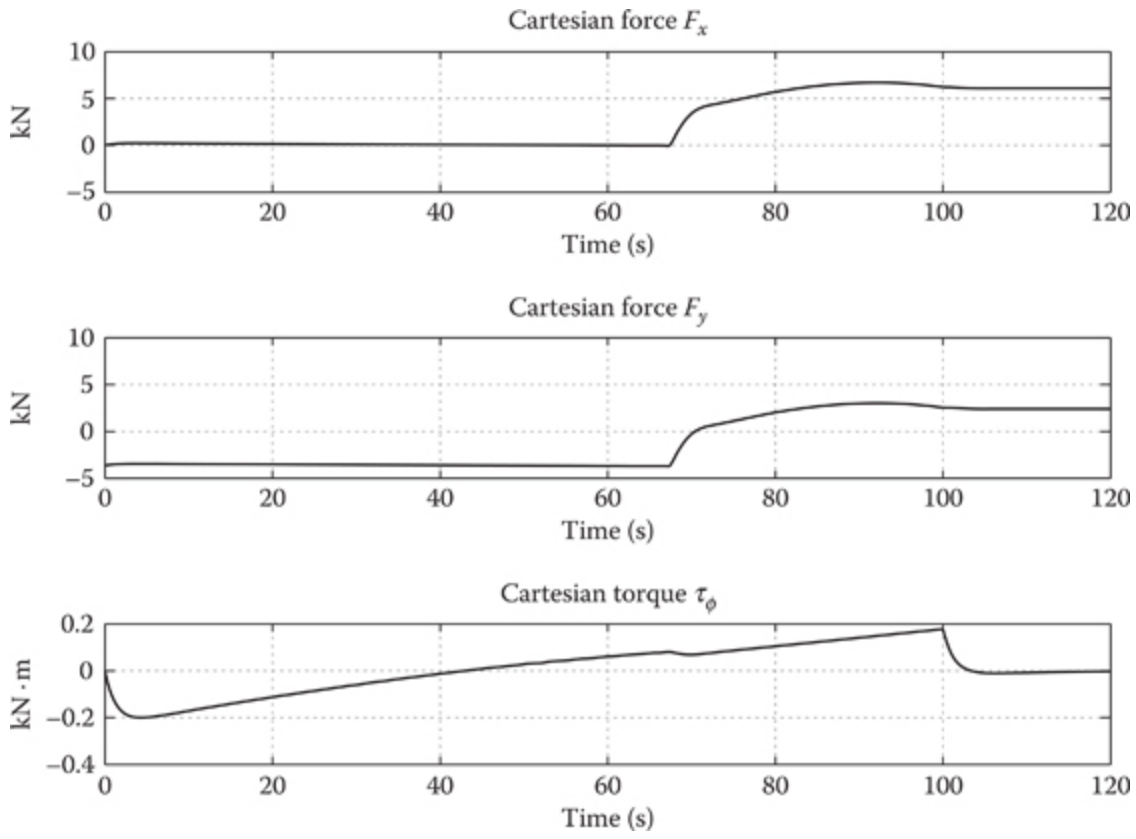

▶ Long Description for Figure 13.12

**FIGURE 13.12**

The closed-loop tracking performance of the planar 4RPR manipulator during a collision with a flat object; utilizing partial IDC stiffness control. ⏎

The Cartesian wrench $\mathcal{F} = [F_x, F_y, \tau_\phi]^T$ generated by the stiffness controller is shown in Figure 13.13. As observed in Figure 13.13, the Cartesian forces in the $x$ and $y$ directions experience a rapid increase following the collision at $67\,s$, while no interaction torques are detected in the $\phi$ direction. Similarly, the actuator forces of the manipulator are presented in Figure 13.14. In this figure, two sets of actuator forces are depicted: the dash-dotted lines

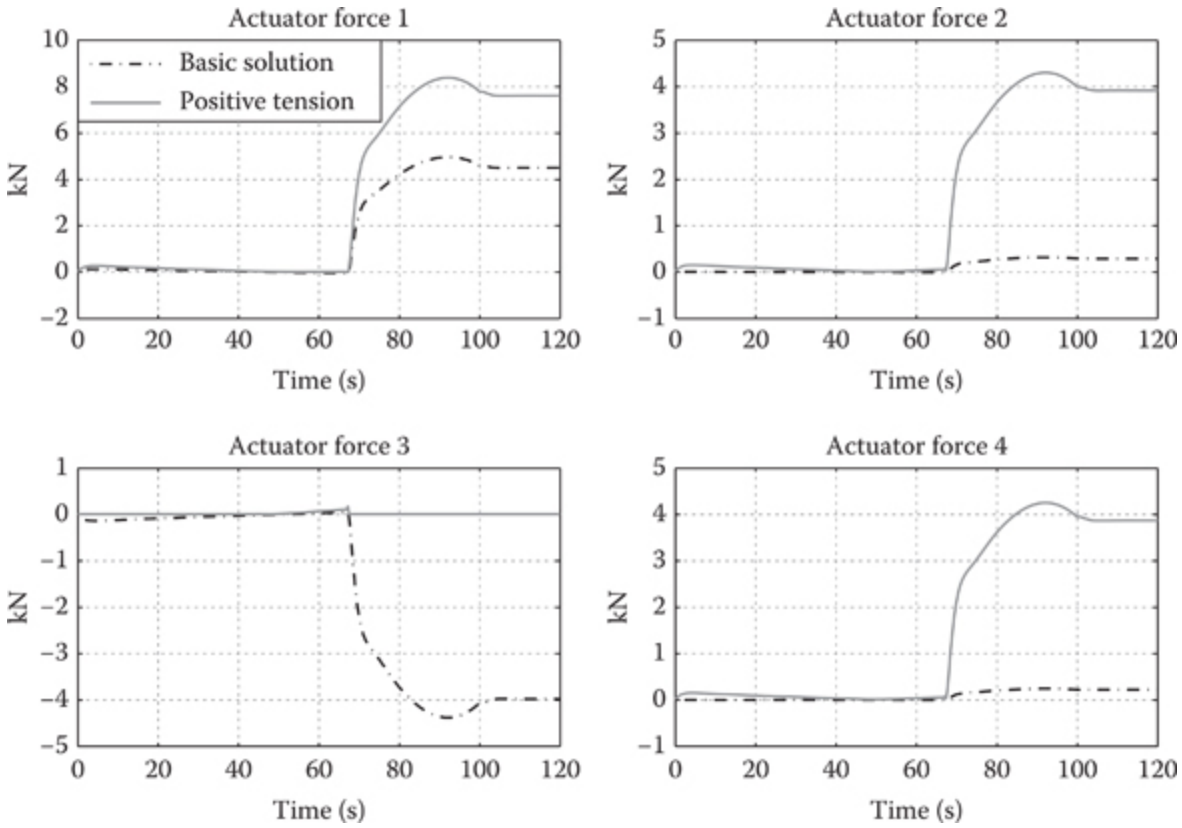
represent the base solution of the optimization problem introduced in the redundancy resolution scheme via Equation 6.84, while the solid line indicates the final optimization solution for the redundancy resolution problem outlined in Equation 6.87. The redundancy resolution solution is derived using an iterative numerical method. As shown in Figure 13.14, the actuator forces increase rapidly after the collision, while the tension forces obtained from the redundancy resolution scheme remain positive throughout the entire duration.




▶ Long Description for Figure 13.13

**FIGURE 13.13**

The Cartesian wrench of the planar 4RPR manipulator during a collision with a flat object; utilizing partial IDC stiffness control. ⏎



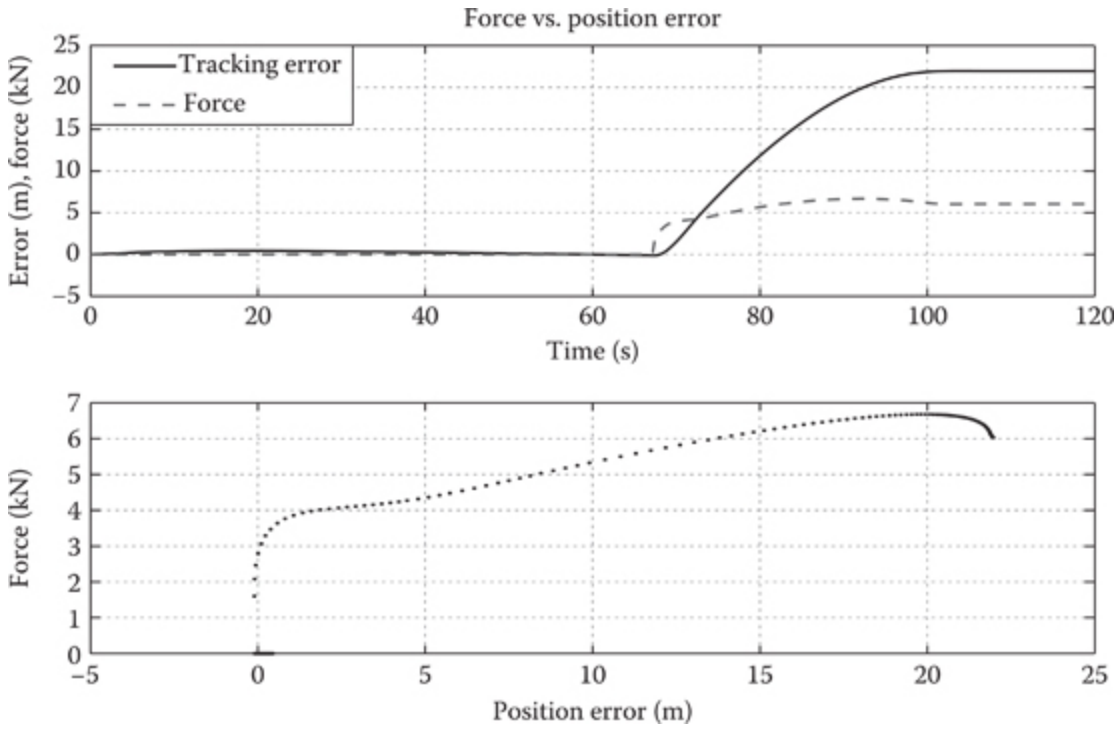

▶ Long Description for Figure 13.14

**FIGURE 13.14**

The actuator forces of the planar $4RPR$ manipulator colliding with a flat object; partial IDC stiffness control. ⏎

To investigate the stiffness characteristics of the manipulator in more detail, the tracking error $e_x$ and the interacting force $F_x$ are plotted together in Figure 13.15. In the top plot of Figure 13.15, the tracking error and force trajectories are displayed as functions of time, while the

bottom plot illustrates the tracking error and force trajectories in relation to one another. As shown in the top figure, the tracking error aligns with the interacting force, except during the transient period immediately following the collision. The force-motion relationship is clearly evident in the bottom figure, which exhibits a quasi-linear behavior except during the transient phase. In the steady state, the force-position behavior converges to a specific stiffness value for the system, which can be appropriately adjusted by tuning the parameters of the stiffness controller.

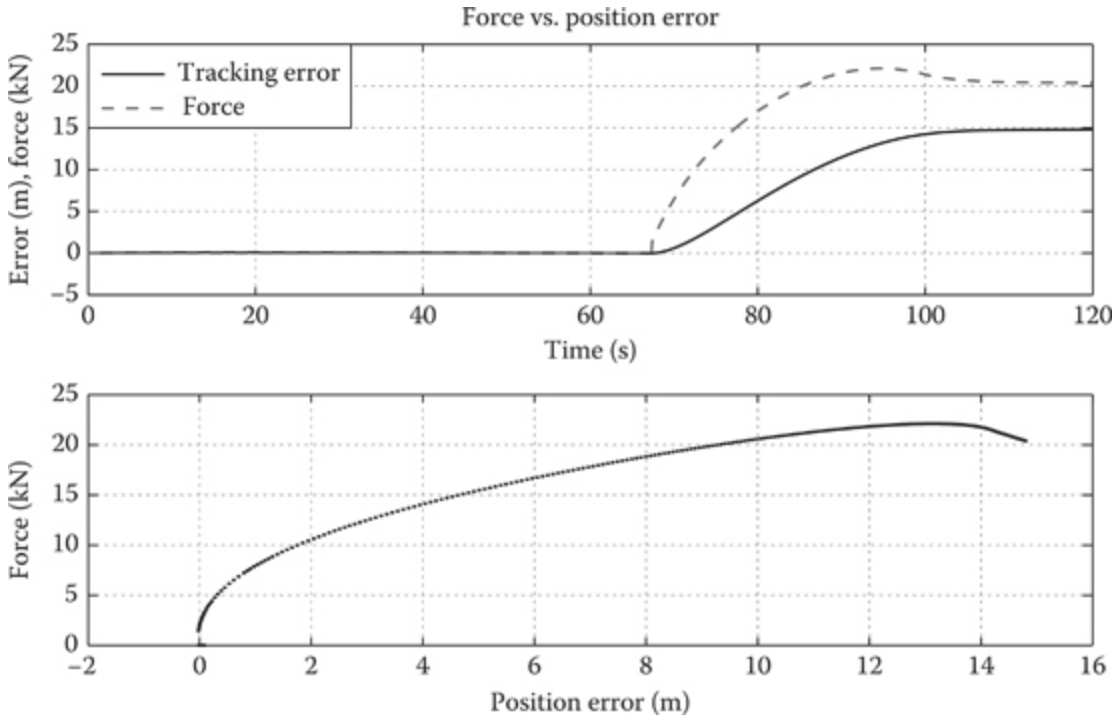



▶ Long Description for Figure 13.15

**FIGURE 13.15**

The tracking error $e_x$ versus the interacting force $\boldsymbol{F}_x$ of the planar 4RPR manipulator during a collision with a flat object; utilizing partial IDC stiffness control. ⏎

To demonstrate how a designer can effectively assign the required stiffness, consider another simulation where all parameters remain the same as before, but the stiffness coefficients are increased by a factor of 5, specifically, $\boldsymbol{K}_d = 0.5 \cdot \boldsymbol{M}(\boldsymbol{\chi})$ and $\boldsymbol{K}_p = 5 \cdot \boldsymbol{M}(\boldsymbol{\chi})$. The tracking error $e_x$ and interacting force $\boldsymbol{F}_x$ for this case are illustrated in Figure 13.16. As observed in Figure 13.16, the steady-state tracking error decreases from approximately $21.96\,m$ in the previous scenario to about $14.8\,m$ in this case. This indicates that by increasing the stiffness of the manipulator, the closed-loop system becomes stiffer, thereby reducing the object's deflection. However, the interaction force has risen from $6.05\,kN$ in the previous case to around $20.38\,kN$ in this condition. The steady-state stiffness of the system can be calculated using the formula $K = F_x/e_x$, yielding values of $0.275\,kN/m$ in the previous case and $1.38\,kN/m$ in this scenario, representing an increase by a factor of 5 as intended.




▶ Long Description for Figure 13.16

**FIGURE 13.16**

The tracking error $e_x$ versus the interacting force $\boldsymbol{F}_x$ of the planar 4RPR manipulator during a collision with a flat object; implementing partial IDC stiffness control with a stiffness requirement increased by a factor of 5. ⏎

### 13.3.4 Case Study II: Stewart–Gough Platform

In this section, the stiffness control of the Stewart–Gough platform is examined in detail. The structure of this manipulator is thoroughly described in Section 4.6.1 and illustrated in Figure 4.16. As previously mentioned, the spatial motion of the moving platform is generated by six piston-cylinder actuators. A complete kinematic analysis of

this manipulator is provided in Section 4.6, while its Jacobian analysis is presented in Section 5.7.

The dynamic analysis of this manipulator, utilizing both the Newton–Euler and virtual work methods, is provided in Section 7.5 and Section 9.6, respectively. The closed-form dynamic formulation employed for the simulations in this section is derived in Section 8.5. Various motion control strategies are implemented in Section 11.7. The subsequent discussion focuses on the implementation of the stiffness control strategy for this manipulator.

The stiffness control strategy implemented on the system incorporates partial feedback linearization alongside PD control. The control effort applied to the system is specified in Equation 13.14. For a fully parallel manipulator with six degrees of freedom, the motion variable is defined as $\boldsymbol{\chi} = [\boldsymbol{x}_p\, \boldsymbol{\theta}]$, which consists of the linear position vector $\boldsymbol{x}_p = [x_p,\, y_p,\, z_p]^T$ and the orientation vector represented by screw coordinates $\boldsymbol{\theta} = \theta[s_x,\, s_y,\, s_z]^T$. The positional tracking error is defined as $\boldsymbol{e} = [e_x,\, e_y,\, e_z,\, e_{\theta_x},\, e_{\theta_y},\, e_{\theta_z}]^T$.

The PD controller is represented by $(\boldsymbol{K}_d \boldsymbol{s} + \boldsymbol{K}_p)\boldsymbol{e}$, where $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ are $6 \times 6$ diagonal matrices that denote the derivative and proportional stiffness controller gains, respectively. The output of the controller is denoted by $\boldsymbol{\mathcal{F}}$, which consists of $\boldsymbol{\mathcal{F}} = [F_x,\, F_y,\, F_z,\, \tau_x,\, \tau_y,\, \tau_z]^T$. In practice, the computed output wrench is converted into actuator forces using the inverse of Jacobian transpose projection.

As depicted schematically in Figure 13.17, a flat surface is assumed to be positioned in the workspace of the robot,

such that the center of the robot's moving platform comes into contact with a stiff environment during part of its maneuver. In the previously analyzed maneuvers for the planar manipulator, the desired trajectory of the moving platform was modeled as a smooth cubic polynomial function for the motion variables $x_d, y_d,$ and $z_d,$ as illustrated in Figure 11.28, with a simulation duration of $2\,s$. This trajectory is modified in the simulations conducted for stiffness control to investigate collisions with the flat object within the workspace.

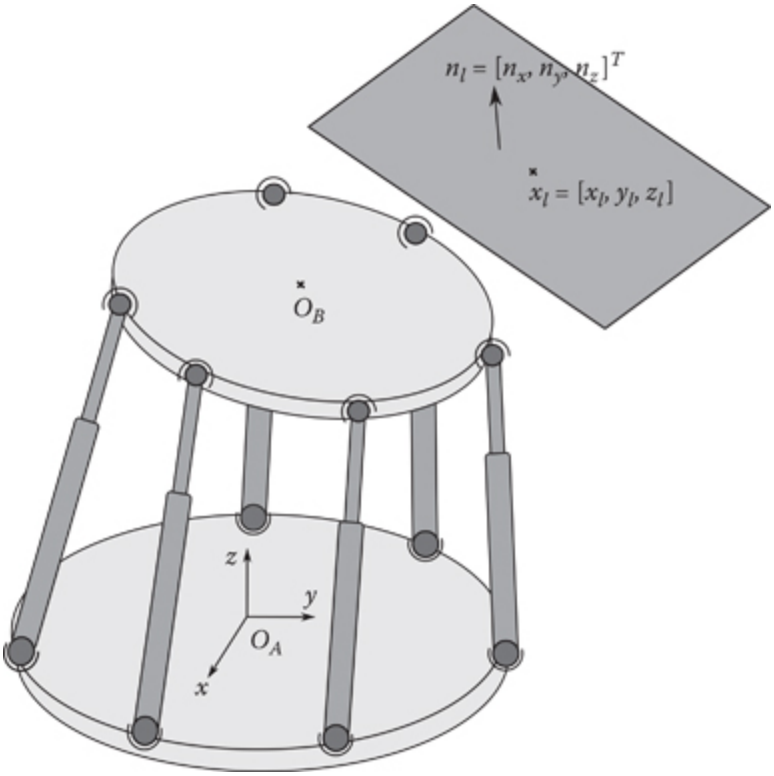
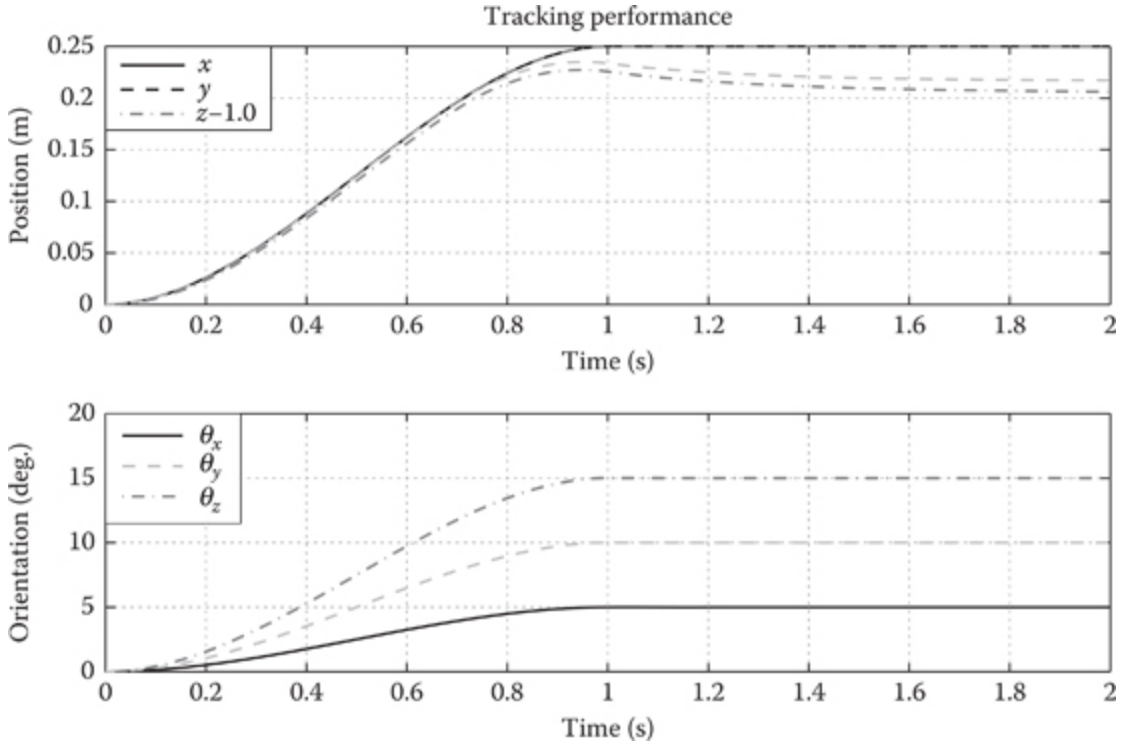



**FIGURE 13.17**
Schematic representation of a flat object located within the workspace of the Stewart–Gough platform. ⏎

In the following simulations, as shown in Figure 13.18, the desired trajectory begins at zero for $x_d$ and $y_d$, and starts from $1\,m$ for $z_d$, with a positive motion of $0.25\,m$ in $1\,s$, remaining at this position until $2\,s$. The object is positioned within the robot's workspace at a height of $0.2\,m$ above the initial position of the moving platform.




▶ Long Description for Figure 13.18

**FIGURE 13.18**
The desired and actual output trajectories of the Stewart–Gough platform during the collision with a flat object, utilizing partial IDC stiffness control. ⏎

As schematically represented in Figure 13.17, the flat object is modeled as having a planar contact surface,

described by a plane that passes through a point denoted by $\boldsymbol{x}_l = (x_l, y_l, z_l)$ and is normal to a vector $\hat{\boldsymbol{n}}_l = [n_x, n_y, n_z]^T$ in space. Consequently, the surface of the flat object is represented by the following plane equation:

$$n_x(x - x_l) + n_y(y - y_l) + n_z(z - z_l) = 0. \tag{13.27}$$

For the simulations, it is assumed that $\boldsymbol{x}_l = [0.2, 0.2, 1.2]^T$ and the normal vector to the plane is $\boldsymbol{n}_l = [1, 1, 1]^T$. This setup indicates that in the specified trajectory illustrated in Figure 13.18, the center of the moving platform makes contact with the flat object at approximately $0.7\,s$, and this contact is maintained until the simulation concludes. A subsequent duration of $1\,s$ is analyzed at the end of the simulation, during which all desired trajectories are treated as constant to assess the steady-state performance of the closed-loop force and position trajectories. The stiffness of the object is modeled using a linear stiffness gain of $\boldsymbol{K}_e = 10^5 \cdot \boldsymbol{I}_{6\times6}(\mathrm{N/m})$, along with a linear structural damping represented by $\boldsymbol{C}_e = 10^4 \cdot \boldsymbol{I}_{6\times6}(\mathrm{N \cdot m/s})$. Consequently, the contact force is defined by

$$\boldsymbol{\mathcal{F}}_e = \boldsymbol{K}_e(\boldsymbol{\chi} - \boldsymbol{\chi}_e) + \boldsymbol{C}_e(\dot{\boldsymbol{\chi}} - \dot{\boldsymbol{\chi}}_e). \tag{13.28}$$

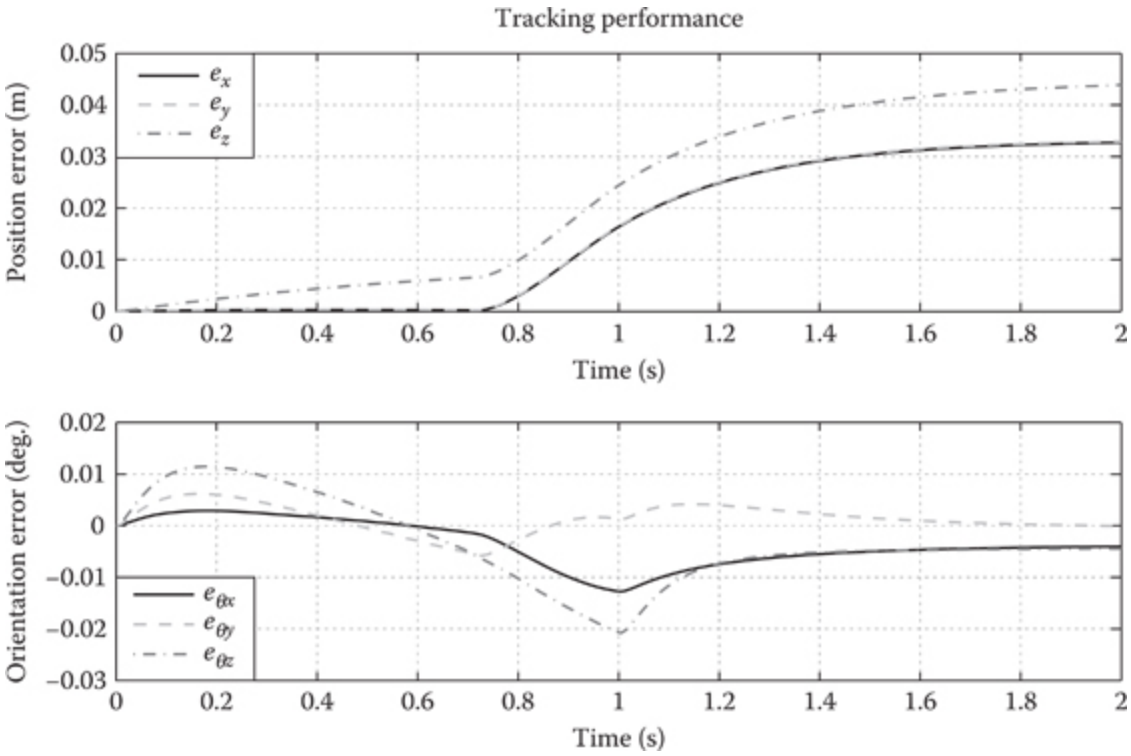
The geometric and inertial parameters employed in the simulations are presented in Table 10.2, while all initial conditions for the states are initialized to zero, except for the height of the moving platform, which is set to $z_o = 1\,m$. In the subsequent simulations, the disturbance wrench acting on the moving platform is assumed to be zero, and the gravity vector is defined as $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$. The control effort is derived from Equation 13.14. Within this controller, a weight compensation torque $\boldsymbol{\mathcal{G}}(\boldsymbol{\chi})$ is integrated into the feedback loop of the closed-loop system.

It is important to note that generating the feedback linearizing term requires knowledge of the robot's dynamic formulation, along with its kinematic and dynamic parameters. In practice, precise information about the dynamic matrices is not accessible, which leads to a 10% perturbation in all kinematic and inertial parameters being implemented in the subsequent simulations.

The PD gains applied in the simulations are configured as $\boldsymbol{K}_p = 100 \cdot \mathrm{diag}[1, 1, 1, 10, 10, 10]$ and $\boldsymbol{K}_d = 100 \cdot \boldsymbol{I}_{6\times 6}$, following several iterations of trial and error. The controller gains utilized here are considerably lower than those used for pure position control, as the aim is to manage the force-motion relationship while the moving platform is in contact with a rigid environment. Consequently, achieving high accuracy in positioning is not the primary goal with these controllers.

The final movement of the moving platform and the tracking errors are depicted in Figures 13.18 and 13.19. As

shown in Figure 13.18, the desired trajectory and the actual closed-loop motion of the system closely align prior to the collision of the moving platform with the object. At approximately $0.7\,s$, the collision occurs, and the motion of the moving platform becomes nearly constant on the object's surface.




▶ Long Description for Figure 13.19

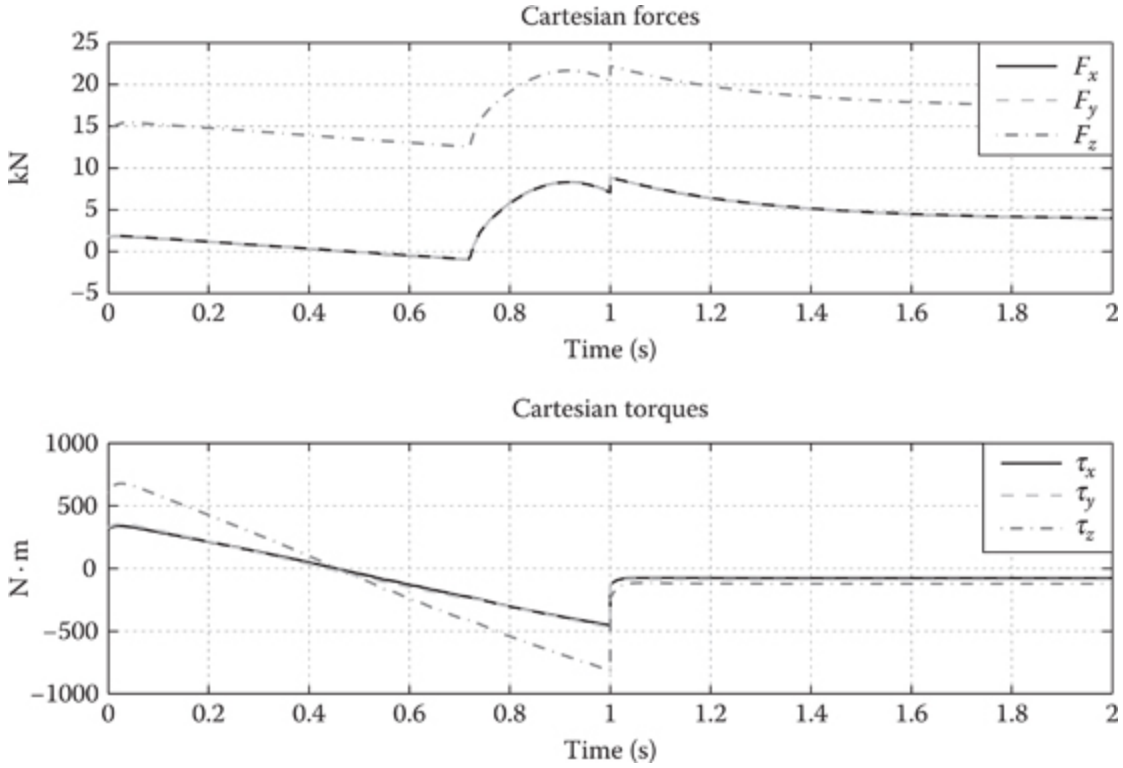
**FIGURE 13.19**

The closed-loop tracking performance of the Stewart–Gough platform during its collision with a flat object, employing partial IDC stiffness control. ⏎

More detailed observations of the moving platform's motion at this moment can be seen in Figure 13.19, which

illustrates that after the collision, the moving platform exhibits a smooth and nondestructive motion across the object's surface, causing only minimal deflection. Additionally, it is important to note that no interaction torques are induced into the simulations, which is evidenced by the relatively consistent tracking error in orientation before and after the collision. The tracking performance of this controller is inferior to that of a pure position controller, as the PD gains are set to lower values to ensure a smooth and nondestructive force-motion interaction during contact.

The Cartesian wrench $\mathcal{F}$, produced by the stiffness controller, is shown in Figure 13.20. As depicted in Figure 13.20, the Cartesian forces $F_x, F_y,$ and $F_z$ experience a rapid increase following the collision at around $0.7\,s$, whereas no interaction torques $\tau_x, \tau_y,$ and $\tau_z$ are observed in this figure. The torques displayed correspond to the linear accelerations required for the cubic polynomial orientation trajectories of the moving platform.

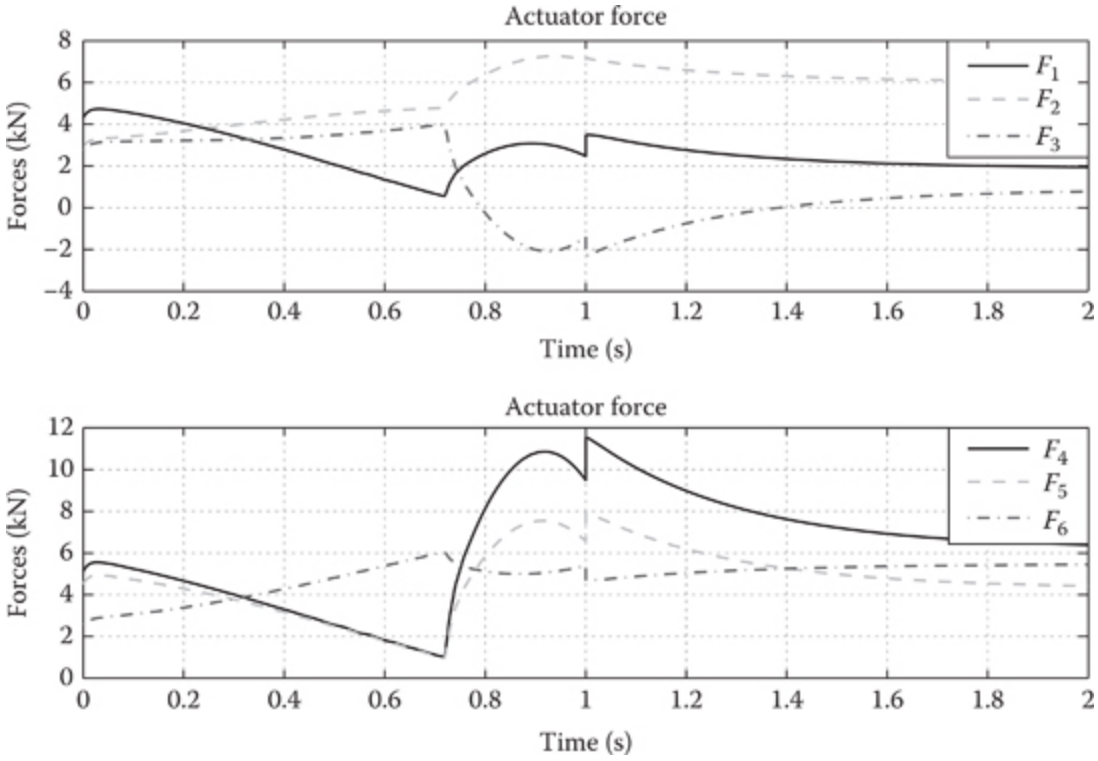



▶ Long Description for Figure 13.20

**FIGURE 13.20**

The Cartesian wrench exerted by the Stewart–Gough platform during its collision with a flat object, using partial IDC stiffness control. ⏎

Similarly, the actuator forces of the manipulator are presented in Figure 13.21. As it can be seen in this figure, the actuator forces also increase sharply after the collision. The sudden change in the actuator forces observed in this figure at $1\,s$ results from the abrupt shift in the requested acceleration for the cubic polynomial trajectories.

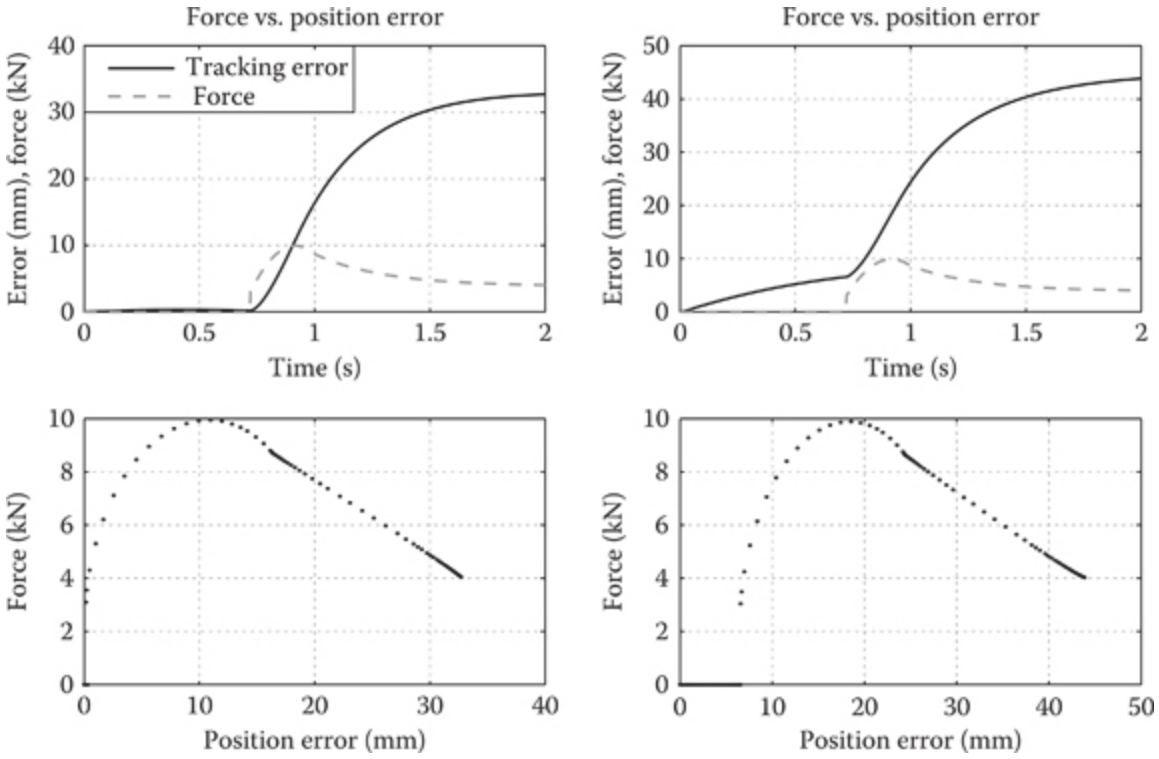



▶ Long Description for Figure 13.21

**FIGURE 13.21**

The actuator forces of the Stewart–Gough platform during its collision with a flat object, utilizing partial IDC stiffness control. ⏎

To analyze the stiffness characteristics of the manipulator in more detail, the tracking error and interacting force are plotted together in Figure 13.22. The figures on the left illustrate the force and tracking errors in the $x$ direction, while those on the right show the corresponding data for the $z$ direction. In the top plots of Figure 13.22, the tracking error and force trajectories are displayed over time, while the bottom plots depict force trajectories in relation to the tracking errors.




▶ Long Description for Figure 13.22

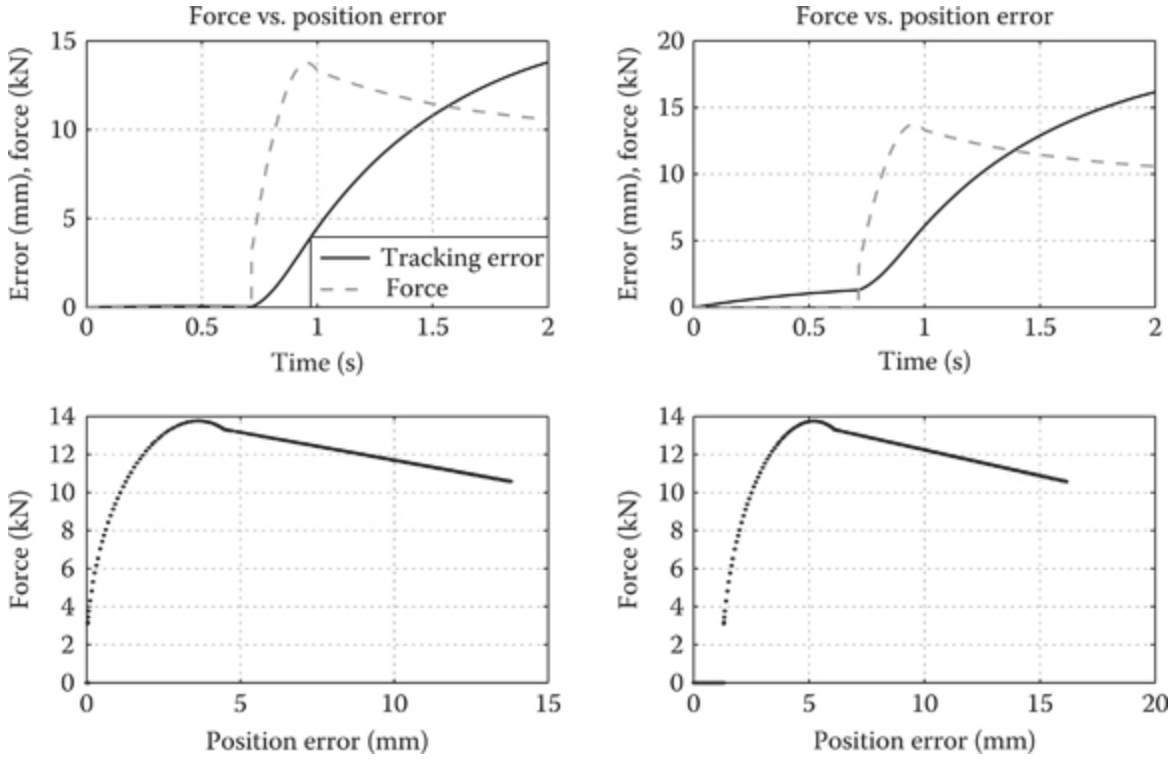
**FIGURE 13.22**

The tracking errors $e_x$ and $e_z$ plotted against the interacting forces $F_x$ and $F_z$ for the Stewart–Gough platform during its collision with a flat object, employing partial IDC stiffness control. ⏎

As observed in the top figures, the tracking errors increase monotonically, while the interacting forces surge rapidly after the collision before converging to steady-state values. The force-motion relationship is clearly represented in the bottom figures, demonstrating a quasi-linear behavior except during the transient phase. At a steady state, the force-position behavior aligns with a specific

stiffness value for the system, which can be appropriately adjusted through the parameters of the stiffness controller.

To demonstrate how a designer can effectively assign the necessary stiffness, consider another simulation where all parameters remain the same as before, but the stiffness coefficients are increased by a factor of 5: $\boldsymbol{K}_p = 500 \cdot \mathrm{diag}[1, 1, 1, 10, 10, 10]$ and $\boldsymbol{K}_d = 500 \cdot \boldsymbol{I}_{6\times 6}$. The tracking errors and interacting forces in the $x$ and $z$ directions are illustrated in Figure 13.23 for this scenario.




▶ Long Description for Figure 13.23

**FIGURE 13.23**

The tracking errors $e_x$ and $e_z$ plotted against the interacting forces $F_x$ and $F_z$ for the Stewart–Gough platform during its collision with a flat object, utilizing

partial IDC stiffness control with a fivefold increase in stiffness requirement. ⏎

As shown in Figure 13.23, the steady-state tracking error in the *x* direction decreases from approximately $32.7\,mm$ in the previous case to about $14\,mm$ in this scenario. This indicates that by increasing the required stiffness for the manipulator, the closed-loop system becomes stiffer, resulting in a reduction in object deflection. However, the interaction force in the *x* direction has risen from $4.08\,kN$ in the previous case to approximately $10.5\,kN$ in this condition. The steady-state stiffness of the system in the prior case can be calculated as $K = F_x/e_x$, yielding $0.125\,kN/mm$ in the previous scenario and $0.75\,kN/mm$ in this condition, indicating a fivefold increase as desired.

## 13.4 Direct Force Control

The stiffness control scheme developed in Section 13.3 is preferable to pure motion control strategies in situations where the robot interacts with a stiff environment. In this control approach, the force-motion relationship is finely tuned to achieve the desired stiffness requirement at a steady state, eliminating the need for force measurement. However, stiffness control has several drawbacks that limit its broader application in scenarios where the robot must precisely control contact forces during motion.

Since this control scheme does not incorporate force measurements, accurate information on contact forces can only be derived from highly precise position sensors with high resolution. Additionally, the contact force is estimated using linear compliance models of the environment, meaning that its accuracy is directly influenced by the elastic characteristics of the contact object and the precise knowledge of its linear and angular compliance coefficients.

Furthermore, the steady-state relationship between the force and position of the robot's moving platform adheres to a desired stiffness relation through set-point control. In other words, for the required stiffness relationship to be achieved, both the desired end-effector manipulator position and the force exerted on the environment must remain constant. In many robotic applications, such as grinding, the moving platform must follow a specific motion trajectory along the object surface while simultaneously applying a desired force trajectory to the object. In these cases, the stiffness control scheme often fails to meet the necessary objectives, prompting practitioners to resort to direct control methods.

If accurate monitoring and control of the contact force is required, it becomes essential to directly measure the interaction force and implement control schemes based on these force measurements. Numerous force control strategies have been developed for such applications. In this section, a typical direct force control scheme based on

inverse dynamics control is presented. As previously mentioned, the inverse dynamics control scheme serves as the foundation for the further development of robust and adaptive force control strategies, making it the primary control scheme to be discussed in this section.

It's important to note that while the primary goal of the direct force control scheme is to achieve desired force tracking, the motion of the moving platform is regarded as a secondary objective. To illustrate this, consider the cascade control topology depicted in Figure 13.2, where the outer loop is centered on force measurement and the inner loop relies on position feedback. As previously mentioned, this force control topology assumes that both force and position measurements are conducted in the task space. In this setup, the wrench $\boldsymbol{\mathcal{F}}$ acting on the moving platform can be measured using a six-degrees-of-freedom wrench sensor, while the motion variable $\boldsymbol{\mathcal{X}}$ can be obtained from a position measurement unit.

Consider a parallel robot with multiple degrees of freedom interacting with a stiff environment. In this scenario, the motion of the robot's moving platform is represented by the motion vector $\boldsymbol{\mathcal{X}}$. For a general six-degrees-of-freedom manipulator, such as the Stewart–Gough platform, this motion vector comprises both the linear motion vector $\boldsymbol{x}$ and the orientation variable $\boldsymbol{\theta}$, preferably expressed in screw coordinates: $\boldsymbol{\mathcal{X}} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{\theta} \end{bmatrix}^T$.

During the robot's interaction with the environment, a wrench $\boldsymbol{\mathcal{F}}_e$ is applied to the end-effector, which includes

the contact force $\boldsymbol{F}_e$ and torque $\boldsymbol{n}_e$, expressed as $\boldsymbol{\mathcal{F}}_e = [\boldsymbol{F}_e \quad \boldsymbol{n}_e]^T$. It is assumed that the interacting force $\boldsymbol{\mathcal{F}}_e$ is measured in the task space and utilized in the outer force feedback loop of a cascade controller structure. Additionally, it is also assumed that the motion variable $\boldsymbol{\mathcal{X}}$ is measured and employed in the inner feedback loop.

The general dynamic formulation of parallel robots, presented in Equation 13.1, is rewritten here for the reader's convenience in the absence of any disturbance wrench:

$$\boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) + \boldsymbol{\mathcal{F}}_e = \boldsymbol{J}^T\boldsymbol{\tau} = \boldsymbol{\mathcal{F}}, \tag{13.29}$$

In this equation, $\boldsymbol{M}(\boldsymbol{\mathcal{X}})$ represents the robot mass matrix, $\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})$ denotes the Coriolis and centrifugal matrix, and $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})$ indicates the gravity vector. The term $\boldsymbol{\tau}$ corresponds to the actuator force vector, while $\boldsymbol{\mathcal{F}}$ is the projection of the actuator force vector into the task space. Lastly, $\boldsymbol{\mathcal{F}}_e$ denotes the interaction wrench of the robot with the stiff environment.

As a potential method for direct force control, examine the closed-loop system illustrated in Figure 13.24. In this configuration, force feedback operates in the outer loop, while position feedback is utilized in the inner loop. This figure demonstrates that the force-tracking error is directly calculated from force measurements as $\boldsymbol{e}_F = \boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{F}}_m$ in

the outer loop, and the force controller is crafted to achieve the necessary force-tracking performance.

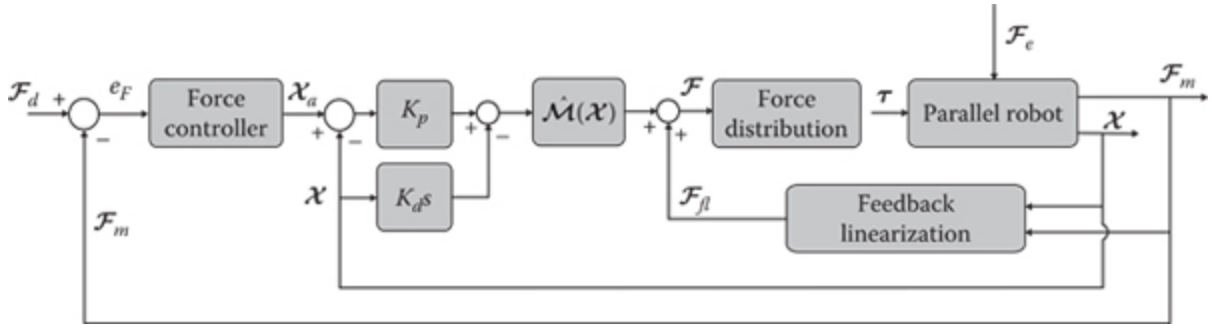



**FIGURE 13.24**
Direct force control strategy, featuring force feedback in the outer loop and motion feedback in the inner loop. ⏎

However, this control strategy does not involve a direct motion-tracking goal, as the desired motion trajectory $\boldsymbol{\mathcal{X}}_d$ is not included. Instead, an auxiliary motion trajectory $\boldsymbol{\mathcal{X}}_a$ is derived from the force control law and serves as the reference input to motion control in the inner loop. Consequently, while no specified motion trajectory is followed, the force control system suggests a motion trajectory for the robot to ensure the desired force is maintained. As previously discussed in the stiffness control strategy, when the robot interacts with a rigid environment, the auxiliary trajectory is automatically formulated along the surface of the object, such that to facilitate appropriate force interaction.

In Figure 13.24, the feedback control law produces the necessary wrench in the task space, denoted as $\boldsymbol{\mathcal{F}}$, to be exerted by the actuators, represented as $\boldsymbol{\tau}$. To apply the

control effort $\boldsymbol{\tau}$, a suitable force distribution scheme is essential in practice. As noted earlier, the force distribution block translates the required wrench in the task space $\boldsymbol{\mathcal{F}}$ to its corresponding actuator force vector $\boldsymbol{\tau}$.

It's important to mention that for a fully parallel manipulator like the Stewart–Gough platform, where the number of actuators matches the number of degrees of freedom, this mapping can be established from the Jacobian transpose of the manipulator $\boldsymbol{J}^T$ using the equation $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$, provided the configurations are nonsingular. However, if the manipulator is not entirely parallel, the force distribution scheme may require additional considerations. Redundancy resolution techniques, which are thoroughly discussed in Section 6.6, will be employed to effectively implement the force distribution scheme for redundant manipulators.

The required wrench $\boldsymbol{\mathcal{F}}$ in the direct force control strategy illustrated in Figure 13.24 relies on inverse dynamics control and comprises three primary components. Within the inner loop, the motion control strategy incorporates a feedback linearization component alongside a linear control approach, while the outer loop typically employs a linear force controller for effective force tracking. Although more sophisticated techniques, such as robust control methods, can be utilized to design the linear aspects of both the motion and force controllers for performance enhancement and analysis, basic PD and

proportional-integrator (PI) controllers are used for the motion and force controllers, respectively.

Therefore, the output wrench $\boldsymbol{\mathcal{F}}$ from the controller that is applied to the manipulator can be derived as follows:

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \boldsymbol{\mathcal{F}}_{fl}, \tag{13.30}$$

which is detailed as follows:

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}) + \boldsymbol{\mathcal{F}}_m \tag{13.31}$$

$$\boldsymbol{a} = \boldsymbol{K}_p(\boldsymbol{\chi}_a - \boldsymbol{\chi}) - \boldsymbol{K}_d\dot{\boldsymbol{\chi}}, \tag{13.32}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ represent the proportional and derivative gain matrices of the motion controller, respectively. It is important to note that for a six-degrees-of-freedom manipulator, $\boldsymbol{a}$ is a $6 \times 1$ vector. Additionally, the auxiliary motion trajectory $\boldsymbol{\chi}_a$ is produced from the force-tracking error $\boldsymbol{e}_F = \boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{F}}_m$ through the force control law. We will refer to the general force controller as the function $\boldsymbol{C}_F$. Thus, in general, the auxiliary motion trajectory $\boldsymbol{\chi}_a$ can be expressed as

$$\boldsymbol{\chi}_a = \boldsymbol{C}_F(\boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{F}}_m) = \boldsymbol{C}_F(\boldsymbol{e}_F).$$

(13.33)

While advanced control schemes, such as robust controller designs, can be employed to define the function $\boldsymbol{C}_F$, we will focus on simple proportional (P) and proportional-integrator (PI) controllers for the force feedback loop in order to develop the concept and analyze performance.

The most straightforward linear controller suitable for the force feedback loop is a pure gain controller. It is not advisable to use the derivative of the force trajectory error in this loop for two main reasons. First, force measurements are often affected by measurement noise, and introducing derivative action can amplify this noise significantly. Second, in comparison to motion variables, force is linked to the acceleration of the motion, which means that force measurement is inherently similar to acceleration measurement. Therefore, considering measurement characteristics, there is no necessity to incorporate the derivative of the force-tracking error in the feedback loop. This is akin to using motion jerk feedback, which is generally not recommended in standard motion control strategies.

On the other hand, utilizing an integrator controller is advisable, as it regulates the steady-state performance of the force trajectory. By incorporating integrator action into the proportional controller, the steady-state error of the

force trajectory can be ensured to converge to zero. Additionally, since force is comparable to acceleration, the force integral action functions similarly to velocity feedback in the motion controller, which is beneficial for appropriately shaping the performance of the force trajectory. Thus, for such scenarios, the controller function $\boldsymbol{C}_F(\boldsymbol{e}_F)$ can be derived in the time domain from the following equation:

$$\boldsymbol{C}_F(\boldsymbol{e}_F) = \boldsymbol{K}_P^{-1} \quad \boldsymbol{K}_{F_p}\boldsymbol{e}_F(t) + \boldsymbol{K}_{F_i}\int_0^t \boldsymbol{e}_F(\eta)\mathrm{d}\eta \quad . \tag{13.34}$$

Apply the Laplace transform to Equation 13.34 to express the controller in the s-domain as follows:

$$\boldsymbol{C}_F(\boldsymbol{e}_F) = \boldsymbol{K}_P^{-1}\left(\boldsymbol{K}_{F_p} + \frac{1}{\boldsymbol{s}}\boldsymbol{K}_{F_i}\right)\boldsymbol{e}_F(s), \tag{13.35}$$

where $\boldsymbol{K}_{F_p}$ and $\boldsymbol{K}_{F_i}$ denote the proportional and integrator controller gain matrices, respectively. The controller is scaled by $\boldsymbol{K}_P^{-1}$, where $\boldsymbol{K}_p$ represents the proportional gain matrix of the motion controller, to offset this gain in the forward path of the feedback control system, as depicted in Figure 13.24. It is important to note that this controller

simplifies to a pure gain controller for the force feedback when $\boldsymbol{K}_{F_i} = \mathbf{0}$ is assumed.

Now, let's express the closed-loop dynamic formulation for the manipulator in the absence of any disturbance wrench as follows:

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\chi}) + \boldsymbol{\mathcal{F}}_e &= \hat{\boldsymbol{M}}(\boldsymbol{\chi})\boldsymbol{a} + \boldsymbol{\mathcal{F}}_{fl} \\ &= \hat{\boldsymbol{M}}(\boldsymbol{\chi})\{\boldsymbol{K}_p(\boldsymbol{\chi}_a - \boldsymbol{\chi}) - \boldsymbol{K}_d\dot{\boldsymbol{\chi}}\} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}\end{aligned}$$

If the dynamic matrices are fully known, we can assume that $\hat{\boldsymbol{M}} = \boldsymbol{M}$, $\hat{\boldsymbol{\mathcal{C}}} = \boldsymbol{\mathcal{C}}$, and $\hat{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}$. Additionally, if the force measurements are free from noise, then $\boldsymbol{\mathcal{F}}_m = \boldsymbol{\mathcal{F}}_e$. Under these conditions, the closed-loop dynamic formulation simplifies to

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} = \boldsymbol{M}(\boldsymbol{\chi})\{-\boldsymbol{K}_d\dot{\boldsymbol{\chi}} - \boldsymbol{K}_p\boldsymbol{\chi} + \boldsymbol{K}_p\boldsymbol{C}_F(\boldsymbol{e}_F)\}. \tag{13.36}$$

Finally, substitute 13.34 into 13.36 and simplify to obtain the closed-loop dynamic equation as follows:

$$\ddot{\boldsymbol{\chi}} + \boldsymbol{K}_d\dot{\boldsymbol{\chi}} + \boldsymbol{K}_p\boldsymbol{\chi} = \boldsymbol{K}_{F_p}\boldsymbol{e}_F(t) + \boldsymbol{K}_{F_i}\int_0^t \boldsymbol{e}_F(\eta)\mathrm{d}\eta. \tag{13.37}$$

This equation indicates that if we assume complete knowledge of the dynamic matrices and if force measurements are free of noise, the closed-loop motion dynamics can be described by a set of second-order systems with proportional-integrator action on the force error. Thus, by selecting suitable positive definite gains for the motion PD controllers, we can shape the transient performance of the motion for quick yet stable operation. The motion will converge to zero only if the force-tracking error becomes zero; otherwise, the manipulator will continue moving to reduce the force trajectory error toward zero.

This can be understood through a physical interpretation of the control structure shown in Figure 13.24. As previously noted, the purpose of motion control in the inner loop of this control structure is the auxiliary motion variable $\boldsymbol{\chi}_a$, which is directly generated from the force error $\boldsymbol{e}_F$. This purpose is dynamically represented by the right-hand side of Equation 13.37. The inverse dynamics control structure employed in the inner loop ensures stable motion for the manipulator as long as the auxiliary variable $\boldsymbol{\chi}_a$ is not zero. Moreover, this auxiliary variable tends to zero only when the force error converges to zero. Consequently, as long as the force error is not zero, the manipulator will move toward or away from the object to achieve zero force tracking. Therefore, it can be concluded that the force tracking will asymptotically converge to zero. Incorporating integral action in the force controller ensures

zero steady-state error, even in the presence of input disturbances or measurement noise.

This control technique is widely used in practice when force tracking is the primary objective. This is due to its ability to significantly linearize and decouple the dynamic formulation of the closed-loop system for motion dynamics. The advantages of such linearization include simplifying the design process and enabling more intuitive controller tuning. Additionally, the closed-loop motion dynamic terms are independent of configuration, making it much easier to adjust the controller gains for optimal performance throughout the robot's entire workspace.

However, it's important to note that achieving good performance requires an accurate model of the system, and the overall procedure is not robust to modeling uncertainty. Furthermore, in this approach, direct motion trajectory tracking is not explicitly considered as a secondary objective, since there is no specified desired motion trajectory in the closed-loop system.

Consequently, this control topology is not advisable in situations where the robot may lose contact with the environment. In such cases, the interacting forces diminish, potentially resulting in very high accelerations being applied to the robot to reach the desired force trajectory, which can be quite dangerous in practice. If the robot loses contact, it may also lead to unpredictable movements that could cause damage to the robot itself or the surrounding environment. Therefore, it is essential to implement

sufficient measures to ensure that the robot maintains contact with the environment throughout all anticipated maneuvers.

Moreover, it is advisable to incorporate sensors and feedback mechanisms that can detect when contact is lost, enabling the system to react appropriately to avoid unsafe conditions. For instance, if a sudden change in force measurement indicates a loss of contact, the control system could be designed to switch to a safe state, reducing the robot's speed or halting its motion until contact is re-established. This proactive approach not only enhances safety but also improves the reliability and robustness of the overall control strategy in dynamic environments where contact conditions may change rapidly.

By integrating these considerations into the control design, we can create a more resilient system that effectively balances the dual objectives of force tracking and motion control, ensuring safe and efficient operation in a variety of applications, from industrial automation to robotic surgery.

### 13.4.1 Case Study I: Planar Cable Robot

This section provides a detailed examination of direct force control schemes for a planar 4-DOF RPR manipulator. The architecture of the planar manipulator under consideration is thoroughly outlined in Section 4.4.1 and illustrated in Figure 4.4. The analyses of the Jacobian and singularity for this manipulator are discussed in Section 5.5, while the

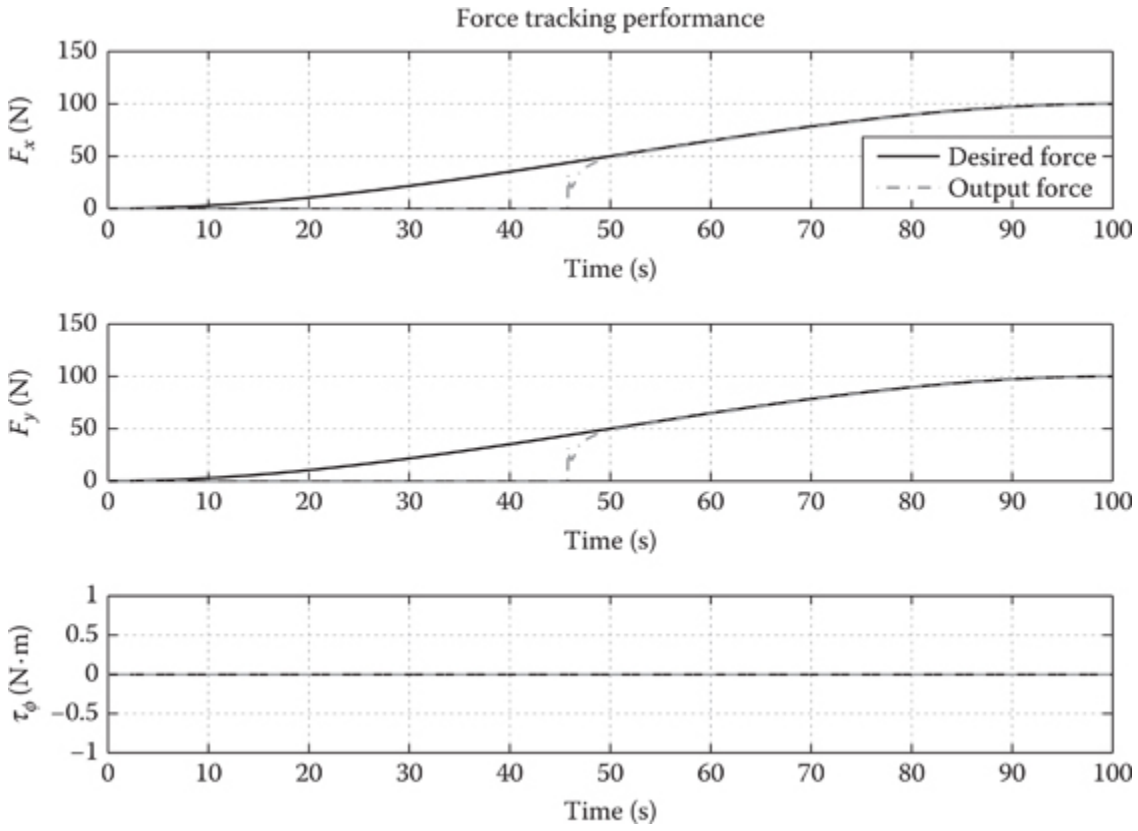
dynamic analysis is covered in Section 7.4. Various motion control structures have been implemented for this manipulator, and their performance has been thoroughly evaluated in Section 12.7. The stiffness control of this manipulator is addressed in Section 13.3.3. Following this, the direct force control scheme developed in Section 13.4 will be applied to the manipulator.

The force control scheme implemented in the system follows the structure illustrated in Figure 13.24, where force feedback is integrated into the outer loop and position feedback is utilized in the inner loop. As depicted in this figure, the force-tracking error is calculated from the force measurement using the equation $\boldsymbol{e}_F = \boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{F}}_m$ in the outer loop, with the force controller designed to achieve the desired force-tracking performance. However, this control scheme does not assign a direct objective for motion tracking, and the desired motion trajectory $\boldsymbol{\chi}_d$ is not defined. Instead, an auxiliary motion trajectory $\boldsymbol{\chi}_a$ is generated based on the force control law and serves as a reference for motion tracking. Consequently, while there is no predefined motion trajectory, the force control scheme effectively suggests a motion trajectory for the robot to ensure that the desired force tracking is maintained.

The required wrench $\boldsymbol{\mathcal{F}}$ within this force control framework is founded on inverse dynamics control and comprises three primary components. The inner loop motion control scheme incorporates feedback linearization along with a PD controller, whereas the outer loop employs

a PI force controller to achieve appropriate force tracking. The control effort applied to the system is detailed in Equation 13.31, with $\boldsymbol{a}$ specified in Equation 13.32, and $\boldsymbol{\chi}_a$ provided in Equation 13.33. It is important to note that generating these terms necessitates knowledge of the robot's dynamic formulation and its kinematic and dynamic parameters. In practical applications, precise information regarding the dynamic matrices is often unavailable, leading to the use of a 50% perturbation in all kinematic and dynamic parameters during the subsequent simulations.

As illustrated in Figure 13.25, the desired force trajectory in the $x$ and $y$ directions is set to transition from zero initial conditions to a final condition of $100\,N$ over a period of 100 seconds. Additionally, as schematically depicted in Figure 13.10, a flat surface is positioned within the robot's workspace, such that the center of the robot's moving platform collides with it during a portion of its maneuver. The shape and location of the object are identical to those used in the stiffness control scheme. For the simulations, the parameters are defined with $\boldsymbol{x}_l = (75, 75)$ and a contact surface slope of $s_l = -1$. Consequently, as seen in Figure 13.25, the moving platform makes contact with the object at approximately 46 seconds, maintaining contact until the simulation concludes.




▶ Long Description for Figure 13.25

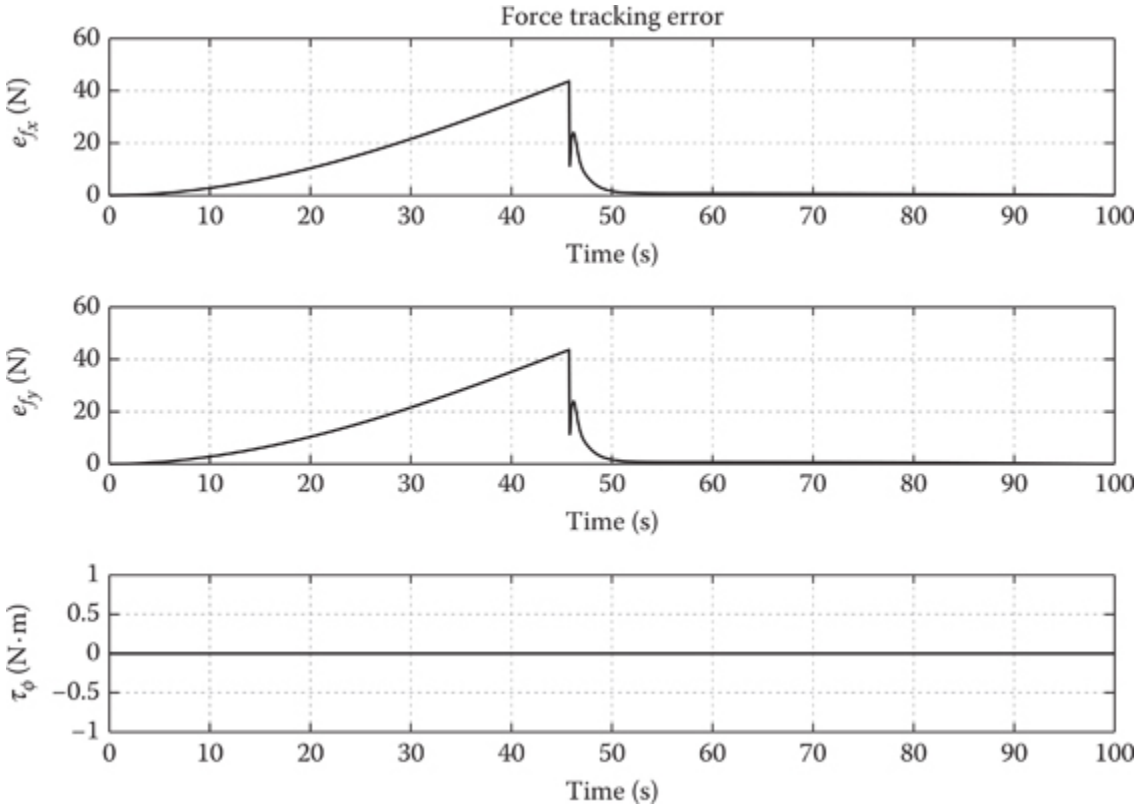
**FIGURE 13.25**

The desired and actual force trajectories of the planar 4RPR manipulator during its collision with a flat object; direct force control implementation. ⏎

The stiffness of the object is represented by Equation 13.26, where the linear stiffness gain is defined as $\boldsymbol{K}_e = 10 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N/m})$, and the linear structural damping is set to $\boldsymbol{C}_e = 10 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N \cdot m/s})$. The geometric and inertial parameters used in the simulations are outlined in Table 10.1, with all initial state conditions initialized to zero. For the subsequent simulations, the disturbance wrench acting on the moving platform is assumed to be zero, while the

gravity vector is taken as $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$. The control effort applied to the system is defined in Equation 13.31. The gains for the force controller are set to $\boldsymbol{K}_{F_i} = 0.1 \cdot \boldsymbol{I}_{3\times3}$ and $\boldsymbol{K}_{F_p} = 0.05 \cdot \boldsymbol{I}_{3\times3}$, while the gains for the inner loop position controller are adjusted to $\boldsymbol{K}_p = 20 \cdot \boldsymbol{I}_{3\times3}$ and $\boldsymbol{K}_d = 5 \cdot \boldsymbol{I}_{3\times3}$.

The final force trajectory of the robot and the corresponding force-tracking errors are depicted in Figures 13.25 and 13.26. As illustrated in Figure 13.25, while the desired force trajectory steadily increases before the collision, the forces experienced by the moving platform remain minimal and do not track the desired values. The collision occurs around 46 seconds, at which point the interaction forces quickly align with the desired force trajectory following a brief transient phase.




▶ Long Description for Figure 13.26

**FIGURE 13.26**

The closed-loop force tracking performance of the planar 4RPR manipulator during its collision with a flat object; implementation of direct force control. ⏎

Further insights into the force-tracking performance are shown in Figure 13.26, which indicates that after the collision, there is a rapid convergence of the force to the desired values. It is important to note that the desired forces in the $x$ and $y$ directions are identical, leading to nearly equal final forces in these directions. Additionally, no interaction torque is considered in the simulations,

which is evidenced by the zero torque observed in the $\phi$ direction both before and after the collision.

The motion trajectory of the robot is depicted in Figure 13.27. In this figure, the solid line represents the actual motion of the moving platform, while the dash-dotted line illustrates the auxiliary motion $\boldsymbol{x}_a$, which is generated by the force feedback law. It's important to note that in the direct force control approach, motion trajectory tracking is not explicitly designated as a secondary objective, as there is no specified desired motion trajectory in the closed-loop system. Consequently, the motion of the moving platform tends to follow the auxiliary motion created by the force feedback law.

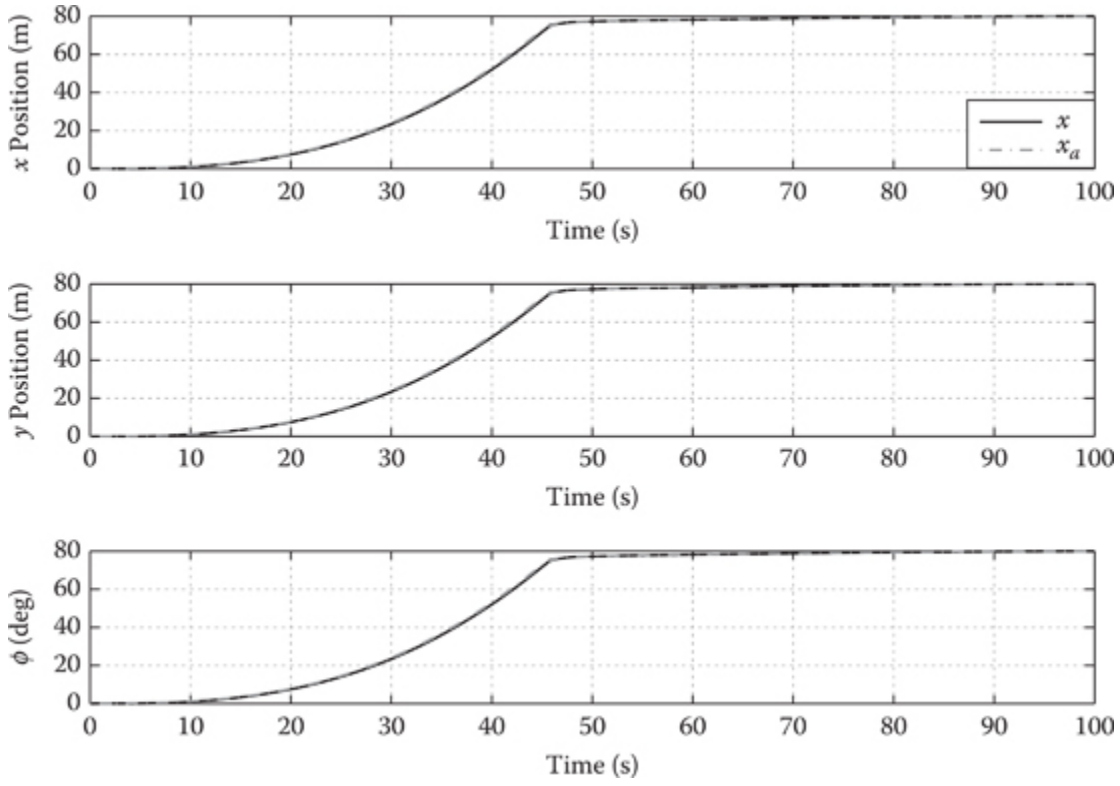
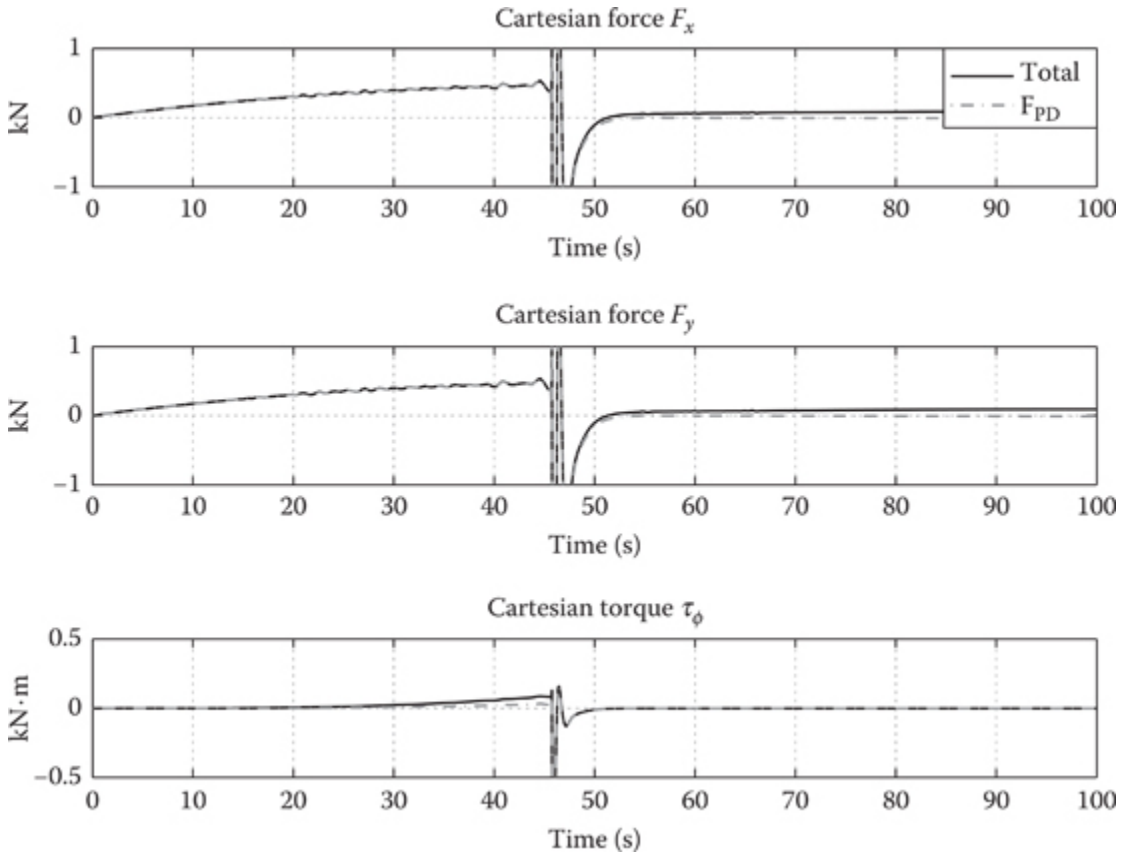



**FIGURE 13.27**

The motion trajectories of the planar 4RPR manipulator during its collision with a flat object; implementation of direct force control. ⏎

As shown in Figure 13.27, the moving platform closely adheres to the auxiliary motion variable $\boldsymbol{\chi}_a$ both before and after the collision. Prior to the collision, the interaction forces are minimal, resulting in a relatively large force tracking error. This leads to an increase in the auxiliary motion variable in the $x$, $y$, and $\phi$ directions, causing the moving platform to perform comparatively large motions. However, once the collision occurs, the force-tracking error quickly converges to zero, which significantly reduces the motion of the manipulator. This minimal motion results in a slight deflection of the object, allowing the elastic forces to adjust and to meet the required force command.

The Cartesian wrench $\boldsymbol{\mathcal{F}} = [F_x, F_y, \tau_\phi]^T$ produced by the direct force controller is depicted in Figure 13.28. In this figure, the total force is represented by a solid line, whereas the PD term is shown with a dash–dotted line. As illustrated in this figure, the Cartesian wrench rises steadily before a collision, leading to considerable movement of the manipulator. During the collision's transient phase, the robot experiences high forces, and the force controller responds by generating relatively high forces initially. This allows the interacting forces to be quickly adjusted to the desired level, after which the actuator forces needed to maintain those interacting forces near their target values are comparatively small.



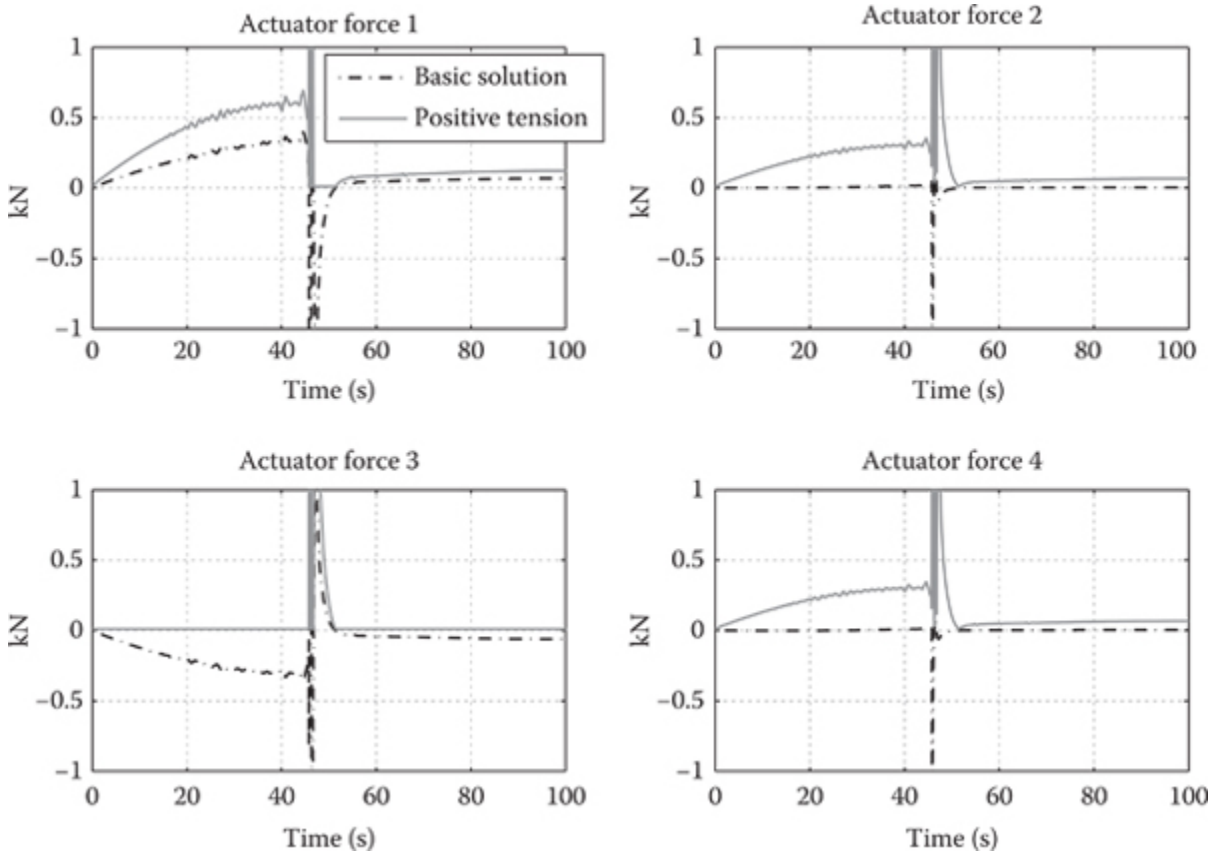

▶ Long Description for Figure 13.28

**FIGURE 13.28**

The Cartesian wrench experienced by the planar $4RPR$ manipulator during its collision with a flat object; direct force control. ⏎

Moreover, it is noted that the PD term constitutes the primary component of the control effort prior to the collision but represents a smaller fraction afterward. This is because the feedback-linearizing force $\boldsymbol{\mathcal{F}}_{fl}$, as defined in Equation 13.31, encompasses the measured force $\boldsymbol{\mathcal{F}}_m$, which quickly takes precedence over the controller force following the collision. The actuator forces for the

manipulator are also shown in Figure 13.29, where two sets of actuator forces are presented. The dash–dotted lines represent the base solution of the optimization problem introduced in the redundancy resolution scheme, as seen in Equation 6.84, while the solid line indicates the final optimization solution for the redundancy resolution problem described in Equation 6.87. The redundancy resolution process is conducted through an iterative numerical approach. As demonstrated in Figure 14.6, the actuator forces rise sharply during the transient of the collision, while the tension forces generated from the redundancy resolution scheme remain positive throughout.




▶ Long Description for Figure 13.29

**FIGURE 13.29**
The forces exerted by the actuators of the planar $4RPR$ manipulator during its collision with a flat object; direct force control. ⏎

## 13.4.2 Case Study II: Stewart–Gough Platform

This section provides a comprehensive study of direct force control for the Stewart–Gough platform. The structure of this manipulator is thoroughly explained in Section 4.6.1 and illustrated in Figure 4.16. As discussed previously, the manipulator's spatial motion is achieved through six piston–cylinder actuators. A complete kinematic analysis is available in Section 4.6, with the Jacobian analysis detailed in Section 5.7. Dynamic analyses based on both the Newton–Euler and virtual work methods are presented in Sections 7.5 and 9.6, respectively. The closed-form dynamic formulation used for simulations in this section is derived in Section 8.5. Various motion control strategies are implemented in Section 12.8, and the stiffness control strategy for this manipulator is examined in detail in Section 13.3.4. In the following discussions, the direct force control strategy is applied to this manipulator and the simulation results are reported.

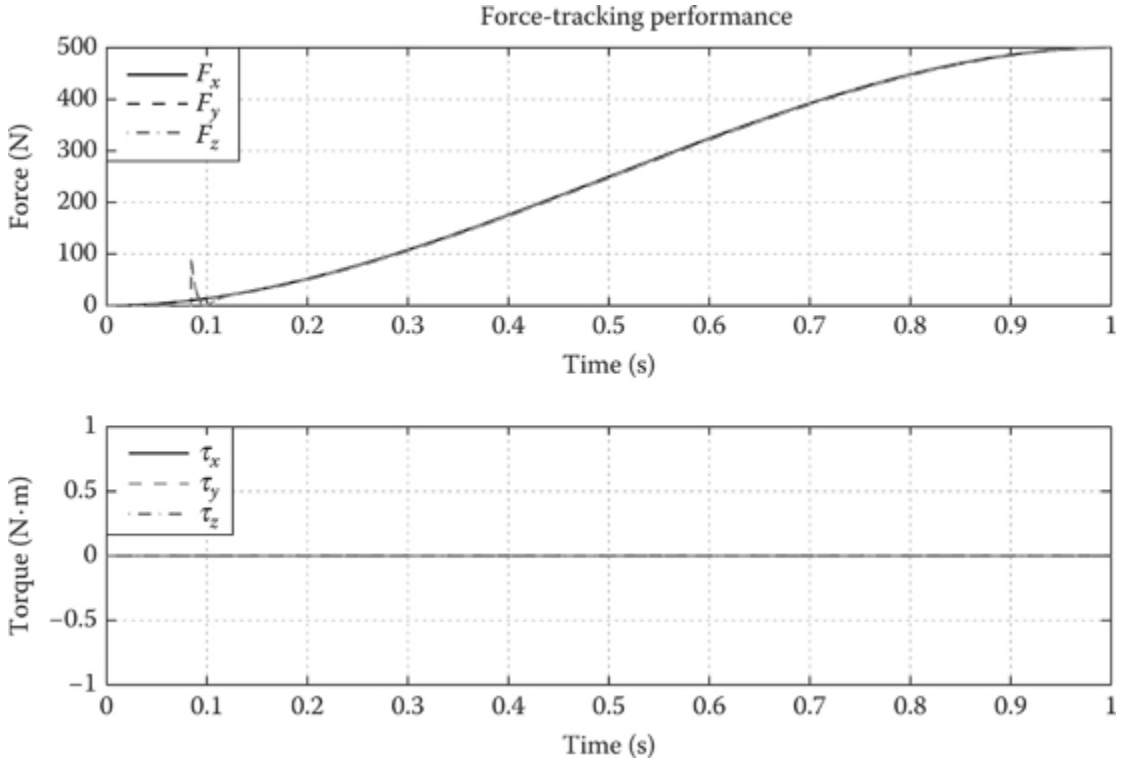
The force control scheme applied to the system is structured as shown in Figure 13.24, with force feedback in the outer loop and position feedback in the inner loop. As depicted in this figure, the force-tracking error is calculated directly from the force measurement as

$\boldsymbol{e}_F = \boldsymbol{\mathcal{F}}_d - \boldsymbol{\mathcal{F}}_m$ in the outer loop, and the force controller is designed to meet the necessary force-tracking performance. Unlike traditional approaches, this control scheme does not assign a direct motion-tracking objective, so the desired motion trajectory $\boldsymbol{\chi}_d$ is not included. Instead, an auxiliary motion trajectory $\boldsymbol{\chi}_a$ is generated from the force control law, serving as the motion-tracking reference. Through this approach, no predetermined motion trajectory is followed; rather, the force control scheme generates a motion trajectory for the robot to achieve the desired force tracking.

In this force control scheme, the required wrench $\boldsymbol{\mathcal{F}}$ is derived using inverse dynamics control and comprises three primary components. In the inner loop, the motion control scheme includes a feedback linearization term combined with a PD controller, while the outer loop utilizes a PI force controller to achieve effective force tracking. The control input applied to the system is specified in Equation 13.31, where $\boldsymbol{a}$ is defined in Equation 13.32 and $\boldsymbol{\chi}_a$ in Equation 13.33. To compute these terms, the dynamic model of the robot along with its kinematic and dynamic parameters are necessary. However, exact information on the dynamic matrices is typically unavailable, therefore, a 50% perturbation in all kinematic and inertial parameters is introduced in the following simulations.

As illustrated in Figure 13.30, the desired force trajectory along the $x$, $y$, and $z$ directions is set to increase from an initial condition of zero to a final condition of $500\,N$ within

$1\,s$. Additionally, as shown schematically in Figure 13.17, a flat surface is placed within the robot's workspace, allowing the center of the moving platform to contact a rigid environment during part of its maneuver. In these simulations, $\boldsymbol{x}_l = [0.2, 0.2, 1.2]^T$ is used as the contact point, and the plane's normal vector is set to $\boldsymbol{n}_l = [1, 1, 1]^T$. Consequently, in the defined force trajectory shown in Figure 13.30, the moving platform collides with the flat surface at approximately $0.85\,s$, with contact maintained until the end of the simulation. The object's stiffness is represented by a linear stiffness gain of $\boldsymbol{K}_e = 10^5 \cdot \boldsymbol{I}_{6\times6}, (\mathrm{N/m})$ and a structural damping factor of $\boldsymbol{C}_e = 10^4 \cdot \boldsymbol{I}_{6\times6}, (\mathrm{N \cdot m/s})$.



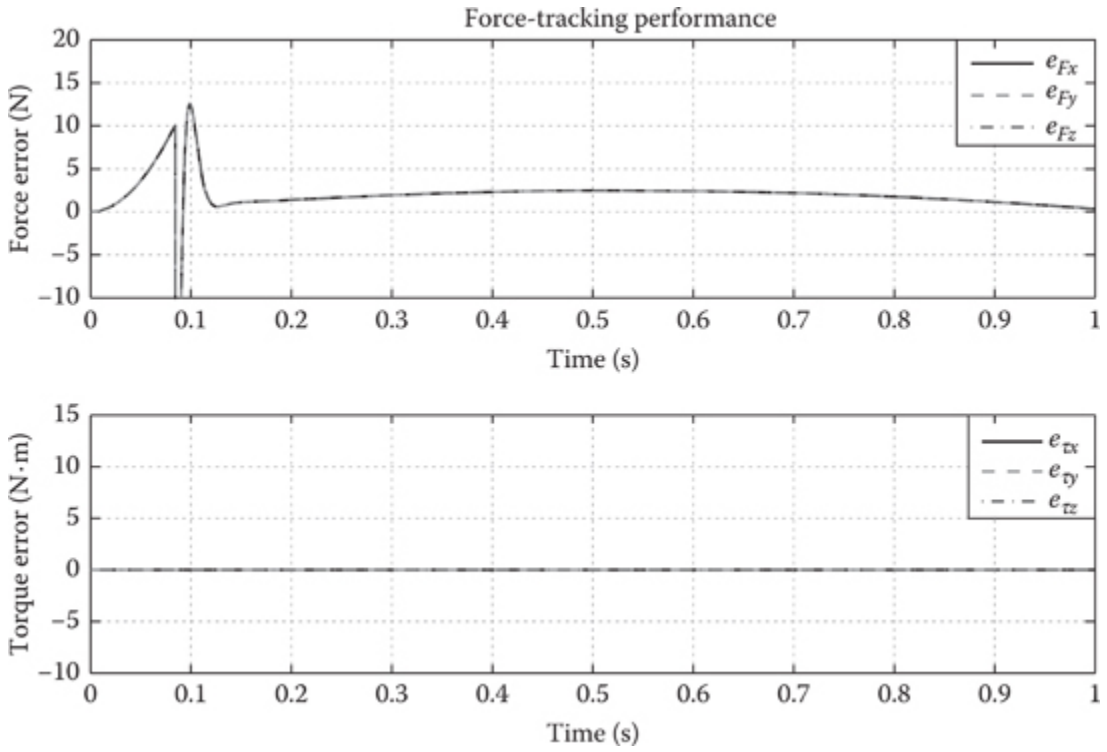

▶ Long Description for Figure 13.30

**FIGURE 13.30**
Desired and output force trajectories of the Stewart–Gough platform during collision with a flat object under direct force control. ⏎

The geometric and inertial parameters used in these simulations are listed in Table 10.2. All initial conditions for the states are set to zero, except for the initial height of the moving platform, set to $z_o = 1$ m. Throughout the simulations, the disturbance wrench on the moving platform is assumed to be zero, and the gravity vector is set as $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$. The control input applied to the system is given in Equation 13.31. The force controller gains are configured as $\boldsymbol{K}_{F_i} = \boldsymbol{I}_{6\times 6}$ and $\boldsymbol{K}_{F_p} = 0.01 \cdot \boldsymbol{I}_{6\times 6}$, while the inner loop position controller gains are set to $\boldsymbol{K}_p = 5000 \cdot \boldsymbol{I}_{6\times 6}$ and $\boldsymbol{K}_d = 50 \cdot \boldsymbol{I}_{6\times 6}$.

The final force trajectory of the robot and the force-tracking errors are shown in Figures 13.30 and 13.31. As depicted in Figure 13.30, while the desired force trajectory steadily increases prior to the collision, the forces exerted on the moving platform remain minimal, with no significant tracking of the desired forces. At approximately $0.085\,s$, the collision occurs, and the interaction forces quickly converge to the desired trajectory following a brief transient response. Further details of the force-tracking performance are shown in Figure 13.31, where the actual forces are observed to converge to the desired values shortly after the collision, exhibiting a minor transient. It is

noted that the desired forces in the $x$, $y$, and $z$ directions are set to be identical, resulting in nearly equal interaction forces along these directions. Additionally, no interaction torque is modeled in the simulations, as confirmed by zero torques in all three directions both before and after the collision.



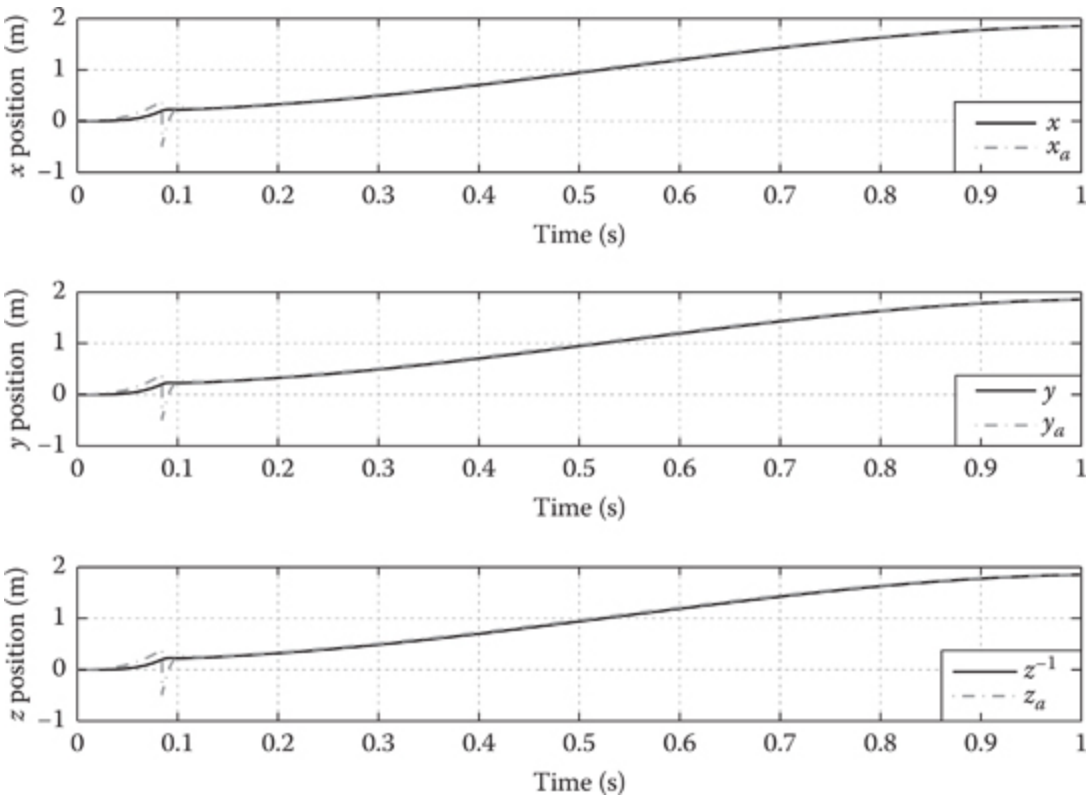

▶ Long Description for Figure 13.31

**FIGURE 13.31**

Closed-loop force-tracking performance of the Stewart–Gough platform during collision with a flat object under direct force control. ⏎

The motion trajectory of the robot is depicted in Figure 13.32. In this figure, the actual motion of the moving

platform is represented by a solid line, while the auxiliary motion variable $\boldsymbol{\chi}_a$, generated by the force feedback law, is shown with a dash-dotted line.




▶ Long Description for Figure 13.32

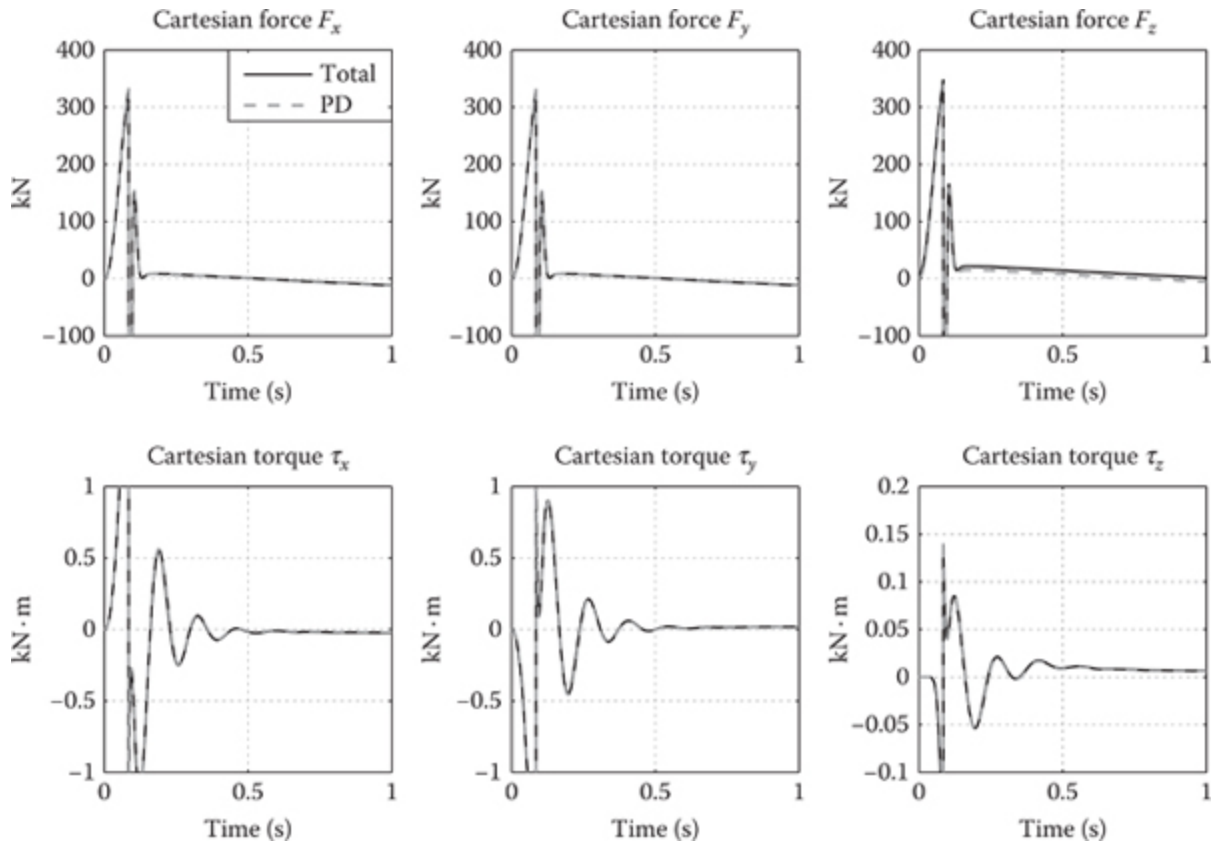
**FIGURE 13.32**

Motion trajectories of the Stewart–Gough platform during collision with a flat object under direct force control. ⏎

As observed in Figure 13.32, the moving platform successfully tracks the auxiliary motion variable $\boldsymbol{\chi}_a$ following the collision. Before the collision, the interacting forces are minimal, leading to relatively large force-tracking errors. This results in an increase in the auxiliary motion variable in the $x$, $y$, and $z$ directions prior to the

collision, causing the moving platform to execute relatively rapid motions. However, after the collision, the force-tracking error decreases significantly, allowing the manipulator's motion to effectively track the auxiliary motion variable. This movement induces a slight deflection of the object, ensuring that the elastic forces align with the required force command.

The Cartesian wrench $\mathcal{F}$ produced by the direct force controller is shown in Figure 13.33, where the total wrench is represented by a solid line, and the PD term is displayed as a dashed line. As illustrated in this figure, the Cartesian forces and torques increase monotonically before the collision, resulting in relatively rapid movements of the manipulator. During the transient phase of the collision, the robot experiences high forces, prompting the force controller to initially generate significant forces. This approach ensures that the interacting forces quickly adjust to the desired values, and afterward, the actuator forces necessary to maintain these interacting forces near their desired levels are relatively small.




▶ Long Description for Figure 13.33

**FIGURE 13.33**

Cartesian wrench of the Stewart–Gough platform during collision with a flat object under direct force control. ⏎

Additionally, it is noted that the PD term constitutes the primary component of the control effort. The actuator forces of the manipulator are also presented in Figure 13.34. As seen in this figure, the actuator forces are quite large prior to the collision, with a significant increase occurring during the transient phase of the collision. However, after collision the actuator forces required to keep the interacting forces close to their desired values remain relatively small.

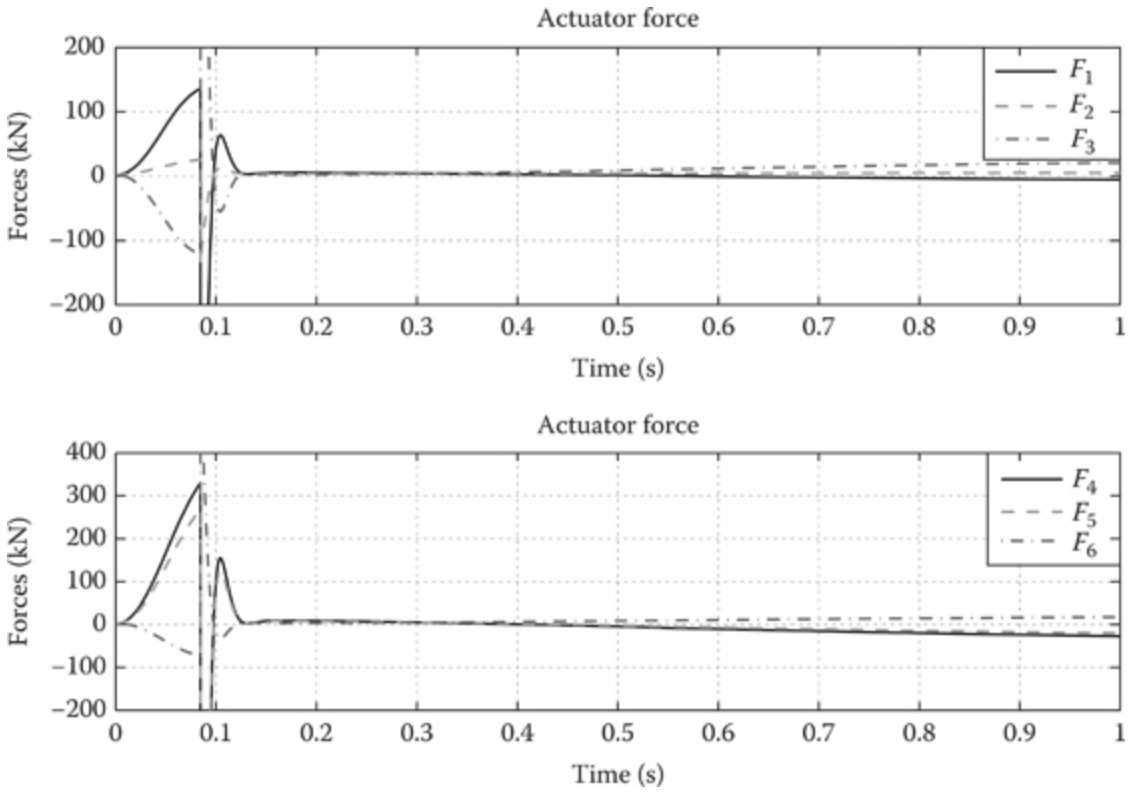



**FIGURE 13.34**

Actuator forces of the Stewart–Gough platform during collision with a flat object under direct force control. ⏎

## Problems

13.1 Consider the five-bar mechanism illustrated in Figure 4.27. The dynamic formulation of this mechanism, along with its kinematic and inertial parameters, as well as a typical motion trajectory, is provided in Problem 10.1. A flat surface is incorporated within the workspace of the mechanism, as modeled in detail in Section 13.3.3. This is utilized

in Equation 13.26, where the contact forces are represented by $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N} \cdot \mathrm{m/s})$.

a. For this mechanism, design a stiffness controller that utilizes partial feedback linearization while considering a 10% perturbation in all geometric and inertial parameters. Position the object such that a collision occurs during the motion. Simulate the closed-loop performance and adjust the stiffness controller gains to achieve an appropriate stiffness during contact.

b. Plot and analyze the closed-loop motion trajectories, tracking errors, Cartesian forces, and actuator forces over time.

c. Plot and analyze the tracking errors in the $x$ and $y$ directions against the corresponding contact forces. Evaluate the stiffness behavior of the closed-loop system during the contact transient and at steady state.

d. Modify the desired stiffness requirements and repeat part (c) to assess the performance of the controller.

13.2 Consider the five-bar mechanism depicted in Figure 4.28. The dynamic formulation of this mechanism, along with its kinematic and inertia parameters and a typical motion trajectory, is detailed in Problem 10.2.

A flat surface is positioned within the workspace of the mechanism, modeled in depth in Section 13.3.3. This surface is utilized in Equation 13.26 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\text{N} \cdot \text{m/s})$ to represent the contact forces. Apply the approach outlined in Problem 13.1 to this mechanism.

13.3 Consider the planar pantograph mechanism illustrated in Figure 4.29. The dynamic analysis of this mechanism, along with its kinematic and inertial parameters and a representative motion trajectory, is detailed in Problem 10.3. Take into account a flat surface located within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.3, and utilized in Equation 13.26 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\text{N} \cdot \text{m/s})$ to represent the contact forces. Implement the approach detailed in Problem 13.1 for this mechanism.

13.4 Consider the planar three-degrees-of-freedom 3RRR parallel manipulator depicted in Figure 4.30. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters, and a representative motion trajectory, is provided in Problem 10.4. Take into account a flat surface located within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.3, and utilized in Equation 13.26 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and

$\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times3}, (\mathrm{N} \cdot \mathrm{m/s})$ to depict the contact forces. Apply the control method described in Problem 13.1 to this manipulator.

13.5 Consider the planar three-degrees-of-freedom 4RRR parallel manipulator illustrated in Figure 4.31. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is detailed in Problem 10.5. Take into account a flat surface situated within the mechanism's workspace, similar to the one extensively modeled in Section 13.3.3, and employed in Equation 13.26 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times3}, (\mathrm{N} \cdot \mathrm{m/s})$ to illustrate the contact forces. Implement the control technique described in Problem 13.1 for this manipulator.

13.6 Consider the 3UPU parallel manipulator depicted in Figure 4.33. The dynamic formulation of this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is presented in Problem 10.7. Take into account a flat object located within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.4, and utilized in Equation 13.28 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times3}, (\mathrm{N} \cdot \mathrm{m/s})$ to depict the contact forces. Apply the control method described in Problem 13.1 to this manipulator.

13.7 Consider the three-degrees-of-freedom 3RPS manipulator illustrated in Figure 4.35. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is detailed in Problem 10.9. Take into account a flat object located within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.4, and utilized in Equation 13.28 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N} \cdot \mathrm{m/s})$ to depict the contact forces. Implement the control technique described in Problem 13.1 for this manipulator.

13.8 Consider the three-degrees-of-freedom 3PRC translational parallel manipulator depicted in Figure 4.35. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is provided in Problem 10.10. Take into account a flat surface located within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.3, and utilized in Equation 13.26 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N} \cdot \mathrm{m/s})$ to illustrate the contact forces. Apply the control method described in Problem 13.1 to this manipulator.

13.9 Consider the Delta robot illustrated in Figure 4.38. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a

representative motion trajectory, is detailed in Problem 10.11. Take into account a flat object situated within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.4, and utilized in Equation 13.28 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N} \cdot \mathrm{m}/\mathrm{s})$ to depict the contact forces. Implement the control technique described in Problem 13.1 for this manipulator.

13.10 Consider the Quadrupteron robot depicted in Figure 4.39. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is provided in Problem 10.12. Take into account a flat object located within the mechanism's workspace, similar to the one thoroughly modeled in Section 13.3.4, and utilized in Equation 13.28 with $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{4\times 4}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{4\times 4}, (\mathrm{N} \cdot \mathrm{m}/\mathrm{s})$ to depict the contact forces. Apply the control method described in Problem 13.1 to this manipulator.

13.11 Consider the five-bar mechanism described in Problem 13.1.

a. For this mechanism, specify a desired force trajectory using cubic polynomials, design a direct force controller employing complete inverse dynamics, and consider a 10% perturbation in all geometric and inertial parameters. Place the object such that a collision occurs during motion. Simulate

the closed-loop performance and fine-tune the controller gains to ensure effective force tracking following the collision.

b. Generate and analyze plots of closed-loop force trajectories, force tracking errors, Cartesian forces, and actuator forces over time.

c. Create plots and evaluate the resulting motion trajectories in relation to the auxiliary motion variables produced by the force feedback law, and intuitively analyze the behavior of the force controller.

13.12 Consider the five-bar mechanism described in Problem 13.2. Apply the control method described in Problem 13.11 to this manipulator.

13.13 Consider a planar pantograph mechanism described in Problem 13.3. Implement the control method described in Problem 13.11 to this manipulator.

13.14 Consider the planar three-degrees-of-freedom 3RRR parallel manipulator described in Problem 13.4. Apply the control method described in Problem 13.11 to this manipulator.

13.15 Consider the planar three-degrees-of-freedom 4RRR parallel manipulator described in Problem 13.5. Implement the control method described in Problem 13.11 to this manipulator.

13.16 Consider the 3UPU parallel manipulator described in Problem 13.6. Apply the control method described in Problem 13.11 to this manipulator.

13.17 Consider the three-degrees-of-freedom 3RPS manipulator described in Problem 13.7. Implement the control method described in Problem 13.11 to this manipulator.

13.18 Consider the three-degrees-of-freedom 3PRC translational parallel manipulator described in Problem 13.8. Apply the control method described in Problem 13.11 to this manipulator.

13.19 Consider the Delta robot described in Problem 13.9. Implement the control method described in Problem 13.11 to this manipulator.

13.20 Consider the Quadrupteron robot described in Problem 13.10. Apply the control method described in Problem 13.11 to this manipulator.

# 14

# *Impedance Control*

DOI: 10.1201/9781003590408-14

This chapter of the book explores the concept of impedance control, initiating with an introduction that establishes a fundamental grasp of the control scheme concept. The subsequent sections define impedance and scrutinize its complexities. The chapter then continues with two motivating demonstrations, outlining the structural framework of impedance control, and providing insights into its organizational facets. The context emphasizes that, through judicious selection of gain matrices in this impedance relation, the interaction dynamics of the robot with its environment can be tuned in both transient and steady-state states. Notably, this approach doesn't pursue the individual control of either motion or force; instead, it focuses on managing the dynamic relationship between force and motion.

Furthermore, the chapter addresses a special case within impedance control, leading to the stiffness control scheme detailed in the previous chapter. The analysis demonstrates

that in stiffness control, the static interaction of force-motion in the robot during the steady-state is examined and adjusted. Conversely, impedance control generalizes this interaction to a dynamic state, allowing for control over both transient and steady-state behavior. A comprehensive summary of force and impedance control methods is presented next in this chapter, providing readers with a synthesized understanding of the discussed concepts in force and impedance control schemes.

To illustrate the real-world application of these methods, the chapter presents two case studies. The impedance control methodologies discussed in the chapter are applied to a planar cable robot and the Stewart-Gough platform, showcasing their utility and effectiveness. Moreover, the inclusion of problems at the chapter's conclusion encourages readers to apply what they have learned, thereby deepening their understanding and skill in implementing these advanced control techniques in practical applications.

## 14.1 Introduction

In the previous chapter, we familiarized ourselves with control techniques tailored for robots interacting with their environment. Among the introduced methods, it was recognized that simultaneously controlling both the robot's configuration vector $\boldsymbol{\mathcal{X}}$ and its interactive wrench $\boldsymbol{\mathcal{F}}$ is a

complex task and, therefore, is not directly pursued. Instead, the stiffness control method was introduced, focusing on adjusting the static relationship between interactive force and resulting deflections. This approach involves a general motion control topology, where no force measurement is necessary to achieve the desired outcome. However, careful design of the desired motion trajectory and tuning of the PD controller gains are essential to adjust the stiffness characteristics of the interaction at steady state.

Conversely, in force control schemes, the primary aim is force tracking, which requires accurate force measurement to be implemented effectively. While this control strategy includes motion feedback in the inner loop of a cascade control structure, treating the motion control of the manipulator as a secondary goal, it is crucial to carefully design both the desired force trajectory and the desired motion trajectory to prevent conflicts between these two objectives in practice.

The main reason why force and position control cannot be performed simultaneously is that the number of actuators in a fully actuated robot with $n$ degrees of freedom is equal to $n$. Consequently, the number of controllable variables is confined to $n$. When aiming to independently and simultaneously control both position and force variables, we require $2n$ inputs. Considering this limitation, the impedance control method, which we will

describe in this chapter, establishes a dynamic relationship between the motion and force variables under control.

The primary concept behind *impedance control* schemes presented in this section is to adjust the dynamic relationship between force and motion variables, rather than focusing solely on a hierarchy of tracking objectives for force and position variables. In this approach, unlike stiffness control schemes, both force and position variables are measured and incorporated into the control structure. Additionally, the aim is to fine-tune the dynamic relationship between force and position quantities, rather than only their static relationship at steady state.

The notion of impedance control was introduced by Hogan in 1985 [259, 260]. Mechanical impedance is defined in analogy to the well-known concept of electrical impedance, describing the relationship between effort and flow variables. Since this relationship can be effectively analyzed in the frequency domain, the dynamic relationship between force and motion variables can be expressed as mechanical impedance. Impedance control schemes offer a control topology that enables the tuning of a system's mechanical impedance to a desired value. The following sections will first define mechanical impedance and its characteristics, then present illustrative demonstrations to elaborate on the concept of impedance control, and finally develop and analyze an impedance control scheme for parallel manipulators.

## 14.2 Definition of Impedance

The notion of impedance in electrical networks is characterized as the resistance of an electrical circuit to the flow of electric current given a specific electric voltage. When extending this concept to other disciplines, the electric voltage is generalized to the broader concept of *effort*, while the electric current is generalized to the general concept of *flow*. To be accurate, electrical impedance can be precisely defined using the voltage-to-current transfer function within a linear electrical network.

Considering that electrical resistance relates voltage to current in a static state, electrical impedance extends this concept to the dynamic behavior of the circuit. As electrical impedance is defined in linear systems with a transfer function, the dynamic characteristics and transient behavior of the system can be explored in its frequency content. Moreover, if a constant electric voltage is applied to an electrical circuit, the electrical impedance will exactly be equal to the electrical resistance of that circuit.

Electrical impedance is represented by the notation $\boldsymbol{Z}(s)$. This quantity, generally a complex variable, can be expressed in polar form using its magnitude and phase angle at each frequency: $|\boldsymbol{Z}|\angle\boldsymbol{Z}$. In this representation, the magnitude of impedance $|\boldsymbol{Z}|$ indicates the ratio of effort amplitude to current amplitude at the desired frequency, and the phase angle $\angle\boldsymbol{Z}$ represents the amount of lead or lag in the phase of input current relative to the phase of

output effort. The inverse of electrical impedance is called electrical *admittance*, representing the ratio of current to effort in the Laplace domain.

In comparison to electrical impedance, mechanical impedance is defined as follows:

---

**Definition**: In linear systems, mechanical impedance $\boldsymbol{Z}$ is defined as the ratio of the Laplace transform of mechanical effort to the Laplace transform of mechanical flow:

$$\boldsymbol{Z}(s) = \frac{\boldsymbol{F}(s)}{\boldsymbol{v}(s)} \tag{14.1}$$

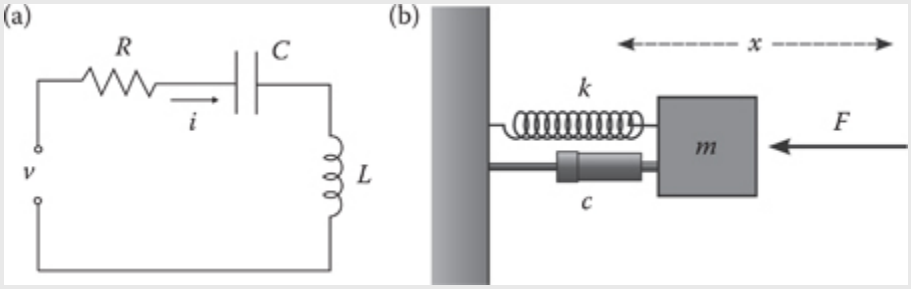
where in this context, effort in mechanical systems is interpreted as the applied force $\boldsymbol{F}$, and mechanical flow is interpreted as the velocity of body motion $\boldsymbol{v}$.

---

This definition has been presented for translational motion of a body in one degree of freedom, and it can be extended to translational motion in other directions and rotational motion about the axes of rotations. In angular motion, mechanical effort is described by torque, and mechanical flow is described by angular velocity.

In the case of a body motion in six degrees of freedom, the multi-variable impedance of the body is defined with the ratio of the six-dimensional array of wrench applied on the body $\boldsymbol{\mathcal{F}}(s)$ to the six-dimensional array representing the rate of the motion twist denoted by $\dot{\boldsymbol{\chi}}(s)$. It should also be noted that in this definition, Laplace transforms of effort and flow have been used, which are applicable only in linear systems. In robots and nonlinear systems, the differential equation relating the effort $\boldsymbol{\mathcal{F}}(t)$ and mechanical motion flow $\dot{\boldsymbol{\chi}}(t)$ can be defined as the system impedance.

**Example 14.1 (Electrical Impedance)** *Obtain the impedance of the RLC electrical circuit shown in Figure 14.1a.*




▶ Long Description for Figure 14.1

**FIGURE 14.1**
Comparison of electrical impedance in (a) an electrical RLC circuit with (b) a mechanical body-spring-damper system. ⏎

**Solution:** The relationship between voltage and current in an RLC electrical circuit can be determined using the following differential equation.

$$v = L\frac{di}{dt} + R\,i + \int_0^t \frac{1}{C}\,i(\tau)d\tau \tag{14.2}$$

In this equation, $L$ represents the inductance of the inductor, $R$ is the electrical resistance of the circuit, and $C$ denotes the capacitance of the capacitor in the circuit. Furthermore, $v$ represents the supply voltage, and $i$ is the current flowing through the circuit. Taking the Laplace transform of this equation determines the electrical impedance of the circuit.

$$Z(s) = \frac{v(s)}{i(s)} = Ls + R + \frac{1}{Cs} \tag{14.3}$$

**Example 14.2 (Mechanical Impedance)** ⏎ *Determine the impedance of the mechanical system composed of a mass-spring-damper, shown in Figure 14.1b.*

**Solution:** The relationship between force and motion in the mass-spring-damper mechanical system can be determined using the following differential equation.

$$m\ddot{x} + c\dot{x} + kx = f$$

(14.4)

In this equation, $m$ represents the mass of the system, $c$ is the linear damping coefficient of the damper, $k$ is the spring stiffness coefficient, and $f$ is the external excitation force applied to the system. If you take the Laplace transform of this equation, the mechanical impedance of the system can be determined as follows.

$$Z(s) = \frac{f(s)}{v(s)} = ms + c + \frac{k}{s}$$

(14.5)

These two examples illustrate that despite structural differences in the physics of electrical and mechanical systems, both exhibit impedance relationships that are similar to each other. This comparison helps electrical engineers comprehend the components of mechanical impedance more easily. By leveraging this analogy, the literature related to the definition of impedance elements in a system is unified. The types of impedance in a system can be classified as follows.

**Inductive Impedance:** The system has inductive impedance, if and only if, the impedance magnitude at $s = 0$ is equal to zero: $|\boldsymbol{Z}(0)| = 0$.

**Resistive Impedance:** The system has resistive impedance, if and only if, the impedance magnitude at $s = 0$ is constant and non-zero: $|\boldsymbol{Z}(0)| = R$.

**Capacitive Impedance:** The system has capacitive impedance, if and only if, the impedance magnitude approaches infinity as $s \to 0$: $\lim_{s \to 0} |\boldsymbol{Z}(s)| = \infty$.

Thus, for the mechanical system in Example 14.2, the mass characterizes the inductive impedance, the viscous damping coefficient characterizes the resistive impedance, and the spring stiffness coefficient characterizes the capacitive impedance. With this definition, the environment with which the robot interacts can be classified into inductive, resistive, and capacitive impedances. An environment with high stiffness can be represented with a large capacitive impedance. On the other hand, an environment with significant structural damping can be represented with a large resistive impedance.

The force-motion relationship in a scenario where a user is holding a surgical haptic device can be modeled by combining inductive, resistive, and capacitive impedances. By defining different impedance elements, the interaction characteristics of users with such a device can be specified. If a user holds the haptic device firmly, its capacitive impedance will dominate over the other two types of impedance. Now, if we want to regulate the dynamic relationship between the robot and the environment, we can adjust the inductive, resistive, and capacitive

impedance properties through a control system. In this way, regardless of the user's interaction with the environment or whether the robot is interacting with a stiff or compliant environment, the impedance characteristics of the system can be brought close to desired values. In the upcoming section, two motivating demonstrations are presented to highlight the importance and nature of controlling impedance.

## 14.3 Impedance Control Concept

So far, it has been explained that in impedance control methods, the goal is to adjust the dynamic interaction relationship between the robot and its surrounding environment. These methods rely on a suitable control topology that stabilizes the system by measuring force and motion, adjusting the dynamic relationship between these two quantities. To better understand this interactive relationship and desirable impedance values in robotic applications, let's consider the development of this concept by presenting a motivating demonstration.

### 14.3.1 Automated Fish Fillet Production

For this purpose, consider the use of industrial robots in the automated production of fish fillets. In a fish production and packaging factory, there are various types of fish with

different sizes, and the production of fish fillets, as one of the premium products of the factory, faces high demand from customers. In the production of fish fillets, the fish is usually sliced along its spinal cord into two parts, and the fish bones are separated from its flesh. Then, it is necessary to carefully remove the fish skin from the flesh. This part of the operation is very precise and delicate, and automating it with robotic systems poses considerable complexity.

To understand the complexities of this operation, let's review how skilled individuals remove the skin from the flesh. A skilled person typically places a piece of the separated fish flesh on the cutting table with the skin side down. Then, using a sharp knife, he or she starts separating the skin from the fish flesh, beginning from the tail section. The movement of the skilled person's hand is controlled with dexterous angular knife cuts in back-and-forth motions. When the knife moves toward the skin, the skilled person feels more speed and less force, adjusting the knife angle toward the fish flesh. Immediately after the knife penetrates the fish's flesh, he or she feels more force and less motion, tilting the knife toward the skin. This way, by applying precise and rapid back-and-forth movements, the skilled person prevents the knife from tilting excessively toward the skin, avoiding tearing the skin, or conversely, preventing the knife from moving too much toward the flesh and wasting a portion of it.

The method of fish skinning effectively demonstrates the simultaneous sensation of force and movement by the skilled person and the control of the relationship during the motion. If the movement of his or her hand results in feeling more speed with less force, the knife tilts excessively toward the skin, requiring a change in the movement direction. On the other hand, if he or she feels slower motion and increased the force, the knife tends to tilt toward the fish flesh. With extensive practice, the skilled person moves their hand with the necessary agility to experience the desired impedance profile during the motion.

To perform this operation automatically by robotic systems, precise measurement of the robot's motion and force is essential. Measuring the robot's motion alone is not sufficient because fish pieces come in various sizes and shapes. Moreover, the skin of the fish is very thin compared to its flesh dimensions, requiring highly accurate measurement systems. Therefore, the possibility of defining a predetermined path for the robot's motion is not feasible. Although force control works better than motion control methods in this application, practical use of them is not possible due to the need for a desired motion path in these approaches, which is not available beforehand.

If the mechanical impedance of the robot can be adjusted to the desired value during motion, this control method, similar to the skilled hand movements, will be able to separate the skin from the flesh without causing tearing.

The desired impedance threshold of the robot in this application can be determined from the impedance of the knife motion on the fish flesh. In a situation where the knife moves more toward the fish skin, the force decreases, and the speed of movement increases. This implies a proportional decrease in mechanical impedance. The appropriate impedance of the robot's motion can be adjusted by the structure of a multivariable mass-spring-damper model.

$$\boldsymbol{Z}_d(s) = \boldsymbol{M}_d s + \boldsymbol{C}_d + \frac{1}{s}\boldsymbol{K}_d \tag{14.6}$$

In this context, $\boldsymbol{Z}_d$ is the desired impedance of the robot, which has the desired inductive $\boldsymbol{M}_d$, the desired resistive $\boldsymbol{C}_d$, and the desired capacitive impedance $\boldsymbol{K}_d$. Using this simple impedance structure, the fish fillet production process can be automated by industrial robots, while the precise determination of the impedance model components is obtained through experiments.

### 14.3.2 Flight Simulator Yoke

For the second case study, let's examine the control system of a flight simulator yoke. Flight simulators are equipped with various subsystems designed to create a realistic flying experience for pilots. They are used worldwide to train pilots for both routine and exceptional flight

situations. Four primary subsystems work together to replicate real flight conditions. A sophisticated visual display simulates the scenery a pilot would see during flight, while a Stewart–Gough platform generates the accelerations felt by the pilot, particularly during takeoff and landing. By providing integrated visual and motion feedback, these systems create a nearly authentic flight experience. The other two subsystems include a flight computer that models aerodynamics and the resulting motions, as well as simulated flight instruments that closely resemble the gauges and controls found in an actual cockpit. One important control in the cockpit is the yoke, which is typically a handle used to manage the aircraft's elevation and attitude.

To simplify the analysis, consider a one-degree-of-freedom yoke used for controlling the elevation of the airplane. In practice, this yoke is a vertical handle that the pilot grips firmly during flight. Pulling the handle toward the pilot increases elevation, while pushing it away decreases flight height. In real aircraft, the yoke is connected to the elevons through hydraulic or mechanical systems, allowing the pilot to sense the aerodynamic forces acting on the elevons.

When the yoke is in its neutral position, the elevon surfaces align with the wing surface, resulting in no elevation commands being sent to the aircraft. By pulling or pushing the yoke, the elevon surfaces tilt relative to the wing, which generates elevation commands for the

airplane. Additionally, the aerodynamic forces acting on the elevons are directly felt by the pilot through the yoke. The greater the elevation command, the more force the pilot experiences.

The relationship between the yoke angle and the force sensed by the pilot can be derived from the aerodynamic interactions with the wings and elevons. However, this relationship can often be approximated as a constant stiffness for typical operational elevation commands during flight. These characteristics are essential considerations for the control system design of a flight simulator yoke.

The second requirement for a flight simulator yoke is to replicate the behavior of a real yoke when the pilot releases it during operation. Upon release, the yoke should naturally return to its idle position due to the aerodynamic forces acting on the elevons. For instance, if the pilot holds the yoke at an angle $\theta \neq 0$ with respect to its neutral position, upon releasing the yoke, it should smoothly move back to the idle position ($\theta = 0$) without chattering or unwanted oscillations. This desired motion can be modeled as the time response of a second-order system, which exhibits a well-defined damping behavior.

In control theory, such a system can be characterized by a damping ratio of $\xi = 0.707$. This specific damping ratio provides a balance between a smooth response and a reasonably fast return to the idle position, resulting in minimal overshoot. Consequently, the dynamics of the yoke's motion can be designed to ensure that the response

is stable and adequately represents the natural behavior observed in real aircraft yokes. This characteristic is crucial for enhancing the realism of the flight simulator experience for the pilot.

A third requirement for the flight simulator yoke is to ensure that it remains stable when accidentally bumped or hit by the pilot while at its idle position. This stability is crucial to prevent any chattering or unwanted oscillations in its motion, allowing for a more realistic and controlled experience during flight simulation.

Lastly, the force-position relationship for the flight simulator yoke must be designed to be independent of how the pilot holds the yoke. This means that the control system should effectively replicate the characteristics of a real yoke for all pilots, regardless of the firmness with which they grip the yoke—whether they hold it tightly or more loosely.

In technical terms, this requirement implies that the impedance of the pilot's hand should not affect the desired impedance characteristics of the flight simulator yoke. The control system must account for variations in grip while still providing consistent feedback that mimics the dynamics of a real aircraft yoke. This approach ensures that all pilots can have a similar experience when using the flight simulator, making it an effective training tool.

In examining the behavior of different motion and force control schemes for the flight simulator yoke, we can

analyze their suitability in meeting the established objectives:

Pure Motion Controller: This type of controller is effective for stability and regulation requirements, addressing the second and third objectives—ensuring a smooth return to the idle position and maintaining stability during accidental bumps. However, a pure motion control scheme fails to address the primary objective of controlling the force or the force-motion relationship. As a result, while it provides satisfactory motion behavior, it does not fulfill the requirement for accurate force feedback and interaction.

Stiffness Control: Stiffness control is a promising approach for achieving the first objective, as it can effectively adjust the force-motion relationship. However, since this scheme does not incorporate force measurement, it cannot reliably satisfy the remaining three objectives. Specifically, without feedback on the force applied by the pilot, the system may struggle to provide a realistic and adaptive response to different grip strengths or interactions, compromising the overall user experience.

Direct Force Controller: This scheme appears to be the most appropriate for this application, as it can simultaneously consider both force and motion objectives. A pure force controller allows for the direct measurement and regulation of forces, enabling the system to adapt to varying pilot interactions. However,

challenges remain regarding the primary force-motion requirement and the independence from human operator characteristics. While it can effectively manage force feedback, it may not fully account for how different pilots interact with the yoke, potentially leading to inconsistencies in the user experience.

Overall, while each control scheme has its strengths, they also have limitations in satisfying all four objectives of the flight simulator yoke system. In conclusion, implementing an impedance-based controller that adjusts the operational impedance of the closed-loop system to a specific value can effectively meet all four previously mentioned objectives. The desired operational impedance can be represented by a combination of resistive and capacitive impedances, resulting in a desired linear impedance for the closed-loop system, expressed as follows:

$$\boldsymbol{Z}_d(s) = \boldsymbol{M}s + \boldsymbol{C}_d + \frac{1}{s}\boldsymbol{K}_d \tag{14.7}$$

In Equation 14.7, $\boldsymbol{Z}_d$ represents the target impedance of the closed-loop system, which consists of the system's inherent inductive impedance matrix $\boldsymbol{M}$, the desired resistive impedance matrix $\boldsymbol{C}_d$, and the desired capacitive impedance matrix $\boldsymbol{K}_d$. This configuration allows for a smooth and stable response with $\xi = 0.707$ from the beginning. The impedance control structures discussed in

the next section can be utilized to adjust the closed-loop impedance of the system appropriately to align with this desired impedance.

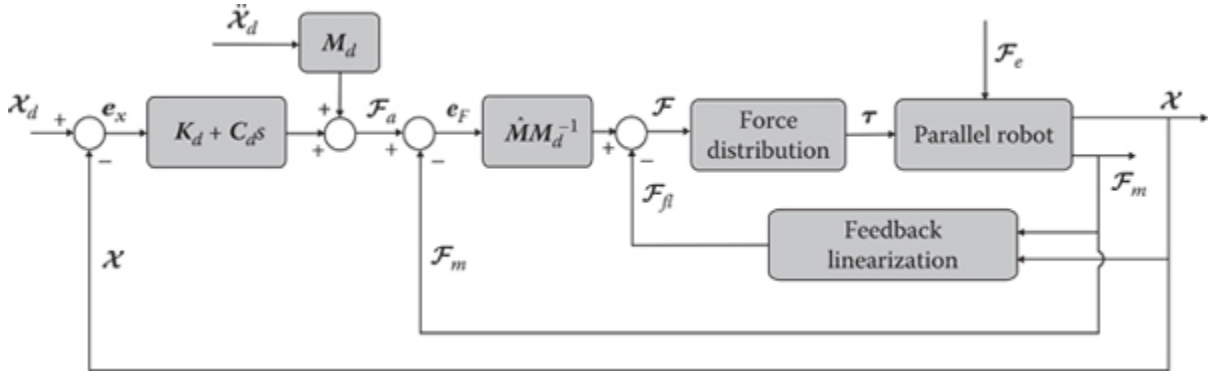
## 14.4 Impedance Control Structure

Consider a parallel robot with multiple degrees of freedom interacting with a rigid environment. In this scenario, the motion of the robot's end effector is represented by the motion vector $\boldsymbol{\mathcal{X}}$, while the wrench interacting with the end effector is denoted by $\boldsymbol{\mathcal{F}}_e$. The interacting force is measured in the task space and is utilized in the inner force feedback loop. Additionally, the motion variable $\boldsymbol{\mathcal{X}}$ is measured and used in the outer feedback loop. The general dynamic formulation of parallel robots, excluding any disturbance wrench, is provided in Equation 13.29.

Within the impedance control scheme, the main goal is to regulate the motion–force dynamic relationship. Since force tracking is not the primary aim, it is recommended to employ a cascade control structure, featuring motion control feedback in the outer loop and force feedback in the inner loop. As previously described, Figure 13.5 can be viewed as a representation of this cascade control topology for the system, where both measured variables exist in the task space. In this configuration, the primary controller is a motion controller, and the secondary controller is a force-tracking controller. Consequently, when the manipulator is

not in contact with a stiff environment, position tracking is ensured by the primary controller. However, when an interacting wrench $\mathcal{F}_e$ is applied to the moving platform, this structure can be designed to manage the force–motion dynamic relationship.

This configuration is recommended when a desired impedance relationship between the force and motion variables is necessary, comprising desired inductive, resistive, and capacitive impedances, as specified in Equation 14.6. As shown in Figure 14.2, the motion-tracking error is directly calculated from motion measurements by $\boldsymbol{e}_x = \boldsymbol{\mathcal{X}}_d - \boldsymbol{\mathcal{X}}$ in the outer loop, and the motion controller is designed to meet the required impedance structure outlined in Equation 14.6.




▶ Long Description for Figure 14.2

**FIGURE 14.2**

Impedance control configuration, featuring motion feedback in the outer loop and force feedback in the inner loop. ⏎

Furthermore, a direct force-tracking objective is not included in this control scheme, which means that the desired force trajectory $\mathcal{F}_d$ is not present. Instead, an auxiliary force trajectory $\mathcal{F}_a$ is generated from the motion control law and serves as the reference for force tracking. This approach allows for no specified force trajectory to be tracked, while the motion control scheme provides a force trajectory for the robot to maintain the desired impedance regulation.

The required wrench $\mathcal{F}$ in the impedance control scheme illustrated in Figure 14.2 is derived from inverse dynamics control and comprises three primary components. In the inner loop, the force control scheme integrates a feedback linearization component along with an adjustment of the mass matrix. Meanwhile, the outer loop typically employs a linear motion controller that adheres to the desired impedance requirements, as indicated in Equation 14.6. Although various impedance structures can serve as the foundation for the control law, in Figure 14.2, a linear impedance relationship between the force and motion variables is established, incorporating desired inductive $\boldsymbol{M}_d$, resistive $\boldsymbol{C}_d$, and capacitive impedances $\boldsymbol{K}_d$, as specified in Equation 14.6.

As illustrated in Figure 14.2, the feedback control law produces the required wrench $\mathcal{F}$ to be exerted by the actuators $\boldsymbol{\tau}$ in the task space. However, to apply the control effort $\boldsymbol{\tau}$, it is essential to utilize an appropriate force distribution scheme in practice. As previously described,

the force distribution block converts the required wrench in the task space, $\boldsymbol{\mathcal{F}}$, into the corresponding actuator force vector $\boldsymbol{\tau}$. It is important to note that for a fully parallel manipulator, such as the Stewart–Gough platform, where the number of actuators matches the number of degrees of freedom, this mapping can be derived from the Jacobian transpose of the manipulator, expressed as $\boldsymbol{\tau} = \boldsymbol{J}^{-T}\boldsymbol{\mathcal{F}}$, at nonsingular configurations.

However, if the manipulator is not completely parallel, the force distribution scheme may require additional computations. Redundancy resolution techniques, which are discussed in detail in Section 6.6, can be employed to effectively implement the force distribution scheme for redundant manipulators.

According to Figure 14.2, the output wrench $\boldsymbol{\mathcal{F}}$ from the controller, which is applied to the manipulator, can be formulated as

$$\boldsymbol{\mathcal{F}} = \hat{\boldsymbol{M}}\boldsymbol{M}_d^{-1} \cdot \boldsymbol{e}_F + \boldsymbol{\mathcal{F}}_{fl}, \tag{14.8}$$

where

$$\boldsymbol{e}_F = \boldsymbol{\mathcal{F}}_a - \boldsymbol{\mathcal{F}}_m \tag{14.9}$$

$$\boldsymbol{\mathcal{F}}_a = \boldsymbol{M}_d\ddot{\boldsymbol{\chi}}_d + \boldsymbol{C}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_d\boldsymbol{e}_x,$$

(14.10)

Similarly, the feedback linearizing term is expressed as

$$\boldsymbol{\mathcal{F}}_{fl} = \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}}) + \boldsymbol{\mathcal{F}}_m.$$

(14.11)

In Equation 14.10, $\boldsymbol{M}_d$ represents the desired inductive impedance, $\boldsymbol{C}_d$ signifies the desired resistive impedance, and $\boldsymbol{K}_d$ indicates the desired capacitive impedance matrices. We will now express the closed-loop dynamic formulation for the manipulator, assuming there is no disturbance wrench, as follows:

$$\begin{aligned}
&\boldsymbol{M}(\boldsymbol{\mathcal{X}})\ddot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}) + \boldsymbol{\mathcal{F}}_e \\
&\quad = \hat{\boldsymbol{M}}\boldsymbol{M}_d^{-1}\left\{\boldsymbol{M}_d\ddot{\boldsymbol{\mathcal{X}}}_d + \boldsymbol{C}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_d\boldsymbol{e}_x - \boldsymbol{\mathcal{F}}_m\right\} + \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{X}}, \dot{\boldsymbol{\mathcal{X}}})\dot{\boldsymbol{\mathcal{X}}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{X}})
\end{aligned}$$

If the information regarding the dynamic matrices is complete, we can assume that $\hat{\boldsymbol{M}} = \boldsymbol{M}$, $\hat{\boldsymbol{\mathcal{C}}} = \boldsymbol{\mathcal{C}}$, and $\hat{\boldsymbol{\mathcal{G}}} = \boldsymbol{\mathcal{G}}$. Additionally, if the force measurements are free from noise, then $\boldsymbol{\mathcal{F}}_m = \boldsymbol{\mathcal{F}}_e$. Under these conditions, the closed-loop dynamic formulation can be simplified to

$$\boldsymbol{M}\boldsymbol{M}_d^{-1}(\boldsymbol{M}_d\ddot{\boldsymbol{\mathcal{X}}}) = \boldsymbol{M}\boldsymbol{M}_d^{-1}\left\{\boldsymbol{M}_d\ddot{\boldsymbol{\mathcal{X}}}_d + \boldsymbol{C}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_d\boldsymbol{e}_x - \boldsymbol{\mathcal{F}}_e\right\}.$$

(14.12)

It is important to note that the manipulator mass matrix $\boldsymbol{M}(\boldsymbol{\chi})$ is a positive definite matrix. By selecting a positive definite matrix $\boldsymbol{M}_d$, the resulting matrix $\boldsymbol{M}\boldsymbol{M}_d^{-1}$ can be definitively determined at all configurations and is nonsingular. Therefore, the closed-loop dynamic formulation can be simplified to

$$\boldsymbol{M}_d\ddot{\boldsymbol{e}}_x + \boldsymbol{C}_d\dot{\boldsymbol{e}}_x + \boldsymbol{K}_d\boldsymbol{e}_x = \boldsymbol{\mathcal{F}}_e.$$

(14.13)

This equation indicates that if complete information about the dynamic matrices is available and the force measurements are free from noise, the closed-loop error dynamic equation adheres to a set of second-order systems characterized by the desired impedance coefficients in relation to the interacting force. In other words, the control structure ensures that the relationship between force and motion aligns with the desired impedance structure specified in Equation 14.6. Consequently, by selecting appropriate positive definite gains for the impedance matrices, the transient performance of the force–motion relationship can be tailored to achieve a fast yet stable interaction. Thus, the focus of control is on the dynamic relationship between the force and motion variables rather than direct position and motion tracking.

However, if the robot is freely moving in space and not interacting with the environment, meaning $\boldsymbol{\mathcal{F}}_e = \mathbf{0}$, the closed-loop system will facilitate suitable motion tracking. This is attributed to the presence of a motion controller in the outer loop.

## 14.5 Special Case in Impedance Control

Another physical interpretation of the impedance control scheme can be derived by simplifying the impedance requirements. Suppose there is no need to modify the inductive impedance matrix of the closed-loop system, and the robot's inherent mass matrix is adequate and does not require adjustment. In this scenario, we can set $\boldsymbol{M}_d = \boldsymbol{M}$, leading to $\boldsymbol{M}\boldsymbol{M}_d^{-1} = \boldsymbol{I}$, where $\boldsymbol{I}$ represents the identity matrix. This modification significantly simplifies the control structure and aligns it more closely with a stiffness control scheme.

Revisiting the control structure illustrated in Figure 14.2, we can note that, in this case, the main force controller block $\boldsymbol{M}\boldsymbol{M}_d^{-1}$ can be eliminated since it simplifies to the identity matrix. Now, if we consider that the measured wrench $\boldsymbol{\mathcal{F}}_m$ is directly subtracted from the feedback linearizing wrench $\boldsymbol{\mathcal{F}}_{fl}$, the result of this subtraction yields

$$\boldsymbol{\mathcal{F}}_{fl} - \boldsymbol{\mathcal{F}}_m = \hat{\boldsymbol{\mathcal{C}}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \hat{\boldsymbol{\mathcal{G}}}(\boldsymbol{\chi}).$$

(14.14)

This suggests that to generate the required wrench in this scenario, no force measurement is necessary. As a result, there is no need for an internal force feedback loop to implement this controller and the feedback linearizing term simplifies to the left-hand side of Equation 14.14. Through these modifications, the impedance control structure depicted in Figure 14.2 reduces to a pure motion inverse dynamics controller as shown in Figure 12.2, which serves as the foundation for the stiffness control scheme. This observation indicates that the impedance controller can be viewed as a generalization of the stiffness control scheme, allowing for suitable tuning of higher-order dynamic terms. Conversely, if only the resistive and capacitive impedance coefficients are taken into account in the force–motion relationship, the dynamic relationship simplifies to stiffness and damping terms.

The impedance control scheme is widely utilized in practice, where the primary goal is to tune the force and motion relationship in a robotic manipulator interacting with a rigid environment. This scheme can be effectively implemented in the applications provided in Section 14.3. However, it is important to note that for optimal performance, an accurate model of the system is essential, as the derived force and motion dynamics are not robust to modeling uncertainties.

## 14.6 Case Study I: Planar Cable Robot

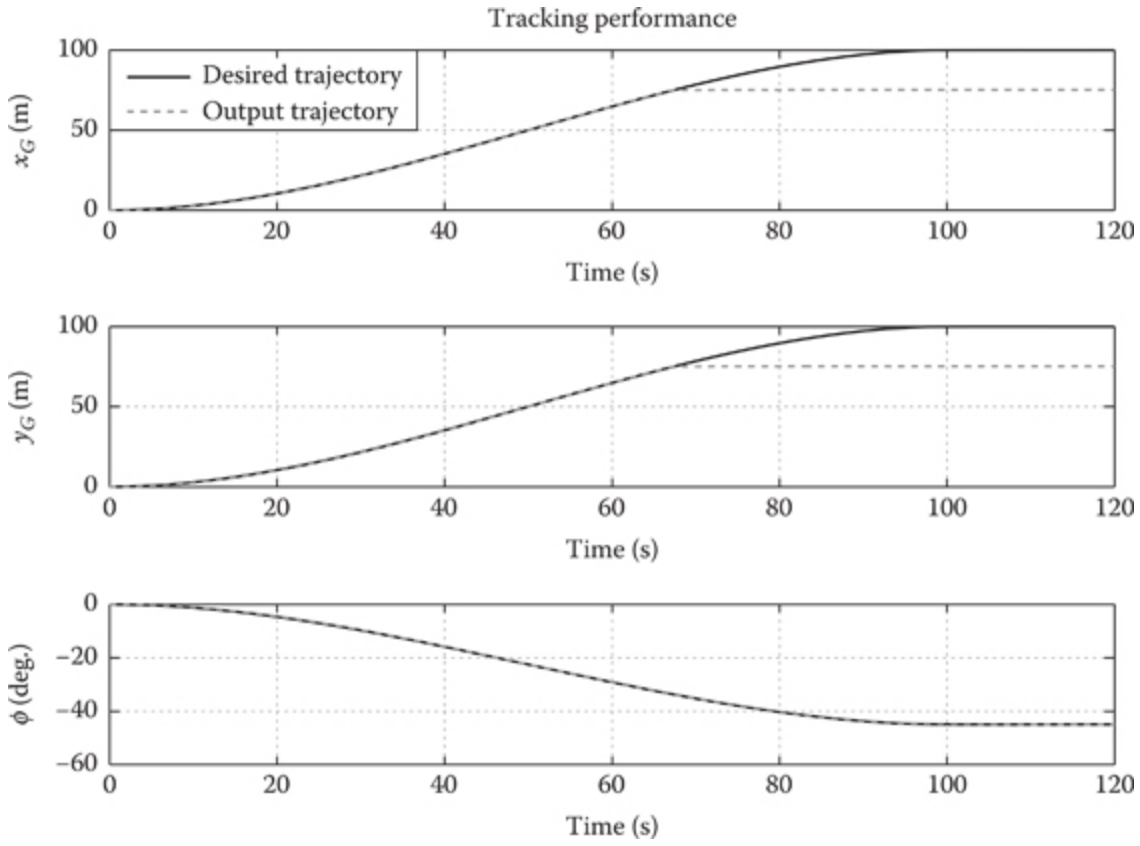
This section provides an in-depth study of impedance control for a planar 4RPR manipulator. The configuration of the planar manipulator is thoroughly explained in Section 4.4.1 and illustrated in Figure 4.4. Analyses of the manipulator's Jacobian and singularities are covered in Section 5.5, while its dynamic analysis is addressed in Section 7.4. Various motion control structures are implemented for this manipulator, with a detailed examination of their performance presented in Section 12.7. Stiffness control and direct force control for the manipulator are discussed in Sections 13.3.3 and 13.4.1, respectively. The results of implementing the impedance control strategy applied to the manipulator will be reported as follows.

The impedance control strategy applied to the system follows the structure shown in Figure 14.2, where position feedback is incorporated in the outer loop and force feedback is utilized in the inner loop. As depicted in Figure 14.2, the motion-tracking error is directly obtained from motion measurements as $\boldsymbol{e}_x = \boldsymbol{\mathcal{X}}_d - \boldsymbol{\mathcal{X}}$ in the outer loop, with the motion controller designed to meet the required impedance structure outlined in Equation 14.6. In the impedance control scheme, the desired wrench $\boldsymbol{\mathcal{F}}$ is based on inverse dynamics control and comprises three primary components. The inner loop's force control scheme includes a feedback linearization component along with a mass matrix adjustment, while the outer loop typically employs a

linear motion controller to satisfy the desired impedance requirements as expressed in Equation 14.6.

A linear impedance relationship between force and motion variables is achieved using this structure, which comprises the desired inductive $\boldsymbol{M}_d$, resistive $\boldsymbol{C}_d$, and capacitive $\boldsymbol{K}_d$ impedance matrices, as specified in Equation 14.6. The control input applied to the system is given in Equation 14.8, where $\boldsymbol{e}_F, \boldsymbol{\mathcal{F}}_a$, and $\boldsymbol{\mathcal{F}}_{fl}$ are defined in Equations 14.9 through 14.11, respectively. It's important to note that generating these terms requires knowledge of the robot's dynamic formulation, along with its kinematic and dynamic parameters. Since exact data on dynamic matrices is typically unavailable, a substantial 50% perturbation in all kinematic and inertial parameters is applied in the subsequent simulations.

As shown schematically in Figure 13.10, a flat surface is positioned within the robot's workspace such that the moving platform encounters a stiff environment during a portion of its maneuver. In Figure 14.3, the desired trajectory in these simulations matches that of the stiffness control scheme illustrated in Figure 13.11, with the object's position set similarly to that in the stiffness control scheme. Therefore, for the simulations, we assume $\boldsymbol{x}_l = (75, 75)$, with a contact surface slope of $s_l = -1$. Along this prescribed trajectory, shown in Figure 14.3, the moving platform makes contact with the flat object around 67 seconds, and this contact is maintained until the end of the simulation.




▶ Long Description for Figure 14.3

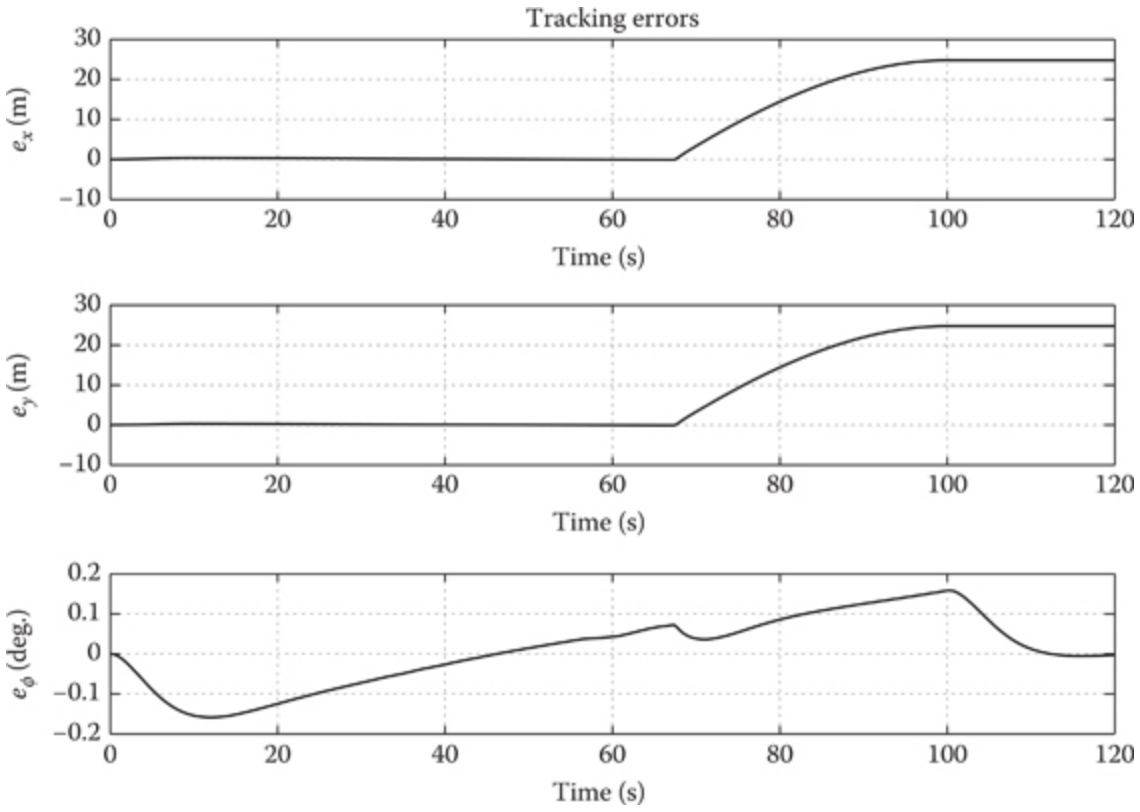
**FIGURE 14.3**

The desired and actual trajectories of the planar $4RPR$ manipulator during collision with a flat object under impedance control. ⏎

The object's stiffness is modeled by Equation 13.28, with a linear stiffness gain of $\boldsymbol{K}_e = 10^3 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and linear structural damping of $\boldsymbol{C}_e = 10^3 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N}\cdot\mathrm{m/s})$. The geometric and inertial parameters used in the simulations are listed in Table 10.1, and the initial state conditions are set to zero. In these simulations, the disturbance wrench acting on the moving platform is assumed to be zero, while

the gravity vector is set as $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$. The control input is calculated using Equation 14.8, where $\boldsymbol{e}_F, \boldsymbol{\mathcal{F}}_a$, and $\boldsymbol{\mathcal{F}}_{fl}$ are specified in Equations 14.9 to 14.11, respectively. The desired impedance matrices are set to $\boldsymbol{M}_d = 100 \cdot \boldsymbol{I}_{3\times3}$, $\boldsymbol{C}_d = 20 \cdot \boldsymbol{I}_{3\times3}$, and $\boldsymbol{K}_d = 100 \cdot \boldsymbol{I}_{3\times3}$.

The final motion of the moving platform and the tracking errors are shown in Figures 14.3 and 14.4. As seen in Figure 14.3, the desired and actual closed-loop motion closely match prior to the collision. The collision occurs at approximately 67 seconds, after which the moving platform's motion remains nearly constant on the object's surface. Figure 14.4 provides a closer view of the platform's motion at this point, showing that the moving platform transitions smoothly along the object's surface after the collision, resulting in only a slight deflection. Additionally, since no interaction torque is considered in the simulations, the tracking error in orientation $\phi$ remains relatively consistent before and after the collision.



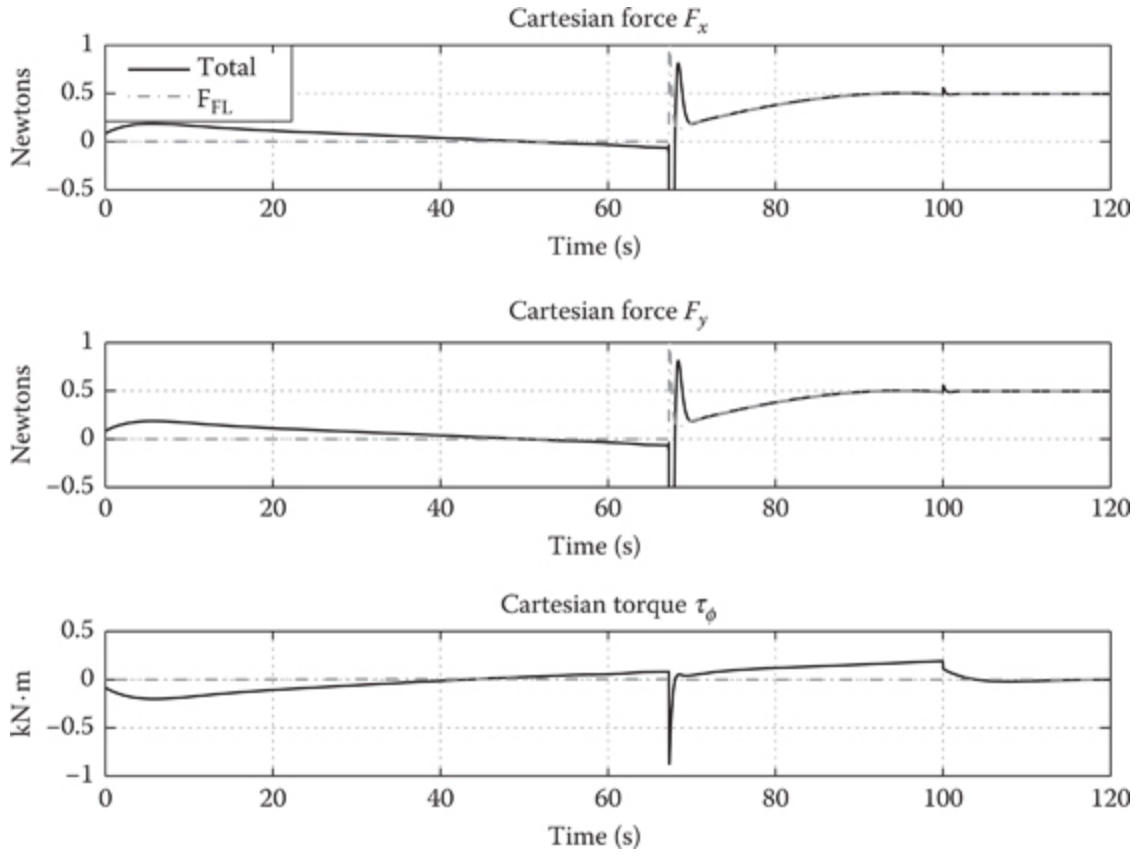

▶ Long Description for Figure 14.4

**FIGURE 14.4**

Closed-loop tracking performance of the planar $4RPR$ manipulator during collision with a flat object under impedance control. ⏎

The Cartesian wrench $\boldsymbol{\mathcal{F}} = [F_x, F_y, \tau_\phi]^T$ generated by the impedance controller is illustrated in Figure 14.5. Here, the total force is shown with a solid line, while the feedback linearizing term appears as a dash-dotted line. In Figure 14.5, the Cartesian forces in the $x$ and $y$ directions increase sharply after the transient at the moment of collision, with no interaction torques observed in the $\phi$ direction. Before the collision, the feedback linearizing term contributes only

a minor portion of the total force, but post-collision, the interaction force $\mathcal{F}_e$ rises quickly, directly impacting the feedback linearizing term. This response occurs because the feedback linearizing wrench $\mathcal{F}_{fl}$, defined in Equation 14.11, includes the measured force $\mathcal{F}_m$.

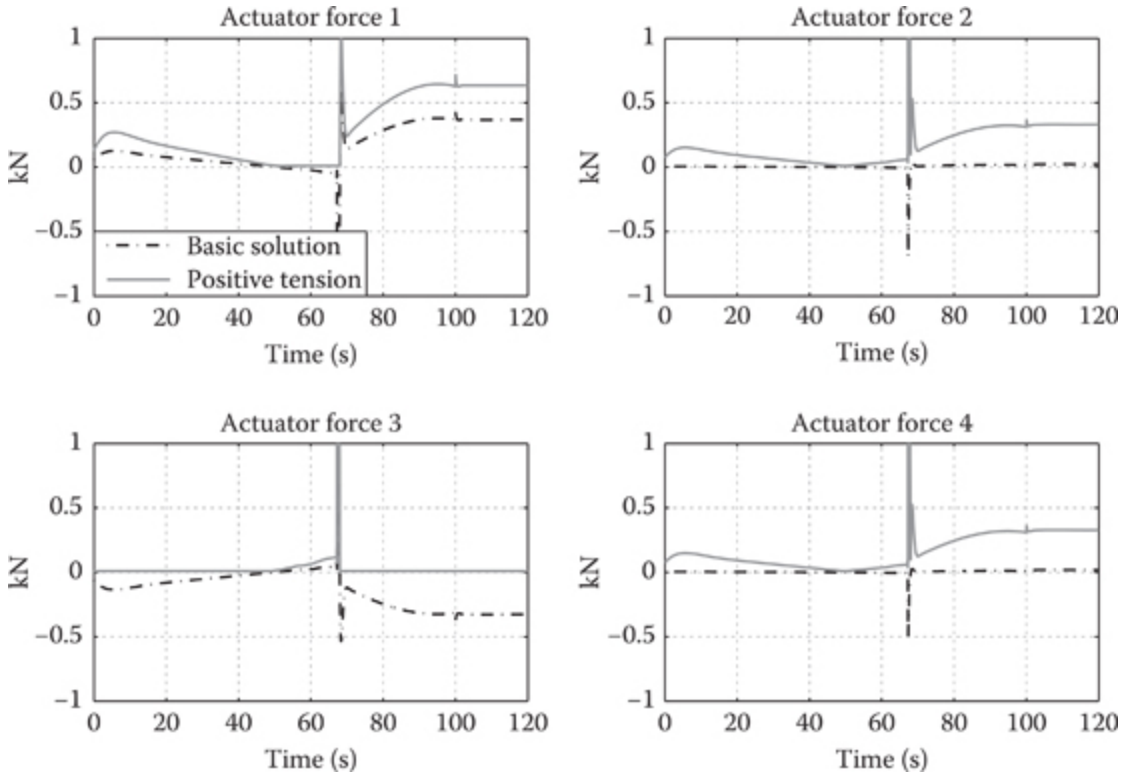



▶ Long Description for Figure 14.5

**FIGURE 14.5**
Cartesian wrench of the planar $4RPR$ manipulator during collision with a flat object under impedance control. ⏎

Similarly, two sets of actuator forces are shown in Figure 14.6. The dash-dotted lines represent the base solution of the optimization problem introduced in the redundancy

resolution scheme by Equation 6.84, while the solid line represents the final optimization solution detailed in Equation 6.87. The redundancy resolution problem is solved using an iterative numerical method. As seen in Figure 14.6, the actuator forces significantly increase post-collision, with tension forces from the redundancy resolution scheme remaining positive throughout.



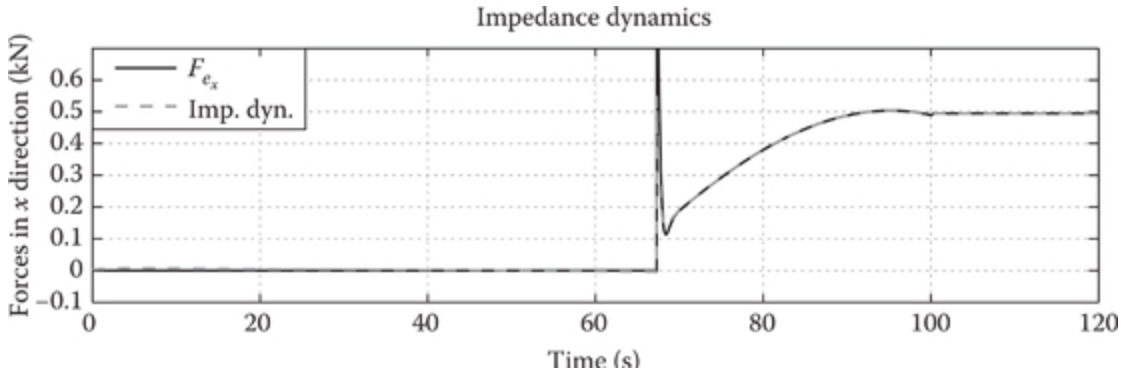

▶ Long Description for Figure 14.6

**FIGURE 14.6**

Actuator forces of the planar $4RPR$ manipulator during collision with a flat object under impedance control. ⏎

To evaluate the performance of the impedance controller and assess its ability to shape the closed-loop dynamics to a

desired impedance, the $x$ component of the interaction force $\mathcal{F}_e$ is plotted in Figure 14.7 alongside the closed-loop system's impedance dynamics, represented by the left-hand side of Equation 14.13. Ideally, if the impedance dynamics are perfectly enforced in the closed-loop system, both sides of Equation 14.13 would match, resulting in identical plots. As shown in Figure 14.7, while the two plots are not identical at all times, they closely track each other with only minor discrepancies. This demonstrates the effectiveness of the impedance control in achieving the desired force-motion relationship and confirms that the desired impedance dynamics have been effectively applied.




▶ Long Description for Figure 14.7

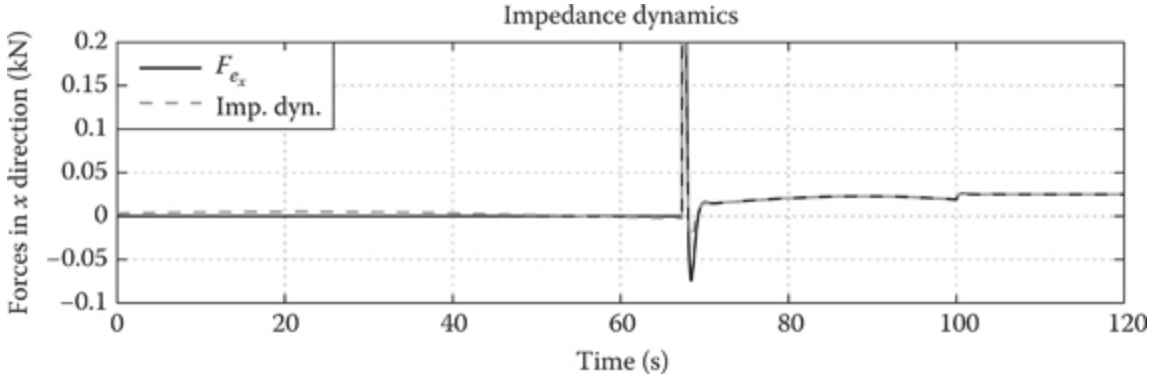
**FIGURE 14.7**

The interaction force $F_{e_x}$ versus the impedance dynamics of the planar $4RPR$ manipulator colliding with a flat object; impedance control. ⏎

To illustrate how a designer can effectively assign impedance requirements, we consider another simulation where all parameters remain unchanged except for the

impedance coefficients, which are set to $\boldsymbol{M}_d = 100 \cdot \boldsymbol{I}_{3\times3}$, $\boldsymbol{C}_d = \boldsymbol{I}_{3\times3}$, and $\boldsymbol{K}_d = 10 \cdot \boldsymbol{I}_{3\times3}$. In this configuration, the impedance stiffness coefficient is decreased by a factor of 10, and the impedance damping coefficient is reduced by 20 times. Although these impedance parameters may not be suitable for practical implementation, this exaggerated adjustment is made to assess the impedance controller's ability to regulate performance toward a desired impedance.

In this scenario, the $x$ component of the interaction force $\boldsymbol{\mathcal{F}}_e$ is plotted in Figure 14.8 alongside the impedance dynamics of the closed-loop system, as represented by the left-hand side of Equation 14.13. As shown in Figure 14.8, despite the significant changes in impedance parameters, the two plots closely follow one another with only minor discrepancies. This observation reinforces the conclusion that the force-motion relationship can be effectively tuned to meet the desired impedance dynamics through the impedance control structure.



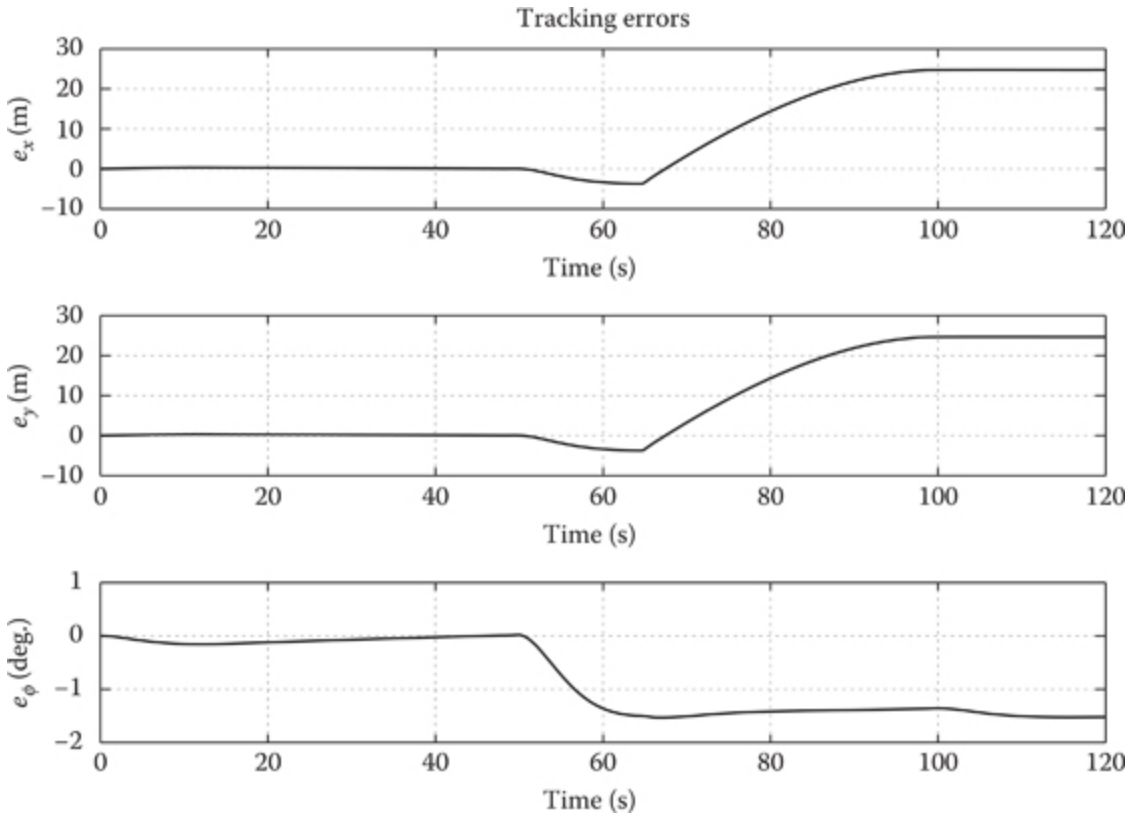

▶ Long Description for Figure 14.8

**FIGURE 14.8**

The interaction force $F_{e_x}$ versus the impedance dynamics of the planar $4RPR$ manipulator colliding with a flat object; impedance control with a different impedance requirement. ⏎

To evaluate the performance of the impedance controller in the presence of external disturbances, an additional simulation is conducted for the closed-loop system. In this simulation, a step disturbance force is applied to the system along the original trajectory shown in Figure 14.3. At time 50 seconds, a $1\,kN$ step disturbance force is applied in both the *x* and *y* directions, along with a $1\,\mathrm{kN}\cdot\mathrm{m}$ disturbance torque in the $\phi$ direction. The desired impedance matrices are configured as $\boldsymbol{M}_d = 100 \cdot \boldsymbol{I}_{3\times 3}$, $\boldsymbol{C}_d = 20 \cdot \boldsymbol{I}_{3\times 3}$, and $\boldsymbol{K}_d = 100 \cdot \boldsymbol{I}_{3\times 3}$, similar to the first simulation.

The closed-loop tracking error of the system is depicted in Figure 14.9. By comparing this figure with Figure 14.4, it is evident that the impedance controller does not completely mitigate the effects of external disturbances, as indicated by the abrupt changes in tracking error when the disturbance wrench is applied. Additionally, due to the persistent disturbances affecting the system, static errors accumulate in the closed-loop response prior to the collision.

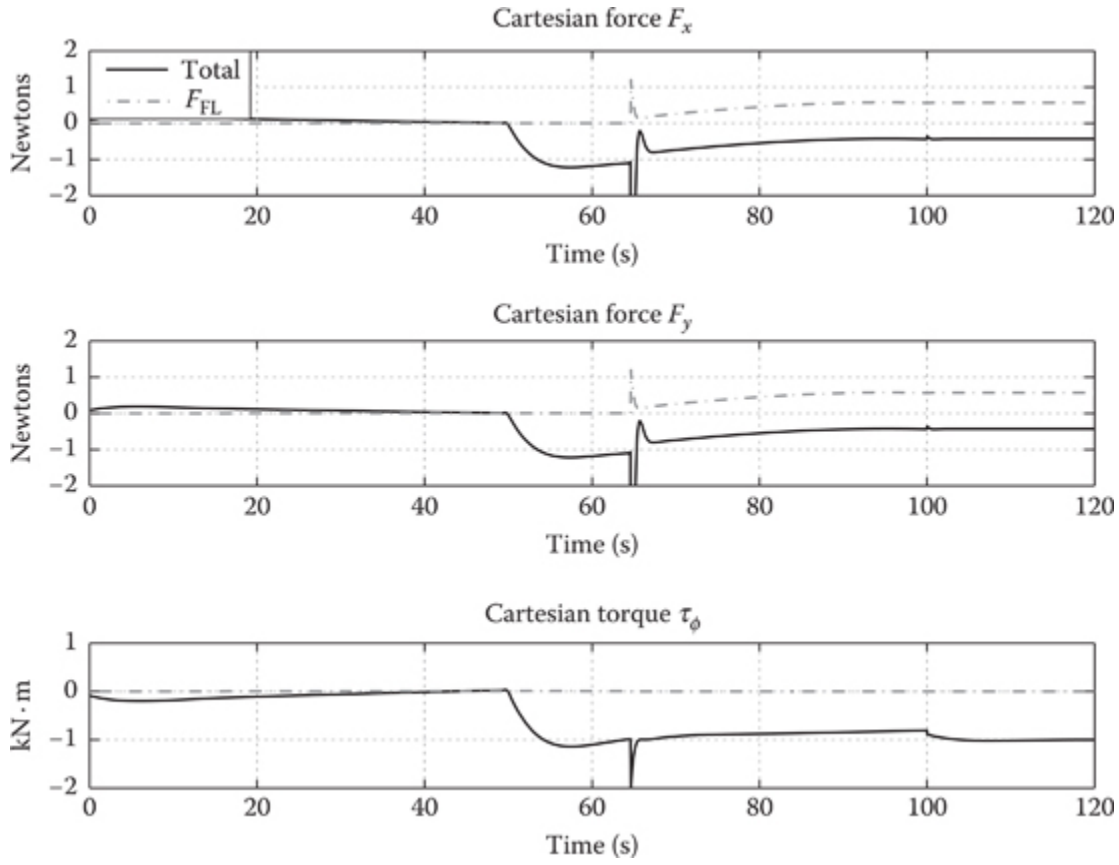



▶ Long Description for Figure 14.9

**FIGURE 14.9**

The closed-loop tracking performance of the planar $4RPR$ manipulator colliding with a flat object under impedance control in the presence of disturbance. ⏎

Figure 14.10 shows the Cartesian wrench generated by the PD controller to perform the required maneuver despite the disturbance. The influence of the step disturbance wrench on the necessary controller forces is also clearly visible after 50 seconds, with the disturbance signature appearing in all generated forces.




▶ Long Description for Figure 14.10

**FIGURE 14.10**

The Cartesian wrench generated by the planar $4RPR$ manipulator colliding with a flat object under impedance control in the presence of disturbance. ⏎

Finally, the effectiveness of the impedance controller in adjusting the closed-loop dynamics to a desired impedance in the presence of disturbances has been analyzed in Figure 14.11. This figure illustrates the $x$ component of the interaction force $\boldsymbol{\mathcal{F}}_e$, alongside the impedance dynamics of the closed-loop system as represented by the left-hand side of Equation 14.13. As observed in this figure, the two plots

exhibit an offset that corresponds to the persistent disturbance acting on the system. It is assumed that the force measurement does not capture the disturbance force, preventing the impedance controller from mitigating its effects. However, if the disturbance force diminishes over time or can be detected by force sensors, its impact on the impedance assignment would subsequently disappear.

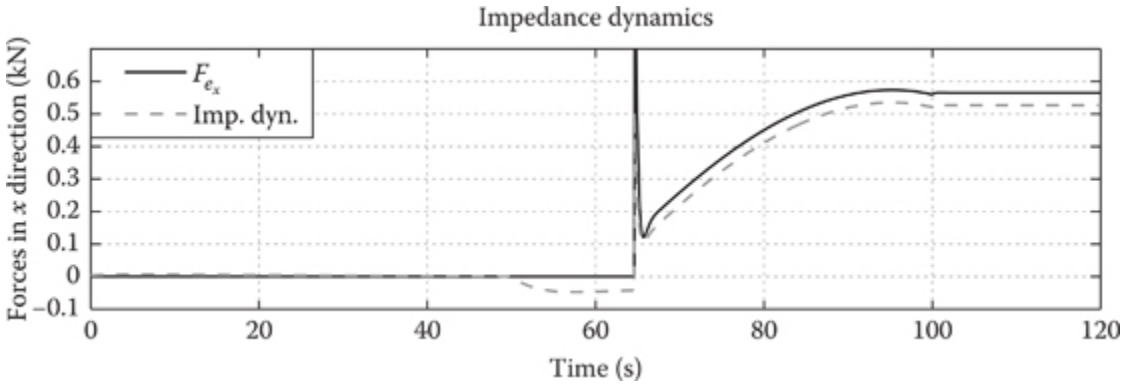



**FIGURE 14.11**
The interaction force $F_{e_x}$ compared to the impedance dynamics of the planar $4RPR$ manipulator colliding with a flat object under impedance control in the presence of disturbance. ⏎

## 14.7 Case Study II: Stewart–Gough Platform

In this section, the impedance control of the Stewart–Gough platform is examined in detail. The architecture of this manipulator is outlined in Section 4.6.1 and illustrated in Figure 4.16. A comprehensive kinematic analysis of this

manipulator is provided in Section 4.6, while its Jacobian analysis is discussed in Section 5.7. The dynamic analysis of this manipulator, utilizing both the Newton–Euler and virtual work methods, is provided in Section 7.5 and Section 9.6, respectively. The closed-form dynamic formulation employed for the simulations in this section is derived in Section 8.5.

Various motion control strategies are implemented in Section 12.8. Stiffness control and direct force control of this manipulator are explored in Sections 13.3.4 and 13.4.2, respectively. The following discussion will focus on the implementation of the impedance control strategy for this manipulator.

The impedance control strategy implemented in the system follows the structure illustrated in Figure 14.2, where position feedback is utilized in the outer loop, and force feedback is incorporated in the inner loop. As depicted in Figure 14.2, the motion-tracking error is directly calculated from motion measurements using the equation $\boldsymbol{e}_x = \boldsymbol{\chi}_d - \boldsymbol{\chi}$ in the outer loop, and the motion controller is designed to meet the required impedance structure outlined in Equation 14.6. The necessary wrench $\boldsymbol{\mathcal{F}}$ in the impedance control scheme is derived from inverse dynamics control and consists of three primary components. In the inner loop, the impedance control scheme incorporates a feedback linearization aspect along with an adjustment of the mass matrix, while the outer loop

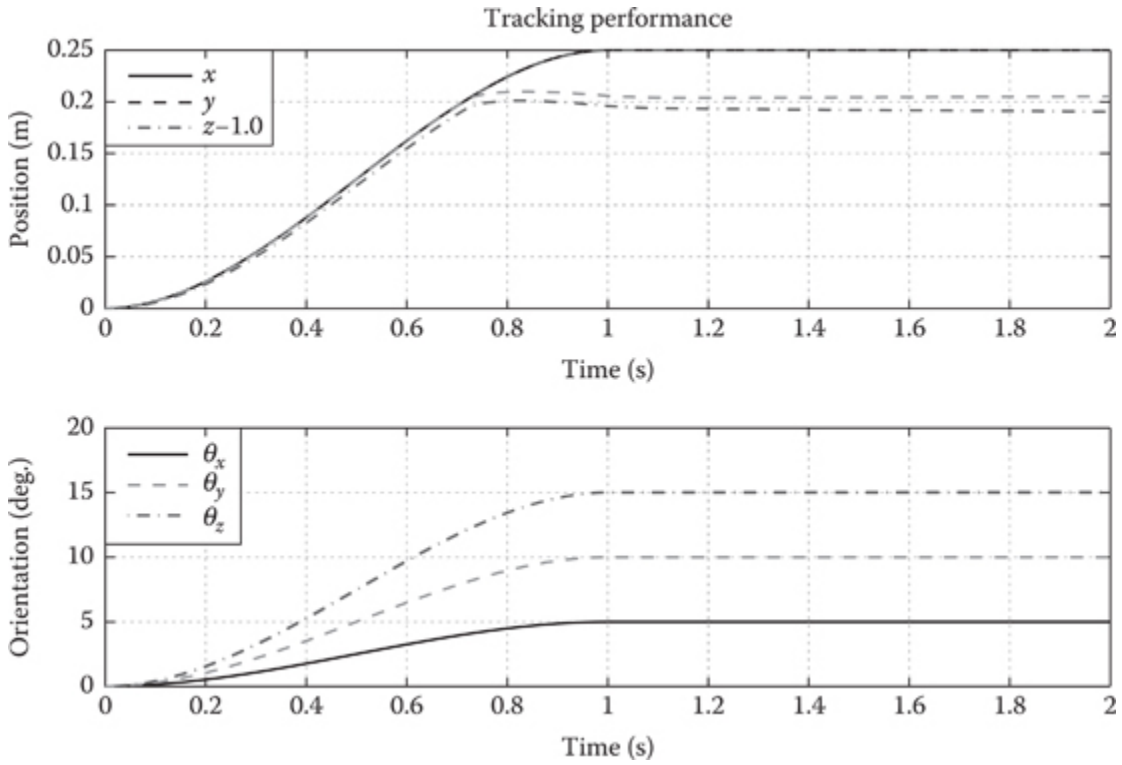
employs a linear motion controller aligned with the desired impedance requirements specified in Equation 14.6.

This framework generates a linear relationship between the force and motion variables. The desired impedance comprises inductive $\boldsymbol{M}_d$, resistive $\boldsymbol{C}_d$, and capacitive $\boldsymbol{K}_d$ matrices, as detailed in Equation 14.6. The control effort exerted on the system is represented in Equation 14.8, where $\boldsymbol{e}_F$, $\boldsymbol{\mathcal{F}}_a$, and $\boldsymbol{\mathcal{F}}_{fl}$ are defined in Equations 14.9 through 14.11. It is important to note that the generation of these terms requires the dynamic formulation of the robot along with its kinematic and dynamic parameters. In practical applications, precise information regarding the dynamic matrices is often unavailable, leading to a consideration of a 10% perturbation in all kinematic and inertial parameters for the subsequent simulations.

As illustrated in Figure 13.17, a flat surface is positioned within the robot's workspace, allowing the robot's moving platform to make contact with it during part of its maneuver. The desired trajectory for the simulations matches that of the stiffness control scheme shown in Figure 13.18, with the object's position also aligned with that in the stiffness control scheme. Specifically, the simulations assume $\boldsymbol{x}_l = [0.2, 0.2, 1.2]^T$ and a normal vector to the plane defined as $\boldsymbol{n}_l = [1,\ 1,\ 1]^T$. Consequently, along the prescribed trajectory depicted in Figure 13.18, the moving platform collides with the flat object approximately at 0.68 seconds, and this contact is maintained until the end of the simulation.

The object's stiffness is modeled with a linear stiffness gain of $\boldsymbol{K}_e = 10^5 \cdot \boldsymbol{I}_{6\times 6}$ (N/m) and a linear structural damping of $\boldsymbol{C}_e = 10^4 \cdot \boldsymbol{I}_{6\times 6}, (\mathrm{N}\cdot\mathrm{m/s})$. The geometric and inertial parameters utilized in these simulations are detailed in Table 10.2, with all initial conditions for the states set to zero, except for the moving platform's initial height, which is established at $z_o = 1$ m. For the simulations that follow, the disturbance wrench applied to the moving platform is assumed to be zero, while the gravity vector is specified as $\boldsymbol{g} = [0,\, 0,\, -9.81]^T$. The control effort is derived from Equation 14.8, with $\boldsymbol{e}_F$, $\boldsymbol{\mathcal{F}}_a$, and $\boldsymbol{\mathcal{F}}_{fl}$ defined in Equations 14.9 through 14.11. The desired impedance matrices are set to $\boldsymbol{M}_d = 100 \cdot \boldsymbol{I}_{6\times 6}$, $\boldsymbol{C}_d = 2 \times 10^3 \cdot \boldsymbol{I}_{6\times 6}$, and $\boldsymbol{K}_d = 10^4 \cdot \boldsymbol{I}_{6\times 6}$.

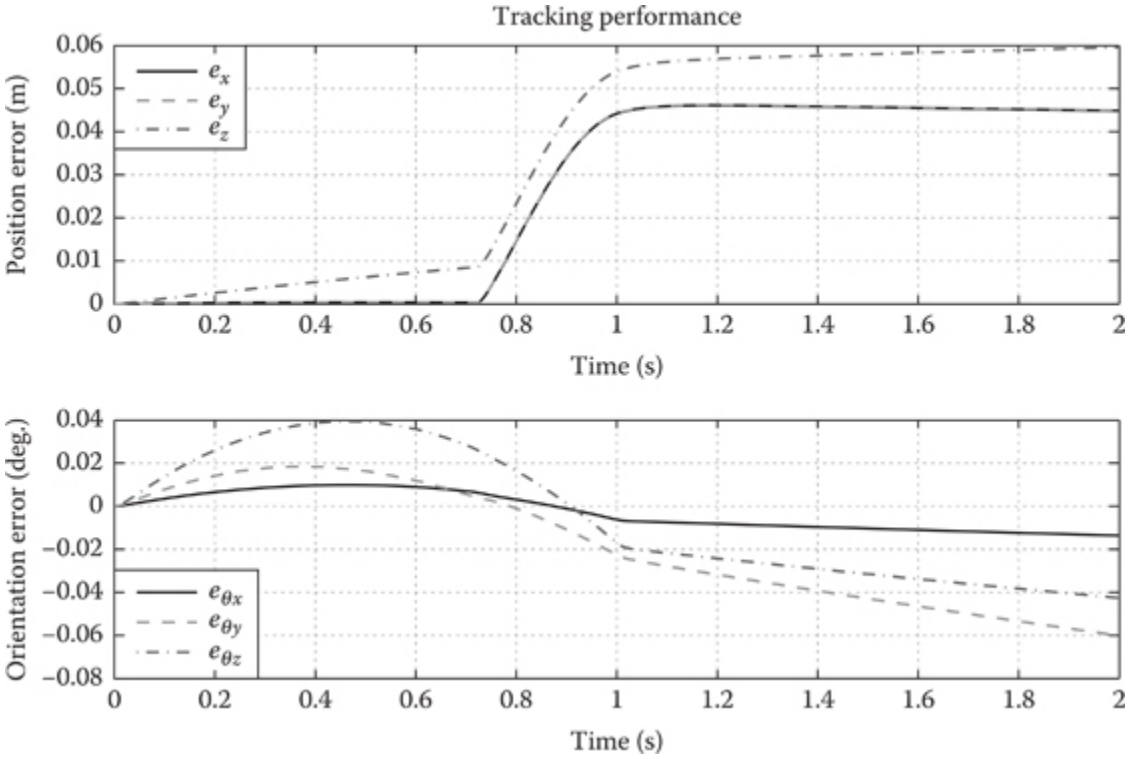
The final motion of the moving platform and the tracking errors are depicted in Figures 14.12 and 14.13. As illustrated in Figure 14.12, the desired and actual closed-loop motions of the system closely align before the collision occurs at approximately $0.68\,s$. Following this event, the motion of the moving platform remains nearly constant on the surface of the object. Further details of the moving platform's behavior at this moment can be found in Figure 14.13, which shows that post-collision, the moving platform exhibits smooth motion across the object's surface, leading to minor deflections of the object.




▶ Long Description for Figure 14.12

**FIGURE 14.12**

Desired and actual trajectories of the Stewart–Gough platform during collision with a flat object, under impedance control. ⏎




▶ Long Description for Figure 14.13

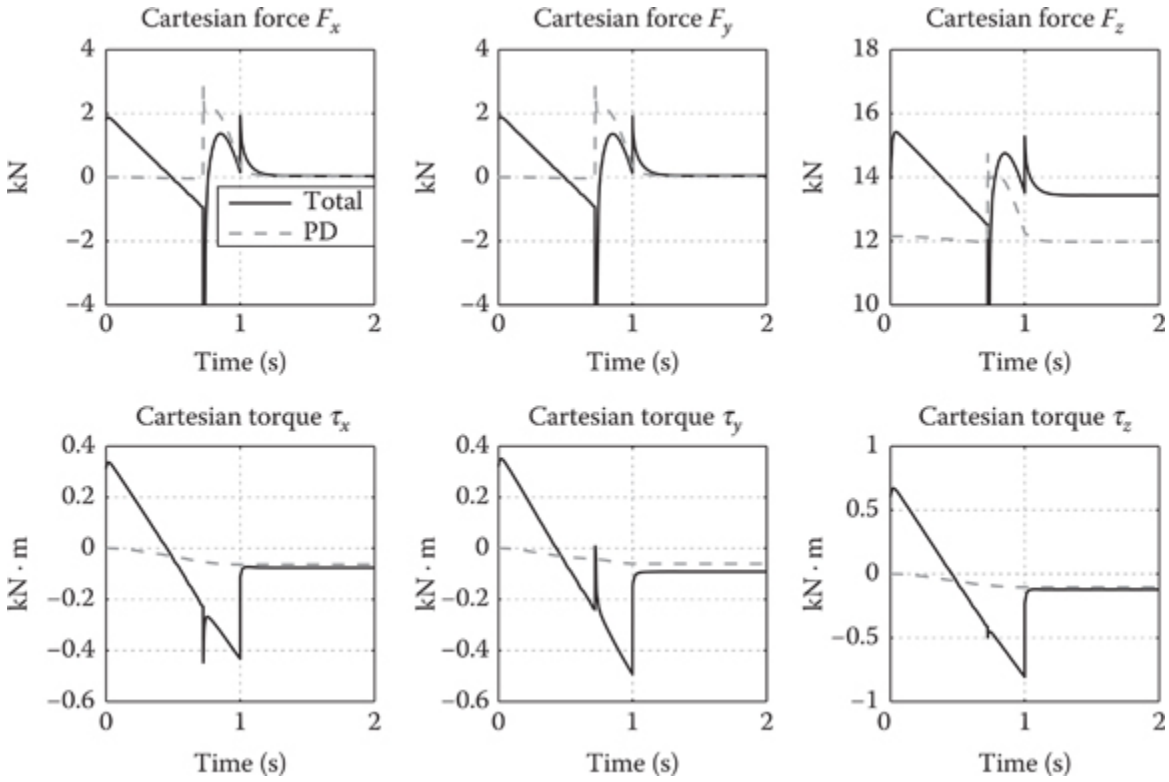
**FIGURE 14.13**

Closed-loop tracking performance of the Stewart–Gough platform during a collision with a flat object under impedance control. ⏎

It's worth noting that the error in the $z$ direction is greater than in the $x$ and $y$ directions due to incomplete gravity compensation, which has a significant effect on the vertical motion. Additionally, since no interaction torque is factored into the simulations, the tracking error in the orientation angles remains relatively uniform before and after the collision.

The Cartesian wrench $\mathcal{F}$ generated by the impedance controller is presented in Figure 14.14, where the total

force is shown with a solid line and the PD term with a dashed line. As observed, the Cartesian forces in the $x$, $y$, and $z$ directions exhibit a linear pattern corresponding to the required acceleration of the moving platform prior to the collision. These forces undergo rapid changes and align with the interaction forces following the collision.



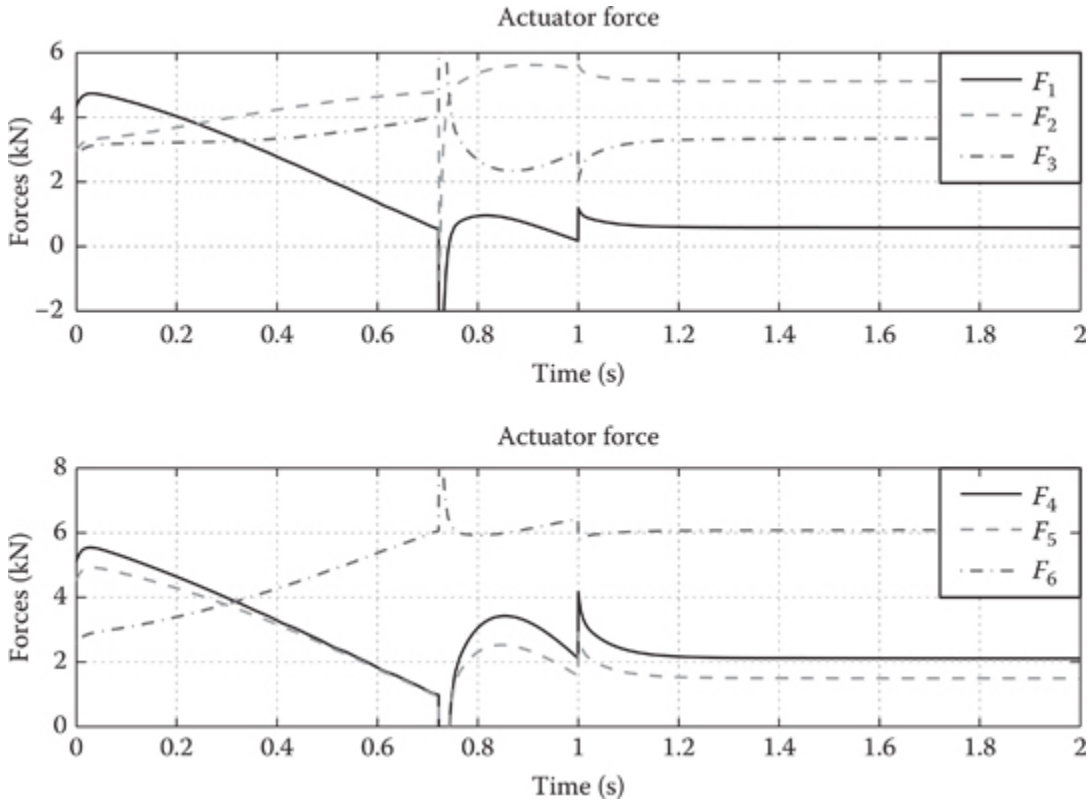

▶ Long Description for Figure 14.14

**FIGURE 14.14**

The Cartesian wrench of the Stewart–Gough platform colliding with a flat object; impedance control. ⏎

Given that no interaction torques are included in the simulation, the Cartesian torques also display a linear shape corresponding to the required angular acceleration

of the moving platform, although a transient behavior is noted at the moment of collision. Furthermore, it can be seen that the PD term contributes only a small fraction of the total force prior to the collision, but after the collision, the interaction force $\mathcal{F}_e$ rises sharply, which immediately influences the PD term. The actuator forces of the manipulator are similarly depicted in Figure 14.15, illustrating a behavior akin to that of the Cartesian wrench both before and after the collision.

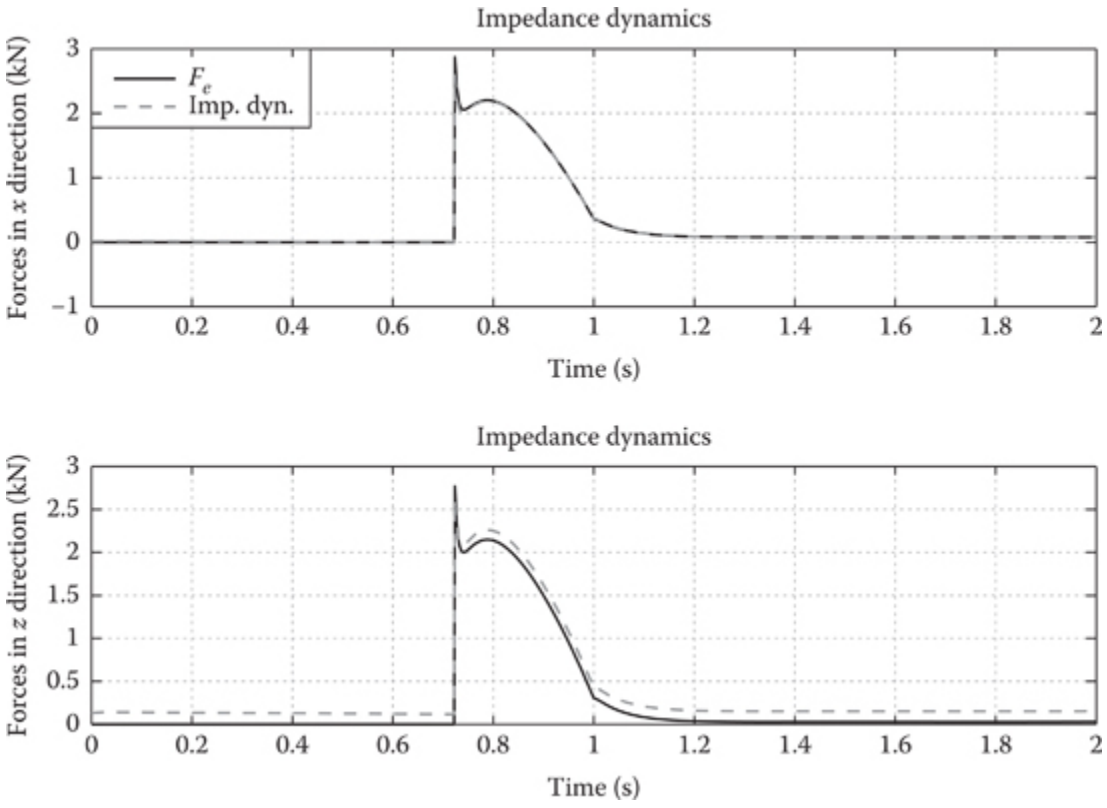



▶ Long Description for Figure 14.15

**FIGURE 14.15**

The actuator forces of the Stewart–Gough platform colliding with a flat object; impedance control. ⏎

To investigate the performance of the impedance controller in adapting the closed-loop dynamics to a desired impedance profile, the $x$ and $z$ components of the interaction force $\mathcal{F}_e$ are depicted in the top and bottom plots of Figure 14.16. In each graph, the interaction force is represented by a solid line, while the impedance dynamics of the closed-loop system, as indicated by the left-hand side of Equation 14.13, is shown with a dashed line. Ideally, if the impedance dynamics are perfectly implemented in the closed-loop system, both sides of Equation 14.16 would match, leading to identical solid and dashed lines. In the upper graph, the two plots align perfectly in the $x$ direction. However, in the lower plot, there is a slight offset between the two, attributed to incomplete gravity compensation, which notably affects the $z$ direction. Despite this minor discrepancy, the plots closely follow each other, indicating effective adjustment of the force–motion relationship to the desired impedance dynamics.



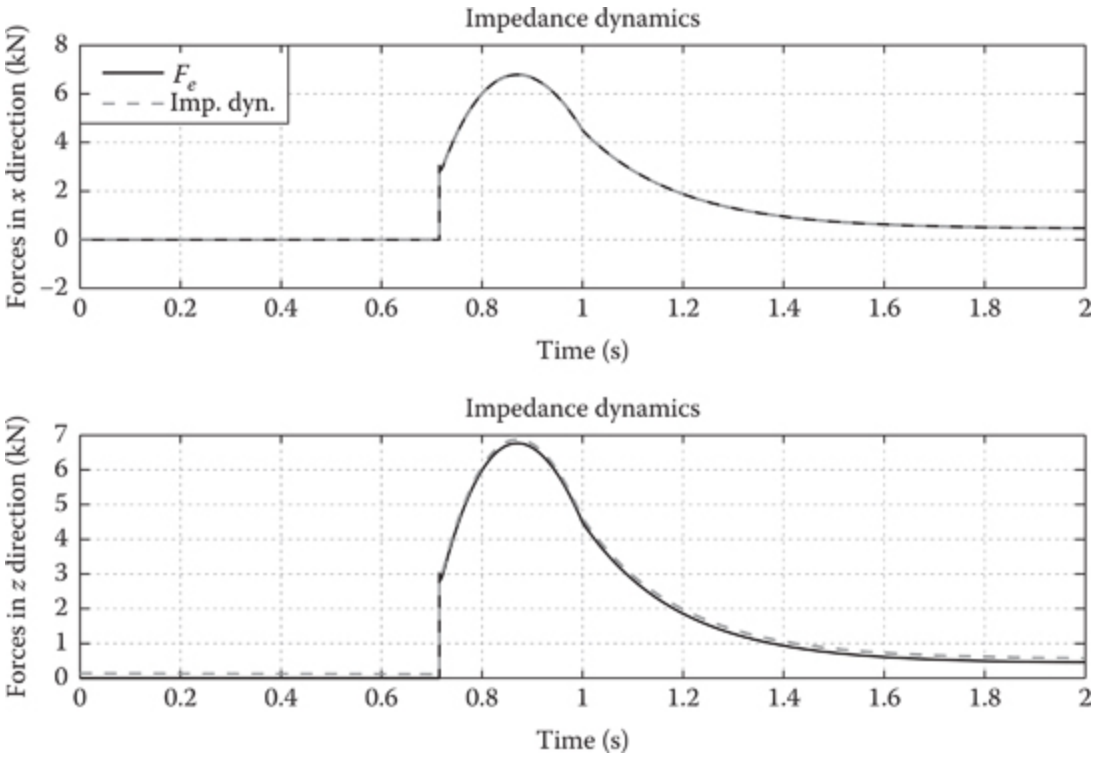

▶ Long Description for Figure 14.16

**FIGURE 14.16**

The interaction force $F_{e_x}$ versus the impedance dynamics of the Stewart–Gough platform colliding with a flat object; impedance control. ⏎

To demonstrate how a designer can effectively specify the necessary impedance requirements, consider a different simulation where all parameters remain consistent, but the impedance coefficients are modified to $\boldsymbol{M}_d = 100 \cdot \boldsymbol{I}_{6\times6}$, $\boldsymbol{C}_d = 10^4 \cdot \boldsymbol{I}_{6\times6}$, and $\boldsymbol{K}_d = 5 \times 10^4 \cdot \boldsymbol{I}_{6\times6}$. In this configuration, the stiffness and damping coefficients are increased by a factor of 5, while the impedance mass matrix remains constant. In this scenario, the $x$ and $z$

components of the interaction force $\mathcal{F}_e$ are plotted in the top and bottom sections of Figure 14.17, respectively, alongside the impedance dynamics of the closed-loop system indicated by the left-hand side of Equation 14.13. Figure 14.17 illustrates that, despite significant changes in the impedance parameters, the solid and dashed lines in both plots remain closely aligned with only a minor difference in the $z$ direction. This observation confirms that the force–motion relationship can be effectively adjusted to meet the required impedance dynamics through the impedance control framework.



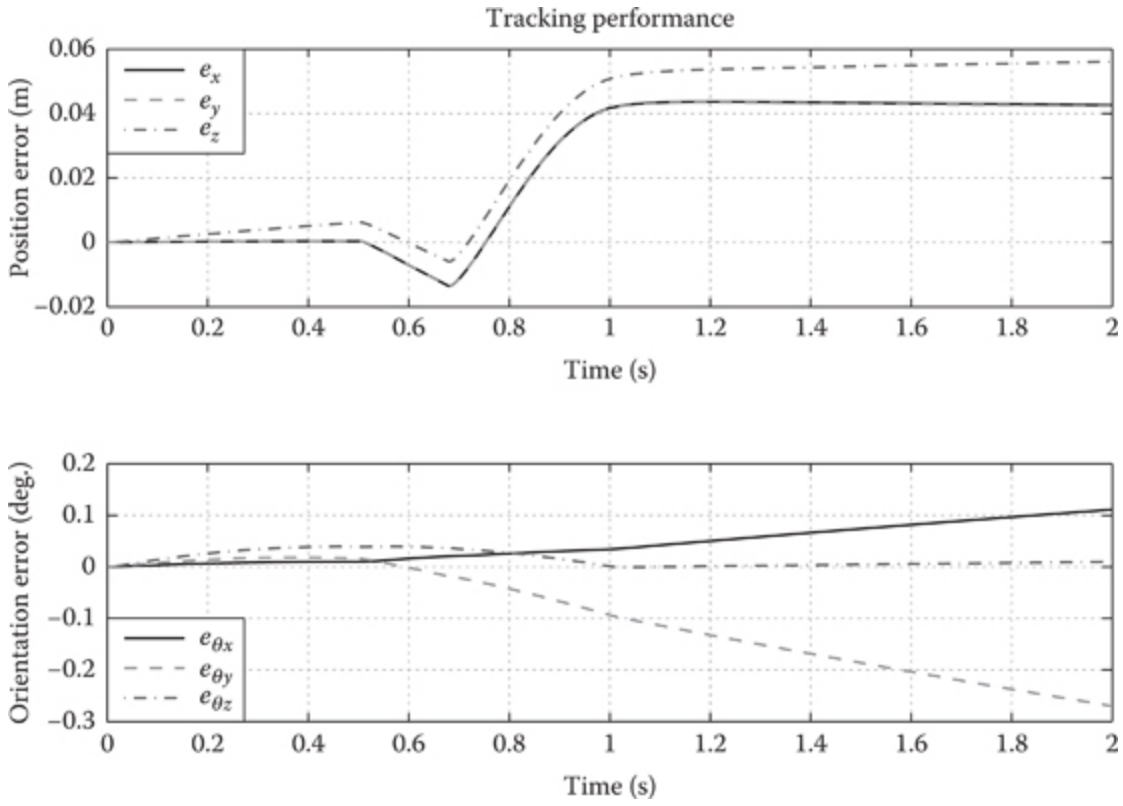

▶ Long Description for Figure 14.17

**FIGURE 14.17**

The interaction force $F_{e_x}$ versus the impedance dynamics of the Stewart–Gough platform colliding with a flat object; impedance control with a different impedance requirement. ⏎

To evaluate the impedance controller's performance under external disturbances, another simulation of the closed-loop system was conducted. In this scenario, a 10kN step disturbance force is applied in the *x*, *y*, and *z* directions at 50,s, while following the original trajectory shown in Figure 14.12. No disturbance torques are included in this simulation. The desired impedance matrices are set to $\boldsymbol{M}_d = 100 \cdot \boldsymbol{I}_{6\times6}, \boldsymbol{C}_d = 2 \times 10^3 \cdot \boldsymbol{I}_{6\times6}$, and $\boldsymbol{K}_d = 10^4 \cdot \boldsymbol{I}_{6\times6}$, consistent with the first simulation.

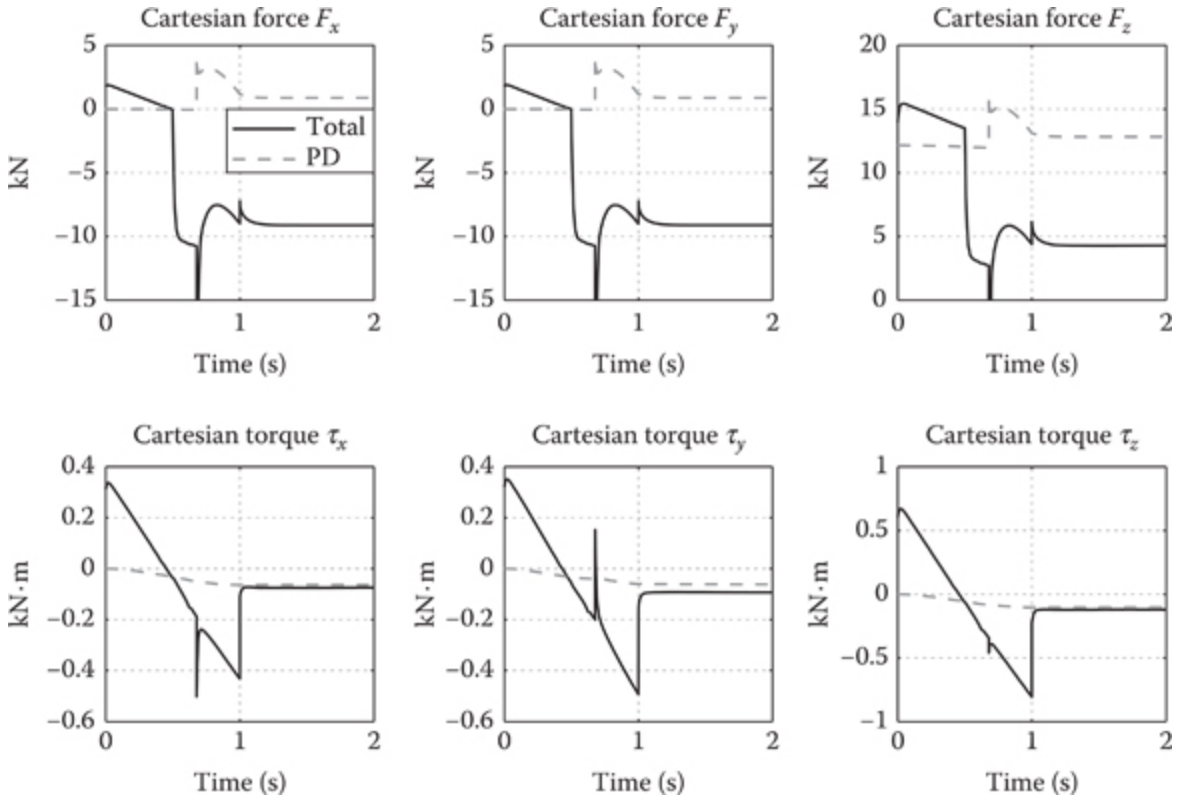
The closed-loop tracking error for this setup is depicted in Figure 14.18. A comparison between Figures 14.19 and Figure 14.13 reveals that the impedance controller does not fully mitigate the effects of the external disturbance; a noticeable abrupt change in the tracking error occurs precisely when the disturbance force is introduced. Additionally, Figure 14.20 shows the Cartesian wrench produced by the PD controller as it attempts to achieve the required maneuver amid the disturbance. The impact of the step disturbance on the required controller forces is evident following the $50\,s$ mark, where the disturbance signature is clearly observable in the forces generated in the *x*, *y*, and *z* directions.




▶ Long Description for Figure 14.18

**FIGURE 14.18**

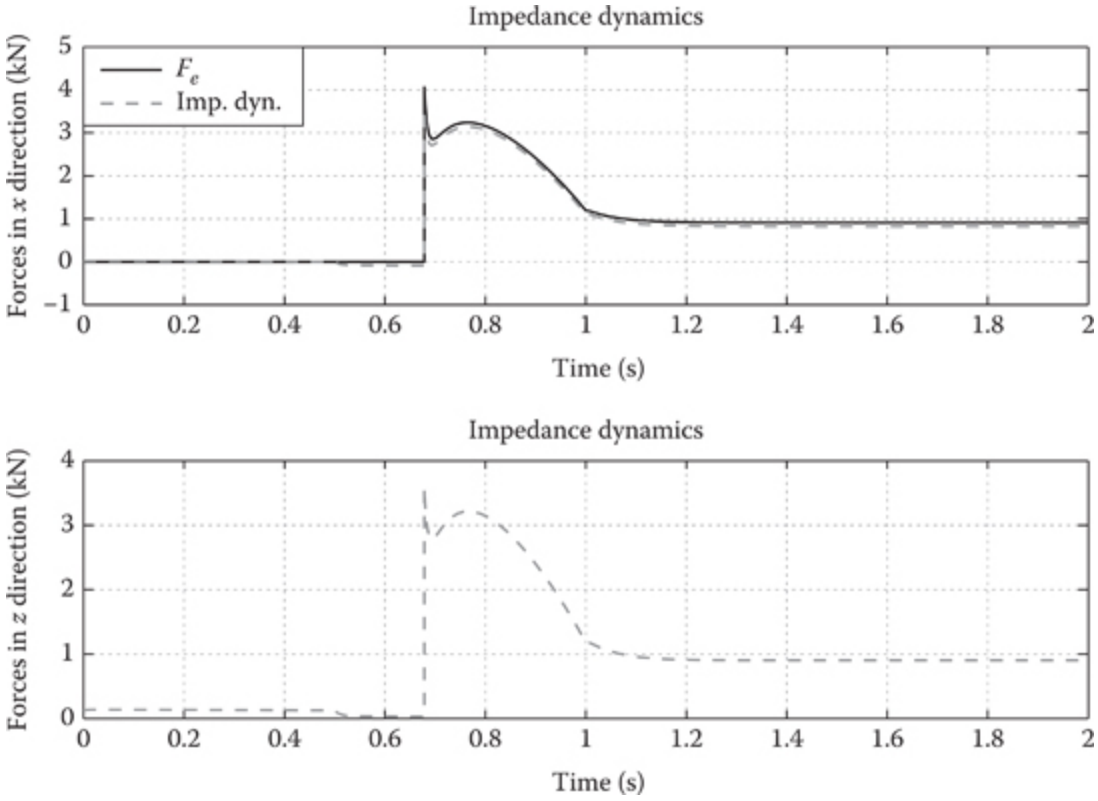
Closed-loop tracking performance of the Stewart–Gough platform during collision with a flat object; impedance control in the presence of disturbance. ⏎




▶ Long Description for Figure 14.19

**FIGURE 14.19**

Cartesian wrench generated by the Stewart–Gough platform during collision with a flat object; impedance control in the presence of disturbance. ⏎




▶ Long Description for Figure 14.20

**FIGURE 14.20**

The interaction force $F_{e_x}$ compared to the impedance dynamics of the Stewart–Gough platform during a collision with a flat object, under impedance control with disturbance. ⏎

The effectiveness of the impedance controller in adjusting the closed-loop dynamics to the desired impedance dynamics under disturbance has been analyzed in Figure 14.20. This figure presents the $x$ and $z$ components of the interacting force $\boldsymbol{f}_e$ in the upper and lower plots, respectively, alongside the impedance dynamics of the closed-loop system, as indicated by the left-

hand side of Equation 14.13. As observed in this figure, there is an offset between the solid and dashed plots, which corresponds to a persistent disturbance acting on the system. It is assumed that the force measurement cannot capture the disturbance force, thus preventing the impedance controller from mitigating its effects. However, if the disturbance force decreases over time or can be detected by the force sensors, its impact on impedance assignment will diminish.

## Problems

14.1 Consider the five-bar mechanism illustrated in Figure 4.27. The dynamic formulation of this mechanism, along with its kinematic and inertial parameters, as well as a typical motion trajectory, is provided in Problem 10.1. A flat surface is incorporated within the workspace of the mechanism, as modeled in detail in Section 13.3.3. This is utilized in Equation 13.26, where the contact forces are represented by $\boldsymbol{K}_e = 100 \cdot \boldsymbol{I}_{3\times 3}$ (N/m) and $\boldsymbol{C}_e = 100 \cdot \boldsymbol{I}_{3\times 3}, (\mathrm{N} \cdot \mathrm{m}/\mathrm{s})$.

a. For this mechanism, design an impedance controller using complete inverse dynamics and account for a 10% perturbation in all geometric and inertial parameters. Position the object so that a collision occurs during motion, and assign appropriate

impedance matrices $\boldsymbol{M}_d, \boldsymbol{C}_d$, and $\boldsymbol{K}_d$ for the closed-loop system. Simulate the closed-loop performance and adjust the impedance controller gains to achieve suitable impedance dynamics during contact.

b. Generate and analyze plots of closed-loop motion trajectories, tracking errors, Cartesian forces, and actuator forces over time.

c. Create plots and assess the interaction forces in the $x$ and $y$ directions relative to the corresponding impedance dynamics, and analyze the impedance behavior of the closed-loop system before and after contact.

d. Modify the desired impedance matrices and repeat part (c) to evaluate the performance of the controller.

14.2 Consider the five-bar mechanism depicted in Figure 4.28. The dynamic formulation of this mechanism, along with its kinematic and inertia parameters and a typical motion trajectory, is detailed in Problem 10.2. Apply the approach outlined in Problem 14.1 to this mechanism.

14.3 Consider the planar pantograph mechanism illustrated in Figure 4.29. The dynamic analysis of this mechanism, along with its kinematic and inertial parameters and a representative motion trajectory, is

detailed in Problem 10.3. Implement the approach detailed in Problem 14.1 for this mechanism.

14.4 Consider the planar three-degrees-of-freedom 3RRR parallel manipulator depicted in Figure 4.30. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters, and a representative motion trajectory, is provided in Problem 10.4. Apply the control method described in Problem 14.1 to this manipulator.

14.5 Consider the planar three-degrees-of-freedom 4RRR parallel manipulator illustrated in Figure 4.31. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is detailed in Problem 10.5. Implement the approach detailed in Problem 14.1 for this manipulator.

14.6 Consider the 3UPU parallel manipulator depicted in Figure 4.33. The dynamic formulation of this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is presented in Problem 10.7. Apply the control method described in Problem 14.1 to this manipulator.

14.7 Consider the three-degrees-of-freedom 3RPS manipulator illustrated in Figure 4.35. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is detailed in

Problem 10.9. Implement the approach detailed in Problem 14.1 for this manipulator.

14.8 Consider the three-degrees-of-freedom 3PR Ctranslational parallel manipulator depicted in Figure 4.35. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is provided in Problem 10.10. Apply the control method described in Problem 14.1 to this manipulator.

14.9 Consider the Delta robot illustrated in Figure 4.38. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is detailed in Problem 10.11. Implement the approach detailed in Problem 14.1 for this manipulator.

14.10 Consider the Quadrupteron robot depicted in Figure 4.39. The dynamic formulation for this manipulator, along with its kinematic and inertial parameters and a representative motion trajectory, is provided in Problem 10.12. Apply the control method described in Problem 14.1 to this manipulator.

# Part V

# Appendices

# A

# *Appendix A: Linear Algebra*

DOI: 10.1201/9781003590408-A

This book assumes that the reader already has a grasp of the fundamental concepts and properties of vectors and matrices, and for this reason, their explanation is omitted from this appendix. However, certain advanced properties in linear algebra that are used throughout the text are briefly reviewed in here. Interested readers are referred to the original sources in this field [261].

## A.1 Vectors and Matrices

A vector is an $n$-dimensional array of numerical elements represented in column form. In order to represent vectors in this book, lowercase letters are used, and they are written in bold format in the text, such as $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$.

An $n$-dimensional vector $\boldsymbol{x}$ belongs to the space $\mathbb{R}^n$ if all its elements belong to the set of real numbers $\mathbb{R}$. Thus, a vector $\boldsymbol{x} \in \mathbb{R}^n$ is defined as follows.

$$\boldsymbol{x} = \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix} \tag{A.1}$$

in which $x_i \in \mathbb{R}, i = 1, 2, \ldots, n$. A row vector is defined as the transpose of a column matrix.

$$\boldsymbol{x}^T = \begin{bmatrix} x_1 & x_2 & \ldots x_n \end{bmatrix} \tag{A.2}$$

Similarly, a matrix with dimensions $m \times n$ is defined as an array consisting of $m$ rows and $n$ columns. To represent matrices in this book, uppercase letters are used, and they are written in bold format in the text, such as $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.

A matrix $\boldsymbol{A}$ with dimensions $m \times n$ belongs to the space $\mathbb{R}^{m \times n}$ if all its elements belong to the set of real numbers $\mathbb{R}$. Thus, a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is defined as follows.

$$\boldsymbol{A} = \begin{matrix} a_{11} & a_{12} & \cdots a_{1n} \\ a_{21} & a_{22} & \cdots a_{2n} \\ \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots a_{mn} \end{matrix} \tag{A.3}$$

In which $a_{ij} \in \mathbb{R}, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n$.

A square matrix $\boldsymbol{A}$ with dimensions $n \times n$ is "upper triangular" if for $i > j$, the elements $a_{ij} = 0$; "lower triangular" if for $i < j$, the elements $a_{ij} = 0$; and "diagonal" if for $i \neq j$, the elements $a_{ij} = 0$.

A "diagonal matrix" with all elements equal to one is called an identity matrix and is denoted by $\boldsymbol{I}$. A diagonal matrix with all elements equal to zero is called a zero matrix and is represented by $\mathbf{0}$. The "transpose" of a matrix $m \times n$ is created by swapping its rows and columns and is denoted by $\boldsymbol{A}^T \in \mathbb{R}^{n \times m}$. A square matrix $\boldsymbol{A}$ is "symmetric" if $\boldsymbol{A}^T = \boldsymbol{A}$; "skew-symmetric" if $\boldsymbol{A}^T = -\boldsymbol{A}$; and "orthogonal" if $\boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{A} \boldsymbol{A}^T = \boldsymbol{I}$.

## A.2 Vectors and Matrices Arithmetic

The inner product of two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, is a scalar value calculated as follows.

$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^T \boldsymbol{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n \tag{A.4}$$

The inner product is commutative if: $\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{y} \cdot \boldsymbol{x}$. Two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, are orthogonal if their inner product is equal to zero: $\boldsymbol{x} \cdot \boldsymbol{y} = 0$. The *Euclidean norm* of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is denoted by $\| \boldsymbol{x} \|$, representing the geometric length of the vector, is determined as follows.

$$\| \boldsymbol{x} \| = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

(A.5)

A unit vector is represented by the notation $\hat{\boldsymbol{x}}$ and is determined by dividing the vector by its own Euclidean norm.

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\| \boldsymbol{x} \|}$$

(A.6)

The Euclidean norm of vectors has the following properties:

$$|\boldsymbol{x} \cdot \boldsymbol{y}| \quad \leq \| \boldsymbol{x} \| \cdot \| \boldsymbol{y} \|$$

(A.7)

$$\| \boldsymbol{x} + \boldsymbol{y} \| \quad \leq \| \boldsymbol{x} \| + \| \boldsymbol{y} \|$$

(A.8)

The Equation A.7 corresponds to the Cauchy-Schwartz inequality, while the Equation A.8 is associated with the well-known triangle inequality.

The cross product of two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, results in another vector and is calculated as follows in $\mathbb{R}^3$.

$$\boldsymbol{z} = \boldsymbol{x} \times \boldsymbol{y} = \begin{matrix} x_2 y_3 - x_3 y_2 \\ x_3 y_1 - x_1 y_3 \\ x_1 y_2 - x_2 y_1 \end{matrix} \tag{A.9}$$

The properties of inner and cross products of vectors can be listed with the following relations.

$$\boldsymbol{x} \times \boldsymbol{x} \quad = \boldsymbol{0} \tag{A.10}$$

$$\boldsymbol{x} \times \boldsymbol{y} \quad = -\boldsymbol{y} \times \boldsymbol{x} \tag{A.11}$$

$$\boldsymbol{x} \times (\boldsymbol{y} + \boldsymbol{z}) \quad = \boldsymbol{x} \times \boldsymbol{y} + \boldsymbol{x} \times \boldsymbol{z} \tag{A.12}$$

$$\alpha(\boldsymbol{x} \times \boldsymbol{y}) \quad = (\alpha \boldsymbol{x}) \times \boldsymbol{y} = \boldsymbol{x} \times (\alpha \boldsymbol{y}) \tag{A.13}$$

$$\boldsymbol{x} \cdot (\boldsymbol{y} \times \boldsymbol{z}) \quad = \boldsymbol{y} \cdot (\boldsymbol{z} \times \boldsymbol{x}) = \boldsymbol{z} \cdot (\boldsymbol{x} \times \boldsymbol{y}) \tag{A.14}$$

$$(\boldsymbol{x} \cdot \boldsymbol{y})\boldsymbol{z} \quad = (\boldsymbol{x}^T\boldsymbol{y})\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x}^T\boldsymbol{y}) = (\boldsymbol{z}\boldsymbol{x}^T)\boldsymbol{y} = \boldsymbol{z}\boldsymbol{x}^T\boldsymbol{y}$$

(A.15)

Equation A.14 describes the property of the triple product, which is widely used in determining the dynamics equations in robotics. The cross product of two vectors, $\boldsymbol{x} \times \boldsymbol{y}$, can be expressed using the product of a skew-symmetric matrix formed by the elements of the vector $\boldsymbol{x}$, denoted by $\boldsymbol{x}_\times$ or $\boldsymbol{x}^\times$, as given in

$$\boldsymbol{x}_\times \text{ or } \boldsymbol{x}^\times = \begin{matrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{matrix}$$

(A.16)

by vector $\boldsymbol{y}$, as follows.

$$\boldsymbol{x} \times \boldsymbol{y} = \boldsymbol{x}^\times \boldsymbol{y} = -\boldsymbol{y}^\times \boldsymbol{x}$$

(A.17)

A set of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ is linearly independent, if and only if, the linear combination of these vectors equals zero only when all the coefficients are zero.

$$\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i = \boldsymbol{0} \Rightarrow \alpha_i = 0 \quad \forall i$$

(A.18)

The rank of a matrix $\boldsymbol{A}$, denoted by $\text{rank}(\boldsymbol{A})$, is the maximum number of columns or rows of this matrix that are linearly independent. With this definition, the rank of a matrix with dimensions $n \times m$ will never be greater than $\min(m, n)$. The inverse of a square matrix $\boldsymbol{A}$ with dimensions $n \times n$, if it exists, is a matrix of the same dimensions that satisfies the following relationship.

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$$

(A.19)

In this equation, $\boldsymbol{I}$ denotes the identity matrix. The inverse of a square matrix, if it exists, could be uniquely computable, if and only if, the matrix is full rank. The inverse of a matrix has the following property.

$$(\boldsymbol{A}^{-1})^{-1} = \boldsymbol{A}$$

(A.20)

$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$$

(A.21)

The inverse of an orthogonal matrix is equal to its transpose: $\boldsymbol{A}^{-1} = \boldsymbol{A}^{T}$. If matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ are invertible

square matrices, and matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ have appropriate dimensions for the following multiplications, then

$$(\boldsymbol{A}+\boldsymbol{BCD})^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{DA}^{-1}\boldsymbol{B}+\boldsymbol{C}^{-1})^{-1}\boldsymbol{DA}^{-1} \tag{A.22}$$

In this equation, the matrix $(\boldsymbol{DA}^{-1}\boldsymbol{B}+\boldsymbol{C}^{-1})$ must be invertible. Furthermore, for a matrix that is partitioned into blocks, the following relationships hold.

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}^{-1} = \begin{bmatrix}\boldsymbol{A}^{-1}+\boldsymbol{A}^{-1}\boldsymbol{B}\Delta^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\Delta^{-1}\\ -\Delta^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & \Delta^{-1}\end{bmatrix} \tag{A.23}$$

$$= \begin{bmatrix}\Omega^{-1} & -\Omega^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\\ -\boldsymbol{D}^{-1}\boldsymbol{C}\Omega^{-1} & \boldsymbol{D}^{-1}+\boldsymbol{D}^{-1}\boldsymbol{C}\Omega^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\end{bmatrix} \tag{A.24}$$

In which $\Delta=\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$ and $\Omega=\boldsymbol{A}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ must be invertible. In the special case where matrices $\boldsymbol{B}$ or $\boldsymbol{C}$ are zero matrices, the inversion of this matrix simplifies as follows. This relationship is known as the “Matrix Inversion Lemma.”

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}^{-1} = \begin{bmatrix}\boldsymbol{A}^{-1} & \boldsymbol{0}\\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{D}^{-1}\end{bmatrix}$$

(A.25)

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \\ \boldsymbol{0} & \boldsymbol{D}^{-1} \end{bmatrix}$$

(A.26)

The null space of matrix $\boldsymbol{A}$, denoted by $\mathcal{N}$, is defined as follows.

$$\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}$$

(A.27)

If the matrix $\boldsymbol{A}$ has dimensions $n \times n$, then

$$\text{rank}(\boldsymbol{A}) + \dim \mathcal{N}(\boldsymbol{A}) = n$$

(A.28)

Thus, a matrix is invertible, if and only if, its null space consists solely of the zero vector.

$$\dim \mathcal{N}(\boldsymbol{A}) = 0$$

The induced norm of a square matrix is defined as follows.

$$\| \boldsymbol{A} \| = \sup_{\|\boldsymbol{x}\| \neq 0} \frac{\| \boldsymbol{A}\boldsymbol{x} \|}{\| \boldsymbol{x} \|}$$

(A.29)

With this definition, it can be shown that:

$$\| \boldsymbol{A}\boldsymbol{x} \| \quad \leq \| \boldsymbol{A} \| \| \boldsymbol{x} \|$$

(A.30)

$$\| \boldsymbol{A}\boldsymbol{B} \| \quad \leq \| \boldsymbol{A} \| \| \boldsymbol{B} \|$$

(A.31)

## A.3 Eigenvalues and Singular Values

Eigenvalues of a square matrix $\boldsymbol{A}$ are obtained from the solution of the characteristic equation of the system, which is defined as follows.

$$\det \left( \lambda \boldsymbol{I} - \boldsymbol{A} \right) = 0$$

(A.32)

The solutions to this equation, $\lambda_1, \lambda_2, \ldots, \lambda_n$, determine the eigenvalues of the matrix $\boldsymbol{A}$. The eigenvalues of the matrix $\boldsymbol{A}$ and its transpose $\boldsymbol{A}^T$ are equal. The vectors that satisfy the equation

$$(\lambda_i \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_i = \boldsymbol{0}$$

(A.33)

are called the eigenvectors of the matrix $\boldsymbol{A}$. If the eigenvalues of matrix $\boldsymbol{A}$ are distinct, then there exist $n$ independent eigenvectors that satisfy Equation A.33. The matrix $\boldsymbol{T}$, composed of the eigenvectors of matrix $\boldsymbol{A}$ as its columns, is referred to as the homogeneous transformation matrix, which is used to convert matrix $\boldsymbol{A}$ into diagonal form. Using this matrix, the converted matrix $\boldsymbol{\Lambda} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}$ will be diagonal, and its diagonal elements will be the eigenvalues of matrix $\boldsymbol{A}$.

Eigenvalues and eigenvectors are not defined for non-square matrices. For this type of matrices, the concept of singular values is utilized. Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ where $m < n$. In this case, a square matrix with dimensions $m \times m$ can be formed using the product $\boldsymbol{A}\boldsymbol{A}^T$. The singular values of matrix $\boldsymbol{A}$ denoted by $\sigma_i$ are obtained from the square root of the eigenvalues of the square matrix $\boldsymbol{A}\boldsymbol{A}^T$.

$$\sigma_i(\boldsymbol{A}) = \sqrt{\lambda_i(\boldsymbol{A}\boldsymbol{A}^T)}$$

(A.34)

In many applications, there is a need to decompose a matrix based on its singular values. In singular value decomposition (SVD), a non-square matrix $\boldsymbol{A}$ is decomposed as follows.

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

(A.35)

In which the orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ are defined as follows.

$$\boldsymbol{U} = [\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \cdots \boldsymbol{u}_m], \boldsymbol{V} = [\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \cdots \boldsymbol{v}_n]$$

(A.36)

Furthermore, the matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ is calculated based on the singular values of the matrix as follows.

$$\boldsymbol{\Sigma} = \left[\begin{array}{cccc|ccc} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \sigma_m & 0 & \cdots & 0 \end{array}\right]$$

(A.37)

Thus, to decompose the singular value matrix $\boldsymbol{A}$, first, the singular values are calculated using Equation A.34 as follows.

$$\det\left(\boldsymbol{A}\boldsymbol{A}^T - \lambda_i \boldsymbol{I}\right) = 0; \quad \sigma_i = \sqrt{\lambda_i}$$

(A.38)

Next, using the obtained singular values, eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$ are calculated, which satisfy the following equation.

$$(\boldsymbol{A}\boldsymbol{A}^T - \sigma_i^2 \boldsymbol{I})\boldsymbol{u}_i = \boldsymbol{0}$$

(A.39)

Now, the orthogonal matrix $\boldsymbol{U}$ is formed using the determined eigenvectors as $\boldsymbol{U} = [\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \cdots \quad \boldsymbol{u}_m]$, and Equation A.39 can be rewritten as follows.

$$\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Sigma}_m^2$$

(A.40)

In this equation, $\boldsymbol{\Sigma}_m$ is the diagonal part of the matrix $\boldsymbol{\Sigma}$, which is obtained from:

$$\boldsymbol{\Sigma}_m = \begin{matrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & \sigma_m \end{matrix}$$

(A.41)

With this equation, one can obtain the $m$ columns of matrix $\boldsymbol{V}$, denoted by $\boldsymbol{V}_m$, by:

$$\boldsymbol{V}_m = \boldsymbol{A}^T\boldsymbol{U}\boldsymbol{\Sigma}_m^{-1}$$

(A.42)

The remaining $n-m$ columns of matrix $\boldsymbol{V}$, denoted by $\boldsymbol{V}_{n-m}$, can be determined from the orthogonal matrix $\boldsymbol{V} = [\boldsymbol{V}_m \mid \boldsymbol{V}_{n-m}]$, although this part of the matrix directly does not affect the singular value decomposition. The singular value decomposition of matrices is widely utilized in various applications, particularly in solving inverse problems for matrices used in robotics.

---

## A.4 Pseudo-Inverse

The pseudo-inverse of matrices is extensively used in the context of solving linear inverse problems. The solution to the matrix equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ can be easily obtained using the pseudo-inverse of matrix $\boldsymbol{A}$: $\boldsymbol{x} = \boldsymbol{A}^\dagger \boldsymbol{y}$. This holds true even when matrix $\boldsymbol{A}$ is not square, as long as it has full rank. The pseudo-inverse of matrix $\boldsymbol{A}$ generalizes this relationship to cases where matrix $\boldsymbol{A}$ is not square or is not full rank. In robotics, the pseudo-inverse of the Jacobian matrix is commonly used in redundancy resolution and robot calibration problems.

**Theorem A.1 (Pseudo-Inverse)** *Consider a real, full-rank matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. The pseudo-inverse of matrix $\boldsymbol{A}$, denoted by $\boldsymbol{A}^\dagger$, is uniquely defined by a matrix that satisfies the following four conditions.*

$$\boldsymbol{A}\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{A}$$

(A.43)

$$\boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\dagger}$$

(A.44)

$$(\boldsymbol{A}\boldsymbol{A}^{\dagger})^{T} = \boldsymbol{A}\boldsymbol{A}^{\dagger}$$

(A.45)

$$(\boldsymbol{A}^{\dagger}\boldsymbol{A})^{T} = \boldsymbol{A}^{\dagger}\boldsymbol{A}$$

(A.46)

To explore the proof of this theorem, please refer to [262].

### A.4.1 Pseudo-Inverse Properties

The pseudo-inverse possesses the following properties:

$$\begin{aligned}
(\boldsymbol{A}^{\dagger})^{\dagger} &= \boldsymbol{A} \\
(\boldsymbol{A}^{T})^{\dagger} &= \boldsymbol{A}^{\dagger T}, \;\; (\boldsymbol{A}\boldsymbol{A}^{T})^{\dagger} = \boldsymbol{A}^{\dagger T}\boldsymbol{A}^{\dagger} \\
\boldsymbol{A}^{\dagger} &= (\boldsymbol{A}^{T}\boldsymbol{A})^{\dagger}\boldsymbol{A}^{T} = \boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{A}^{T})^{\dagger} \\
\boldsymbol{A}^{T}\boldsymbol{A}\boldsymbol{A}^{\dagger} &= \boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{A}^{T} = \boldsymbol{A}^{T}
\end{aligned}$$

- Suppose matrix $\boldsymbol{A}$ has dimensions $m \times n$ and matrix $\boldsymbol{B}$ has dimensions $n \times p$. Then, in general, $(\boldsymbol{A}\boldsymbol{B})^{\dagger}$ is not equal to $\boldsymbol{B}^{\dagger}\boldsymbol{A}^{\dagger}$. However, if the ranks of the two

matrices are equal and given by $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{B}) = n$ , then the equality $(\boldsymbol{AB})^{\dagger} = \boldsymbol{B}^{\dagger}\boldsymbol{A}^{\dagger}$ holds.

- In a matrix $\boldsymbol{A}$ with dimensions $m \times n$, if $m < n$ and $\text{rank}(\boldsymbol{A}) = m$, then the matrix $\boldsymbol{AA}^T$ is an invertible matrix with dimensions $m \times m$. In this case, the pseudo-inverse is obtained by the following equation

$$\boldsymbol{A}^{\dagger} = \boldsymbol{A}^T(\boldsymbol{AA}^T)^{-1} \tag{A.47}$$

  This pseudo-inverse is referred to as the “right inverse.”

- In a matrix $\boldsymbol{A}$ with dimensions $m \times n$, if $m > n$ and $\text{rank}(\boldsymbol{A}) = n$, then the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is an invertible matrix with dimensions $n \times n$. In this case, the pseudo-inverse is obtained by the following relationship:

$$\boldsymbol{A}^{\dagger} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T \tag{A.48}$$

  This pseudo-inverse is referred to as the “left inverse.”

- If we express the singular value decomposition of the matrix $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$ (Equation A.37), then the pseudo-inverse $\boldsymbol{A}^{\dagger} \in \mathbb{R}^{m \times m}$ can be obtained using the following equation.

$$\boldsymbol{A}^{\dagger} = \boldsymbol{V} \begin{bmatrix} \boldsymbol{\Sigma}_m^{-1} & | & \boldsymbol{0} \end{bmatrix} \boldsymbol{U}^T$$

(A.49)

### A.4.2 Linear Inverse Problem

In this section, we use the pseudo-inverse to determine the solution to two inverse linear problems. Suppose we have a set of linear equations represented by the matrix equation $\boldsymbol{Ax} = \boldsymbol{y}$, where $\boldsymbol{A}$ is an $m \times n$ matrix. In this case, the solution to this system of linear equations for an unknown vector $\boldsymbol{x}$ with dimensions $n \times 1$ is the vector $\boldsymbol{y}$ with dimensions $m \times 1$. Now, if the matrix $\boldsymbol{A}$ is square ($m = n$) and non-singular, then with $\text{rank}(\boldsymbol{A}) = n$, the solution to the inverse problem can be determined using the inverse matrix $\boldsymbol{A}^{-1}$, and it is given by $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$.

In the case where the matrix $\boldsymbol{A}$ is non-square, with $m > n$, where the rank of a matrix $\boldsymbol{A}$ is $n$, there is no direct inverse for the matrix $\boldsymbol{A}$. In this scenario, the system of linear equations is over-determined, having more equations than unknowns. In general, such a system of equations does not have a solution that satisfies all equations, and the use of the pseudo-inverse matrix $\boldsymbol{A}^{\dagger}$ provides a solution that minimizes the Euclidean norm of the error vector $\boldsymbol{e} = \boldsymbol{Ax} - \boldsymbol{y}$.

$$V = \| \boldsymbol{Ax} - \boldsymbol{y} \|^2 = \| \boldsymbol{e} \|^2 = \boldsymbol{e}^T\boldsymbol{e}$$

(A.50)

To prove this claim, since in the optimal response, the derivative of the cost function must be equal to zero $\partial V / \partial \boldsymbol{x} = \mathbf{0}$, compute:

$$\begin{aligned} \frac{\partial (\boldsymbol{e}^T \boldsymbol{e})}{\partial \boldsymbol{x}} &= 2\boldsymbol{A}^T \boldsymbol{e} = 0 \\ &= 2\boldsymbol{A}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) = 0 \end{aligned} \tag{A.51}$$

As a result, in the optimal response, we will have:

$$\boldsymbol{A}^T \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T \boldsymbol{y} \tag{A.52}$$

Since $\boldsymbol{A}^T \boldsymbol{A}$ is a square matrix with dimensions $n \times n$, it is invertible. Therefore, the solution to the Equation A.52 can be obtained as follows.

$$\boldsymbol{x}_{opt} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{y} \tag{A.53}$$

As seen, the optimal solution to the inverse problem $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ can be determined by the left inverse of the matrix $\boldsymbol{A}$, as previously defined in Equation A.48.

Now, if the matrix $\boldsymbol{A}$ is non-square with $m < n$, and the rank of the matrix $\boldsymbol{A}$ is equal to $m$, in this case the direct inverse of the matrix $\boldsymbol{A}$ does not exist, and the number of

equations $m$ is less than the number of unknowns $n$. This system of linear equations is under-determined. In this situation, there are countless solutions to the system of linear equations, and one can choose a solution that optimizes a desired criterion. In general, it can be shown that the solution to this category of algebraic equations is obtained from the following equation.

$$\boldsymbol{x} = \boldsymbol{A}^{\dagger}\boldsymbol{y} + (\boldsymbol{I} - \boldsymbol{A}^{\dagger}\boldsymbol{A})\boldsymbol{z} \tag{A.54}$$

In which $\boldsymbol{z}$ is an arbitrary vector with a dimension of $n$, and $\boldsymbol{A}^{\dagger}$ is the right inverse of the matrix $\boldsymbol{A}$. According to the following calculations, this solution satisfies the system of linear equations, and therefore, it is considered as a general solution for this system.

$$\begin{aligned} \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{A}\boldsymbol{A}^{\dagger}\boldsymbol{y} + \boldsymbol{A}(\boldsymbol{I} - \boldsymbol{A}^{\dagger}\boldsymbol{A})\boldsymbol{z} \\ &= \boldsymbol{A}\boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{A}^{T})^{-1}\boldsymbol{y} + (\boldsymbol{A} - \boldsymbol{A}\boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{A}^{T})^{-1}\boldsymbol{A})\boldsymbol{z} = \boldsymbol{y} + ( \end{aligned}$$

Moreover, it can be shown that the base solution of this equation, determined by the relation $\boldsymbol{x}_o = \boldsymbol{A}^{\dagger}\boldsymbol{y}$, minimizes the norm $\| \boldsymbol{x} \|$, among countless solutions to this system of linear equations. The proof of this claim can be derived by considering that other solutions to this system of equations add an expression to the optimal base solution, which is orthogonal to the base solution, thereby increasing the

length of the resulting vector. To prove the orthogonality of these two vectors, it can be shown that their dot product is equal to zero.

$$\begin{aligned}(\boldsymbol{A}^{\dagger}\boldsymbol{y})^{T}(\boldsymbol{I}-\boldsymbol{A}^{\dagger}\boldsymbol{A}) &= \boldsymbol{y}^{T}\boldsymbol{A}^{\dagger^{T}}(\boldsymbol{I}-\boldsymbol{A}^{\dagger}\boldsymbol{A}) \\ &= \boldsymbol{y}^{T}\left[(\boldsymbol{I}-\boldsymbol{A}^{\dagger}\boldsymbol{A})\boldsymbol{A}^{\dagger}\right]^{T}\boldsymbol{z} \\ &= \boldsymbol{y}^{T}(\boldsymbol{A}^{\dagger}-\boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{A}^{\dagger})^{T}\boldsymbol{z}=0\end{aligned}$$

In these calculations, the symmetry property of the matrix $\boldsymbol{I}-\boldsymbol{A}^{\dagger}\boldsymbol{A}$ has been utilized to simplify the first equation, and Equation A.44 has been used to simplify the final equation. In this way, the right inverse is employed to solve under-determined inverse problems.

## A.5 Kronecker Product

The Kronecker product, denoted by $\otimes$, defines an operation for multiplying two matrices of arbitrary dimensions, resulting in a block matrix. If matrix $\boldsymbol{A}$ has dimensions $m \times n$ and matrix $\boldsymbol{B}$ has dimensions $p \times q$, then the Kronecker product of these two matrices, denoted as $\boldsymbol{A} \otimes \boldsymbol{B}$, is a matrix with dimensions $mp \times nq$. It is formed by creating a block matrix from the element-wise multiplication of the first matrix by the second matrix.

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{matrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots a_{2n}\boldsymbol{B} \\ \vdots & \vdots & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \cdots a_{mn}\boldsymbol{B} \end{matrix} \quad . \tag{A.55}$$

The Kronecker product possesses properties of bilinearity and associativity:

$$\alpha(\boldsymbol{A} \otimes \boldsymbol{B}) \quad = (\alpha \boldsymbol{A}) \otimes \boldsymbol{B} = \boldsymbol{A} \otimes (\alpha \boldsymbol{B}) \;\; ; \;\; \alpha \in R \tag{A.56}$$

$$\boldsymbol{A} \otimes (\boldsymbol{B} + \boldsymbol{C}) \quad = \boldsymbol{A} \otimes \boldsymbol{B} + \boldsymbol{A} \otimes \boldsymbol{C} \tag{A.57}$$

$$(\boldsymbol{A} + \boldsymbol{B}) \otimes \boldsymbol{C} \quad = \boldsymbol{A} \otimes \boldsymbol{C} + \boldsymbol{B} \otimes \boldsymbol{C} \tag{A.58}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B}) \otimes \boldsymbol{C} \quad = \boldsymbol{A} \otimes (\boldsymbol{B} \otimes \boldsymbol{C}) \tag{A.59}$$

Furthermore, for matrices with appropriate dimensions, their mixed product has the following property.

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = \boldsymbol{AC} \otimes \boldsymbol{BD}$$

(A.60)

The matrix $(\boldsymbol{A} \otimes \boldsymbol{B})$ is invertible, if and only if, matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible. In this case, the inverse of this matrix is calculated as follows.

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$$

(A.61)

Also, the transpose in this product has the property of distributivity.

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{T} = \boldsymbol{A}^{T} \otimes \boldsymbol{B}^{T}$$

(A.62)

Another property of the Kronecker product used in simplifying dynamic equations is as follows. With the definition of the vector $\boldsymbol{x} \in \mathbb{R}^n$, since the Kronecker product is not commutative in general, i.e., $\boldsymbol{I}_{n\times n} \otimes \boldsymbol{x} \neq \boldsymbol{x} \otimes \boldsymbol{I}_{n\times n}$, in which $\boldsymbol{I}_{n\times n}$ denotes the identity matrix of size $n \times n$. However, it can be shown that the following relationship holds for any arbitrary vector $\boldsymbol{x} \in \mathbb{R}^n$ .

$$(\boldsymbol{I}_{n\times n} \otimes \boldsymbol{x})\boldsymbol{x} = (\boldsymbol{x} \otimes \boldsymbol{I}_{n\times n})\boldsymbol{x}$$

(A.63)

Let us define the time derivative and partial derivative of matrices using the Kronecker product and investigate how the Kronecker product is used in the gradients of composite matrices. Consider the matrix $\boldsymbol{A}(\boldsymbol{q})$, according to the convention, the partial derivative of this matrix with respect to the variable $\boldsymbol{q}$ can be written in the form of a column-block matrix as follows.

$$\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{q}} = \begin{matrix} \frac{\partial \boldsymbol{A}}{\partial q_1} \\ \vdots \\ \frac{\partial \boldsymbol{A}}{\partial q_n} \end{matrix}$$

(A.64)

With this definition, the time derivative of a matrix can be obtained using the Kronecker product, as described in [200].

$$\frac{d\boldsymbol{A}}{dt} = \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{q}}^T (\boldsymbol{I}_{n\times n} \otimes \dot{\boldsymbol{q}})$$

(A.65)

If the relative derivative of the product of two matrices is of interest, use the following relationship.

$$\frac{\partial}{\partial \boldsymbol{q}}(\boldsymbol{A}(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})) = (\boldsymbol{I}_{n\times n} \otimes \boldsymbol{A})\frac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}} + \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{q}}\boldsymbol{B} \tag{A.66}$$

# B

# *Appendix B: Nonlinear Control Review*

DOI: 10.1201/9781003590408-B

This appendix provides a brief introduction to the representation of dynamic systems in state space, along with key definitions and stability theorems for nonlinear systems. The focus is specifically on the stability of time-invariant or autonomous systems. Readers seeking further study are encouraged to consult [233, 263].

## B.1 Dynamical Systems

As detailed in this book, the dynamics formulation of parallel robots are expressed by a set of nonlinear second-order differential equations. It has also been demonstrated that the inertia matrix in robots is positive-definite and invertible. With this consideration and by forming the state variable vector of the robot as the augmentation of the

motion variables and their derivatives, $\boldsymbol{x} = [\boldsymbol{q} \;\; \dot{\boldsymbol{q}}]^T$, the dynamic equations of the manipulator can be rewritten in state space.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

(B.1)

In this equation, $\boldsymbol{x}$ is the state vector of the system, $\boldsymbol{u}$ is the input vector, and $\boldsymbol{f}$ is a continuous and multivariate function $\boldsymbol{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, called as the vector field.

Equation B.1 represents the dynamic equations of the open-loop robot in state space. In this representation, the dynamic equations of robot motion are shown as a set of first-order differential equations, where the vector of joint torques appears as the input vector in the system. If the motion variables of the robot are computed based on control methods discussed in the book and applied to the system, the input vector $\boldsymbol{u}$ becomes a function of the robot state vector $\boldsymbol{x}$, i.e. $\boldsymbol{u} = \boldsymbol{g}(x)$. With this description, the closed-loop system can be simplified into the following equation.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})) = \boldsymbol{f}(\boldsymbol{x})$$

(B.2)

It is evident that in the general case, the dynamic equations of the closed-loop system are multivariable and nonlinear.

To analyze stability in such a system, it is necessary to utilize the theory of nonlinear systems.

## B.2 Stability Definitions

**Definition B.1 (Equilibrium Point)** *Consider the nonlinear system B.2 represented in the state space* $\mathbb{R}^n$. *An equilibrium point of the system refers to a point in the* $\mathbb{R}^n$ *space where the variations in the state variables are equal to zero, or in other words, the system comes to rest at this point,* $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$. *If at the origin,* $\dot{\boldsymbol{x}}$ *becomes zero, then* $\boldsymbol{f}(\mathbf{0}) = \mathbf{0}$. *In this case, the origin of the state space* $\mathbb{R}^n$ *is defined as an equilibrium point of the system.*

Note that if the equilibrium point of the system $\boldsymbol{f}$ is not at the origin, it can easily be transferred to the origin by a change of coordinates. Therefore, assuming $\boldsymbol{f}(\mathbf{0}) = \mathbf{0}$ constrain the generality of this definition. If the initial conditions of the state variables are at the origin, $\boldsymbol{x}(0) = \mathbf{0}$, then the time response of the state variables, known as the trajectory of the system, remains at the origin for all times; we call this the trivial trajectory.

$$\boldsymbol{x}(t) \equiv \mathbf{0} \qquad \forall t > 0$$

The stability analysis of a system essentially examines the characteristics of nontrivial trajectories in the system. In

general terms, it can be said that an equilibrium point in a system is stable if the trajectory created under initial conditions near the origin remains close to the equilibrium point and does not diverge from it. Stability in the sense of *Lyapunov* can be precisely defined as follows.

**Definition B.2 (Lyapunov Stability)** *The equilibrium point of the system B.2 at the origin is said to be:*

- ***Stable:*** *If and only if, for every* $\epsilon > 0$*, there exists a* $\delta(\epsilon) > 0$ *such that*

$$\| \boldsymbol{x}(0) \| < \delta \Rightarrow \| \boldsymbol{x}(t) \| < \epsilon \qquad \forall t > 0 \tag{B.3}$$

- ***Asymptotic Stable:*** *If and only if, it is stable and, in addition, there exists a* $\delta > 0$ *such that*

$$\| \boldsymbol{x}(0) \| < \delta \Rightarrow \lim \| \boldsymbol{x}(t) \| \to 0 \quad \text{as} \quad t \to \infty \tag{B.4}$$

- ***Unstable:*** *If it is not stable.*

Lyapunov stability can be interpreted as every trajectory starting from a neighborhood of the origin remains in the vicinity of the origin for all times. Furthermore, if the trajectory tends toward the origin as time approaches infinity, the equilibrium point is asymptotically stable. With this interpretation, if the system's state is perturbed

slightly from the origin, it returns to its equilibrium state. An equilibrium point is only unstable when, with the smallest deviation from the equilibrium point, the trajectory diverges and does not return toward the equilibrium.

It should be noted that these definitions are interpreted locally and are not global. This means that it may satisfy these conditions in a small neighborhood of the origin but not hold farther away. Stability is only *globally* achieved when these conditions hold for all initial conditions.

Another important concept of stability can be defined by bounding the trajectory. This definition of stability is used when the system is nonlinear and time-varying. In such a case, initial conditions are defined at time $t_0$, and the trajectory's properties are examined from that time onward. In time-varying systems, the concept of uniform stability is defined, where the behavioral properties of the trajectory are not dependent on the time $t_0$.

**Definition B.3 (Uniformly Ultimately Bounded (UUB))** *The trajectory of a system $\boldsymbol{x}(t) : [t_0, \infty) \to \mathbb{R}^n$ with initial conditions $\boldsymbol{x}(t_0) = x_0$ is said to be uniformly ultimately bounded (UUB) with respect to the set $\boldsymbol{S}$ if there exists a time $T(\boldsymbol{x}_0, \boldsymbol{S})$ such that for all $t \geq T(\boldsymbol{x}_0, \boldsymbol{S})$, the trajectory $\boldsymbol{x}(t)$ remains within the set $\boldsymbol{S}$.*

$$\boldsymbol{x}(t) \in \boldsymbol{S} \quad ; \quad \forall t \geq t_0 + T \tag{B.5}$$

This type of trajectory boundedness means that, starting from the initial time $t_0$, the state variables $\boldsymbol{x}(t)$ will eventually enter and remain within the set $\boldsymbol{S}$. Additionally, the uniformity in this definition implies that this boundedness is not a function of time $t_0$. If the set $\boldsymbol{S}$ is a small neighborhood near the equilibrium point, this type of boundedness is very close to the concept of stability, but it lacks the asymptotic property. This means that the trajectory remains in the vicinity of the equilibrium point with a limited but computable steady-state error.

## B.3 Lyapunov Stability

Alexander Mikhailovich Lyapunov (1857–1918) was a Russian mathematician and physicist renowned for advancing the theory of stability of dynamic systems and presenting a mathematical framework for this purpose [264]. The fundamental philosophy of Lyapunov's method is a mathematical generalization of a fundamental physical observation. In a dynamical system, if its total energy is continuously dissipated, then the system, whether linear or nonlinear, must converge to a stationary equilibrium at some point. Since the energy is a scalar function, the stability of a system can be inferred solely by examining changes in an energy-type scalar function.

To provide a brief overview of Lyapunov's direct method in the analysis of nonlinear systems, we will first present a

physical interpretation on a simple system and then articulate the mathematical generalization of this idea, precisely described by Lyapunov, in the form of some theorems.

To this end, consider a simple mechanical system composed of a mass, a spring, and a damper, as illustrated in Figure 14.1. Assume that in this system, both the spring and the damper exhibit nonlinear behavior and are modeled by the relationships $F_s = k_0 x + k_1 x^3$ and $F_d = b\dot{x}|\dot{x}|$, respectively. In this model, $k_0$, $k_1$ are the stiffness coefficients of the spring, and $b$ is the damping coefficient of the damper, all of which are scalar and positive values. Furthermore, assume that no external forces, such as driving forces or disturbances, are applied to the system. In this case, the dynamics formulation of the mass motion can be described using the following equation:

$$m\ddot{x} + b\dot{x}|\dot{x}| + k_0 x + k_1 x^3 = 0 \tag{B.6}$$

In this equation, $m$ represents the mass of the body, and $x$ is its displacement. This dynamic equation can be rewritten in the state space by considering the state vector $\boldsymbol{x} = [x \;\; \dot{x}]^T$, which augment the position and velocity of the mass. Stability analysis of the origin equilibrium point in this system can be interpreted by examining the convergence or divergence of the state vector $\boldsymbol{x}(t)$ with respect to its initial conditions $\boldsymbol{x}(0)$ in the vicinity of the

equilibrium point. It is evident that directly utilizing the stability definition is very challenging in determining the stability of the equilibrium point. For this reason, here, we analyze the total energy of the system and its behavior over time.

The total energy of the system, which is the sum of kinetic and potential energy and is denoted by the scalar variable $V(\boldsymbol{x})$, can be calculated in terms of the system's state vector as follows:

$$\begin{aligned} V(\boldsymbol{x}) &= \frac{1}{2}m\dot{x}^2 + \int_0^x (k_0 y + k_1 y^3) dy \\ &= \frac{1}{2}m\dot{x}^2 + \frac{1}{2}k_0 x^2 + \frac{1}{4}k_1 x^4 \end{aligned}$$

Now, examine the concept of the stability of motion based on the total energy of the system. Using physical observations, it can be stated that:

- The total energy of the system becomes zero only when the system is in its equilibrium state at the origin, $x = \dot{x} = 0$.
- Asymptotic stability implies the convergence of the total energy of the system toward zero over time.
- Instability is directly associated with an increase in the total energy within the system.

With these observations, it can be concluded that the stability of the motion trajectory is related to the rate of change of energy within the system, rather than the energy itself. By calculating the rate of change of energy with respect to time, it can be expressed as:

$$\dot{V}(\boldsymbol{x}) = m\ddot{x}\,\dot{x} + (k_0 x + k_1 x^3)\dot{x} = (m\ddot{x} + k_0 x + k_1 x^3)\dot{x} \tag{B.7}$$

In order to analyze the stability of the motion trajectory of the system $\boldsymbol{x}(t)$, it is essential to examine the solution of the state equations concerning its initial conditions $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Note that the solution of the system's state equations satisfies the relation in B.6. Therefore, substitute Equation B.6 into B.7, and obtain the changes in the total energy of the system as follows:

$$\dot{V}(\boldsymbol{x}) = (-b\,\dot{x}\,|\dot{x}|)\dot{x} = -b\,|\dot{x}^3| \leq 0 \tag{B.8}$$

This equation indicates that over time, the total energy of the system is consistently decreasing until $\dot{x} = 0$. This means that, regardless of the initial excitation, the system will lose energy until it reaches a stationary state, representing the concept of asymptotic stability at the origin.

To mathematically generalize these observations, we analyze the characteristics of the total energy function of the system.

- The total energy function of the system is a scalar function and strictly positive, becoming zero only at the origin: $V(\boldsymbol{x}) > 0 \;\; ; \;\; \forall \boldsymbol{x} \neq \mathbf{0}$.
- The total energy function of the system, $V(\boldsymbol{x})$, is zero at the equilibrium point: $V(\mathbf{0}) = 0$.
- The time derivative of the total energy function of the system, $\dot{V}(\boldsymbol{x})$, is uniformly decreasing.

**Definition B.4 (Locally Positive-Definite:)** *A continuous scalar function $V(\boldsymbol{x})$ is locally positive-definite, if and only if, $V(\mathbf{0}) = 0$, and within a neighborhood $B_{R_0}$ of the origin with radius $R_0$, it is strictly positive*: $V(\boldsymbol{x}) > 0$.

**Definition B.5 (Lyapunov Function Candidate:)** *A locally positive-definite function $V(\boldsymbol{x})$ is called a Lyapunov function candidate if it is continuous and has first-order partial derivatives with respect to the state vector $\boldsymbol{x}$ within a close neighborhood of the origin.*

Note that if the above conditions are satisfied throughout the entire state space, we call the scalar function globally positive-definite. Furthermore, we define $V(\boldsymbol{x})$ as globally negative-definite if $-V(\boldsymbol{x})$ is globally positive-definite. The function $V(\boldsymbol{x})$ is termed as positive semi-definite, if it can

become zero at points other than the origin but remains non-negative elsewhere: $V(\boldsymbol{x}) \geq 0$.

With this introduction, we can present the Lyapunov direct method for analyzing the stability of nonlinear autonomous systems in the form of the Lyapunov stability theorem.

**Theorem B.1 (Lyapunov Local Theorem)** *Suppose $\boldsymbol{x} = \mathbf{0}$ is the equilibrium point of the nonlinear system B.2, and $D \subset \mathbb{R}^n$ is a domain containing the equilibrium point. Furthermore, assume that the scalar function $V : D \to \mathbb{R}$ is continuous and locally positive-definite, such that*

$$V(\mathbf{0}) = 0 \ \text{ and } \ V(\boldsymbol{x}) > 0 \ \text{ in } \ D - \{\mathbf{0}\}$$
$$\dot{V}(\boldsymbol{x}) \leq 0 \ \text{ in } \ D$$

*Then, the equilibrium point $\boldsymbol{x} = \mathbf{0}$ is stable. If*

$$\dot{V}(\boldsymbol{x}) < 0 \ \text{ in } \ D - \{\mathbf{0}\}$$

*Then, the equilibrium point $\boldsymbol{x} = \mathbf{0}$ is asymptotically stable.*

Note that, despite using the total energy of the system as the Lyapunov candidate function in the presented example, this theorem imposes no specific constraints on the choice of this function other than its local positive-definite property within the domain $D$. Furthermore, the proof of the Lyapunov theorem does not have a direct connection to

the physical interpretation provided in this section, and it establishes the stability of the motion trajectory through a mathematical description rather than energy considerations [233]. The proof of this theorem is based on Lyapunov level sets $\boldsymbol{S}(c)$, defined as

$$\boldsymbol{S}(c) = \{\boldsymbol{x} \in \mathbb{R}^n \mid V(\boldsymbol{x}) = c\} \tag{B.9}$$

The general principles of proving this theorem are based on the fact that if $V(\boldsymbol{x})$ satisfies the theorem conditions, then definite level sets can be defined in the state space that do not intersect with each other and converges to the origin as the value of $c$ decreases. Moreover, these level sets also indicate the convergence direction of the motion trajectory, and with decreasing level sets, the motion trajectory asymptotically converges to the origin.

This mathematical interpretation of the stability of the motion trajectory and the Lyapunov stability theorem is very powerful; however, it is both direct but challenging. It is direct in the sense that if a Lyapunov candidate function satisfying these conditions can be found, the stability of the equilibrium point is easily ensured. However, since this theorem only provides sufficient conditions, the failure to satisfy these conditions does not necessarily imply instability. Instead, it means that an appropriate candidate for the Lyapunov function has not been chosen, and the search for a suitable candidate function must continue.

On the other hand, there is currently no comprehensive method for determining a suitable Lyapunov function candidate in nonlinear systems, and the developed methods are all limited to specific types of nonlinear systems [265]. Although the total energy of the system is always an appropriate initial choice for the Lyapunov function, if it does not satisfy the conditions of the theorem, a systematic method for tuning it and reaching a suitable Lyapunov function has not been found.

In the direct Lyapunov method, the stability and asymptotic stability of the equilibrium point are distinguished with a slight difference in the Lyapunov theorem conditions. This is done while finding a Lyapunov function that has a negative-definite derivative and satisfies asymptotic stability conditions is much more challenging than ensuring stability. Fortunately, this limitation has been reasonably addressed in the Krasovskii-Lasalle's invariance theorem, details of which will be reviewed in Section B.4.

Many methods and theorems derived from the Lyapunov stability theorem are applicable to various systems. In the rest of this chapter, only two cases of the extension of the Lyapunov direct method are presented: the Lyapunov's golbal stability theorem and the UUB Stability theorem.

**Theorem B.2 (Global Lyapunov Theorem)** *Suppose* $\boldsymbol{x} = \mathbf{0}$ *is the equilibrium point of the nonlinear system B.2, and there exists a continuous, locally definite positive scalar function* $V : \mathbb{R}^n \to \mathbb{R}$ *such that*

$$
\begin{aligned}
&V(\mathbf{0}) = 0 \ \text{ and } \ V(\boldsymbol{x}) > 0 \ \ , \ \ \forall \boldsymbol{x} \neq \mathbf{0} \\
&\text{If} \quad \| \ \boldsymbol{x} \ \| \rightarrow \infty \Rightarrow V(\boldsymbol{x}) \rightarrow \infty \\
&\dot{V}(\boldsymbol{x}) \leq 0, \ \ \forall \boldsymbol{x} \neq \mathbf{0}
\end{aligned}
\tag{B.10}
$$

*Then, the equilibrium point* $\boldsymbol{x} = \mathbf{0}$ *is "globally stable." Moreover, if*

$$\dot{V}(\boldsymbol{x}) < 0 \ \ in \ \ D - \{\mathbf{0}\}$$

*Then, the equilibrium point* $\boldsymbol{x} = \mathbf{0}$ *is "asymptotically globally stable."*

**Theorem B.3 (Uniformly Ultimately Bounded (UUB))** *Suppose* $\boldsymbol{x} = \mathbf{0}$ *is the equilibrium point of the nonlinear system B.2, and* $D \subset \mathbb{R}^n$ *is a domain containing the equilibrium point. Furthermore, assume that the scalar function* $V : D \rightarrow \mathbb{R}$ *is continuous and locally positive-definite, and* $\boldsymbol{S}(c)$ *is an arbitrary level set of the Lyapunov function V defined for* $c > 0$ *as in Equation B.9. In this case, the motion trajectory of the system* $\boldsymbol{x}(t)$ *will be uniformly ultimately bounded with respect to the set* $\boldsymbol{S}$ *if:*

$$\dot{V}(\boldsymbol{x}) < 0 \ \ : \forall \boldsymbol{x} \ \ \text{Out of Set} \ \ \boldsymbol{S}$$

If $\dot{V}$ is strictly negative outside the set $\boldsymbol{S}$, the system's trajectory outside this set will converge toward it. When the motion trajectory is within this set, convergence to the origin cannot be guaranteed. However, leaving the motion trajectory from this set is also not possible, because the direction of motion is from outside the set toward its interior for all trajectories. Therefore, the motion trajectory will remain uniformly within the set $\boldsymbol{S}$ until the ultimate time.

## B.4 Krasovskii–Lasalle Theorem

As mentioned earlier, in many cases, the trajectory of a nonlinear system is asymptotically stable, but it is not always easy to find a Lyapunov candidate function whose derivative is strictly negative. In cases where $\dot{V}$ is negative semi-definite, the concept of *invariant set* in dynamical systems can be used to analyze the asymptotic stability of motion trajectories and estimate the convergence region.

In this regard, this section focuses on the definition of invariant sets and how to use them to analyze the asymptotic stability of motion trajectories in nonlinear systems. The Krasovskii-Lasalle theorem is employed for this purpose. This theorem has broader applications and is used for analyzing the convergence of trajectories toward limit cycles, as well as estimating the convergence region of these trajectories [233].

**Definition B.6 (Invariant Set)** *The set* $\boldsymbol{M}$ *in the dynamical system described by the Equation B.2 is called an invariant set if any motion trajectory that starts inside this set remains within it for future times.*

$$\boldsymbol{x}(0) \in \boldsymbol{M} \Rightarrow \boldsymbol{x}(t) \in \boldsymbol{M} \quad , \quad \forall t > 0$$

An invariant set is a generalization of the concept of an equilibrium point with higher dimensions in a dynamical system. If a set contains only one equilibrium point of the system, it is called an invariant set, because if the initial conditions of the system conform to the equilibrium point, no new trajectory is generated, and the system remains on the equilibrium point in future times.

A set that includes multiple equilibrium points of the system is also an invariant set. Additionally, a set of points that forms a limit cycle of a dynamic system is also considered an invariant set. A set of points that describes a specific motion trajectory of the system from time zero to infinity is also an invariant set. In both cases, if we start the motion from a point in these sets, the points forming the motion trajectory will remain on the limit cycle or predefined trajectory in future times, and thus they will not leave the invariant set.

The Krasovskii-Lasalle theorem introduces conditions under which all trajectories of the system will converge toward an invariant set. A direct consequence of this theorem in the case where the invariant set only contains

one equilibrium point is the asymptotic stability of the equilibrium point, even if the derivative of the Lyapunov candidate function is not strictly negative-definite.

**Theorem B.4 (Krasovskii-Lasalle)** *For the nonlinear system introduced in Equation B.2, assume $D \subset \mathbb{R}^n$ is a bounded region in the state space, and $\Omega \subset D$ is a compact set that is invariant with respect to the dynamical system. Additionally, suppose $V : D \to \mathbb{R}$ is a continuously differentiable Lyapunov candidate function such that it is negative semi-definite in the region $\Omega$ ($\dot{V} \leq 0$). In this case, let $E$ be a set of points in $\Omega$ where the Lyapunov function's derivative is zero ($\dot{V} = 0$), and let $M$ be the largest invariant set in $E$. Then, any trajectory starting from within $\Omega$ will converge to the invariant set $M$ as time goes to infinity, i.e., $t \to \infty$.*

This theorem does not directly address the asymptotic stability of the equilibrium point in cases where $\dot{V}(\boldsymbol{x})$ is negatively semi-definite, and it is not limited to such applications. However, when the origin $\boldsymbol{x} = \mathbf{0}$ is the only point within the invariant set $M$, asymptotic stability of the equilibrium point is guaranteed even when $V(\boldsymbol{x})$ is negatively semi-definite. This implies that if the Lyapunov function $V$ does not become zero along any trajectory, it must converge to the equilibrium point. In this case, the only solution of the dynamical system satisfying $\dot{V} = 0$ is the equilibrium point at the origin. The result of this

theorem is used to establish asymptotic stability at the origin in cases where $V(\boldsymbol{x})$ is negative semi-definite.

# Bibliography


[1] Hamid D. Taghirad. *Fundamentals of Robotics: Applied Case Studies with MATLAB® and Python*. CRC Press, 2025.

[2] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, second edition, 2020.

[3] John J. Craig. *Introduction to Robotics*. Pearson Education Limited, fourth edition, 2021.

[4] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010. ISBN 1849966346.

[5] Saeed B. Niku. *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2020.

[6] J. P. Merlet. *Parallel Robots*. Springer, second edition, 2006.

[7] X. Kong and C. Gosselin. *Type Synthesis of Parallel Mechanisms*. Springer, 2007.

[8] Lung-Wen Tsai. *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons, 1999.

[9] L. A. Ballard, S. Sabanovic, J. Kaur, and S. Milojevic. George Charles Devol Jr. History. *IEEE Robotics & Automation Magazine*, 19 (3): 114–119, 2012.

[10] Johanna Wallén. *The history of the industrial robot*. Linköping University Electronic Press, 2008.

[11] S. Saeedvand, M. Jafari, H. S. Aghdasi, and J. Baltes. A comprehensive survey on humanoid robot development. *The Knowledge Engineering Review*, 34, 2019.

[12] J. F. Engelberger. *Robotics in Practice*. Amacom, New York, 1980.

[13] International Federation of Robotics. Standardization, Robot definitions at ISO. URL https://ifr.org/standardisation

[14] Karel Čapek. *Rossum's Universal Robots*. Doubleday, New York, 1923. translated by Paul Selver.

[15] M. Grübler. *Getriebelehre*. Springer–Verlag, Berlin, 1917.

[16] I.A. Bonev. The true origins of parallel robots. http://www.parallemic.org/Reviews/Review007.html, January, 24, 2003.

[17] V. E. Gough. Contribution to discussion to papers on research in automobile stability on control and in tyre performance. *Proceedings of the Automotive Division Instrument Engineering*, pages 392–394, 1956.

[18] C.W. French et al. Multi-axial subassemblage testing (MAST) system: description and capabilities. In *13th World Conf. on Earthquake Engineering*, pages 2146–51. Vancouver, Canada, August 2004.

[19] K.L. Cappel. Motion simulator. US Patent No. 3,295,224, January 3, 1967.

[20] D. Stewart. A platform with six degrees of freedom. *Proceedings of the UK Institute of Mechanical Engineering*, 180 (1): 371–386, 1965.

[21] B. Dasgupta and T. Mruthyunjaya. The stewart platform manipulator: a review. *Mechanism and machine theory*, 35 (1): 15–40, 2000.

[22] A. Gasparetto and L. Scalera. From the unimate to the delta robot: the early decades of industrial robotics. In *Explorations in the history and heritage of machines and mechanisms*, pages 284–295. Springer, 2019.

[23] R. Clavel. *Conception d'un robot paralléle rapide á 4 degrés de liberté*. PhD thesis, EPFL, Lausanne, Switzerland, 1991.

[24] R. Clavel. Delta, a fast robot with parallel geometry. In *Proc. of the 18th International Symposium on Industrial Robots*, pages 91–100, April 1988.

[25] S. Zimmermann, R. Poranne, and S. Coros. Go fetch! Dynamic grasps using boston dynamics spot with external robotic arm. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4488–4494. IEEE, 2021.

[26] G. Bai, Y. Ge, D. Scoby, B. Leavitt, V. Stoerger, N. Kirchgessner, S. Irmak, G. Graef, J. Schnable, and T. Awada. Nu-spidercam: A large-scale, cable-driven, integrated sensing and robotic system for advanced phenotyping, remote sensing, and agronomic research. *Computers and Electronics in Agriculture*, 160: 71–81, 2019.

[27] H. Li, C.M. Gosselin, M.J. Richard, and B. Mayer St-Onge. Analytic form of the six-dimensional singularity locus of the general gough-stewart platform. *ASME Journal of Mechanical Design*, 128: 279–287, 2006.

[28] Z. Huang, Y. Cao, Y.W. Li, and L.H. Chen. Structure and property of the singularity loci of the 3/6–Stewart–

Gough platform for general orientations. *Robotica*, 24 (1): 75–84, 2006.

[29] D.H. Chaubert, A.O. Boryssenko, A. van Ardenne, J.G. Bij de Vaate, and C. Craeye. The square kilometer array (SKA) antenna. In *IEEE Int. Symp. Phased Array Systems and Technology*, pages 351–358, Oct. 2003.

[30] M.V. Ivashina, A. Van Ardenne, J.D. Bregman, J.G.B. de Vaate, and M. van Veelen. Activities for the square kilometer array (SKA) in Europe. In *Proc. of the Int. Conf. Antenna Theory and Techniques*, pages 633–636. USA, Sept. 2003.

[31] B. Carlson et al. The large adaptive reflector: A 200-m diameter, wideband, cm-wave radio telescope. In *Radio Telescopes-Proceedings of SPIE Meeting 4015*, pages 33–44. Bellingham,WA, 2000.

[32] C. Lambert, A. Saunders, C. Crawford, and M. Nahon. Design of a one-third scale multi-tethered aerostat system for precise positioning of a radio telescope receiver. In *CASI Flight Mechanics and Operations Symposium*. Montreal, Canada, 2003.

[33] Rendong Nan. Five hundred meter aperture spherical radio telescope (fast). *Science in China series G*, 49: 129–148, 2006.

[34] P. Jiang et al. Commissioning progress of the fast. *Science China Physics, Mechanics & Astronomy*, 62: 1–

22, 2019.

[35] S. Kawamura, H. Kino, and Ch. Won. High-speed manipulation by using parallel wire-driven robots. *Robotica*, pages 13–21, 2000.

[36] S. Qian, B. Zi, W. W. Shang, and Q. S. Xu. A review on cable-driven parallel robots. *Chinese Journal of Mechanical Engineering*, 31 (1): 1–11, 2018.

[37] J. Albus, R. Bostelman, and N. Dagalakis. The NIST roboCrane. *J. Nat. Inst. Standards Technol.*, 97 (3), 1992.

[38] National Institute of Standard and Technology. Robocrane project, large scale manufacturing using cable control. http://www.isd.mel.nist.gov/projects/robocrane, March 2001.

[39] E. R. Davies. *Machine vision: theory, algorithms, practicalities*. Elsevier, 2004.

[40] L. Rabiner and B. H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.

[41] T. Dean, J. Allen, and Y. Aloimonos. *Artificial intelligence: theory and practice*. Benjamin-Cummings Publishing Co., Inc., 1995.

[42] Y. Wang and F. Zhang. *Trends in control and decision-making for human-robot collaboration systems*.

Springer, 2017.

[43] J. P. Laumond et al. *Robot motion planning and control*, volume 229. Springer, 1998.

[44] S. G. Tzafestas. *Introduction to mobile robot control*. Elsevier, 2013.

[45] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[46] S. Thrun, W. Burgard, and D. Fox. *Probalistic robotics*. Emerald Group Publishing Limited, 2006.

[47] C. W. De Silva. *Sensors and actuators: Engineering system instrumentation*. CRC Press, 2015.

[48] S. T. Venkataraman and T. Iberall. *Dextrous robot hands*. Springer Science & Business Media, 2012.

[49] Z. Kappassov, J. A. Corrales, and V. Perdereau. Tactile sensing in dexterous robot hands. *Robotics and Autonomous Systems*, 74: 195–220, 2015.

[50] L. W. Tsai and S. A. Joshi. Kinematics and optimization of a spatial 3-UPU parallel manipulator. *ASME J. Mech. Design*, 122: 439–466, 2000.

[51] A. Fattah and G. Kasaei. Kinematics and dynamics of a parallel manipulator with a new architecture. *Robotica*, 18 (5): 535–543, 2000.

[52] S. Joshi and L. W. Tsai. A comparison study of two 3-DOF parallel manipulators: one with three and the other with four supporting legs. *IEEE Transactions Robotics and Automation*, 19: 200–209, 2002.

[53] A. Sokolov and P. Xirouchakis. Kinematics analysis of a 3–dof parallel manipulator with r–p–s joint structure. *Robotica*, 23 (2): 207–217, 2005.

[54] Y. Yangmin Li and Q. Xu. Kinematic analysis and design of a new 3–dof translational parallel manipulator. *Transactions of the ASME. Journal of Mechanical Design*, 128 (4): 729–737, 2006.

[55] P. L. Richard and C. M. Gosselin. Kinematic analysis and prototyping of a partially decoupled 4–DOF 3T1R parallel manipulator. *ASME Journal of Mechanical Design*, 129 (6): 611–616, 2007.

[56] L. Euler. Nova methodus motum corporum rigidorum degerminandi. *Novi commentarii academiae scientiarum Petropolitanae*, pages 208–238, 1776.

[57] M. Nitschke and E. H. Knickmeyer. Rotation parameters: A survey of techniques. *Journal of surveying engineering*, 126 (3): 83–105, 2000.

[58] M. Ceccarelli. Screw axis defined by giulio mozzi in 1763 and early studies on helicoidal motion. *Mechanism and Machine Theory*, 35 (6): 761–770, 2000.

[59] J. M. McCarthy and G. S. Soh. *Geometric design of linkages*, volume 11. Springer Science & Business Media, 2010.

[60] Giulio Mozzi. *Discorso matematico sopra il rotamento momentaneo dei corpi [Mathematical treatise on the temporally revolving of bodies]*. nella stamperia di Donato Campo, 1763.

[61] M. Chasles. Notes on the general properties of a system of two identical bodies randomly located in space; and on the finite or infinitesimal motion of a free solid body. *Bull. des Sci. Math., Astron., Phys. et Chim*, 14: 321–326, 1830.

[62] R. S. Ball. *A Treatise on the Theory of Screws*. Cambridge University Press, Cambridge, 1900.

[63] J. Gallardo-Alvarado. *Kinematic analysis of parallel manipulators by algebraic screw theory*. Springer, 2016.

[64] C. D Crane, M. Griffis, and J. Duffy. *Screw theory and its application to spatial robot manipulators*. Cambridge University Press, 2022.

[65] J. Pardos-Gotor. *Screw theory in robotics: an illustrated and practicable introduction to modern mechanics*. CRC Press, 2021.

[66] H. Cheng and K. C. Gupta. An historical note on finite rotations. *Journal of Applied Mechanics*, 56: 139–145, 1989.

[67] Z. Huang, Q. Li, and H. Ding. Basics of screw theory. In *Theory of Parallel Mechanisms*, pages 1–16. Springer, 2013.

[68] T. Y. Lam. Hamilton's quaternions. In *Handbook of algebra*, volume 3, pages 429–454. Elsevier, 2003.

[69] S. L. Altmann. *Rotations, quaternions, and double groups*. Courier Corporation, 2005.

[70] S. Sarabandi and F. Thomas. A survey on the computation of quaternions from rotation matrices. *Journal of mechanisms and robotics*, 11 (2), 2019.

[71] S. Zhao. Time derivative of rotation matrices: A tutorial. *arXiv preprint arXiv:1609.06088*, 2016.

[72] John Voight. *Quaternion algebras*. Springer Nature, 2021.

[73] K. W. Spring. Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review. *Mechanism and machine theory*, 21 (5): 365–373, 1986.

[74] J. L. Meriam and L. G. Kraige. *Engineering Mechanics: Dynamics*. Wiley, New York, 5th edition, 2007.

[75] Bernard Roth. Screws, motors, and wrenches that cannot be bought in a hardware store. In *Proc. Int. Symp. Robotics Research*, pages 679–693, 1984.

[76] Fuan Wen and Chonggao Liang. Displacement analysis of the 6–6 stewart platform mechanisms. *Mech. Mach. Theory*, 29 (4): 547–557, 1994.

[77] X. S. Gao, D. Lei, Qizheng Liao, and Gui-Fang Zhang. Generalized stewart-gough platforms and their direct kinematics. *IEEE Transactions Robotics*, 21 (2): 141–151, 2005.

[78] M. Tale Masouleh, C. Gosselin, M. H. Saadatzi, X. Kong, and H. D. Taghirad. Kinematic analysis of 5–RPUR (3T2R) parallel mechanisms. *Meccanica*, 46 (2): 131–146, 2011.

[79] M. Tale Masouleh, M. H. Saadatzi, and H. D. Taghirad. Workspace analysis of 5–PRUR parallel mechanisms (3T2R). *Journal of Robotics and Computer Integrated Manufacturing*, 28 (1): 437–448, 2012.

[80] J. P. Merlet. Direct kinematics of parallel manipulators. *IEEE Transactions on Robotics and Automation*, 9 (6): 842–845, 1993.

[81] Ping Ji and Hongtao Wu. An efficient approach to the forward kinematics of a planar parallel manipulator with similar platforms. *IEEE Transactions Robotics and Automation*, 18: 647–649, Aug. 2002.

[82] I.A. Bonev, J. Ryu, Sung-Gaun Kim, and Sun-Kyu Lee. A closed-form solution to the direct kinematics of nearly general parallel manipulators with optimally located three linear extra sensors. *IEEE Transactions Robotics and Automation*, 18: 148–156, 2001.

[83] B. Siciliano. The tricept robot: inverse kinematics, manipulability analysis and closed-loop direct kinematics algorithm. *Robotica*, 17: 437–445, June 1999.

[84] H. D. Taghirad and M. Nahon. Dynamic analysis of a redundantly actuated parallel manipulator: A virtual work approach. In *Iranian Conference on Electrical Engineering*, pages 54–60. Tehran, Iran, May 2007.

[85] H. D. Taghirad and M. Nahon. Kinematic analysis of a macro-micro redundantly actuated parallel manipulator. *Advanced Robotics*, 22 (2): 657–687, 2008.

[86] H. D. Taghirad and M. Nahon. Dynamic analysis of a macro-micro redundantly actuated parallel manipulator. *Advanced Robotics*, 22 (3): 949–981, 2008.

[87] V. Hayward. *Design of hydraulic robot shoulder based on combinatorial mechanism*, chapter 3, pages 297–310. Experimental Robotics 3, Lecture Notes in Control and Information Sciences 200. Springer-Verlag, New York, 1994.

[88] V. Hayward and R. Kurtz. Modeling of a parallel wrist mechanism with actuator redundancy. *Int. J. Laboratory Robotics and Automation*, 4 (2): 69–76, 1992.

[89] J. C. B. Grant and J. E. Anderson. *Grant's Atlas of anatomy*. Williams & Wilkins, 1978.

[90] M. Okada, Y. Nakamura, and S.I. Hoshino. Development of the cybernetic shoulder–a three dof mechanism that imitates biological shoulder motion. In *Int. Conf. Intelligent Robots and Systems*, pages 543–548, 1999.

[91] V. Hayward. *Design of a hydraulic robot shoulder based on a combinatorial mechanism*, chapter 6, pages 297–310. Lecture Notes in Control & Information Sciences. Springer-Verlag, The 3rd Int. Symp. Experimental Robotics III, Japan, Oct. 1994.

[92] H. Sadjadian and H. D. Taghirad. Comparison of different methods for computing the forward kinematics of a redundant parallel manipulator. *Journal of Intelligent and Robotic Systems*, 44: 225–246, 2005.

[93] H. Sadjadian and H. D. Taghirad. Kinematic, singularity and stiffness analysis of the hydraulic shoulder:a 3–dof redundant parallel manipulator. *Advanced Robotics*, 20 (7): 763–781, 2006.

[94] H. Sadjadian, H. D. Taghirad, and A. Fatehi. Neural networks approaches for computing the forward kinematics of a redundant parallel manipulator. *International Journal of Computational Intelligence*, 2 (1): 40–47, 2005.

[95] P. Dietmaier. The Stewart–Gough platform of general geometry can have 40 real postures. In *Proceedings of ARK*, pages 7–16. Strobl, Austria, June 1998.

[96] M. L. Husty. An algorithm for solving the direct kinematics of general Stewart–Gough platforms. *Mechanism and Machine Theory*, 31 (4): 365–379, 1996.

[97] P. Ji. and H. Wu. A closed-form forward kinematics solution for the 6-6P stewart platform. *IEEE Transactions on Robotics and Automation*, 17 (4): 522–526, 2001.

[98] J. P. Merlet. Solving the forward kinematics of a gough-type parallel manipulator with interval analysis. *The International Journal of Robotics Research*, 23 (3): 221–235, 2004.

[99] C. Innocenti. Forward kinematics in polynomial form of the general stewart platform. *J. Mech. Des.*, 123 (2): 254–260, 2001.

[100] T. Y. Lee and J. K. Shim. Forward kinematics of the general 6-6 stewart platform using algebraic

elimination. *Mechanism and Machine Theory*, 36 (9): 1073–85, 2001.

[101] M. Raghavan. The stewart platform of general geometry has 40 configurations. In *Proc. of the ASME Design and Automation Conference*, pages 397–402. Chicago, USA, Sept. 1991.

[102] C. W. Wampler. Forward displacement analysis of general six-in-parallel SPS (stewart) platform manipulators using soma coordinates. *Mechanism and Machine Theory*, 31 (3): 331–337, 1996.

[103] C. Innocenti. A novel numerical approach to the closure of the 6-6 stewart platform mechanism. In *Proc. of the Fifth International Conference on Advanced Robotics: Robots in Unstructured Environments*, pages 852–855, 1991.

[104] D. M. Ku. Direct displacement analysis of a stewart platform mechanism. *Mechanism and Machine Theory*, 34: 453–465, 1999.

[105] L. C. T. Wang and C. C. Chen. On the numerical kinematic analysis of general parallel robotic manipulators. *IEEE Transactions on Robotics and Automation*, 9 (3): 272–285, 1993.

[106] L. Baron and J. Angeles. The direct kinematics of parallel manipulators under joint-sensor redundancy.

*IEEE Transactions Robotics and Automation*, 16 (1): 12–19, 2000.

[107] Y. J. Chiu and M. H. Perng. Forward kinematics of a general fully parallelmanipulator with auxiliary sensors. *The International Journal of Robotics Research*, 20 (5): 401–414, 2001.

[108] K. R. Han, W. K. Chung, and Y. Youm. New resolution scheme of the forward kinematics of parallel manipulators using extra sensors. *ASME Journal of Mechanical Design*, 118 (2): 214–219, 1996.

[109] V. Parenti-Castelli and R. D. Gregorio. A new algorithm based on two extra-sensors for real-time computation for the actual configuration of the generalized Stewart–Gough manipulator. *ASME Journal of Mechanical Design*, 112: 294–298, 2000.

[110] Z. Geng and L. S. Haynes. Neural network solution for the forward kinematics problem of a stewart platform. *Robotics and Computer-Integrated Manufacturing*, 9 (6): 485–495, 1992.

[111] C. S. Yee. Forward kinematics solution of stewart platform using neural networks. *Neurocomputing*, 16 (4): 333–349, 1997.

[112] P. J. Parikh and S. Y. Lam. A hybrid strategy to solve the forward kinematics problem in parallel

manipulators. *IEEE Transactions on Robotics*, 21 (1): 18–25, 2005.

[113] Y. Wang. A direct numerical solution to forward kinematics of general Stewart-Gough platforms. *Robotica*, 25 (1): 121–128, 2007.

[114] M. Griffis and J. Duffy. A forward displacement analysis of a class of stewart platforms. *Journal of Robotic Systems*, 6 (6): 703–720, 1989.

[115] P. Nanua, K. J. Waldron, and V. Murthy. Direct kinematic solution of a stewart platform. *IEEE Transactions on Robotics and Automation*, 6 (4): 438–444, 1990.

[116] I.D. Akcali and H. Mutlu. A novel approach in the direct kinematics of stewart platform mechanisms with planar platforms. *Transactions of the ASME. Journal of Mechanical Design*, 128 (1): 252–263, 2006.

[117] K. Liu, J.M. Fitzgerald, and F.L. Lewis. Kinematic analysis of a stewart platform manipulator. *IEEE Transactions on Industrial Electronics*, 40 (2): 282–293, 1993.

[118] S. V. Sreenivasan, K. J. Waldron, and P. Nanua. Closed-form direct displacement analysis of a 6-6 stewart platform. *Mechanism and Machine Theory*, 29 (6): 855–864, 1994.

[119] J. Yang and Z. J. Geng. Closed form forward kinematics solution to a class of hexapod robots. *IEEE Transactions on Robotics and Automation*, 14: 503–508, 1998.

[120] J. J. Moré. *Numerical Analysis*, pages 105–116. Lecture Notes in Mathematics. Springer Verlag, 1977.

[121] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall Series in Computational Mathematics. Prentice-Hall, 1983.

[122] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2 (2): 164–168, 1944.

[123] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11 (2): 431–441, 1963.

[124] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical Analysis*, pages 105–116. Springer, 1978.

[125] P. Ji and Wu H. Algebraic solution to forward kinematics of a 3–DOF spherical parallel manipulator. *Journal of Robotic Systems*, 18 (5): 251–257, 2001.

[126] M. M. Aref, H. D. Taghirad, and S. Barissi. Optimal design of dexterous cable driven parallel

manipulators. *International Journal of Robotics Research:Theory and Applications*, 1 (1): 29–47, 2009.

[127] F. C. Park. Optimal Robot Design and Differential Geometry. *Journal of Mechanical Design*, 117 (B): 87–92, 06 1995.

[128] C. A. Klein and B. E. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6 (2): 72–83, 1987.

[129] T. Yoshikawa. The manipulability of robotic mechanisms. *International Journal of Robotics Research*, 4 (2): 3–9, 1985.

[130] C.M. Gosselin and J. Angeles. A global performance index for the kinematic optimization of robotic manipulators. *ASME Journal of Mechanical Design*, 113 (3): 220–226, 1991.

[131] M. A. Hosseini, H. R. M. Daniali, and H. D. Taghirad. Dexterous workspace optimization of a tricept parallel manipulator. *Advanced Robotics*, 25 (3): 1697–1721, 2011.

[132] C. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Transactions Robotics and Automation*, 6 (3): 281–290, 1990.

[133] K. Cleary and M. Uebel. Jacobian formulation for a novel 6-DOF parallel manipulator. In *IEEE Conf.*

*Robotics and Automation*, pages 2377–82, May 1994.

[134] D. Kim, W. Chung, and Y. Youm. Analytic jacobian of in-parallel manipulators. In *IEEE Conf. Robotics and Automation*, pages 2376–81, April 2000.

[135] O. Ma and J. Angeles. Architecture singularities of platform manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1542–1547. Sacramento, CA, 1991.

[136] J.P. Merlet. Singular configurations of parallel manipulators and grassmann geometry. *The International Journal of Robotics Research*, 8 (5): 45–56, 1989.

[137] M.G. Mohamed and J. Duffy. A direct determination of the instantaneous kinematics of fully parallel robot manipulators. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 107 (2): 226–229, 1985.

[138] D. Zlatanov, R. G. Fenton, and B. Benhabib. Singularity analysis of mechanisms and robots via a velocity-equation model of the instantaneous kinematics. In *IEEE International Conference on Robotics and Automation*, pages 986–991. San Diego, CA, USA, 1994.

[139] D. Zlatanov, I. A. Bonev, and C. M. Gosselin. Constraint singularities of parallel mechanisms. In

*IEEE International Conference on Robotics and Automation*, pages 496–502. Washington DC, USA, 2002.

[140] K.J. Waldron and K.H. Hunt. Series-parallel dualities in actively coordinated mechanisms. *Robotics Research*, 4: 175–181, 1988.

[141] K. H. Hunt. *Kinematic Geometry of Mechanisms*. Oxford University Press, Cambridge, 1978.

[142] E. F. Fichter. A stewart platform-based manipulator: General theory and practical construction. *The International Journal of Robotics Research*, 5 (2): 157–182, 1986.

[143] Z. Huang, L. H. Chen, and Y. W. Li. The singularity principle and property of stewart parallel manipulator. *Journal of Robotic Systems*, 20 (4): 163–176, 2003.

[144] R. Di Gregorio. Analytic formulation of the 6-3 fully-parallel manipulators singularity determination. *Robotica*, 19: 663–667, 2001.

[145] R. Di Gregorio. Singularity-locus expression of a class of parallel mechanisms. *Robotica*, 20: 323–328, 2002.

[146] A. Wolf and M. Shoham. Investigation of parallel manipultors using linear complex approximation. *Journal of Mechanical Design*, 125: 564–572, 2003.

[147] D. Kim and W. Y. Chung. Analytic singularity equation and analysis of six-dof parallel manipulators using local structurization method. *IEEE Transactions Rob. Autom.*, 5 (4): 612–622, 1999.

[148] B. Mayer St-Onge and C. M. Gosselin. Singularity analysis and representation of the general gough-stewart platform. *The International Journal of Robotics Research*, 19 (3): 271–288, 2000.

[149] T. Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4 (2): 3–9, 1985.

[150] T. Yoshikawa. Dynamic manipulability of robot manipulators. *Transactions of the Society of Instrument and Control Engineers*, 21 (9): 970–975, 1985.

[151] L. El Ghaoui. Inversion error, condition number, and approximate inverses of uncertain matrices. *Linear algebra and its applications*, 343: 171–193, 2002.

[152] C. Gosselin and J. Angeles. A global performance index for the kinematic optimization of robotic manipulators. *Journal of Mechanical Design*, 113 (3): 220–226, 09 1991.

[153] B.S. El-Khasawneh and P.M. Ferreira. Computation of stiffness and stiffness bounds for parallel link

manipulators. *International Journal of Machine Tools and Manufacture*, 39 (2): 321–342, 1999.

[154] C. Gosselin. Stiffness mapping for parallel manipulators. *IEEE Transactions Robotics and Automation*, 6 (3): 377–382, 1990.

[155] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME Journal of Dynamic Systems, Measurement and Control*, 108 (3): 163–171, 1986.

[156] C. A. Klein and C. H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 13 (2): 245–250, 1983.

[157] K. Glass, R. Colbaugh, D. Lim, and H. Seraji. Real-time collision avoidance for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 11 (3): 448–457, 1995.

[158] H. Seraji. Task-based configuration control of redundant manipulators. *Journal of Robotic Systems*, 9 (3): 411–451, 1992.

[159] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10 (2): 47–53, 1969.

[160] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7 (12): 868–871, 1977.

[161] J. Hollerbach and Ki Suh. Redundancy resolution of manipulators through torque optimization. *IEEE Journal of Robotics and Automation*, 3 (4): 308–316, 1988.

[162] O. Khatib. Unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Transactions on Robotics and Automation*, 3 (1): 43–53, 1987.

[163] R. Dubey and J. Y. S. Luh. Redundant robot control using task based performance measures. *Journal of Robotic Systems*, 5 (5): 409–432, 1988.

[164] S. L. Chiu. Task compatibility of manipulator postures. *International Journal of Robotics Research*, 7 (5): 13–21, 1988.

[165] H. Seraji. Configuration control of redundant manipulators: theory and implementation. *IEEE Transactions on Robotics and Automation*, 5 (4): 472–490, 1989.

[166] P. Bosscher, A. Riechel, and I. Ebert-Uphoff. Wrench-feasible workspace generation for cable-driven robots.

*IEEE Transaction on Robotics*, 22 (5): 890–902, Oct. 2006.

[167] S. R. Oh and S. K. Agrawal. Cable suspended planar robots with redundant cables: Controllers with positive tensions. *IEEE Transactions on Robotics*, 21 (3): 457–465, 2005.

[168] G. Barrette and C.M. Gosselin. Determination of the dynamic workspace of cable-driven planar parallel mechanisms. *Transactions ASME Journal of Mechanical Design*, 127 (2): 242–248, 2005.

[169] M. Hassan and A. Khajepour. Optimization of actuator forces in cable-based parallel manipulators using convex analysis. *IEEE Transactions on Robotics*, 24 (3): 736–740, 2008.

[170] M. Agahi and L. Notash. Redundancy resolution of wire-actuated parallel manipulators. *Transactions of CSME*, 33 (4): 561–573, 2009.

[171] M. J. D. Powell. *Variable Metric Methods for Constrained Optimization*. Mathematical Programming: The State of the Art. Springer Verlag, 1983.

[172] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 1987.

[173] H. D. Taghirad and Y. B. Bedoustani. An analytic-iterative redundancy resolution scheme for cable-

driven redundant parallel manipulators. *IEEE Transactions on Robotics*, 27 (6): 1137–43, 2011.

[174] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *The Proceedings of 2nd Berkeley Symposium, Berkeley*, pages 481–492, 1951.

[175] W. Karush. *Minima of functions of several variables with inequalities as side constraints*. Masters dissertation, Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois, 1939.

[176] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization*. Addison Wesley, 1991.

[177] Thomas F. Coleman and Yin Zhang. *Optimization Toolbox 6, User's Guide*. The MathWorks, Inc., MA, USA, April 2011.

[178] F. Sabahi, M. A. Khosravi, and H. D. Taghirad. Implementation of analytic iterative redundancy resolution technique on KNTU cable robot. In *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 690–694. IEEE, 2014.

[179] C. Gosselin. Parallel computational algorithms for the kinematics and dynamics of planar and spatial parallel manipulators. *Transactions ASME Journal of*

*Dynamic Systems, Measurement and Control*, 118 (1): 22–28, 1996.

[180] B. Dasgupta and T.S. Mruthyunjaya. A newton-euler formulation for the inverse dynamics of the stewart platform manipulator. *Mechanism and Machine Theory*, 33 (8): 1135–52, 1998.

[181] J. Wang and C.M. Gosselin. A new approach for the dynamic analysis of parallel manipulators. *Multibody System Dynamics*, 2 (3): 317–334, 1998.

[182] J. McPhee, P. Shi, and J.C. Piedboeuf. Dynamics of multibody systems using virtual work and symbolic programming. *Mathematical and Computer Modelling of Dynamical Systems*, 8 (2): 137–155, 2002.

[183] C. C. Nguyen and F. J Pooran. Dynamic analysis of a 6 DOF CKCM robot end-effector for dual-arm telerobot systems. *Robotics and Autonomous Systems*, 5 (4): 377–394, 1989.

[184] G. Lebret, K. Liu, and F. L. Lewis. Dynamic analysis and control of a stewart platform manipulator. *Journal of Robotic Systems*, 10 (5): 629–655, 1993.

[185] J. L. Meriam, L. G. Kraige, and J. N. Bolton. *Engineering mechanics: dynamics*. John Wiley & Sons, 2020.

[186] A. R. Abdulghany. Generalization of parallel axis theorem for rotational inertia. *American Journal of*

*Physics*, 85 (10): 791–795, 2017.

[187] J. H. Ginsberg. *Advanced engineering dynamics*. Cambridge University Press, 1998.

[188] F. L. Lewis, D. M. Dawson, and C. T. Abdallah. *Robot manipulator control: theory and practice*. Marcel Dekker, New York, 2nd edition, 2004.

[189] W. Khalil and O. Ibrahim. General solution for the dynamic modeling of parallel robots. *Journal of Intelligent and Robotic Systems*, 49 (1): 19–37, 2007.

[190] B. Dasgupta and P. Choudhury. A general strategy based on the Newton-Euler approach for the dynamic formulation of parallel manipulators. *Mechanism and Machine Theory*, 34 (6): 801–824, 1999.

[191] M. Carricato and C. Gosselin. On the Modeling of Leg Constraints in the Dynamic Analysis of Gough/Stewart-Type Platforms. *Journal of Computational and Nonlinear Dynamics*, 4 (1): 1–8, 2009.

[192] R. Oftadeh, M. M. Aref, and H.D. Taghirad. Explicit Dynamics Formulation of Stewart-Gough Platform: A Newton–Euler Approach. In *Proceedings of the IEEE International Conference on IROS*, pages 2773–77. IEEE, Taiwan, Oct. 2010.

[193] H. C. Lin, T. C. Lin, and K. H. Yae. On the skew-symmetric property of the Newton-Euler formulation

for open chain robot manipulators. In *Proc. of American Control Conference*, pages 2322–2326. IEEE, 1995.

[194] H.R. Harrison and T. Nettleton. *Advanced Engineering Dynamics*. Wiley, New York, 1997.

[195] R. Dvorak, F. Freistetter, and J. Kurths. *Chaos and Stability in Planetary Systems*. Springer, 2005.

[196] Oliver M. OReilly. *Intermediate Dynamics for Engineers: A Unified Treatment of Newton-Euler and Lagrangian Mechanics*. Cambridge University Press, Cambridge, 2008.

[197] H. Abdellatif and B. Heimann. Computational efficient inverse dynamics of 6-DOF fully parallel manipulators by using the Lagrangian formalism. *Mechanism and Machine Theory*, 44 (1): 192–207, 2009.

[198] D. T. Greenwood. *Advanced Dynamics*. Cambridge University Press, Cambridge, 2003.

[199] J.W. Brewer. Kronecker products and matrix calculus in system theory functions in dynamics of mechanical systems. *IEEE Transactions On Circuits and Systems*, CAS-25 (9): 772–781, Sept. 1978.

[200] V. K. Nguyen. Consistent definition of partial derivatives of matrix functions in dynamics of

mechanical systems. *Mechanism and Machine Theory*, 45 (7): 981–988, 2010.

[201] C. G. Atkeson, C. H. An, and J. M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5 (3): 101–119, 1986.

[202] H. Mayeda, K. Yoshida, and K. Osuka. Base parameters of manipulator dynamic models. *IEEE Transactions on Robotics and Automation*, 6 (3): 312–321, 1990.

[203] W. S. Lu and Q. H. Meng. Regressor formulation of robot dynamics: computation and applications. *IEEE Transactions on Robotics and Automation*, 9 (3): 323–333, 1993.

[204] G. Calafiore, M. Indri, and B. Bona. Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation. *Journal of Robotic Systems*, 18 (2): 55–68, 2001.

[205] A. Hassani, S. A. Khalilpour, A. Bataleblu, and Hamid D. Taghirad. Full dynamic model of 3-UPU translational parallel manipulator for model-based control schemes. *Robotica*, page 1¨C16, 2022.

[206] A. Hassani, M. R. Dindarloo, R. Khorrambakht, A. Bataleblu, R. Heidari, M. Motaharifar, S. F. Mohammadi, and H. D. Taghirad. On the dynamic

calibration and trajectory control of ARASH:ASiST. In *2022 8th International Conference on Control, Instrumentation and Automation (ICCIA)*, pages 1–6. IEEE Xplore, 2022.

[207] A. Hassani, A. Bataleblu, S. A. Khalilpour, and Hamid D. Taghirad. Dynamic modeling and identification of aras-diamond: A vitreoretinal eye surgery robot. *Modares Mechanical Engineering*, 21 (11), 2021.

[208] Jean Le Rond d'Alembert. *Traité de dynamique*. David l'ané, 1743.

[209] Alain Codourey. Dynamic modeling of parallel robots for computed-torque control implementation. *The International Journal of Robotics Research*, 17 (12): 1325–1336, 1998.

[210] S. El Hraiech, A. H. Chebbi, Z. Affi, and L. Romdhane. Error estimation and sensitivity analysis of the 3-UPU translational parallel robot due to design parameter uncertainties. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233 (8): 2713–2727, Aug. 2018.

[211] Z. Xiangzhou, L. Yougao, and B. Hongzan. Inverse dynamics of 3-upu parallel mechanism with pure rotation based on d'alembert principle. In *2007 International Conference on Mechatronics and Automation*, pages 2842–2847. IEEE, 2007.

[212] M. Vakil, H. Pendar, and H. Zohoor. Comments to the: “closed-form dynamic equations of the general stewart platform through the newton–euler approach” and “a newton–euler formulation for the inverse dynamics of the stewart platform manipulator”. *Mechanism and Machine Theory*, 43 (10): 1349–1351, oct 2008.

[213] Yongjie Zhao and Feng Gao. Inverse dynamics of the 6-dof out-parallel manipulator by means of the principle of virtual work. *Robotica*, 27 (2): 259–268, mar 2009.

[214] C. Alexopoulos and P. M. Griffin. Path planning for a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 22 (2): 318–322, 1992.

[215] H. Zhang, W. Lin, and A. Chen. Path planning for the mobile robot: A review. *Symmetry*, 10 (10): 450, 2018.

[216] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi. A comparative review on mobile robot path planning: Classical or meta-heuristic methods? *Annual Reviews in Control*, 50: 233–252, 2020.

[217] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni. *Path Planning and Trajectory Planning Algorithms: A General Overview*, pages 3–27. Springer International Publishing, Cham, 2015.

[218] L. F. Shampine. *Numerical Solution of Ordinary Differential Equations*. Routledge, 2018.

[219] K. Atkinson, W. Han, and D. E. Stewart. *Numerical Solution of Ordinary Differential Equations*, volume 108. John Wiley & Sons, 2011.

[220] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2016.

[221] John H Seinfeld, Leon Lapidus, and Myungkyu Hwang. Review of numerical integration techniques for stiff ordinary differential equations. *Industrial & Engineering Chemistry Fundamentals*, 9 (2): 266–275, 1970.

[222] HDD Watson and AR Gourlay. Implicit integration for CSMP III and the problem of stiffness. *Simulation*, 26 (2): 57–61, 1976.

[223] M. Gautier. Numerical calculation of the base inertial parameters of robots. *Journal of Robotic Systems*, 8 (4): 485–506, 1991.

[224] J. Wu, J. Wang, and Z. You. An overview of dynamic parameter identification of robots. *Robotics and Computer-integrated Manufacturing*, 26 (5): 414–419, 2010.

[225] Hamid D. Taghirad and P. R. Belanger. Modeling and parameter identification of harmonic drive systems.

*Journal of Dynamic Systems, Measurements, and Control, ASME Pub.*, 120 (4): 439–444, 1998.

[226] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Force Control*. Springer, 2009.

[227] M. A. Khosravi and H. D. Taghirad. Dynamic analysis and control of cable driven robots with elastic cables. *Transactions of CSME*, 35 (4): 543–577, 2011.

[228] F. Golnaraghi and B. C. Kuo. *Automatic Control Systems*. McGraw-Hill, 10th edition, 2017.

[229] K. J. Åstrom and T. Hägglund. *Advanced PID Control*. ISA, Instrumentation Systems and Automation Society, 2006.

[230] A. O'dwyer. *Handbook of PI and PID Controller Tuning Rules*. World Scientific, 2009.

[231] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback Control Theory*. Courier Corporation, 2013.

[232] C. An, C. Atkeson, J. Griffiths, and J. M. Hollerbach. Experimental evaluation of feedforward and computed torque control. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 165–168. IEEE, 1987.

[233] Hassan K. Khalil. *Nonlinear Control*. Pearson New York, 2015.

[234] J. J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice hall Englewood Cliffs, NJ, 1991.

[235] Se-Han Lee, Jae-Bok Song, Woo-Chun Choi, and Daehie Hong. Position control of a stewart platform using inverse dynamics control with approximate dynamics. *Mechatronics*, 13 (6): 605–619, 2003.

[236] Paolo Righettini, Roberto Strada, Filippo Cortinovis, Federico Tabaldi, Jasmine Santinelli, and Andrea Ginammi. An experimental investigation of the dynamic performances of a high speed 4-dof 5r parallel robot using inverse dynamics control. *Robotics*, 13 (3): 54, 2024.

[237] A. Fumagalli and P. Masarati. Real-time inverse dynamics control of parallel manipulators using general-purpose multibody software. *Multibody System Dynamics*, 22: 47–68, 2009.

[238] P. A. Ioannou and J. Sun. *Robust Adaptive Control*. Courier Corporation, 2012.

[239] S. Roy and I. N. Kar. *Adaptive-Robust Control with Limited Knowledge on Systems Dynamics*. Springer, 2020.

[240] M. T. Ravichandran and A. D. Mahindrakar. Robust stabilization of a class of underactuated mechanical systems using time scaling and lyapunov redesign.

*IEEE Transactions on Industrial Electronics*, 58 (9): 4299–4313, 2010.

[241] C. Silva and E. Trélat. Smooth regularization of bang-bang optimal control problems. *IEEE Transactions on Automatic Control*, 55 (11): 2488–2499, 2010.

[242] R. A. De Carlo, S. H. Zak, and G. P. Matthews. Variable structure control of nonlinear multivariable systems: a tutorial. *Proceedings of the IEEE*, 76 (3): 212–232, 1988.

[243] C. Edwards and S. Spurgeon. *Sliding Mode Control: Theory and Applications*. Crc Press, 1998.

[244] N. K. Sharma and J. Sivaramakrishnan. *Discrete-time Higher Order Sliding Mode: The Concept and the Control*. Springer, 2018.

[245] Y. Shtessel, C. Edwards, L. Fridman, A. Levant, et al. *Sliding Mode Control and Observation*, volume 10. Springer, 2014.

[246] N. B. Cheng, L. W. Guan, L. P. Wang, and J. Han. Chattering reduction of sliding mode control by adopting nonlinear saturation function. In *Advanced Materials Research*, volume 143, pages 53–61. Trans Tech Publ, 2011.

[247] Q. P. Ha, Q. H. Nguyen, D. C. Rye, and Hugh F Durrant-Whyte. Fuzzy sliding-mode controllers with

applications. *IEEE Transactions on Industrial Electronics*, 48 (1): 38–46, 2001.

[248] L. Fridman, A. Levant, et al. Higher order sliding modes. *Sliding Mode Control in Engineering*, 11: 53–102, 2002.

[249] V. Utkin. Discussion aspects of high-order sliding mode control. *IEEE Transactions on Automatic Control*, 61 (3): 829–833, 2015.

[250] R. Rembala and C. Ower. Robotic assembly and maintenance of future space stations based on the ISS mission operations experience. *Acta Astronautica*, 65 (7-8): 912–920, Oct.-Nov. 2009.

[251] U. Mohrlen, D. Weber, R. Gonzalez, D. M. Schmid, T. Sulser, and R. Gobet. Robot-assisted minimal invasive pediatric urology. *Journal of Pediatric Urology*, 5 (1): S45–S46, 2009.

[252] Hamid D. Taghirad and P. R. Belanger. Intelligent built-in torque sensor of harmonic drive systems. *IEEE Transactions on Instrumentation and Measurements*, 48 (6): 1201–1207, 1999.

[253] Hamid D. Taghirad and P. R. Belanger. H-infinity based robust torque control of harmonic drive systems. *Journal of Dynamic Systems, Measurements, and Control, ASME Pub.*, 123 (3): 338–345, 2001.

[254] R. Ghorbani and Q. Wu. Adjustable stiffness artificial tendons: Conceptual design and energetics study in bipedal walking robots. *Mechanism and Machine Theory*, 44 (1): 140–161, 2009.

[255] Y. T. Lee, H. R. Choi, W. K. Chung, and Y. Youm. Stiffness control of a coupled tendon-driven robot hand. *IEEE Transactions on Control Systems Technology*, 14 (5): 10–19, 1994.

[256] T. S. Chung. An inherent stability problem in cartesian compliance and an alternative structure of compliance control. *IEEE Transactions on Robotics and Automation*, 7 (1): 21–30, Feb 1991.

[257] L. Zollo, B. Siciliano, C. Laschi, G. Teti, and Paolo Dario. An experimental study on compliance control for a redundant personal robot arm. *Robotics and Autonomous Systems*, 44 (2): 101–129, 2003.

[258] R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Prentice Hall, 12th edition, 2010.

[259] Neville Hogan. Impedance control: An approach to manipulation: Part 1 theory. *Journal of Dynamics Systems, Measurements, and Control*, 107: 1–16, 1985.

[260] Neville Hogan. Impedance control: An approach to manipulation: Part 2 implementation. *Journal of*

*Dynamics Systems, Measurements, and Control*, 107: 17–24, 1985.

[261] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, MA, 3rd edition, 2005.

[262] G. Wang, Y. Wei, S. Qiao, P. Lin, and Y. Chen. *Generalized Inverses: Theory and Computations*, volume 53. Springer, 2018.

[263] A. Isidori. *Nonlinear Control Systems*. Springer Verlag, New York, 1999.

[264] P. S. Shcherbakov. Alexander mikhailovitch lyapunov: On the centenary of his doctoral dissertation on stability of motion. *Automatica*, 28 (5): 865–871, 1992.

[265] Amir Ali Ahmadi and Pablo A Parrilo. On higher order derivatives of lyapunov functions. In *Proceedings of the 2011 American Control Conference*, pages 1313–1314. IEEE, 2011.

# *Index*


A 3PRC manipulator
- Direct force control, 519
- Dynamic analysis, 261, 296, 321
- Dynamic simulation, 370
- Geometry, 35
- Impedance control, 544
- Jacobian analysis, 174
- Kinematic analysis, 143
- Lagrange, 296
- Linear control, 423
- Motion control, 472
- Newton–Euler, 261
- Robust control, 473
- Static force, 227
- Stiffness, 227
- Stiffness control, 518
- Virtual work, 321

A 3RPS manipulator
- Direct force control, 519
- Dynamic analysis, 261, 296, 321
- Dynamic simulation, 370

Geometry, 34
Impedance control, 544
Jacobian analysis, 173
Kinematic analysis, 142
Lagrange, 296
Linear control, 423
Motion control, 472
Newton–Euler, 261
Robust control, 473
Static force, 227
Stiffness, 227
Stiffness control, 518
Virtual work, 321
A 3RRR manipulator
Direct force control, 519
Dynamic analysis, 260, 295, 320
Dynamic simulation, 368
Geometry, 31
Impedance control, 544
Jacobian analysis, 171
Kinematic analysis, 134
Lagrange, 295
Linear control, 423
Motion control, 472
Newton–Euler, 260
Robust control, 473
Static force, 225
Stiffness, 225

Stiffness control, 517
Virtual work, 320
A 3UPS manipulator
Dynamic analysis, 261, 296, 321
Dynamic simulation, 370
Geometry, 34
Jacobian analysis, 173
Kinematic analysis, 140
Lagrange, 296
Linear control, 423
Motion control, 472
Newton–Euler, 261
Robust control, 473
Static force, 226
Stiffness, 226
Virtual work, 321
A 3UPU manipulator
Direct force control, 519
Dynamic analysis, 261, 296
Dynamic simulation, 369
Geometry, 32
Impedance control, 544
Jacobian analysis, 173
Kinematic analysis, 139
Lagrange, 296
Linear control, 423
Motion control, 472
Newton–Euler, 261

Robust control, 473
Static force, 226
Stiffness, 226
Stiffness control, 518
A 4RRR manipulator
Direct force control, 519
Dynamic analysis, 260, 296, 320
Dynamic simulation, 369
Geometry, 32
Impedance control, 544
Jacobian analysis, 172
Kinematic analysis, 137
Lagrange, 296
Linear control, 423
Motion control, 472
Newton–Euler, 260
Robust control, 473
Static force, 225
Stiffness, 225
Stiffness control, 517
Virtual work, 320
Acceleration analysis
Planar manipulator, 241
Stewart–Gough platform, 247
Active set method, 219
Adaptive control, 432
Adaptive IDC
Lyapunov stability, 444

Planar manipulator, 456, 457
Task space, 441
Adaptive inverse dynamics control, 441, 446
Adaptive law, 444
Passivity-based, 444
Adaptive law, 444
Analytic forward kinematics, 121
Angular acceleration, 91
Angular momentum, 237
Angular velocity, 81
Absolute derivative, 85
Quaternion, 83
Rate of Euler angles, 83
Relative derivative, 85
Rotation matrix, 82
Screw representation, 81

Cable driven robot, 24
Cable robot
Dynamics simulation, 344
Forward dynamics simulation, 344
Forward kinematics, 106
Geometry, 105
Kinematic analysis, 104
Kinematics simulation, 108
Kinematics verification, 108
Mechanism description, 104
Static force analysis, 181
Cable robots

Constant mass dynamics, 240
Dynamic analysis, 240
FAST, 26
LAR, 25
Redundancy resolution, 27
RoboCrane, 26
WARP, 26
Calibration
Angular acceleration, 365
Base inertial parameters (BIP), 364
Consistency measure, 367
Least squares, 366
Model evaluation, 367
Orthogonal decomposition, 367
Capacitive impedance, 523
Cascade control, 477
Force control, 478
Impedance control, 481
Cauchy-Schwartz inequality, 550
Center of mass, 234
Chattering, 439
UUB stability, 441
Chebychev–Grübler–Kutzbach criterion, 12
Christoffel matrix, 273, 274
Closed–form dynamics
Interatcing force, 476
Stewart–Gough platform, 255, 258, 259
Closed–form Lagrange formulation, 276

Closed–loop formulation
Direct force control, 507
Impedance control, 529
closed-loop dynamics simulation
Stewart–Gough platform, 355
Combined singularity, 155
Compliance matrix, 198, 199
Consistency measure, 367
Control topology, 377, 476
Joint space, 380
Task space, 378, 379
Coriolis and centrifugal
Matrix, 292
Vector, 272
Cosine directions, 42
Cubic polynomials, 324
Cubic polynomials with via points, 326
Cubic trajectory planning, 325

D’Alembert principle, 298
Damping ratio, 386
Degrees of freedom, 11, 12
Delta robot, 22
Adept Quattro, 23
Direct force control, 519
Dynamic analysis, 261, 296, 321
Dynamic simulation, 370
Falcon, 23
FlexPicker, 22

Geometry, 36, 145
Impedance control, 544
Jacobian analysis, 174
Kinematic analysis, 145
Lagrange, 296
Linear control, 423
Motion control, 473
Newton–Euler, 261
Robust control, 473
Static force, 228
Stiffness, 228
Stiffness control, 518
Virtual work, 321
Derivative filter, 365
Dexterity ellipsoid, 187
Dexterity matrix, 189
Dexterity measure, 191
Direct force control, 503, 505
Closed–loop formulation, 507
Planar manipulator, 508, 511
Stewart–Gough platform, 511, 516
Disturbance
Planar manipulator, 400, 405
Stewart-Gough platform, 414, 418
Disturbance rejection, 387
DOF, 11, 12
Piston-crank mechanism, 13
Planar 4RPR mechanism, 14

Stewart–Gough platform, 15
Dynamic analysis, 231, 263, 297
Planar cable robot, 278
Planar manipulator, 240, 278, 305
Stewart–Gough platform, 246, 287, 308
Dynamic calibration, 364
Dynamic formulation
Closed–form, 246
Linearity in parameters, 277
Mass matrix properties, 272
Properties, 270
Skew symmetricity, 273
Dynamical systems, 558
Dynamics simulation
Cable robot, 344
Implicit integration, 347
Planar manipulator, 344, 350
Runge–Kutta, 347
Stewart–Gough platform, 352
Dynamics verification, 307

Eigenvalue, 552
Eigenvector, 552
Electrical impedance, 522
Environment model, 492, 498
Equilibrium point, 559
Global stability, 559
Euler law, 239
Euler-Lagrange formulation, 265

Euler-Lagrange Method
Skew-Symmetricity, 276
Euler-Lagrange method
Christoffel matrix, 273, 274

Feed-forward control, 425, 445
Planar manipulator, 449
Stewart–Gough platform, 461
Feed-forward control, 391
Five bar 4R1P mechanism
Direct force control, 519
Dynamic analysis, 260, 295, 320
Dynamic simulation, 368
Geometry, 31
Impedance control, 544
Jacobian analysis, 171
Kinematic analysis, 134
Lagrange, 295
Linear control, 422
Motion control, 472
Newton–Euler, 260
Robust control, 473
Static force, 224
Stiffness, 224
Stiffness control, 517
Virtual work, 320
Five bar mechanism
Direct force control, 518
Dynamic analysis, 259, 295, 320

Dynamic simulation, 368
Geometry, 31
Impedance control, 543
Jacobian analysis, 170
Kinematic analysis, 134
Lagrange, 295
Linear control, 422
Motion control, 471
Newton–Euler, 259
Robust control, 473
Static force, 224
Stiffness, 224
Stiffness control, 517
Virtual work, 320
Flexible link, 7
Flight simulator, 21
Force control, 474, 480
Cascade control, 477, 478
Closed–form formulation, 476
Control topology, 476
Joint space, 481
Force distribution, 348
Forward dynamics simulation, 340, 352, 360, 364
Stewart–Gough platform, 352
Forward kinematics, 102
6-6$^p$ Stewart–Gough platform, 121
Numerical solution, 108
Shoulder manipulator, 114

Stewart–Gough platform, 119
Forward kinematics singularity, 154
Free body diagram
Dynamic analysis, 240
Static force analysis, 179
Fully parallel manipulator, 15

General Stewart–Gough platform, 126
Numerical forward kinematics, 129
Generalized coordinates, 265
Generalized force
Planar manipulator, 287
Stewart–Gough platform, 295
Generalized forces, 265
Global measures, 195
Gough platform, 21
Grübler criterion, 12
Gravity vector
Potential energy, 272
Stewart–Gough platform, 154, 290

Hexapod, 21
Holonomic, 265
Homogeneous coordinates
Free vectors, 59
Line vectors, 59
Homogeneous transformation, 58
Arithmetic, 61
Inverse, 64

Homogeneous transformation matrix, 59
Hybrid robot, 16, 23

IDC
    Task space, 427
Impedance, 521
    Capacitive, 523, 524
    Definition, 521
    Desired, 527
    Electrical, 522
    Inductive, 523, 524
    Mechanical, 523
    Resistive, 523, 524
Impedance control, 474, 481, 520, 528
    Cascade control, 481
    Case study, 525
    Closed–loop formulation, 529
    Compliance control, 529
    Concept, 524
    Conceptual idea, 524
    Control topology, 476
    Desired dynamics, 525
    Desired impedance, 527
    Example, 524
    Formulation, 528
    Joint space, 483
    Planar manipulator, 530, 534
    Special case, 529
    Stewart–Gough platform, 535, 539

Structure, 527, 528
Implicit formulation, 255
Induced Norm, 552
Inductive impedance, 523
Inertia matrix, 235
Parallel axis theorem, 236
Principal axes, 235
Rotation, 236
Transformation, 236
Interior point method, 220
Intrinsic features of Jacobian matrix, 187
Invarient set, 563
Inverse Dynamics Control
Model Uncertainty, 433
Inverse dynamics control, 445
Planar manipulator, 452
Stewart–Gough platform, 465
Task space, 427
Inverse dynamics simulation, 342
Planar manipulator, 350
Stewart–Gough platform, 358
Inverse kinematics, 102
Kinematic loop, 103
Loop closure, 102
Shoulder manipulator, 113
Stewart–Gough platform, 119
Inverse kinematics singularity, 154
Inverse problems, 209

Forces, 210
Velocities, 209
Isotropy to singularity, 189

Jacobian analysis
Planar manipulator, 155
Shoulder manipulator, 160
Stewart–Gough platform, 165
Jacobian Matrix
Global measures, 195
Jacobian matrix
Definition, 152
Dexterity ellipsoid, 187
Dexterity measure, 191
Force mapping, 181
Intrinsic features, 187
Inverse problems, 209
Isotropy, 189
Manipulability measure, 193
Planar manipulator, 158
Shoulder manipulator, 162, 163
Singularity, 189
Singularity analysis, 153
Velocity loop closure, 152
Joint space
Force control, 481
Impedance control, 483

Karush-Kuhn-Tucker theorem, 218

Kinematic analysis
3–3 Stewart–Gough platform, 121
6–3 Stewart–Gough platform, 121
Cable robot, 104
Planar manipulator, 104
Shoulder manipulator, 110
Stewart–Gough platform, 116
Kinematic chain, 10
Kinematics simulation
Cable robot, 108
Planar manipulator, 108
Shoulder manipulator, 115
Stewart–Gough platform, 127
Kinetic energy, 237, 238, 271
KKT theorem, 218
Klaus Cappel, 21
Kronecker product, 273, 556

Lagrange formulation, 264
Centrifugal matrix, 275
Centrifugal vector, 272
Closed-form, 276
Coriolis and centrifugal matrix, 292
Coriolis matrix, 275
Coriolis vector, 272
First kind, 269, 270
Generalized coordinates, 265
Generalized force, 287, 295
Generalized forces, 265

Gravity vector, 272, 290
Lagrangian, 269
Limb dynamics, 280
Mass matrix, 271, 289
Matrix form, 271
Moving platform dynamics, 284
Naive approach, 279
Planar manipulator, 285
Second kind, 267, 269, 271
Stewart–Gough platform, 287
Systematic approach, 280
Lagrange multiplier, 216
Lagrangian, 269
Large adaptive reflector (LAR), 25
Left pseudo-inverse, 554
Limb closed–form dynamics, 255
Stewart–Gough platform, 257
Limb dynamics
Planar manipulator, 242, 280
Stewart–Gough platform, 250, 287
Linear algebra, 549
Linear control
PID controller, 388
Damping ratio, 386
Disturbance rejection, 387
Error integral, 387
Feed-forward control, 391
Measurement noise, 390

Natural frequency, 386
PD control, 384
PID control, 388
Simulation, 395
Steady-state error, 386
Time response, 386
Linear inverse problem, 554
Linear Momentum, 237
Linear velocity, 85
Linear-parabolic trajectory, 332
Linearity in parameters, 277
Link, 7, 10
Loop closure, 102
Lyaponov stability
Definition, 559
Lyapunov stability, 560
Adaptive IDC, 444
Concept, 560
Global theorem, 562
Krasovskii–Lasalle theorem, 563
Krasovskii-lasalle, 564
Local theorem, 561
Robust IDC, 437
UUB, 563

Machine, 10
Magic of feedback, 382
Manipulability measure, 193
Mass matrix

Kinetic energy, 271
Stewart–Gough platform, 289
Mass matrix properties, 272
Mass properties, 233
Matrix, 549
Matrix inversion lemma, 551
Mechanical impedance, 522, 523
Mechanism, 10
Model Uncertainty, 433
Moments of inertia, 235
Motion control
Adaptive control, 432
Adaptive IDC, 441, 446
Control topology, 377
Dynamic formulation, 445
Feed-forward control, 425, 445
IDC, 427, 445
Partial IDC, 429, 446
PD control, 392, 445
Planar manipulator, 447
Redundancy resolution, 211
Robust control, 432
Robust IDC, 434, 446
Stewart–Gough platform, 408, 459
Summary, 445
Task space, 379
Motion planning
Task space, 324

Moving platform dynamics
Planar manipulator, 245, 284
Stewart–Gough platform, 254, 293
Multivariable control, 424
Multivariable PID control, 394

Natural frequency, 386
Newton law, 238
Newton–Euler formulation, 240
Stewart–Gough platform, 246
Newton–Euler laws, 238
Non-holonomic, 265
Nonlinear control, 424, 558
Null Space, 552
Numerical forward kinematics
Stewart–Gough platform, 129
Numerical simulation, 341

Octahedral hexapod, 21
Optimization
Active set method, 219
Equality constraint, 215
Full constraints, 217
Interior point method, 220
Iterative analytic, 222
Iterative solutions, 218
KKT theorem, 218
Lagrange multiplier, 216
Regular point, 215

Stationary point, 215
Trust region method, 220
Orientation
Definition, 41
Exponential, 95
Quaternion, 75, 95
Rodrigues' formula, 95
Rodrigues' rotation formula, 73
Screw axis, 72
Orthogonal matrix, 551

Pantograph mechanism
Direct force control, 519
Dynamic analysis, 260, 295, 320
Dynamic simulation, 368
Geometry, 31
Impedance control, 544
Jacobian analysis, 171
Kinematic analysis, 134
Lagrange, 295
Linear control, 422
Motion control, 472
Newton–Euler, 260
Robust control, 473
Static force, 224
Stiffness, 224
Stiffness control, 517
Virtual work, 320
Parallel axis theorem, 236

Parallel robot, 16, 20
Partial IDC
    Planar manipulator, 452
    Stewart–Gough platform, 467
    Task space, 429
Partial inverse dynamics control, 446
Passivity-based, 444
PD control, 384, 445
    Planar manipulator, 395
    Stewart–Gough platform, 408, 414
    Task space, 392
PID control, 388
    Planar manipulator, 402, 405, 415, 448
    Stewart–Gough platform, 459
PID controller, 388
Piston-crank mechanism, 13
Planar 4RPR mechanism, 14
Planar cable robot
    Dynamic analysis, 278
    Lagrangian formulation, 278
    Stiffness control, 492
    Virtual work formulation, 305
Planar manipulator
    Acceleration analysis, 241
    Adaptive IDC, 456, 457
    Closed–loop simulation, 350
    Direct force control, 508, 511
    Disturbancel, 400, 405

Dynamic analysis, 240, 278, 305
Dynamics simulation, 344, 350
Dynamics verification, 307
Feed-forward control, 449
Force distribution, 348
Forward dynamics simulation, 344
Forward kinematics, 106
Generalized force, 287
Geometry, 105
IDC, 452
Impedance control, 530, 534
Inverse dynamics control, 452
Inverse dynamics simulation, 350
Inverse kinematics, 106
Jacobian analysis, 155
Jacobian matrix, 158
Kinematic analysis, 104
Kinematics simulation, 108
Kinematics verification, 108
Lagrange formulation, 285
Lagrangian formulation, 278
Limb dynamics, 242, 280
Mechanism description, 104
Motion control, 447
Moving platform dynamics, 245, 284
Newton–Euler formulation, 240
Partial IDC, 452
PD control, 395

PID control, 402, 405, 415, 448
Robust IDC, 454
Sensitivity analysis, 159
Singularity analysis, 159
Static force analysis, 181
Stiffness control, 495
Velocity loop closure, 156
Virtual work verification, 307
Planer cable robots
Dynamic analysis, 305
Potential Energy, 238
Potential energy, 272
Principle of virtual work, 299
Pseudo-inverse, 553

Quadrupteron robot
Direct force control, 519
Dynamic analysis, 262, 296, 321
Dynamic simulation, 371
Geometry, 37
Impedance control, 545
Jacobian analysis, 174
Kinematic analysis, 146
Lagrange, 296
Linear control, 423
Motion control, 473
Newton–Euler, 262
Robust control, 473
Static force, 228

Stiffness, 228
Stiffness control, 518
Virtual work, 321
Quaternion, 75, 95
Quaternion rates, 83
Quintic polynomials, 329
Quintic Trajectory Planning, 330

Radio telescope
FAST, 26
Radio telescopes, 25
Redundancy resolution, 15, 211
Active set method, 219
Base solution, 212
Cable robot formulation, 213
Constrained optimization, 215
Full constrained optimization, 217
General formulation, 214
General solution, 213
Interior point method, 220
Iterative analytic, 222
Iterative solutions, 218
KKT multipliers, 215
KKT theorem, 218
Lagrange multipliers, 215, 216
Main formulation, 212
Null space, 213
Problem formulation, 212
Regular point, 215

Stationary point, 215
Trust region method, 220
Redundant manipulator, 15
Regular point, 215
Resistive impedance, 523
Right pseudo-inverse, 554
Rigid body
Angular acceleration, 91
Dynamics, 233
Rigid body dynamics, 233
Rigid link, 7
Robot
Components, 5
History, 5
Robot compliance
Flexible elements, 196
Robot control
Multivariable, 424
Nonlinear, 424
Robot motion
Simulation, 339
Robot motion simulation, 339
Robust control, 432
Robust IDC
Lyapunov stability, 437
Planar manipulator, 454
Stewart–Gough platform, 469
Task space, 434

UUB stability, 441
Variable structure control, 439
Robust Inverse Dynamics Control
Asymptotic Stability, 437
Chattering, 439
Robust inverse dynamics control, 434, 446
Rodrigues' formula, 95
Rodrigues' rotation formula, 73
Rotation matrix, 41
Definition, 41
Determinant, 46
Eigenvalues, 46
Inverse, 44
Mapping, 44
Ortonormal, 43
Transpose, 44

Screw axis, 72
Screw coordinates, 88
General motion, 90
Pitch, 88
Translational motion, 89
Second order system time response, 386
Sensitivity analysis, 159
Sensitivity transfer function, 383
Serial robot, 16
Characteristics, 18
Shoulder manipulator
Forward kinematics, 114

Geometry, 111
Inverse kinematics, 113
Jacobian analysis, 160
Jacobian matrix, 162, 163
Kinematic analysis, 110
Kinematics simulations, 115
Mechanism description, 111
Singularity analysis, 163
Static force analysis, 183
Stiffness, 203
Velocity loop closure, 161
Simulation
Inverse dynamics, 342
Numerical, 341
Simulation of forward dynamics, 340
Simulation of linear control, 395
Singular configurations, 150
Singular value, 552
Singular value decomposition, 553
Singularity
Combined, 155
Forward kinematics, 154
Inverse kinematics, 154
Singularity analysis, 153
3–6 Stewart–Gough platform, 169
Background literature, 167
Planar manipulator, 159
Shoulder manipulator, 163

Stewart–Gough platform, 167
Singularity to isotropy, 189
Skew symmetricity property, 273
Skew-Symmetric Property, 276
Skycam, 24
Spatial orientation manipulator
Dynamic analysis, 261, 296, 321
Dynamic simulation, 369
Geometry, 32
Jacobian analysis, 172
Kinematic analysis, 138
Lagrange, 296
Linear control, 423
Motion control, 472
Newton–Euler, 261
Robust control, 473
Static force, 225
Stiffness, 225
Virtual work, 321
Spidercam, 24
Static force analysis, 178
Cable robot, 181
Free body diagram, 179
Planar manipulator, 181
Shoulder manipulator, 183
Stewart–Gough platform, 185
Virtual work, 179
Stationary point, 215

Steady-state rrror, 386
Stewart–Gough platform, 21
6-6$^p$, 121
Acceleration analysis, 247
Analytic forward kinematics, 121
Closed–form dynamics, 255, 258, 259
Closed–form formulation, 255
closed-loop dynamics simulation, 355
Degrees of freedom, 15
Direct force control, 511, 516
Dynamic analysis, 246, 287, 308
Dynamics simulation, 352
Feed-forward control, 461
Forward dynamics simulation, 352, 364
Forward kinematics, 119
Generalized force, 295
Geometry, 118
IDC, 465
Impedance control, 535, 539
Implicit dynamics, 255
Inverse dynamics control, 465
Inverse dynamics simulation, 358
Inverse kinematics, 119
Jacobian analysis, 165
Kinematic analysis, 116
Kinematics literature, 119
Kinematics simulation, 127
Lagrange formulation, 287

Limb closed–form dynamics, 255, 257
Limb dynamics, 250, 287
Mechanism description, 117
Motion control, 408, 459
Moving platform closed–form dynamics, 258
Moving platform dynamics, 254, 293
Newton–Euler formulation, 246
Numerical forward Kinematics, 126
Numerical forwardkKinematics, 129
Partial IDC, 467
PD control, 408, 414
PID control, 459
Robust IDC, 469
Singularity analysis, 167
Static force analysis, 185
Stiffness, 206
Stiffness control, 495
Virtual work formulation, 308, 311
Stewart-Gough platform
Disturbance, 418
Disturbancel, 414
Stiffness analysis
Parallel manipulator, 196
Shoulder manipulator, 203
Stewart–Gough platform, 206
Stiffness control, 474, 484
Environment model, 492, 498
Force–motion relation, 491

Lyapunov analysis, 490
Multiple DOF, 488
Parallel manipulator, 484
Planar cable robot, 492
Planar manipulator, 495
Single DOF, 485
Stewart–Gough platform, 495
Stiffness matrix, 198
Symmetrical parallel manipulator, 100

Time response, 386
Trajectory Planning
Linear-Parabolic, 335
Optimal Time, 337
Quintic Polynomials, 330
Via Points, 328
Trajectory planning, 323
Cubic ploynomials, 324
Cubic polynimial, 325
Joint space, 339
Linear-parabolic, 332
Maximum speed, 334
Optimal time, 334
Quintic polynomials, 329
Teaching pendant, 339
Via points, 326
Trajectory planning in joint space, 339
Trajectory planning in task space, 324
Trajectory Planning with Via Points, 328

Triangular inequality, 550
Trivial trajectory, 559
Trust region method, 220
Twist, 90

Uncertainty, 433
    Structured, 434
    Unstructured, 434
Unit quaternion, 75, 95
UUB stability, 441, 559

Variable structure control, 439
Vector, 549
    Linear independence, 551
    Skew-symmetric matrix, 551
Velocity loop closure, 152
    Planar manipulator, 156
    Shoulder manipulator, 161
Virtual work, 298, 302
    Static force analysis, 179
    Stewart–Gough platform, 308, 311
Virtual work principle, 179, 299