MARK W. SPONG | SETH HUTCHINSON
M. VIDYASAGAR

ROBOT MODELING AND CONTROL

SECOND EDITION

WILEY

Robot Modeling and Control

Robot Modeling and Control

Second Edition

Mark W. Spong
Seth Hutchinson
M. Vidyasagar

WILEY

This edition first published 2020

Edition History
John Wiley & Sons, Ltd (1e, 2006)

Registered Office(s)
John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Office
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Library of Congress Cataloging-in-Publication Data
Names: Spong, Mark W, author. | Hutchinson, Seth, author. | Vidyasagar, M. (Mathukumalli), 1947- author.
Title: Robot modeling and control / Mark W Spong, Seth Hutchinson, and M. Vidyasagar.
Description: Second edition. | Hoboken, NJ : John Wiley & Sons, Inc., 2020. | Includes bibliographical references and index.
Identifiers: LCCN 2019055413 (print) | LCCN 2019055414 (ebook) | ISBN 9781119523994 (hardback) | ISBN 9781119524076 (adobe pdf) | ISBN 9781119524045 (epub)
Subjects: LCSH: Robots–Control systems. | Robots–Dynamics. | Robotics.
Classification: LCC TJ211.35 .S75 2020 (print) | LCC TJ211.35 (ebook) | DDC 629.8/92–dc23
LC record available at https://lccn.loc.gov/2019055413
LC ebook record available at https://lccn.loc.gov/2019055414

Cover image: © Patrick Palej/EyeEm/Getty Images
Cover design by Wiley

Set in 11/13.5pt Computer Modern Roman by Aptara Inc., New Delhi, India

Printed by CPI Antony Rowe Ltd

10 9 8 7 6 5 4 3 2 1

Preface

This text is a second edition of our book, *Robot Modeling and Control*, John Wiley & Sons, Inc., 2006, which grew out of the earlier text, M.W. Spong and M. Vidyasagar, *Robot Dynamics and Control*, John Wiley & Sons, Inc., 1989. The second edition reflects some of the changes that have occurred in robotics and robotics education in the past decade. In particular, many courses are now treating mobile robots on an equal footing with robot manipulators. As a result, we have expanded the discussion on mobile robots into a full chapter. In addition, we have added a new chapter on underactuated robots. We have also revised the material on vision, vision-based control, and motion planning to reflect changes in those topics.

Organization of the Text

After the introductory first chapter, which introduces the terminology and history of robotics and discusses the most common robot design and applications, the text is organized into four parts. Part I consists of four chapters dealing with the geometry of rigid motions and the kinematics of manipulators.

Chapter 2 presents the mathematics of rigid motions; rotations, translations, and homogeneous transformations.

Chapter 3 presents solutions to the forward kinematics problem using the Denavit–Hartenberg representation, which gives a very straightforward and systematic way to describe the forward kinematics of manipulators.

Chapter 4 discuses velocity kinematics and the manipulator Jacobian. The geometric Jacobian is derived in the cross product form. We also introduce the so-called analytical Jacobian for later use in task space control. We have reversed the order of our treatment of velocity kinematics and inverse kinematics from the presentation in the first edition in order to include a new section in Chapter 5 on numerical inverse kinematics algorithms, which rely on the Jacobian for their implementation.

Chapter 5 deals with the inverse kinematics problem using the geometric

approach, which is especially suited for manipulators with spherical wrists. We show how to solve the inverse kinematics in closed form for the most common manipulator designs. We also discuss numerical search algorithms for solving inverse kinematics. Numerical algorithms are increasingly popular because of both the increasing power of computers and the availability of open-source software for numerical algorithms.

Part II deals with dynamics and motion planning and consists of two chapters.

Chapter 6 is a detailed account of robot dynamics. The Euler–Lagrange equations are derived from first principles and their structural properties are discussed in detail. The recursive Newton–Euler formulation of robot dynamics is also presented.

Chapter 7 is an introduction to the problems of path and trajectory planning. Several of the most popular methods for motion planning and obstacle avoidance are presented, including the method of artificial potential fields, randomized algorithms, and probabilistic roadmap methods. The problem of trajectory generation is presented as essentially a problem of polynomial spline interpolation. Trajectory generation based on cubic and quintic polynomials as well as trapezoidal velocity trajectories are derived for interpolation in joint space.

Part III deals with the control of manipulators.

Chapter 8 is an introduction to independent joint control. Linear models and linear control methods based on PD, PID, and state space methods are presented for set-point regulation, trajectory tracking, and disturbance rejection. The concept of feedforward control, including the method of computed torque control, is introduced as a method for nonlinear disturbance rejection and for tracking of time-varying reference trajectories.

Chapter 9 discusses nonlinear and multivariable control. This chapter summarizes much of the research in robot control that took place in the late 1980s and early 1990s. Simple derivations of the most common robust and adaptive control algorithms are presented that prepare the reader for the extensive literature in robot control.

Chapter 10 treats the force control problem. Both impedance control and hybrid control are discussed. We also present the lesser known hybrid impedance control method, which allows one to control impedance and regulate motion and force at the same time. To our knowledge this is the first textbook that discusses the hybrid impedance control approach to robot force control.

Chapter 11 is an introduction to visual servo control, which is the problem of controlling robots using feedback from cameras mounted either on

the robot or in the workspace. We present those aspects of vision that are most useful for vision-based control applications, such as imaging geometry and feature extraction. We then develop the differential kinematics that relate camera motion to changes in extracted features and we discuss the main concepts in visual servo control.

Chapter 12 is a tutorial overview of geometric nonlinear control and the method of feedback linearization of nonlinear systems. Feedback linearization generalizes the methods of computed torque and inverse dynamics control that are covered in Chapters 8 and 9. We derive and prove the necessary and sufficient conditions for local feedback linearization of single-input/single-output nonlinear systems, which we then apply to the flexible joint control problem. We also introduce the notion of nonlinear observers with output injection.

Part IV is a completely new addition to the second edition and treats the control problems for underactuated robots and nonholonomic systems.

Chapter 13 deals with underactuated serial-link robots. Underactuation arises in applications such as bipedal locomotion and gymnastic robots. In fact, the flexible-joint robot models presented in Chapters 8 and 12 are also examples of underactuated robots. We present the ideas of partial feedback linearization and transformation to normal forms, which are useful for controller design. We also discuss energy and passivity methods to control this class of systems.

Chapter 14 deals primarily with wheeled mobile robots, which are examples of systems subject to nonholonomic constraints. Many of the control design methods presented in the chapters leading up to Chapter 14 do not apply to nonholonomic systems. Thus, we cover some new techniques applicable to these systems. We present two fundamental results, namely Chow's theorem and Brockett's theorem, that provide conditions for controllability and stabilizability, respectively, of mobile robots.

Finally, the appendices have been expanded to give much of the necessary background mathematics to be able to follow the development of the concepts in the text.

A Note to the Instructor

This text is suitable for several quarter-long or semester-long courses in robotics, either as a two- or three- course sequence or as stand-alone courses. The first five chapters can be used for a junior/senior-level introduction to robotics for students with at least a minimal background in linear algebra. Chapter 8 may also be included in an introductory course for students with some exposure to linear control systems. The independent joint control

problem largely involves the control of actuator and drive-train dynamics; hence most of the subject can be taught without prior knowledge of Euler–Lagrange dynamics.

A graduate-level course on robot dynamics and control can be taught using all or parts of Chapters 6 through 12.

Finally, one or more special topics courses can be taught using Chapters 9 through 14. Below we outline several possible courses that can be taught from this book:

Course 1: Introduction to Robotics
Level: Junior/Senior undergraduate
For a one quarter course (10 weeks):

Chapter 1: Introduction
Chapter 2: Rigid Motions and Homogeneous Transformations
Chapter 3: Forward Kinematics
Chapter 4: Velocity Kinematics and Jacobians
Chapter 5: Inverse Kinematics

For a one semester course (16 weeks) add:

Chapter 7: Motion Planning and Trajectory Generation
Chapter 8: Independent Joint Control

Course 2: Robot Dynamics and Control
Level: Senior undergraduate/graduate
For a one quarter course (10 weeks):

Chapters 1–5: Rapid Review of Kinematics (selected sections)
Chapter 6: Dynamics
Chapter 7: Path and Trajectory Planning
Chapter 9: Nonlinear and Multivariable Control
Chapter 10: Force Control

For a one semester course (16 weeks) add:

Chapter 11: Vision-Based Control
Chapter 12: Feedback Linearization

Course 3: Advanced Topics in Robot Control
Level: Graduate
For a one semester course (16 weeks):

Chapter 6: Dynamics
Chapter 7: Motion Planning and Trajectory Generation
Chapter 9: Nonlinear and Multivariable Control
Chapter 11: Vision-Based Control
Chapter 12: Feedback Linearization

Chapter 13: Underactuated Robots
Chapter 14: Mobile Robots

The instructor may wish to supplement the material in any of these courses with additional material to delve deeper into a particular topic. Also, either of the last two chapters can be covered in Course 2 by eliminating the Force Control chapter or the Vision-Based Control chapter.

Acknowledgements

We would like to offer a special thanks to Nick Gans, Peter Hokayem, Benjamin Sapp, and Daniel Herring, who did an outstanding job of producing most of the figures in the first edition, and to Andrew Messing for figure contributions to the current edition. We would like to thank Francois Chaumette for discussions regarding the formulation of the interaction matrix in Chapter 11 and to Martin Corless for discussion on the robust control problem in Chapter 9. We are indebted to several reviewers for their very detailed and thoughtful reviews, especially Brad Bishop, Jessy Grizzle, Kevin Lynch, Matt Mason, Eric Westervelt. We would like to thank our students, Nikhil Chopra, Chris Graesser, James Davidson, Nick Gans, Jon Holm, Silvia Mastellone, Adrian Lee, Oscar Martinez, Erick Rodriguez, and Kunal Srivastava, for constructive feedback on the first edition.

We would like to acknowledge Lila Spong for proofreading the manuscript of the second edition, and also the many people who sent us lists of typographical errors and corrections to the first edition, especially Katherine Kuchenbecker and her students, who provided numerous corrections.

Mark W. Spong
Seth Hutchinson
M. Vidyasagar

To my wife Lila – MWS
To my wife Wendy – SH
To my grandson Niyuddh Anand – MV

Contents

PREFACE **v**

CONTENTS **xi**

1 INTRODUCTION **1**

1.1 Mathematical Modeling of Robots 5
1.1.1 Symbolic Representation of Robot Manipulators . . . 5
1.1.2 The Configuration Space 5
1.1.3 The State Space 6
1.1.4 The Workspace 7
1.2 Robots as Mechanical Devices 7
1.2.1 Classification of Robotic Manipulators 8
1.2.2 Robotic Systems 10
1.2.3 Accuracy and Repeatability 10
1.2.4 Wrists and End Effectors 12
1.3 Common Kinematic Arrangements 13
1.3.1 Articulated Manipulator (RRR) 13
1.3.2 Spherical Manipulator (RRP) 14
1.3.3 SCARA Manipulator (RRP) 14
1.3.4 Cylindrical Manipulator (RPP) 15
1.3.5 Cartesian Manipulator (PPP) 15
1.3.6 Parallel Manipulator 18
1.4 Outline of the Text 18
1.4.1 Manipulator Arms 18
1.4.2 Underactuated and Mobile Robots 27
Problems 27
Notes and References 29

I THE GEOMETRY OF ROBOTS 33

2 RIGID MOTIONS 35
2.1 Representing Positions . . . 36
2.2 Representing Rotations . . . 38
2.2.1 Rotation in the Plane . . . 38
2.2.2 Rotations in Three Dimensions . . . 41
2.3 Rotational Transformations . . . 44
2.4 Composition of Rotations . . . 48
2.4.1 Rotation with Respect to the Current Frame . . . 48
2.4.2 Rotation with Respect to the Fixed Frame . . . 50
2.4.3 Rules for Composition of Rotations . . . 51
2.5 Parameterizations of Rotations . . . 52
2.5.1 Euler Angles . . . 53
2.5.2 Roll, Pitch, Yaw Angles . . . 55
2.5.3 Axis-Angle Representation . . . 57
2.5.4 Exponential Coordinates . . . 59
2.6 Rigid Motions . . . 61
2.6.1 Homogeneous Transformations . . . 62
2.6.2 Exponential Coordinates for General Rigid Motions . 65
2.7 Chapter Summary . . . 65
Problems . . . 67
Notes and References . . . 73

3 FORWARD KINEMATICS 75
3.1 Kinematic Chains . . . 75
3.2 The Denavit–Hartenberg Convention . . . 78
3.2.1 Existence and Uniqueness . . . 80
3.2.2 Assigning the Coordinate Frames . . . 83
3.3 Examples . . . 87
3.3.1 Planar Elbow Manipulator . . . 87
3.3.2 Three-Link Cylindrical Robot . . . 89
3.3.3 The Spherical Wrist . . . 90
3.3.4 Cylindrical Manipulator with Spherical Wrist . . . 91
3.3.5 Stanford Manipulator . . . 93
3.3.6 SCARA Manipulator . . . 95
3.4 Chapter Summary . . . 96
Problems . . . 96
Notes and References . . . 99

4 VELOCITY KINEMATICS 101
4.1 Angular Velocity: The Fixed Axis Case . . . 102
4.2 Skew-Symmetric Matrices . . . 103
4.2.1 Properties of Skew-Symmetric Matrices . . . 104
4.2.2 The Derivative of a Rotation Matrix . . . 105
4.3 Angular Velocity: The General Case . . . 107
4.4 Addition of Angular Velocities . . . 108
4.5 Linear Velocity of a Point Attached to a Moving Frame . . . 110
4.6 Derivation of the Jacobian . . . 111
4.6.1 Angular Velocity . . . 112
4.6.2 Linear Velocity . . . 113
4.6.3 Combining the Linear and Angular Velocity Jacobians 115
4.7 The Tool Velocity . . . 119
4.8 The Analytical Jacobian . . . 121
4.9 Singularities . . . 122
4.9.1 Decoupling of Singularities . . . 123
4.9.2 Wrist Singularities . . . 125
4.9.3 Arm Singularities . . . 125
4.10 Static Force/Torque Relationships . . . 129
4.11 Inverse Velocity and Acceleration . . . 131
4.12 Manipulability . . . 133
4.13 Chapter Summary . . . 136
Problems . . . 138
Notes and References . . . 140

5 INVERSE KINEMATICS 141
5.1 The General Inverse Kinematics Problem . . . 141
5.2 Kinematic Decoupling . . . 143
5.3 Inverse Position: A Geometric Approach . . . 145
5.3.1 Spherical Configuration . . . 146
5.3.2 Articulated Configuration . . . 148
5.4 Inverse Orientation . . . 151
5.5 Numerical Inverse Kinematics . . . 156
5.6 Chapter Summary . . . 158
Problems . . . 160
Notes and References . . . 162

II DYNAMICS AND MOTION PLANNING 163

6 DYNAMICS 165
6.1 The Euler–Lagrange Equations . . . 166
6.1.1 Motivation . . . 166
6.1.2 Holonomic Constraints and Virtual Work . . . 170
6.1.3 D'Alembert's Principle . . . 174
6.2 Kinetic and Potential Energy . . . 177
6.2.1 The Inertia Tensor . . . 178
6.2.2 Kinetic Energy for an n-Link Robot . . . 180
6.2.3 Potential Energy for an n-Link Robot . . . 181
6.3 Equations of Motion . . . 181
6.4 Some Common Configurations . . . 184
6.5 Properties of Robot Dynamic Equations . . . 194
6.5.1 Skew Symmetry and Passivity . . . 194
6.5.2 Bounds on the Inertia Matrix . . . 196
6.5.3 Linearity in the Parameters . . . 196
6.6 Newton–Euler Formulation . . . 198
6.6.1 Planar Elbow Manipulator Revisited . . . 206
6.7 Chapter Summary . . . 209
Problems . . . 211
Notes and References . . . 214

7 PATH AND TRAJECTORY PLANNING 215
7.1 The Configuration Space . . . 216
7.1.1 Representing the Configuration Space . . . 217
7.1.2 Configuration Space Obstacles . . . 218
7.1.3 Paths in the Configuration Space . . . 221
7.2 Path Planning for $\mathcal{Q} = \mathbb{R}^2$ . . . 221
7.2.1 The Visibility Graph . . . 222
7.2.2 The Generalized Voronoi Diagram . . . 224
7.2.3 Trapezoidal Decompositions . . . 226
7.3 Artificial Potential Fields . . . 229
7.3.1 Artificial Potential Fields for $\mathcal{Q} = \mathbb{R}^n$ . . . 230
7.3.2 Potential Fields for $\mathcal{Q} \neq \mathbb{R}^n$ . . . 235
7.4 Sampling-Based Methods . . . 245
7.4.1 Probabilistic Roadmaps (PRM) . . . 246
7.4.2 Rapidly-Exploring Random Trees (RRTs) . . . 250
7.5 Trajectory Planning . . . 252
7.5.1 Trajectories for Point-to-Point Motion . . . 253
7.5.2 Trajectories for Paths Specified by Via Points . . . 261

7.6 Chapter Summary 263
Problems 265
Notes and References 267

III CONTROL OF MANIPULATORS 269

8 INDEPENDENT JOINT CONTROL 271
8.1 Introduction 271
8.2 Actuator Dynamics 273
8.3 Load Dynamics 276
8.4 Independent Joint Model 278
8.5 PID Control 281
8.6 Feedforward Control 288
8.6.1 Trajectory Tracking 289
8.6.2 The Method of Computed Torque 291
8.7 Drive-Train Dynamics 292
8.8 State Space Design 297
8.8.1 State Feedback Control 299
8.8.2 Observers 301
8.9 Chapter Summary 304
Problems 307
Notes and References 309

9 NONLINEAR AND MULTIVARIABLE CONTROL 311
9.1 Introduction 311
9.2 PD Control Revisited 313
9.3 Inverse Dynamics 317
9.3.1 Joint Space Inverse Dynamics 317
9.3.2 Task Space Inverse Dynamics 320
9.3.3 Robust Inverse Dynamics 322
9.3.4 Adaptive Inverse Dynamics 327
9.4 Passivity-Based Control 329
9.4.1 Passivity-Based Robust Control 331
9.4.2 Passivity-Based Adaptive Control 332
9.5 Torque Optimization 333
9.6 Chapter Summary 337
Problems 341
Notes and References 343

10 FORCE CONTROL **345**
10.1 Coordinate Frames and Constraints 347
10.1.1 Reciprocal Bases 347
10.1.2 Natural and Artificial Constraints 349
10.2 Network Models and Impedance 351
10.2.1 Impedance Operators 353
10.2.2 Classification of Impedance Operators 354
10.2.3 Thévenin and Norton Equivalents 355
10.3 Task Space Dynamics and Control 355
10.3.1 Impedance Control 356
10.3.2 Hybrid Impedance Control 358
10.4 Chapter Summary 361
Problems 362
Notes and References 364

11 VISION-BASED CONTROL **365**
11.1 Design Considerations 366
11.1.1 Camera Configuration 366
11.1.2 Image-Based vs. Position-Based Approaches 367
11.2 Computer Vision for Vision-Based Control 368
11.2.1 The Geometry of Image Formation 369
11.2.2 Image Features 373
11.3 Camera Motion and the Interaction Matrix 378
11.4 The Interaction Matrix for Point Features 379
11.4.1 Velocity Relative to a Moving Frame 380
11.4.2 Constructing the Interaction Matrix 381
11.4.3 Properties of the Interaction Matrix for Points 384
11.4.4 The Interaction Matrix for Multiple Points 385
11.5 Image-Based Control Laws 386
11.5.1 Computing Camera Motion 387
11.5.2 Proportional Control Schemes 389
11.5.3 Performance of Image-Based Control Systems 390
11.6 End Effector and Camera Motions 393
11.7 Partitioned Approaches 394
11.8 Motion Perceptibility 397
11.9 Summary 399
Problems 401
Notes and References 405

12 FEEDBACK LINEARIZATION 409
12.1 Background 410
12.1.1 Manifolds, Vector Fields, and Distributions 410
12.1.2 The Frobenius Theorem 414
12.2 Feedback Linearization 417
12.3 Single-Input Systems 419
12.4 Multi-Input Systems 429
12.5 Chapter Summary 433
Problems 433
Notes and References 435

IV CONTROL OF UNDERACTUATED SYSTEMS 437

13 UNDERACTUATED ROBOTS 439
13.1 Introduction 439
13.2 Modeling 440
13.3 Examples of Underactuated Robots 443
13.3.1 The Cart-Pole System 443
13.3.2 The Acrobot 445
13.3.3 The Pendubot 446
13.3.4 The Reaction-Wheel Pendulum 447
13.4 Equilibria and Linear Controllability 448
13.4.1 Linear Controllability 450
13.5 Partial Feedback Linearization 456
13.5.1 Collocated Partial Feedback Linearization 457
13.5.2 Noncollocated Partial Feedback Linearization 459
13.6 Output Feedback Linearization 461
13.6.1 Computation of the Zero Dynamics 463
13.6.2 Virtual Holonomic Constraints 466
13.7 Passivity-Based Control 466
13.7.1 The Simple Pendulum 467
13.7.2 The Reaction-Wheel Pendulum 471
13.7.3 Swingup and Balance of The Acrobot 473
13.8 Chapter Summary 474
Problems 476
Notes and References 477

14 MOBILE ROBOTS **479**
14.1 Nonholonomic Constraints 480
14.2 Involutivity and Holonomy 484
14.3 Examples of Nonholonomic Systems 487
14.4 Dynamic Extension 493
14.5 Controllability of Driftless Systems 495
14.6 Motion Planning 499
14.6.1 Conversion to Chained Forms 499
14.6.2 Differential Flatness 506
14.7 Feedback Control of Driftless Systems 509
14.7.1 Stabilizability 509
14.7.2 Nonsmooth Control 511
14.7.3 Trajectory Tracking 513
14.7.4 Feedback Linearization 515
14.8 Chapter Summary 519
Problems 520
Notes and References 521

A TRIGONOMETRY **523**
A.1 The Two-Argument Arctangent Function 523
A.2 Useful Trigonometric Formulas 523

B LINEAR ALGEBRA **525**
B.1 Vectors 525
B.2 Inner Product Spaces 526
B.3 Matrices 528
B.4 Eigenvalues and Eigenvectors 530
B.5 Differentiation of Vectors 533
B.6 The Matrix Exponential 534
B.7 Lie Groups and Lie Algebras 534
B.8 Matrix Pseudoinverse 536
B.9 Schur Complement 536
B.10 Singular Value Decomposition (SVD) 537

C LYAPUNOV STABILITY **539**
C.1 Continuity and Differentiability 539
C.2 Vector Fields and Equilibria 541
C.3 Lyapunov Functions 545
C.4 Stability Criteria 545
C.5 Global and Exponential Stability 546
C.6 Stability of Linear Systems 547

C.7 LaSalle's Theorem 548
C.8 Barbalat's Lemma 549

D OPTIMIZATION **551**
D.1 Unconstrained Optimization 551
D.2 Constrained Optimization 552

E CAMERA CALIBRATION **555**
E.1 The Image Plane and the Sensor Array 555
E.2 Extrinsic Camera Parameters 556
E.3 Intrinsic Camera Parameters 557
E.4 Determining the Camera Parameters 557

BIBLIOGRAPHY **561**

INDEX **576**

Chapter 1

INTRODUCTION

Robotics is a relatively young field of modern technology that crosses traditional engineering boundaries. Understanding the complexity of robots and their application requires knowledge of electrical engineering, mechanical engineering, systems and industrial engineering, computer science, economics, and mathematics. New disciplines of engineering, such as manufacturing engineering, applications engineering, and knowledge engineering have emerged to deal with the complexity of the field of robotics and factory automation. More recently, mobile robots are increasingly important for applications like autonomous vehicles and planetary exploration.

This book is concerned with fundamentals of robotics, including **kinematics**, **dynamics**, **motion planning**, **computer vision**, and **control**. Our goal is to provide an introduction to the most important concepts in these subjects as applied to industrial robot manipulators, mobile robots and other mechanical systems.

The term **robot** was first introduced by the Czech playwright Karel Čapek in his 1920 play Rossum's Universal Robots, the word **robota** being the Czech word for worker. Since then the term has been applied to a great variety of mechanical devices, such as teleoperators, underwater vehicles, autonomous cars, drones, etc. Virtually anything that operates with some degree of autonomy under computer control has at some point been called a robot. In this text we will focus on two types of robots, namely industrial manipulators and mobile robots.

Industrial Manipulators

An industrial manipulator of the type shown in Figure 1.1 is essentially a mechanical arm operating under computer control. Such devices, though

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

Figure 1.1: A six-axis industrial manipulator, the KUKA 500 FORTEC robot. (Photo courtesy of KUKA Robotics.)

far from the robots of science fiction, are nevertheless extremely complex electromechanical systems whose analytical description requires advanced methods, presenting many challenging and interesting research problems.

An official definition of such a robot comes from the **Robot Institute of America** (RIA):

A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.

The key element in the above definition is the reprogrammability, which gives a robot its utility and adaptability. The so-called robotics revolution is, in fact, part of the larger computer revolution.

Even this restricted definition of a robot has several features that make it attractive in an industrial environment. Among the advantages often cited in favor of the introduction of robots are decreased labor costs, increased precision and productivity, increased flexibility compared with specialized machines, and more humane working conditions as dull, repetitive, or hazardous jobs are performed by robots.

The industrial manipulator was born out of the marriage of two earlier technologies: **teleoperators** and **numerically controlled milling machines**. Teleoperators, or master-slave devices, were developed during the second world war to handle radioactive materials. Computer numerical control (CNC) was developed because of the high precision required in the ma-

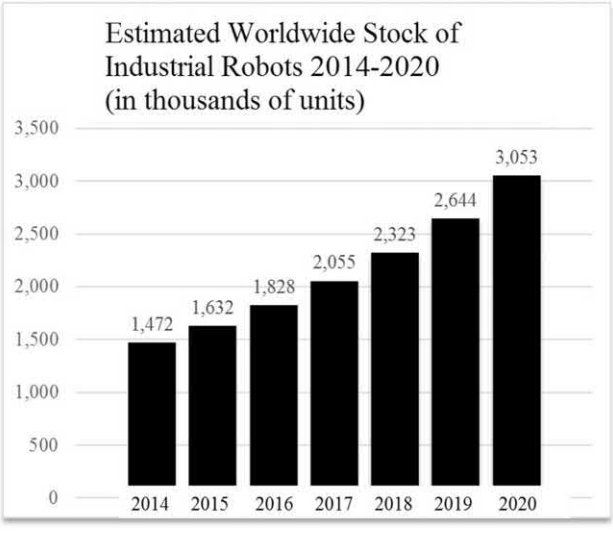


Figure 1.2: Estimated number of industrial robots worldwide 2014–2020. The industrial robot market has been growing around 14% per year. (Source: International Federation of Robotics 2018.)

chining of certain items, such as components of high performance aircraft. The first industrial robots essentially combined the mechanical linkages of the teleoperator with the autonomy and programmability of CNC machines.

The first successful applications of robot manipulators generally involved some sort of material transfer, such as injection molding or stamping, in which the robot merely attended a press to unload and either transfer or stack the finished parts. These first robots could be programmed to execute a sequence of movements, such as moving to a location A, closing a gripper, moving to a location B, etc., but had no external sensor capability. More complex applications, such as welding, grinding, deburring, and assembly, require not only more complex motion but also some form of external sensing such as vision, tactile, or force sensing, due to the increased interaction of the robot with its environment.

Figure 1.2 shows the estimated number of industrial robots worldwide between 2014 and 2020. In the future the market for service and medical robots will likely be even greater than the market for industrial robots. Service robots are defined as robots outside the manufacturing sector, such as robot vacuum cleaners, lawn mowers, window cleaners, delivery robots, etc. In 2018 alone, more than 30 million service robots were sold worldwide. The future market for robot assistants for elderly care and other medical robots will also be strong as populations continue to age.

Mobile Robots

Mobile robots encompass wheel and track driven robots, walking robots, climbing, swimming, crawling and flying robots. A typical wheeled mobile robot is shown in Figure 1.3. Mobile robots are used as household robots for vacuum cleaning and lawn mowing robots, as field robots for surveillance, search and rescue, environmental monitoring, forestry and agriculture, and other applications. Autonomous vehicles, for example self-driving cars and trucks, is an emerging area of robotics with great interest and promise.

There are many other applications of robotics in areas where the use of humans is impractical or undesirable. Among these are undersea and planetary exploration, satellite retrieval and repair, the defusing of explosive devices, and work in radioactive environments. Finally, prostheses, such as artificial limbs, are themselves robotic devices requiring methods of analysis and design similar to those of industrial manipulators.

The science of robotics has grown tremendously over the past twenty years, fueled by rapid advances in computer and sensor technology as well as theoretical advances in control and computer vision. In addition to the topics listed above, robotics encompasses several areas not covered in this text such as legged robots, flying and swimming robots, grasping, artificial intelligence, computer architectures, programming languages, and computer-aided design. In fact, the new subject of **mechatronics** has emerged over the past four decades and, in a sense, includes robotics as a subdiscipline.

Figure 1.3: Example of a typical mobile robot, the Fetch series. The figure on the right shows the mobile robot base with an attached manipulator arm. (Photo courtesy of Fetch Robotics.)

Mechatronics has been defined as the synergistic integration of mechanics, electronics, computer science, and control, and includes not only robotics, but many other areas such as automotive control systems.

1.1 Mathematical Modeling of Robots

In this text we will be primarily concerned with developing and analyzing mathematical models for robots. In particular, we will develop methods to represent basic geometric aspects of robotic manipulation and locomotion. Equipped with these mathematical models, we will develop methods for planning and controlling robot motions to perform specified tasks. We begin here by describing some of the basic notation and terminology that we will use in later chapters to develop mathematical models for robot manipulators and mobile robots.

1.1.1 Symbolic Representation of Robot Manipulators

Robot manipulators are composed of **links** connected by **joints** to form a **kinematic chain**. Joints are typically rotary (revolute) or linear (prismatic). A **revolute** joint is like a hinge and allows relative rotation between two links. A **prismatic** joint allows a linear relative motion between two links. We denote revolute joints by R and prismatic joints by P, and draw them as shown in Figure 1.4. For example, a three-link arm with three revolute joints will be referred to as an RRR arm.

Each joint represents the interconnection between two links. We denote the axis of rotation of a revolute joint, or the axis along which a prismatic joint translates by z_i if the joint is the interconnection of links i and $i+1$. The **joint variables**, denoted by θ for a revolute joint and d for the prismatic joint, represent the relative displacement between adjacent links. We will make this precise in Chapter 3.

1.1.2 The Configuration Space

A **configuration** of a manipulator is a complete specification of the location of every point on the manipulator. The set of all configurations is called the **configuration space**. In the case of a manipulator arm, if we know the values for the joint variables (i.e., the joint angle for revolute joints, or the joint offset for prismatic joints), then it is straightforward to infer the position of any point on the manipulator, since the individual links of the manipulator are assumed to be rigid and the base of the manipulator is

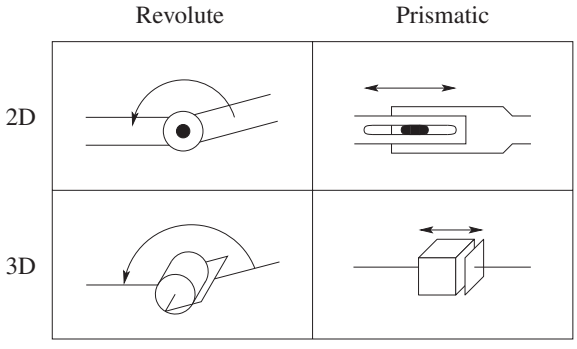


Figure 1.4: Symbolic representation of robot joints. Each joint allows a single degree of freedom of motion between adjacent links of the manipulator. The revolute joint (shown in 2D and 3D on the left) produces a relative rotation between adjacent links. The prismatic joint (shown in 2D and 3D on the right) produces a linear or telescoping motion between adjacent links.

assumed to be fixed. Therefore, we will represent a configuration by a set of values for the joint variables. We will denote this vector of values by q, and say that the robot is in configuration q when the joint variables take on the values $q_1, \ldots, q_n$, with $q_i = \theta_i$ for a revolute joint and $q_i = d_i$ for a prismatic joint.

An object is said to have n **degrees of freedom** (DOF) if its configuration can be minimally specified by n parameters. Thus, the number of DOF is equal to the dimension of the configuration space. For a robot manipulator, the number of joints determines the number of DOF. A rigid object in three-dimensional space has six DOF: three for **positioning** and three for **orientation**. Therefore, a manipulator should typically possess at least six independent DOF. With fewer than six DOF the arm cannot reach every point in its work space with arbitrary orientation. Certain applications such as reaching around or behind obstacles may require more than six DOF. A manipulator having more than six DOF is referred to as a **kinematically redundant** manipulator.

1.1.3 The State Space

A configuration provides an instantaneous description of the geometry of a manipulator, but says nothing about its dynamic response. In contrast, the **state** of the manipulator is a set of variables that, together with a description of the manipulator's dynamics and future inputs, is sufficient to determine the future time response of the manipulator. The **state space** is

Figure 1.5: The Kinova® Gen3 Ultra lightweight arm, a 7-degree-of-freedom redundant manipulator. (Photo courtesy of Kinova, Inc.)

the set of all possible states. In the case of a manipulator arm, the dynamics are Newtonian, and can be specified by generalizing the familiar equation $F = ma$. Thus, a state of the manipulator can be specified by giving the values for the joint variables q and for the joint velocities $\dot{q}$ (acceleration is related to the derivative of joint velocities). The dimension of the state space is thus $2n$ if the system has n DOF.

1.1.4 The Workspace

The **workspace** of a manipulator is the total volume swept out by the end effector as the manipulator executes all possible motions. The workspace is constrained by the geometry of the manipulator as well as mechanical constraints on the joints. For example, a revolute joint may be limited to less than a full 360° of motion. The workspace is often broken down into a **reachable workspace** and a **dexterous workspace**. The reachable workspace is the entire set of points reachable by the manipulator, whereas the dexterous workspace consists of those points that the manipulator can reach with an arbitrary orientation of the end effector. Obviously the dexterous workspace is a subset of the reachable workspace. The workspaces of several robots are shown later in this chapter.

1.2 Robots as Mechanical Devices

There are a number of physical aspects of robotic manipulators that we will not necessarily consider when developing our mathematical models. These include mechanical aspects (e.g., how are the joints actually constructed),

accuracy and repeatability, and the tooling attached at the end effector. In this section, we briefly describe some of these.

1.2.1 Classification of Robotic Manipulators

Robot manipulators can be classified by several criteria, such as their **power source**, meaning the way in which the joints are actuated; their **geometry**, or kinematic structure; their **method of control**; and their intended **application area**. Such classification is useful primarily in order to determine which robot is right for a given task. For example, an hydraulic robot would not be suitable for food handling or clean room applications whereas a SCARA robot would not be suitable for automobile spray painting. We explain this in more detail below.

Power Source

Most robots are either electrically, hydraulically, or pneumatically powered. Hydraulic actuators are unrivaled in their speed of response and torque producing capability. Therefore hydraulic robots are used primarily for lifting heavy loads. The drawbacks of hydraulic robots are that they tend to leak hydraulic fluid, require much more peripheral equipment (such as pumps, which require more maintenance), and they are noisy. Robots driven by DC or AC motors are increasingly popular since they are cheaper, cleaner and quieter. Pneumatic robots are inexpensive and simple but cannot be controlled precisely. As a result, pneumatic robots are limited in their range of applications and popularity.

Method of Control

Robots are classified by control method into **servo** and **nonservo** robots. The earliest robots were nonservo robots. These robots are essentially **open-loop** devices whose movements are limited to predetermined mechanical stops, and they are useful primarily for materials transfer. In fact, according to the definition given above, fixed stop robots hardly qualify as robots. Servo robots use **closed-loop** computer control to determine their motion and are thus capable of being truly multifunctional, reprogrammable devices.

Servo controlled robots are further classified according to the method that the controller uses to guide the end effector. The simplest type of robot in this class is the **point-to-point** robot. A point-to-point robot can be taught a discrete set of points but there is no control of the path of the end effector in between taught points. Such robots are usually taught

a series of points with a **teach pendant**. The points are then stored and played back. Point-to-point robots are limited in their range of applications. With **continuous path** robots, on the other hand, the entire path of the end effector can be controlled. For example, the robot end effector can be taught to follow a straight line between two points or even to follow a contour such as a welding seam. In addition, the velocity and/or acceleration of the end effector can often be controlled. These are the most advanced robots and require the most sophisticated computer controllers and software development.

Application Area

Robot manipulators are often classified by application area into **assembly** and **nonassembly robots**. Assembly robots tend to be small and electrically driven with either revolute or SCARA geometries (described below). Typical nonassembly application areas are in welding, spray painting, material handling, and machine loading and unloading.

One of the primary differences between assembly and nonassembly applications is the increased level of precision required in assembly due to significant interaction with objects in the workspace. For example, an assembly task may require part insertion (the so-called **peg-in-hole** problem) or gear meshing. A slight mismatch between the parts can result in wedging and jamming, which can cause large interaction forces and failure of the task. As a result, assembly tasks are difficult to accomplish without special fixtures and jigs, or without controlling the interaction forces.

Geometry

Most industrial manipulators at the present time have six or fewer DOF. These manipulators are usually classified kinematically on the basis of the first three joints of the arm, with the wrist being described separately. The majority of these manipulators fall into one of five geometric types: **articulated (RRR)**, **spherical (RRP)**, **SCARA (RRP)**, **cylindrical (RPP)**, or **Cartesian (PPP)**. We discuss each of these below in Section 1.3.

Each of these five manipulator arms is a **serial link** robot. A sixth distinct class of manipulators consists of the so-called **parallel robot**. In a parallel manipulator the links are arranged in a closed rather than open kinematic chain. Although we include a brief discussion of parallel robots in this chapter, their kinematics and dynamics are more difficult to derive than those of serial link robots and hence are usually treated only in more advanced texts.

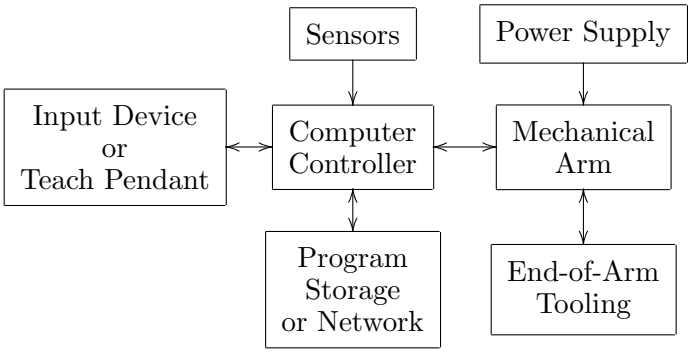


Figure 1.6: The integration of a mechanical arm, sensing, computation, user interface and tooling forms a complex robotic system. Many modern robotic systems have integrated computer vision, force/torque sensing, and advanced programming and user interface features.

1.2.2 Robotic Systems

A robot manipulator should be viewed as more than just a series of mechanical linkages. The mechanical arm is just one component in an overall **robotic system**, illustrated in Figure 1.6, which consists of the **arm**, **external power source**, **end-of-arm tooling**, **external and internal sensors**, **computer interface**, and **control computer**. Even the programmed software should be considered as an integral part of the overall system, since the manner in which the robot is programmed and controlled can have a major impact on its performance and subsequent range of applications.

1.2.3 Accuracy and Repeatability

The **accuracy** of a manipulator is a measure of how close the manipulator can come to a given point within its workspace. **Repeatability** is a measure of how close a manipulator can return to a previously taught point. The primary method of sensing positioning errors is with position encoders located at the joints, either on the shaft of the motor that actuates the joint or on the joint itself. There is typically no direct measurement of the end-effector position and orientation. One relies instead on the assumed geometry of the manipulator and its rigidity to calculate the end-effector position from the measured joint positions. Accuracy is affected therefore by computational errors, machining accuracy in the construction of the manipulator, flexibility effects such as the bending of the links under gravitational and other loads, gear backlash, and a host of other static and dynamic effects.

It is primarily for this reason that robots are designed with extremely high rigidity. Without high rigidity, accuracy can only be improved by some sort of direct sensing of the end-effector position, such as with computer vision.

Once a point is taught to the manipulator, however, say with a teach pendant, the above effects are taken into account and the proper encoder values necessary to return to the given point are stored by the controlling computer. Repeatability therefore is affected primarily by the controller resolution. **Controller resolution** means the smallest increment of motion that the controller can sense. The resolution is computed as the total distance traveled divided by 2^n, where n is the number of bits of encoder accuracy. In this context, linear axes, that is, prismatic joints, typically have higher resolution than revolute joints, since the straight-line distance traversed by the tip of a linear axis between two points is less than the corresponding arc length traced by the tip of a rotational link.

In addition, as we will see in later chapters, rotational axes usually result in a large amount of kinematic and dynamic coupling among the links, with a resultant accumulation of errors and a more difficult control problem. One may wonder then what the advantages of revolute joints are in manipulator design. The answer lies primarily in the increased dexterity and compactness of revolute joint designs. For example, Figure 1.7 shows that for the same range of motion, a rotational link can be made much smaller than a link with linear motion.

Thus, manipulators made from revolute joints occupy a smaller working volume than manipulators with linear axes. This increases the ability of the manipulator to work in the same space with other robots, machines, and people. At the same time, revolute-joint manipulators are better able to maneuver around obstacles and have a wider range of possible applications.

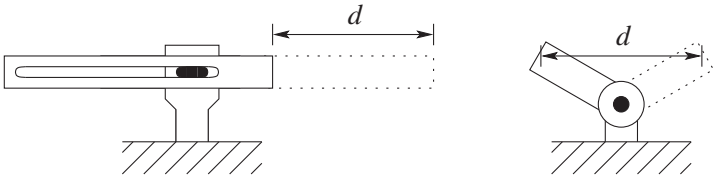


Figure 1.7: Linear vs. rotational link motion showing that a smaller revolute joint can cover the same distance d as a larger prismatic joint. The tip of a prismatic link can cover a distance equal to the length of the link. The tip of a rotational link of length a, by contrast, can cover a distance of $2a$ by rotating 180 degrees.

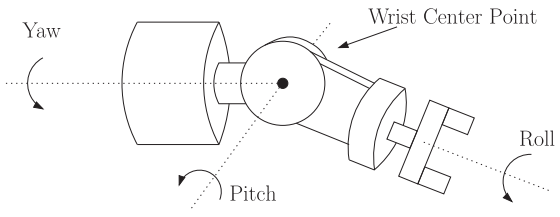


Figure 1.8: The spherical wrist. The axes of rotation of the spherical wrist are typically denoted roll, pitch, and yaw and intersect at a point called the wrist center point.

1.2.4 Wrists and End Effectors

The joints in the kinematic chain between the arm and end effector are referred to as the **wrist**. The wrist joints are nearly always all revolute. It is increasingly common to design manipulators with **spherical wrists**, by which we mean wrists whose three joint axes intersect at a common point, known as the **wrist center point**. Such a spherical wrist is shown in Figure 1.8.

The spherical wrist greatly simplifies kinematic analysis, effectively allowing one to decouple the position and orientation of the end effector. Typically the manipulator will possess three DOF for position, which are produced by three or more joints in the arm. The number of DOF for orientation will then depend on the DOF of the wrist. It is common to find wrists having one, two, or three DOF depending on the application. For example, the SCARA robot shown in Figure 1.14 has four DOF: three for the arm, and one for the wrist, which has only a rotation about the final z-axis.

The arm and wrist assemblies of a robot are used primarily for positioning the **hand**, **end effector**, and any **tool** it may carry. It is the end effector or tool that actually performs the task. The simplest type of end effector is a gripper, such as shown in Figure 1.9, which is usually capable of only two actions, **opening** and **closing**. While this is adequate for materials transfer, some parts handling, or gripping simple tools, it is not adequate for other tasks such as welding, assembly, grinding, etc.

A great deal of research is therefore devoted to the design of special purpose end effectors as well as of tools that can be rapidly changed as the task dictates. Since we are concerned with the analysis and control of the manipulator itself and not in the particular application or end effector, we will

Figure 1.9: A two-finger gripper. (Photo courtesy of Robotiq, Inc.)

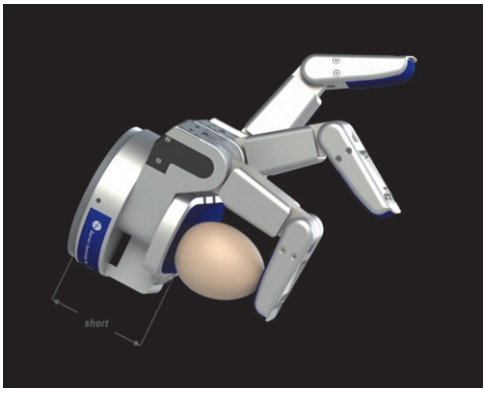


Figure 1.10: Anthropomorphic hand developed by Barrett Technologies. Such grippers allow for more dexterity and the ability to manipulate objects of various sizes and geometries. (Photo courtesy of Barrett Technologies.)

not discuss the design of end effectors or the study of grasping and manipulation. There is also much research on the development of anthropomorphic hands such as that shown in Figure 1.10.

1.3 Common Kinematic Arrangements

There are many possible ways to construct kinematic chains using prismatic and revolute joints. However, in practice, only a few kinematic designs are commonly used. Here we briefly describe the most typical arrangements.

1.3.1 Articulated Manipulator (RRR)

The articulated manipulator is also called a **revolute**, **elbow**, or **anthropomorphic** manipulator. The KUKA 500 articulated arm is shown in Fig-

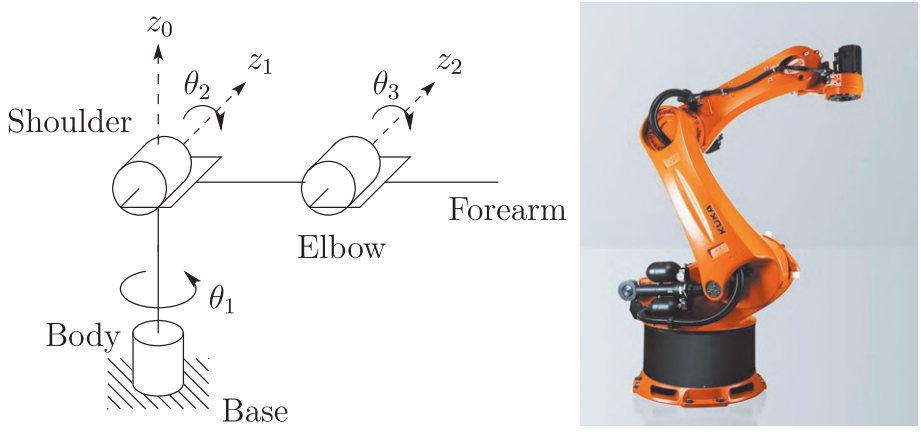


Figure 1.11: Symbolic representation of an RRR manipulator (left), and the KUKA 500 arm (right), which is a typical example of an RRR manipulator. The links and joints of the RRR configuration are analogous to human joints and limbs. (Photo courtesy of KUKA Robotics.)

ure 1.11. In the anthropomorphic design the three links are designated as the body, upper arm, and forearm, respectively, as shown in Figure 1.11. The joint axes are designated as the **waist** (z_0), **shoulder** (z_1), and **elbow** (z_2). Typically, the joint axis z_2 is parallel to z_1 and both z_1 and z_2 are perpendicular to z_0. The workspace of the elbow manipulator is shown in Figure 1.12.

1.3.2 Spherical Manipulator (RRP)

By replacing the third joint, or elbow joint, in the revolute manipulator by a prismatic joint, one obtains the spherical manipulator shown in Figure 1.13. The term **spherical manipulator** derives from the fact that the joint coordinates coincide with the spherical coordinates of the end effector relative to a coordinate frame located at the shoulder joint. Figure 1.13 shows the Stanford Arm, one of the most well-known spherical robots.

1.3.3 SCARA Manipulator (RRP)

The **SCARA** arm (for **S**elective **C**ompliant **A**rticulated **R**obot for **A**ssembly) shown in Figure 1.14 is a popular manipulator, which, as its name suggests, is tailored for pick-and-place and assembly operations. Although the SCARA has an RRP structure, it is quite different from the spherical manipulator in both appearance and in its range of applications. Unlike the spherical

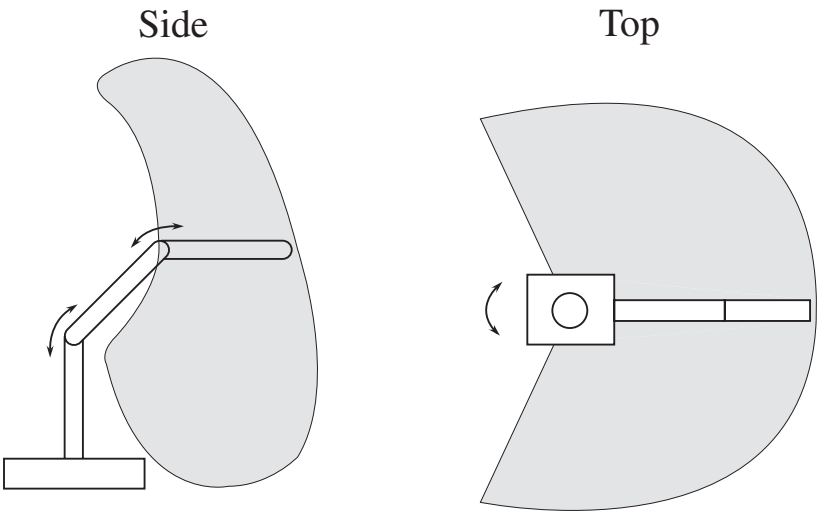


Figure 1.12: Workspace of the elbow manipulator. The elbow manipulator provides a larger workspace than other kinematic designs relative to its size.

design, which has z_0 perpendicular to z_1, and z_1 perpendicular to z_2, the SCARA has z_0, z_1, and z_2 mutually parallel. Figure 1.14 shows the symbolic representation of a SCARA arm and the Yamaha YK-XC manipulator.

1.3.4 Cylindrical Manipulator (RPP)

The cylindrical manipulator is shown in Figure 1.15. The first joint is revolute and produces a rotation about the base, while the second and third joints are prismatic. As the name suggests, the joint variables are the cylindrical coordinates of the end effector with respect to the base.

1.3.5 Cartesian Manipulator (PPP)

A manipulator whose first three joints are prismatic is known as a Cartesian manipulator. The joint variables of the Cartesian manipulator are the Cartesian coordinates of the end effector with respect to the base. As might be expected, the kinematic description of this manipulator is the simplest of all manipulators. Cartesian manipulators are useful for table-top assembly applications and, as gantry robots, for transfer of material or cargo. The symbolic representation of a Cartesian robot is shown in Figure 1.16.

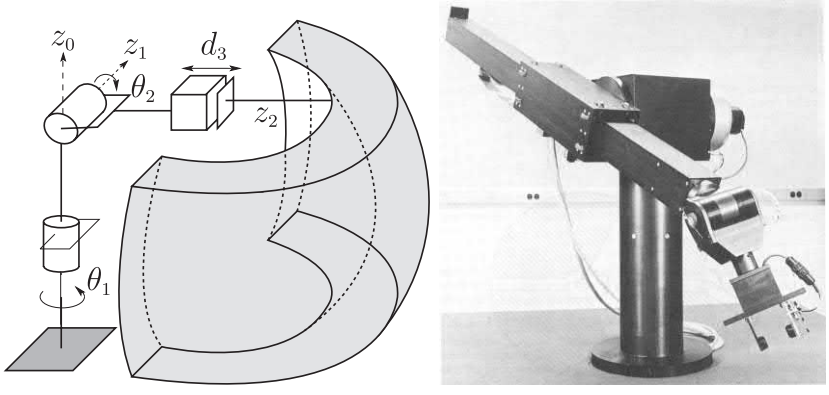


Figure 1.13: Schematic representation of an RRP manipulator, referred to as a spherical robot (left), and the Stanford Arm (right), an early example of a spherical arm. (Photo courtesy of the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

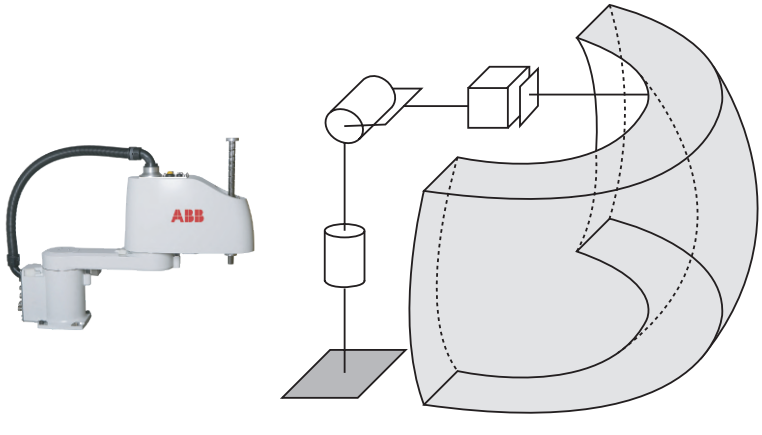


Figure 1.14: The ABB IRB910SC SCARA robot (left) and the symbolic representation showing a portion of its workspace (right). (Photo courtesy of ABB.)

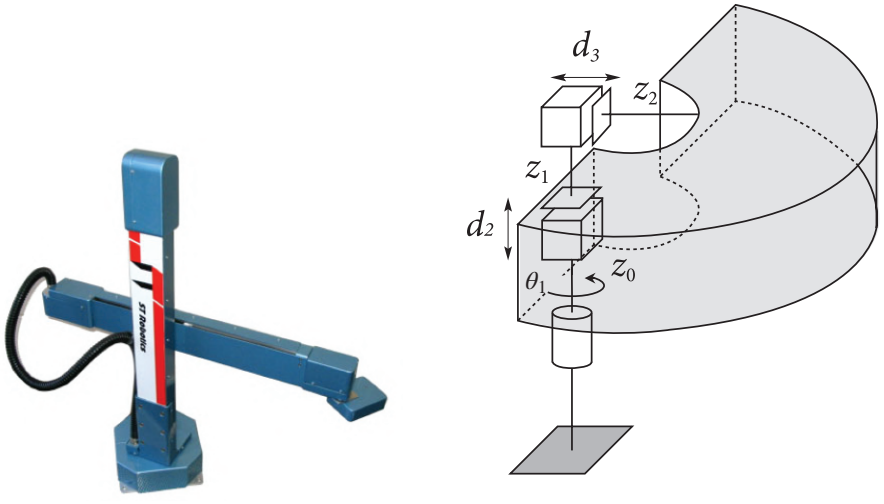


Figure 1.15: The ST Robotics R19 cylindrical robot (left) and the symbolic representation showing a portion of its workspace (right). Cylindrical robots are often used in materials transfer tasks. (Photo courtesy of ST Robotics.)

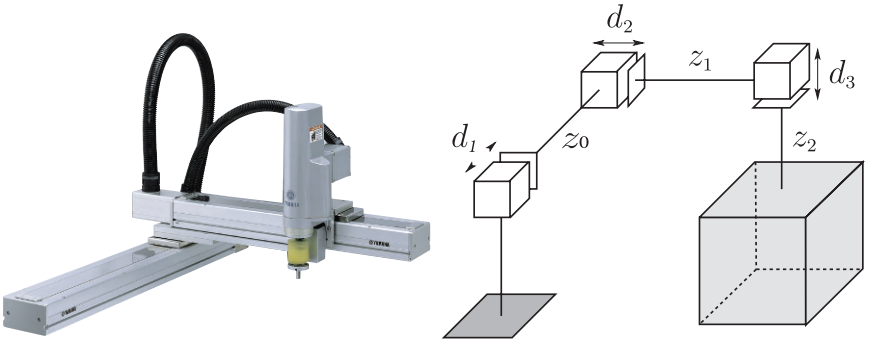


Figure 1.16: The Yamaha YK-XC Cartesian robot (left) and the symbolic representation showing a portion of its workspace (right). Cartesian robots are often used in pick-and-place operations. (Photo courtesy of Yamaha Robotics.)

1.3.6 Parallel Manipulator

A **parallel manipulator** is one in which some subset of the links form a closed chain. More specifically, a parallel manipulator has two or more kinematic chains connecting the base to the end effector. Figure 1.17 shows the ABB IRB360 robot, which is a parallel manipulator. The closed-chain kinematics of parallel robots can result in greater structural rigidity, and hence greater accuracy, than open chain robots. The kinematic description of parallel robots is fundamentally different from that of serial link robots and therefore requires different methods of analysis.

1.4 Outline of the Text

The present textbook is divided into four parts. The first three parts are devoted to the study of manipulator arms. The final part treats the control of underactuated and mobile robots.

1.4.1 Manipulator Arms

We can use the simple example below to illustrate some of the major issues involved in the study of manipulator arms and to preview the topics covered. A typical application involving an industrial manipulator is shown in Figure 1.18. The manipulator is shown with a grinding tool that it must use to remove a certain amount of metal from a surface. Suppose we wish to move the manipulator from its **home** position to position A, from which

Figure 1.17: The ABB IRB360 parallel robot. Parallel robots generally have higher structural rigidity than serial link robots. (Photo courtesy of ABB.)

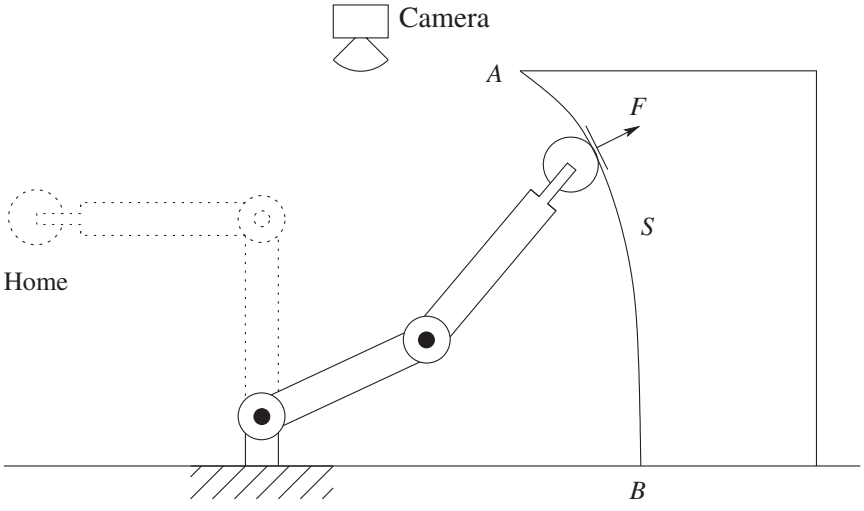


Figure 1.18: Two-link planar robot example. Each chapter of the text discusses a fundamental concept applicable to the task shown.

point the robot is to follow the contour of the surface S to the point B, at constant velocity, while maintaining a prescribed force F normal to the surface. In so doing the robot will cut or grind the surface according to a predetermined specification. To accomplish this and even more general tasks, we must solve a number of problems. Below we give examples of these problems, all of which will be treated in more detail in the text.

Chapter 2: Rigid Motions

The first problem encountered is to describe both the position of the tool and the locations A and B (and most likely the entire surface S) with respect to a common coordinate system. In Chapter 2 we describe representations of coordinate systems and transformations among various coordinate systems. We describe several ways to represent rotations and rotational transformations and we introduce so-called **homogeneous transformations**, which combine position and orientation into a single matrix representation.

Chapter 3: Forward Kinematics

Typically, the manipulator will be able to sense its own position in some manner using internal sensors (position encoders located at joints 1 and 2) that can measure directly the joint angles θ_1 and θ_2. We also need therefore to express the positions A and B in terms of these joint angles. This leads

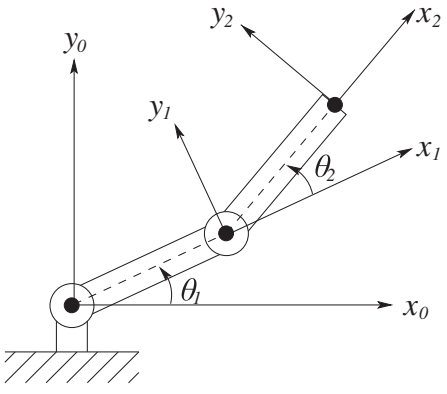


Figure 1.19: Coordinate frames attached to the links of a two-link planar robot. Each coordinate frame moves as the corresponding link moves. The mathematical description of the robot motion is thus reduced to a mathematical description of moving coordinate frames.

to the **forward kinematics problem** studied in Chapter 3, which is to determine the position and orientation of the end effector or tool in terms of the joint variables.

It is customary to establish a fixed coordinate system, called the **world** or **base** frame to which all objects including the manipulator are referenced. In this case we establish the base coordinate frame $o_0x_0y_0$ at the base of the robot, as shown in Figure 1.19. The coordinates (x, y) of the tool are expressed in this coordinate frame as

$$x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \tag{1.1}$$
$$y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2) \tag{1.2}$$

in which a_1 and a_2 are the lengths of the two links, respectively. Also the **orientation of the tool frame** relative to the base frame is given by the direction cosines of the x_2 and y_2 axes relative to the x_0 and y_0 axes, that is,

$$\begin{array}{ll} x_2 \cdot x_0 = \cos(\theta_1 + \theta_2) \ ; & y_2 \cdot x_0 = -\sin(\theta_1 + \theta_2) \\ x_2 \cdot y_0 = \sin(\theta_1 + \theta_2) \ ; & y_2 \cdot y_0 = \ \cos(\theta_1 + \theta_2) \end{array} \tag{1.3}$$

which we may combine into a **rotation matrix**

$$\begin{bmatrix} x_2 \cdot x_0 & y_2 \cdot x_0 \\ x_2 \cdot y_0 & y_2 \cdot y_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{1.4}$$

Equations (1.1), (1.2), and (1.4) are called the **forward kinematic equations** for this arm. For a six-DOF robot these equations are quite complex and cannot be written down as easily as for the two-link manipulator. The general procedure that we discuss in Chapter 3 establishes coordinate frames at each joint and allows one to transform systematically among these frames using matrix transformations. The procedure that we use is referred to as the **Denavit–Hartenberg** convention. We then use **homogeneous coordinates** and **homogeneous transformations**, developed in Chapter 2, to simplify the transformation among coordinate frames.

Chapter 4: Velocity Kinematics

To follow a contour at constant velocity, or at any prescribed velocity, we must know the relationship between the tool velocity and the joint velocities. In this case we can differentiate Equations (1.1) and (1.2) to obtain

$$\begin{aligned} \dot{x} &= -a_1 \sin\theta_1 \cdot \dot{\theta}_1 - a_2 \sin(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2) \\ \dot{y} &= a_1 \cos\theta_1 \cdot \dot{\theta}_1 + a_2 \cos(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2) \end{aligned} \tag{1.5}$$

Using the vector notation $x = \begin{bmatrix} x \\ y \end{bmatrix}$ and $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$, we may write these equations as

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -a_1 \sin\theta_1 - a_2 \sin(\theta_1+\theta_2) & -a_2 \sin(\theta_1+\theta_2) \\ a_1 \cos\theta_1 + a_2 \cos(\theta_1+\theta_2) & a_2 \cos(\theta_1+\theta_2) \end{bmatrix} \dot{\theta} \\ &= J\dot{\theta} \end{aligned} \tag{1.6}$$

The matrix J defined by Equation (1.6) is called the **Jacobian** of the manipulator and is a fundamental object to determine for any manipulator. In Chapter 4 we present a systematic procedure for deriving the manipulator Jacobian.

The determination of the joint velocities from the end-effector velocities is conceptually simple since the velocity relationship is linear. Thus, the joint velocities are found from the end-effector velocities via the inverse Jacobian

$$\dot{\theta} = J^{-1}\dot{x} \tag{1.7}$$

where J^{-1} is given by

$$J^{-1} = \frac{1}{a_1 a_2 \sin\theta_2} \begin{bmatrix} a_2 \cos(\theta_1+\theta_2) & a_2 \sin(\theta_1+\theta_2) \\ -a_1 \cos\theta_1 - a_2 \cos(\theta_1+\theta_2) & -a_1 \sin\theta_1 - a_2 \sin(\theta_1+\theta_2) \end{bmatrix}$$

The determinant of the Jacobian in Equation (1.6) is equal to $a_1 a_2 \sin\theta_2$. Therefore, this Jacobian does not have an inverse when $\theta_2 = 0$ or $\theta_2 = \pi$, in

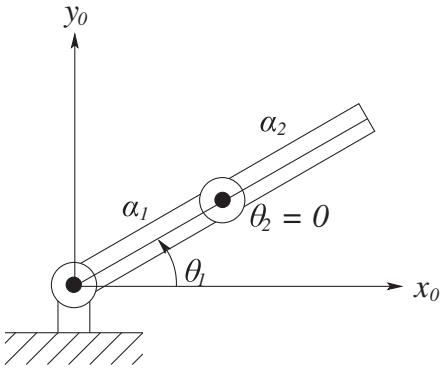


Figure 1.20: A singular configuration results when the elbow is straight. In this configuration the two-link robot has only one DOF.

which case the manipulator is said to be in a **singular configuration**, such as shown in Figure 1.20 for $\theta_2 = 0$. The determination of such singular configurations is important for several reasons. At singular configurations there are infinitesimal motions that are unachievable; that is, the manipulator end effector cannot move in certain directions. In the above example the end effector cannot move in the positive x_2 direction when $\theta_2 = 0$. Singular configurations are also related to the nonuniqueness of solutions of the inverse kinematics. For example, for a given end-effector position of the two-link planar manipulator, there are in general two possible solutions to the inverse kinematics. Note that a singular configuration separates these two solutions in the sense that the manipulator cannot go from one to the other without passing through a singularity. For many applications it is important to plan manipulator motions in such a way that singular configurations are avoided.

Chapter 5: Inverse Kinematics

Now, given the joint angles θ_1, θ_2 we can determine the end-effector coordinates x and y from Equations (1.1) and (1.2). In order to command the robot to move to location A we need the inverse; that is, we need to solve for the joint variables θ_1, θ_2 in terms of the x and y coordinates of A. This is the problem of **inverse kinematics**. Since the forward kinematic equations are nonlinear, a solution may not be easy to find, nor is there a unique solution in general. We can see in the case of a two-link planar mechanism that there may be no solution, for example if the given (x, y) coordinates are out of reach of the manipulator. If the given (x, y) coordinates are within the manipulator's reach there may be two solutions as shown in Figure 1.21,

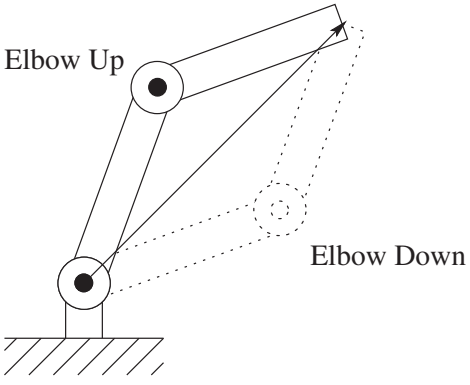


Figure 1.21: The two-link elbow robot has two solutions to the inverse kinematics except at singular configurations, the elbow up solution and the elbow down solution.

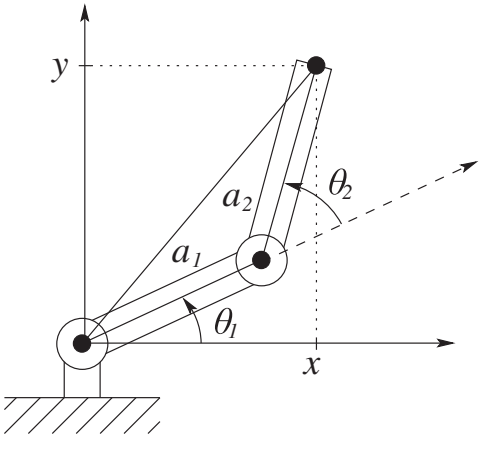


Figure 1.22: Solving for the joint angles of a two-link planar arm.

the so-called **elbow up** and **elbow down** configurations, or there may be exactly one solution if the manipulator must be fully extended to reach the point. There may even be an infinite number of solutions in some cases (Problem 1–19).

Consider the diagram of Figure 1.22. Using the **law of cosines**[1] we see that the angle θ_2 is given by

$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2} := D \tag{1.8}$$

[1]See Appendix A.

We could now determine θ_2 as $\theta_2 = \cos^{-1}(D)$. However, a better way to find θ_2 is to notice that if $\cos(\theta_2)$ is given by Equation (1.8), then $\sin(\theta_2)$ is given as

$$\sin(\theta_2) = \pm\sqrt{1 - D^2} \tag{1.9}$$

and, hence, θ_2 can be found by

$$\theta_2 = \tan^{-1}\frac{\pm\sqrt{1 - D^2}}{D} \tag{1.10}$$

The advantage of this latter approach is that both the elbow-up and elbow-down solutions are recovered by choosing the negative and positive signs in Equation (1.10), respectively.

It is left as an exercise (Problem 1–17) to show that θ_1 is now given as

$$\theta_1 = \tan^{-1}(y/x) - \tan^{-1}\left(\frac{a_2 \sin\theta_2}{a_1 + a_2\cos\theta_2}\right) \tag{1.11}$$

Notice that the angle θ_1 depends on θ_2. This makes sense physically since we would expect to require a different value for θ_1, depending on which solution is chosen for θ_2.

Chapter 6: Dynamics

In Chapter 6 we develop techniques based on **Lagrangian dynamics** for systematically deriving the equations of motion for serial-link manipulators. Deriving the dynamic equations of motion for robots is not a simple task due to the large number of degrees of freedom and the nonlinearities present in the system. We also discuss the so-called **recursive Newton–Euler** method for deriving the robot equations of motion. The Newton–Euler formulation is well-suited for real-time computation for both simulation and control applications.

Chapter 7: Path Planning and Trajectory Generation

The robot control problem is typically decomposed hierarchically into three tasks: **path planning**, **trajectory generation**, and **trajectory tracking**. The path planning problem, considered in Chapter 7, is to determine a path in task space (or configuration space) to move the robot to a goal position while avoiding collisions with objects in its workspace. These paths encode position and orientation information without timing considerations, that is, without considering velocities and accelerations along the planned

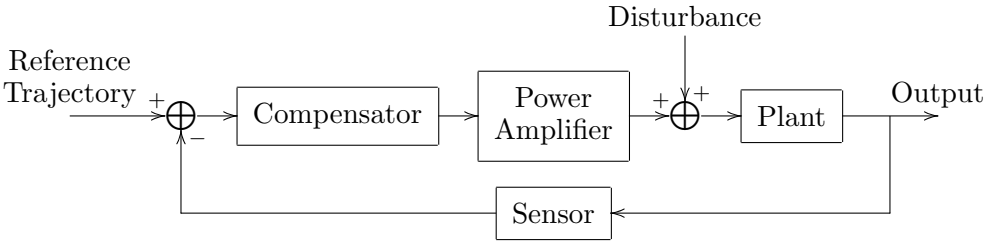


Figure 1.23: Basic structure of a feedback control system. The compensator measures the error between a reference and a measured output and produces a signal to the plant that is designed to drive the error to zero despite the presences of disturbances.

paths. The trajectory generation problem, also considered in Chapter 7, is to generate reference trajectories that determine the time history of the manipulator along a given path or between initial and final configurations. These are typically given in joint space as polynomial functions of time. We discuss the most common polynomial interpolation schemes used to generate these trajectories.

Chapter 8: Independent Joint Control

Once reference trajectories for the robot are specified, it is the task of the control system to track them. In Chapter 8 we discuss the motion control problem. We treat the **twin problems of tracking and disturbance rejection**, which are to determine the control inputs necessary to follow, or **track**, a reference trajectory, while simultaneously **rejecting** disturbances due to unmodeled dynamic effects such as friction and noise. We first model the actuator and drive-train dynamics and discuss the design of independent joint control algorithms.

A block diagram of a single-input/single-output (SISO) feedback control system is shown in Figure 1.23. We detail the standard approaches to robot control based on both frequency domain and state space techniques. We also introduce the notion of **feedforward control** for tracking time-varying trajectories. We also introduce the fundamental notion of **computed torque**, which is a feedforward disturbance cancellation scheme.

Chapter 9: Nonlinear and Multivariable Control

In Chapter 9 we discuss more advanced control techniques based on the Lagrangian dynamic equations of motion derived in Chapter 6. We introduce

the notion of **inverse dynamics** control as a means for compensating the complex nonlinear interaction forces among the links of the manipulator. Robust and adaptive control of manipulators are also introduced using the **direct method of Lyapunov** and so-called **passivity-based control**.

Chapter 10: Force Control

In the example robot task above, once the manipulator has reached location A, it must follow the contour S maintaining a constant force normal to the surface. Conceivably, knowing the location of the object and the shape of the contour, one could carry out this task using position control alone. This would be quite difficult to accomplish in practice, however. Since the manipulator itself possesses high rigidity, any errors in position due to uncertainty in the exact location of the surface or tool would give rise to extremely large forces at the end effector that could damage the tool, the surface, or the robot. A better approach is to measure the forces of interaction directly and use a **force control** scheme to accomplish the task. In Chapter 10 we discuss force control and compliance, along with common approaches to force control, namely **hybrid control** and **impedance control**.

Chapter 11: Vision-Based Control

Cameras have become reliable and relatively inexpensive sensors in many robotic applications. Unlike joint sensors, which give information about the internal configuration of the robot, cameras can be used not only to measure the position of the robot but also to locate objects in the robot's workspace. In Chapter 11 we discuss the use of computer vision to determine position and orientation of objects.

In some cases, we may wish to control the motion of the manipulator relative to some target as the end effector moves through free space. Here, force control cannot be used. Instead, we can use computer vision to close the control loop around the vision sensor. This is the topic of Chapter 11. There are several approaches to vision-based control, but we will focus on the method of Image-Based Visual Servo (IBVS). With IBVS, an error measured in image coordinates is directly mapped to a control input that governs the motion of the camera. This method has become very popular in recent years, and it relies on mathematical development analogous to that given in Chapter 4.

Chapter 12: Feedback Linearization

Chapter 12 discusses the method of **nonlinear feedback linearization**. Feedback linearization relies on more advanced tools from differential geometry. We discuss the **Frobenius theorem**, from which we prove necessary and sufficient conditions for a single-input nonlinear system to be equivalent under coordinate transformation and nonlinear feedback to a linear system. We apply the feedback linearization method to the control of elastic-joint robots, for which previous methods such as inverse dynamics cannot be applied.

1.4.2 Underactuated and Mobile Robots

Chapter 13: Underactuated Systems

Chapter 13 treats the control problem for **underactuated** robots, which have fewer actuators than degrees of freedom. Unlike the control of fully-actuated manipulators considered in Chapters 8–11, underactuated robots cannot track arbitrary trajectories and, consequently, the control problems are more challenging for this class of robots than for fully-actuated robots. We discuss the method of partial feedback linearization and introduce the notion of **zero dynamics**, which plays an important role in understanding how to control underactuated robots.

Chapter 14: Mobile Robots

Chapter 14 is devoted to the control of **mobile robots**, which are examples of so-called **nonholonomic systems**. We discuss controllability, stabilizability, and tracking control for this class of systems. Nonholonomic systems require the development of new tools for analysis and control, not treated in the previous chapters. We introduce the notions of **chain-form** systems, and **differential flatness**, which provide methods to transform nonholonomic systems into forms amenable to control design. We discuss two important results, **Chow's theorem** and **Brockett's theorem**, that can be used to determine when a nonholonomic system is controllable or stabilizable, respectively.

Problems

1–1 What are the key features that distinguish robots from other forms of automation such as CNC milling machines?

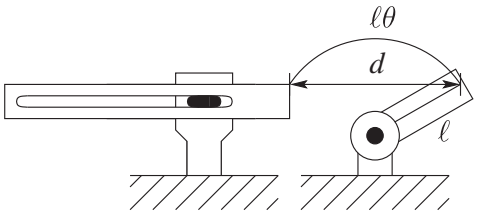


Figure 1.24: Diagram for Problem 1–12, 1–13, and 1–14.

1–2 Briefly define each of the following terms: forward kinematics, inverse kinematics, trajectory planning, workspace, accuracy, repeatability, resolution, joint variable, spherical wrist, end effector.

1–3 What are the main ways to classify robots?

1–4 Make a list of 10 robot applications. For each application discuss which type of manipulator would be best suited; which least suited. Justify your choices in each case.

1–5 List several applications for nonservo robots; for point-to-point robots; for continuous path robots.

1–6 List five applications that a continuous path robot could do that a point-to-point robot could not do.

1–7 List five applications for which computer vision would be useful in robotics.

1–8 List five applications for which either tactile sensing or force feedback control would be useful in robotics.

1–9 Suppose we could close every factory today and reopen them tomorrow fully automated with robots. What would be some of the economic and social consequences of such a development?

1–10 Suppose a law were passed banning all future use of industrial robots. What would be some of the economic and social consequences of such an act?

1–11 Discuss applications for which redundant manipulators would be useful.

1–12 Referring to Figure 1.24, suppose that the tip of a single link travels a distance d between two points. A linear axis would travel the distance

d while a rotational link would travel through an arc length $\ell\theta$ as shown. Using the law of cosines, show that the distance d is given by

$$d = \ell\sqrt{2(1 - \cos\theta)}$$

which is of course less than $\ell\theta$. With 10-bit accuracy, $\ell = 1$ meter, and $\theta = 90°$, what is the resolution of the linear link? of the rotational link?

1–13 For the single-link revolute arm shown in Figure 1.24, if the length of the link is 50 cm and the arm travels 180 degrees, what is the control resolution obtained with an 8-bit encoder?

1–14 Repeat Problem 1–13 assuming that the 8-bit encoder is located on the motor shaft that is connected to the link through a 50:1 gear reduction. Assume perfect gears.

1–15 Why is accuracy generally less than repeatability?

1–16 How could manipulator accuracy be improved using endpoint sensing? What difficulties might endpoint sensing introduce into the control problem?

1–17 Derive Equation (1.11).

1–18 For the two-link manipulator of Figure 1.19 suppose $a_1 = a_2 = 1$.

(a) Find the coordinates of the tool when $\theta_1 = \frac{\pi}{6}$ and $\theta_2 = \frac{\pi}{2}$.

(b) If the joint velocities are constant at $\dot{\theta}_1 = 1$, $\dot{\theta}_2 = 2$, what is the velocity of the tool? What is the instantaneous tool velocity when $\theta_1 = \theta_2 = \frac{\pi}{4}$?

(c) Write a computer program to plot the joint angles as a function of time given the tool locations and velocities as a function of time in Cartesian coordinates.

(d) Suppose we desire that the tool follow a straight line between the points (0,2) and (2,0) at constant speed s. Plot the time history of joint angles.

1–19 For the two-link planar manipulator of Figure 1.19 is it possible for there to be an infinite number of solutions to the inverse kinematic equations? If so, explain how this can occur.

1–20 Explain why it might be desirable to reduce the mass of distal links in a manipulator design. List some ways this can be done. Discuss any possible disadvantages of such designs.

Notes and References

We give below some of the important milestones in the history of modern robotics.

1947 — The first servoed electric powered teleoperator is developed.

1948 — A teleoperator is developed incorporating force feedback.

1949 — Research on numerically controlled milling machine is initiated.

1954 — George Devol designs the first programmable robot

1956 — Joseph Engelberger, a Columbia University physics student, buys the rights to Devol's robot and founds the Unimation Company.

1961 — The first Unimate robot is installed in a Trenton, New Jersey plant of General Motors to tend a die casting machine.

1961 — The first robot incorporating force feedback is developed.

1963 — The first robot vision system is developed.

1971 — The Stanford Arm is developed at Stanford University.

1973 — The first robot programming language (WAVE) is developed at Stanford.

1974 — Cincinnati Milacron introduced the T^3 robot with computer control.

1975 — Unimation Inc. registers its first financial profit.

1976 — The Remote Center Compliance (RCC) device for part insertion in assembly is developed at Draper Labs in Boston.

1976 — Robot arms are used on the Viking I and II space probes and land on Mars.

1978 — Unimation introduces the PUMA robot, based on designs from a General Motors study.

1979 — The SCARA robot design is introduced in Japan.

1981 — The first direct-drive robot is developed at Carnegie-Mellon University.

1982 — Fanuc of Japan and General Motors form GM Fanuc to market robots in North America.

1983 — Adept Technology is founded and successfully markets the direct-drive robot.

1986 — The underwater robot, Jason, of the Woods Hole Oceanographic Institute, explores the wreck of the Titanic, found a year earlier by Dr. Robert Barnard.

1988 — Stäubli Group purchases Unimation from Westinghouse.

1988 — The IEEE Robotics and Automation Society is formed.

1993 — The experimental robot, ROTEX, of the German Aerospace Agency (DLR) was flown aboard the space shuttle Columbia and performed a variety of tasks under both teleoperated and sensor-based offline programmed modes.

1996 — Honda unveils its Humanoid robot; a project begun in secret in 1986.

1997 — The first robot soccer competition, RoboCup-97, is held in Nagoya, Japan and draws 40 teams from around the world.

1997 — The Sojourner mobile robot travels to Mars aboard NASA's Mars PathFinder mission.

2001 — Sony begins to mass produce the first household robot, a robot dog named Aibo.

2001 — The Space Station Remote Manipulation System (SSRMS) is launched in space on board the space shuttle Endeavor to facilitate continued construction of the space station.

2001 — The first telesurgery is performed when surgeons in New York perform a laparoscopic gall bladder removal on a woman in Strasbourg, France.

2001 — Robots are used to search for victims at the World Trade Center site after the September 11th tragedy.

2002 — Honda's Humanoid Robot ASIMO rings the opening bell at the New York Stock Exchange on February 15th.

2004 — The Mars rovers Spirit and Opportunity both landed on the surface of Mars in January of this year. Both rovers far outlived their planned missions of 90 Martian days. Spirit was active until 2010 and Opportunity stayed active until 2018, and holds the record for having driven farther than any off-Earth vehicle in history.

2005 — ROKVISS (Robotic Component Verification on board the International Space Station), the experimental teleoperated arm built by the German Aerospace Center (DLR), undergoes its first tests in space.

2005 — Boston Dynamics releases the quadrupedal robot Big Dog.

2007 — Willow Garage develops the Robot Operating System (ROS).

2011 — Robonaut 2 was launched to the International Space Station.

2017 — A robot called Sophia was granted Saudi Arabian citizenship, becoming the first robot ever to have a nationality.

Part I

THE GEOMETRY OF ROBOTS

Chapter 2

RIGID MOTIONS

A large part of robot kinematics is concerned with establishing various coordinate frames to represent the positions and orientations of rigid objects, and with transformations among these coordinate frames. Indeed, the geometry of three-dimensional space and of rigid motions plays a central role in all aspects of robotic manipulation. In this chapter we study the operations of rotation and translation, and introduce the notion of homogeneous transformations.

Homogeneous transformations combine the operations of rotation and translation into a single matrix multiplication, and are used in Chapter 3 to derive the so-called forward kinematic equations of rigid manipulators. Since we make extensive use of elementary matrix theory, the reader may wish to review Appendix B before beginning this chapter.

We begin by examining representations of points and vectors in a Euclidean space equipped with multiple coordinate frames. Following this, we introduce the concept of a rotation matrix to represent relative orientations among coordinate frames. We then combine these two concepts to build homogeneous transformation matrices, which can be used to simultaneously represent the position and orientation of one coordinate frame relative to another. Furthermore, homogeneous transformation matrices can be used to perform coordinate transformations. Such transformations allow us to represent various quantities in different coordinate frames, a facility that we will often exploit in subsequent chapters.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

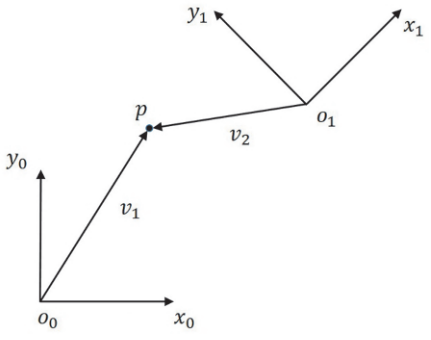


Figure 2.1: Two coordinate frames, a point p, and two vectors v_1 and v_2.

2.1 Representing Positions

Before developing representation schemes for points and vectors, it is instructive to distinguish between the two fundamental approaches to geometric reasoning: the **synthetic** approach and the **analytic** approach. In the former, one reasons directly about geometric entities (e.g., points or lines), while in the latter, one represents these entities using coordinates or equations, and reasoning is performed via algebraic manipulations. The latter approach requires the choice of a reference coordinate frame. A coordinate frame consists of an origin (a single point in space), and two or three orthogonal coordinate axes, for two- and three-dimensional spaces, respectively.

Consider Figure 2.1, which shows two coordinate frames that differ in orientation by an angle of 45°. Using the synthetic approach, without ever assigning coordinates to points or vectors, one can say that x_0 is perpendicular to y_0, or that $v_1 \times v_2$ defines a vector that is perpendicular to the plane containing v_1 and v_2, in this case pointing out of the page.

In robotics, one typically uses analytic reasoning, since robot tasks are often defined using Cartesian coordinates. Of course, in order to assign coordinates it is necessary to specify a reference coordinate frame. Consider again Figure 2.1. We could specify the coordinates of the point p with respect to either frame $o_0x_0y_0$ or frame $o_1x_1y_1$. In the former case we might assign to p the coordinate vector $(5, 6)$ and in the latter case $(-3, 3)$. So that the reference frame will always be clear, we will adopt a notation in which a superscript is used to denote the reference frame. Thus, we write

$$p^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \qquad p^1 = \begin{bmatrix} -3 \\ 3 \end{bmatrix}$$

Geometrically, a point corresponds to a specific location in space. We stress here that p is a geometric entity, a point in space, while both p^0 and p^1 are coordinate vectors that represent the location of this point in space with respect to coordinate frames $o_0x_0y_0$ and $o_1x_1y_1$, respectively. When no confusion can arise, we may simply refer to these coordinate frames as frame 0 and frame 1, respectively.

Since the origin of a coordinate frame is also a point in space, we can assign coordinates that represent the position of the origin of one coordinate frame with respect to another. In Figure 2.1, for example, we may have

$$o_1^0 = \begin{bmatrix} 12 \\ 8 \end{bmatrix}, \qquad o_0^1 = \begin{bmatrix} -16 \\ 3 \end{bmatrix}$$

Thus, o_1^0 specifies the coordinates of the point o_1 relative to frame 0 and o_0^1 specifies the coordinates of the point 0_0 relative to frame 1. In cases where there is only a single coordinate frame, or in which the reference frame is obvious, we will often omit the superscript. This is a slight abuse of notation, and the reader is advised to bear in mind the difference between the geometric entity called p and any particular coordinate vector that is assigned to represent p. The former is independent of the choice of coordinate frames, while the latter obviously depends on the choice of coordinate frames.

While a point corresponds to a specific location in space, a **vector** specifies a direction and a magnitude. Vectors can be used, for example, to represent displacements or forces. Therefore, while the point p is not equivalent to the vector v_1, the displacement from the origin o_0 to the point p is given by the vector v_1. In this text, we will use the term **vector** to refer to what are sometimes called **free vectors**, that is, vectors that are not constrained to be located at a particular point in space. Under this convention, it is clear that points and vectors are not equivalent, since points refer to specific locations in space, but a free vector can be moved to any location in space. Thus, two vectors are equal if they have the same direction and the same magnitude.

When assigning coordinates to vectors, we use the same notational convention that we used when assigning coordinates to points. Thus, v_1 and v_2 are geometric entities that are invariant with respect to the choice of coordinate frames, but the representation by coordinates of these vectors depends directly on the choice of reference coordinate frame. In the example of Figure 2.1, we would obtain

$$v_1^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \qquad v_1^1 = \begin{bmatrix} 8 \\ 2 \end{bmatrix}, \qquad v_2^0 = \begin{bmatrix} -6 \\ 2 \end{bmatrix}, \qquad v_2^1 = \begin{bmatrix} -3 \\ 3 \end{bmatrix}$$

In order to perform algebraic manipulations using coordinates, it is essential that all coordinate vectors be defined with respect to the same coordinate frame. In the case of free vectors, it is enough that they be defined with respect to "parallel" coordinate frames, that is, frames whose respective coordinate axes are parallel, since only their magnitude and direction are specified and not their absolute locations in space.

Using this convention, an expression of the form $v_1^1 + v_2^2$, where v_1^1 and v_2^2 are as in Figure 2.1, is not defined since the frames $o_0x_0y_0$ and $o_1x_1y_1$ are not parallel. Thus, we see a clear need not only for a representation system that allows points to be expressed with respect to various coordinate frames, but also for a mechanism that allows us to transform the coordinates of points from one coordinate frame to another. Such coordinate transformations are the topic for much of the remainder of this chapter.

2.2 Representing Rotations

In order to represent the relative position and orientation of one rigid body with respect to another, we attach coordinate frames to each body, and then specify the geometric relationship between these coordinate frames. In Section 2.1 we saw how one can represent the position of the origin of one frame with respect to another frame. In this section, we address the problem of describing the orientation of one coordinate frame relative to another frame. We begin with the case of rotations in the plane, and then generalize our results to the case of rotations in a three-dimensional space.

2.2.1 Rotation in the Plane

Figure 2.2 shows two coordinate frames, with frame $o_1x_1y_1$ obtained by rotating frame $o_0x_0y_0$ by an angle θ. Perhaps the most obvious way to represent the relative orientation of these two frames is merely to specify the angle of rotation θ. There are two immediate disadvantages to such a representation. First, there is a discontinuity in the mapping from relative orientation to the value of θ in a neighborhood of $\theta = 0$. In particular, for $\theta = 2\pi - \epsilon$, small changes in orientation can produce large changes in the value of θ, for example, a rotation by ϵ causes θ to "wrap around" to zero. Second, this choice of representation does not scale well to the three-dimensional case.

A slightly less obvious way to specify the orientation is to specify the coordinate vectors for the axes of frame $o_1x_1y_1$ with respect to coordinate

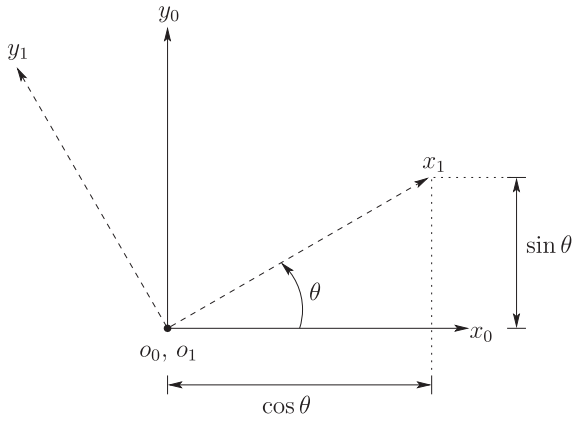


Figure 2.2: Coordinate frame $o_1x_1y_1$ is oriented at an angle θ with respect to $o_0x_0y_0$.

frame $o_0x_0y_0$:

$$R_1^0 = \left[x_1^0 \mid y_1^0\right]$$

in which x_1^0 and y_1^0 are the coordinates in frame $o_0x_0y_0$ of unit vectors x_1 and y_1, respectively.[1] A matrix in this form is called a **rotation matrix**. Rotation matrices have a number of special properties that we will discuss below.

In the two-dimensional case, it is straightforward to compute the entries of this matrix. As illustrated in Figure 2.2,

$$x_1^0 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \qquad y_1^0 = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

which gives

$$R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.1}$$

Note that we have continued to use the notational convention of allowing the superscript to denote the reference frame. Thus, R_1^0 is a matrix whose column vectors are the coordinates of the unit vectors along the axes of frame $o_1x_1y_1$ expressed relative to frame $o_0x_0y_0$.

Although we have derived the entries for R_1^0 in terms of the angle θ, it is not necessary that we do so. An alternative approach, and one that

[1] We will use x_i, y_i to denote both coordinate axes and unit vectors along the coordinate axes depending on the context.

scales nicely to the three-dimensional case, is to build the rotation matrix by projecting the axes of frame $o_1x_1y_1$ onto the coordinate axes of frame $o_0x_0y_0$. Recalling that the dot product of two unit vectors gives the projection of one onto the other, we obtain

$$x_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix}, \qquad y_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix}$$

which can be combined to obtain the rotation matrix

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix}$$

Thus, the columns of R_1^0 specify the direction cosines of the coordinate axes of $o_1x_1y_1$ relative to the coordinate axes of $o_0x_0y_0$. For example, the first column $(x_1 \cdot x_0, x_1 \cdot y_0)$ of R_1^0 specifies the direction of x_1 relative to the frame $o_0x_0y_0$. Note that the right-hand sides of these equations are defined in terms of geometric entities, and not in terms of their coordinates. Examining Figure 2.2 it can be seen that this method of defining the rotation matrix by projection gives the same result as we obtained in Equation (2.1).

If we desired instead to describe the orientation of frame $o_0x_0y_0$ with respect to the frame $o_1x_1y_1$ (that is, if we desired to use the frame $o_1x_1y_1$ as the reference frame), we would construct a rotation matrix of the form

$$R_0^1 = \begin{bmatrix} x_0 \cdot x_1 & y_0 \cdot x_1 \\ x_0 \cdot y_1 & y_0 \cdot y_1 \end{bmatrix}$$

Since the dot product is commutative, (that is, $x_i \cdot y_j = y_j \cdot x_i$), we see that

$$R_0^1 = (R_1^0)^T$$

In a geometric sense, the orientation of $o_0x_0y_0$ with respect to the frame $o_1x_1y_1$ is the inverse of the orientation of $o_1x_1y_1$ with respect to the frame $o_0x_0y_0$. Algebraically, using the fact that coordinate axes are mutually orthogonal, it can readily be seen that

$$(R_1^0)^T = (R_1^0)^{-1}$$

The above relationship implies that $(R_1^0)^T R_1^0 = I$ and it is easily shown that the column vectors of R_1^0 are of unit length and mutually orthogonal (Problem 2–4). Thus R_1^0 is an orthogonal matrix. It also follows from the above that (Problem 2–5) $\det R_1^0 = \pm 1$. If we restrict ourselves to right-handed coordinate frames, as defined in Appendix B, then $\det R_1^0 = +1$ (Problem 2–5).

More generally, these properties extend to higher dimensions, which can be formalized as the so-called **special orthogonal group of order** n.

Definition 2.1. *The* **special orthogonal group of order** n*, denoted* $SO(n)$*, is the set of* $n \times n$ *real-valued matrices*

$$SO(n) = \{R \in \mathbb{R}^{n \times n} \mid R^T R = R R^T = I \text{ and } \det R = +1\} \tag{2.2}$$

Thus, for any $R \in SO(n)$ *the following properties hold*

- $R^T = R^{-1} \in SO(n)$
- *The columns (and therefore the rows) of* R *are mutually orthogonal*
- *Each column (and therefore each row) of* R *is a unit vector*
- $\det R = 1$

The special case, $SO(2)$*, respectively,* $SO(3)$*, is called the* **rotation group** *of order* 2*, respectively* 3*.*

To provide further geometric intuition for the notion of the inverse of a rotation matrix, note that in the two-dimensional case, the inverse of the rotation matrix corresponding to a rotation by angle θ can also be easily computed simply by constructing the rotation matrix for a rotation by the angle $-\theta$:

$$\begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T$$

2.2.2 Rotations in Three Dimensions

The projection technique described above scales nicely to the three-dimensional case. In three dimensions, each axis of the frame $o_1 x_1 y_1 z_1$ is projected onto coordinate frame $o_0 x_0 y_0 z_0$. The resulting rotation matrix $R \in SO(3)$ is given by

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

As was the case for rotation matrices in two dimensions, matrices in this form are orthogonal, with determinant equal to 1 and therefore elements of $SO(3)$.

Example 2.1. *Suppose the frame* $o_1 x_1 y_1 z_1$ *is rotated through an angle* θ *about the* z_0*-axis, and we wish to find the resulting transformation matrix* R_1^0*. By convention, the right hand rule (see Appendix B) defines the positive*

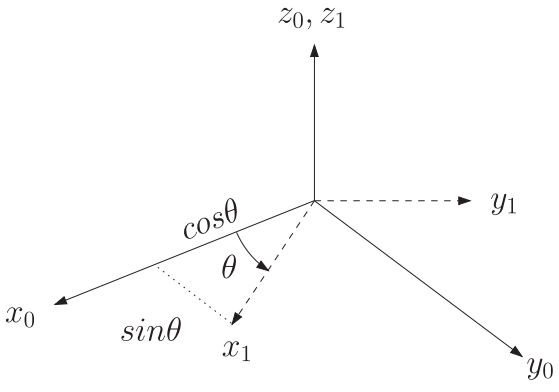


Figure 2.3: Rotation about z_0 by an angle θ.

sense for the angle θ to be such that rotation by θ about the z-axis would advance a right-hand threaded screw along the positive z-axis. From Figure 2.3 we see that

$$x_1 \cdot x_0 = \cos\theta, \qquad y_1 \cdot x_0 = -\sin\theta,$$
$$x_1 \cdot y_0 = \sin\theta, \qquad y_1 \cdot y_0 = \cos\theta$$

and

$$z_0 \cdot z_1 \;=\; 1$$

while all other dot products are zero. Thus, the rotation matrix R_1^0 has a particularly simple form in this case, namely

$$R_1^0 \;=\; \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The rotation matrix given in Equation (2.3) is called a **basic rotation matrix** (about the z-axis). In this case we find it useful to use the more descriptive notation $R_{z,\theta}$ instead of R_1^0 to denote the matrix. It is easy to verify that the basic rotation matrix $R_{z,\theta}$ has the properties

$$R_{z,0} \;=\; I \tag{2.4}$$
$$R_{z,\theta}R_{z,\phi} \;=\; R_{z,\theta+\phi} \tag{2.5}$$

which together imply

$$\left(R_{z,\theta}\right)^{-1} \;=\; R_{z,-\theta} \tag{2.6}$$

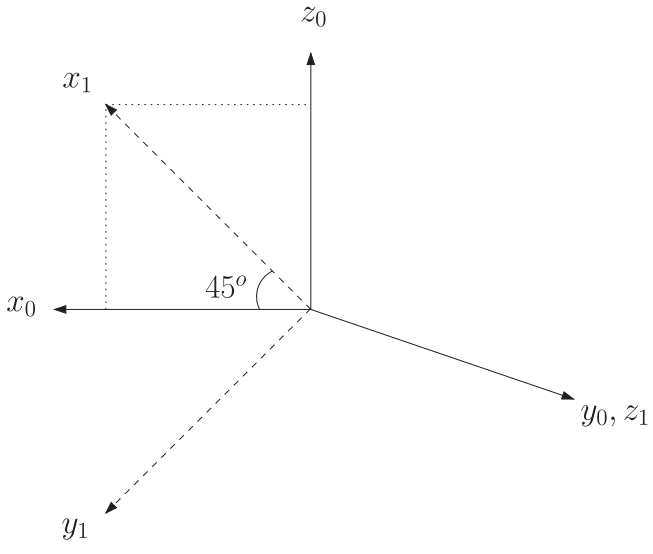


Figure 2.4: Defining the relative orientation of two frames.

Similarly, the basic rotation matrices representing rotations about the x and y-axes are given as (Problem 2–8)

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.7}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.8}$$

which also satisfy properties analogous to Equations (2.4)–(2.6).

Example 2.2. *Consider the frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ shown in Figure 2.4. Projecting the unit vectors x_1, y_1, z_1 onto x_0, y_0, z_0 gives the coordinates of x_1, y_1, z_1 in the $o_0x_0y_0z_0$ frame as*

$$x_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad y_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{-1}{\sqrt{2}} \end{bmatrix}, \quad z_1^0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

The rotation matrix R_1^0 specifying the orientation of $o_1x_1y_1z_1$ relative to

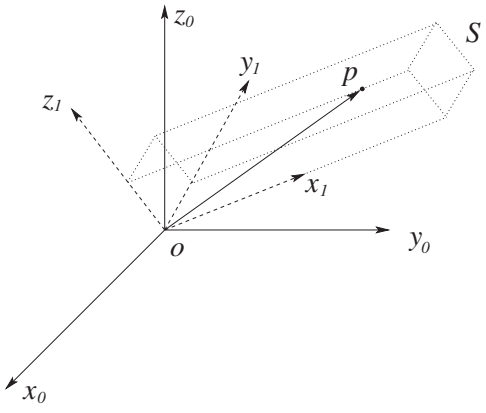


Figure 2.5: Coordinate frame attached to a rigid body.

$o_0x_0y_0z_0$ has these as its column vectors, that is,

$$R_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}$$

2.3 Rotational Transformations

Figure 2.5 shows a rigid object S to which a coordinate frame $o_1x_1y_1z_1$ is attached. Given the coordinates p^1 of the point p (in other words, given the coordinates of p with respect to the frame $o_1x_1y_1z_1$), we wish to determine the coordinates of p relative to a fixed reference frame $o_0x_0y_0z_0$. The coordinates $p^1 = (u, v, w)$ satisfy the equation

$$p^1 = ux_1 + vy_1 + wz_1$$

In a similar way, we can obtain an expression for the coordinates p^0 by projecting the point p onto the coordinate axes of the frame $o_0x_0y_0z_0$, giving

$$p^0 = \begin{bmatrix} p \cdot x_0 \\ p \cdot y_0 \\ p \cdot z_0 \end{bmatrix}$$

Combining these two equations we obtain

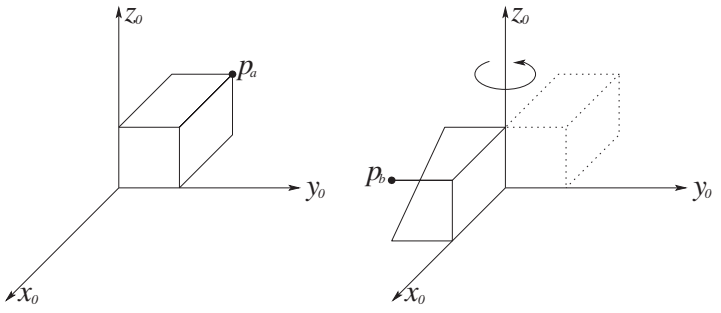


Figure 2.6: The block in (b) is obtained by rotating the block in (a) by π about z_0.

$$
\begin{aligned}
p^0 &= \begin{bmatrix} (ux_1 + vy_1 + wz_1) \cdot x_0 \\ (ux_1 + vy_1 + wz_1) \cdot y_0 \\ (ux_1 + vy_1 + wz_1) \cdot z_0 \end{bmatrix} \\
&= \begin{bmatrix} ux_1 \cdot x_0 + vy_1 \cdot x_0 + wz_1 \cdot x_0 \\ ux_1 \cdot y_0 + vy_1 \cdot y_0 + wz_1 \cdot y_0 \\ ux_1 \cdot z_0 + vy_1 \cdot z_0 + wz_1 \cdot z_0 \end{bmatrix} \\
&= \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}
\end{aligned}
$$

But the matrix in this final equation is merely the rotation matrix R_1^0, which leads to

$$p^0 = R_1^0 p^1 \tag{2.9}$$

Thus, the rotation matrix R_1^0 can be used not only to represent the orientation of coordinate frame $o_1x_1y_1z_1$ with respect to frame $o_0x_0y_0z_0$, but also to transform the coordinates of a point from one frame to another. If a given point is expressed relative to $o_1x_1y_1z_1$ by coordinates p^1, then $R_1^0p^1$ represents the **same point** expressed relative to the frame $o_0x_0y_0z_0$.

We can also use rotation matrices to represent rigid motions that correspond to pure rotation. For example, in Figure 2.6(a) one corner of the block is located at the point p_a in space. Figure 2.6(b) shows the same block after it has been rotated about z_0 by the angle π. The same corner of the block is now located at point p_b in space. It is possible to derive the coordinates for p_b given only the coordinates for p_a and the rotation matrix that

corresponds to the rotation about z_0. To see how this can be accomplished, imagine that a coordinate frame is rigidly attached to the block in Figure 2.6(a), such that it is coincident with the frame $o_0x_0y_0z_0$. After the rotation by π, the block's coordinate frame, which is rigidly attached to the block, is also rotated by π. If we denote this rotated frame by $o_1x_1y_1z_1$, we obtain

$$R_1^0 = R_{z,\pi} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In the local coordinate frame $o_1x_1y_1z_1$, the point p_b has the coordinate representation p_b^1. To obtain its coordinates with respect to frame $o_0x_0y_0z_0$, we merely apply the coordinate transformation Equation (2.9), giving

$$p_b^0 = R_{z,\pi} p_b^1$$

It is important to notice that the local coordinates p_b^1 of the corner of the block do not change as the block rotates, since they are defined in terms of the block's own coordinate frame. Therefore, when the block's frame is aligned with the reference frame $o_0x_0y_0z_0$ (that is, before the rotation is performed), the coordinates p_b^1 equals p_a^0, since before the rotation is performed, the point p_a is coincident with the corner of the block. Therefore, we can substitute p_a^0 into the previous equation to obtain

$$p_b^0 = R_{z,\pi} p_a^0$$

This equation shows how to use a rotation matrix to represent a rotational motion. In particular, if the point p_b is obtained by rotating the point p_a as defined by the rotation matrix R, then the coordinates of p_b with respect to the reference frame are given by

$$p_b^0 = R p_a^0$$

This same approach can be used to rotate vectors with respect to a coordinate frame, as the following example illustrates.

Example 2.3. *The vector v with coordinates $v^0 = (0, 1, 1)$ is rotated about y_0 by $\frac{\pi}{2}$ as shown in Figure 2.7. The resulting vector v_1 is given by*

$$v_1^0 = R_{y,\frac{\pi}{2}} v^0 \tag{2.10}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \tag{2.11}$$

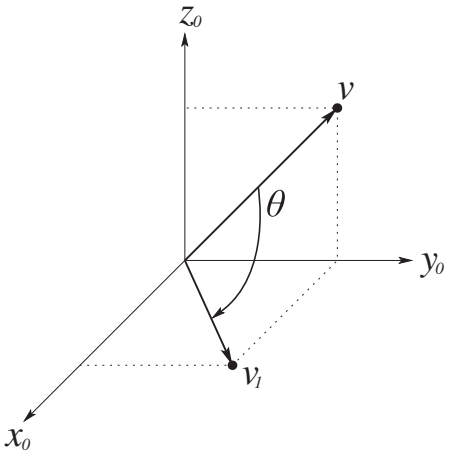


Figure 2.7: Rotating a vector about axis y_0.

Thus, a third interpretation of a rotation matrix R is as an operator acting on vectors in a fixed frame. In other words, instead of relating the coordinates of a fixed vector with respect to two different coordinate frames, Equation (2.10) can represent the coordinates in $o_0x_0y_0z_0$ of a vector v_1 that is obtained from a vector v by a given rotation.

As we have seen, rotation matrices can serve several roles. A rotation matrix, either $R \in SO(3)$ or $R \in SO(2)$, can be interpreted in three distinct ways:

1. It represents a coordinate transformation relating the coordinates of a point p in two different frames.

2. It gives the orientation of a transformed coordinate frame with respect to a fixed coordinate frame.

3. It is an operator taking a vector and rotating it to give a new vector in the same coordinate frame.

The particular interpretation of a given rotation matrix R should be made clear by the context.

Similarity Transformations

A coordinate frame is defined by a set of **basis vectors**, for example, unit vectors along the three coordinate axes. This means that a rotation matrix,

as a coordinate transformation, can also be viewed as defining a change of basis from one frame to another. The matrix representation of a general linear transformation is transformed from one frame to another using a so-called **similarity transformation**. For example, if A is the matrix representation of a given linear transformation in $o_0x_0y_0z_0$ and B is the representation of the same linear transformation in $o_1x_1y_1z_1$ then A and B are related as

$$B = (R_1^0)^{-1} A R_1^0 \tag{2.12}$$

where R_1^0 is the coordinate transformation between frames $o_1x_1y_1z_1$ and $o_0x_0y_0z_0$. In particular, if A itself is a rotation, then so is B, and thus the use of similarity transformations allows us to express the same rotation easily with respect to different frames.

Example 2.4. *Henceforth, whenever convenient we use the shorthand notation $c_\theta = \cos\theta$, $s_\theta = \sin\theta$ for trigonometric functions. Suppose frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ are related by the rotation*

$$R_1^0 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

If $A = R_{z,\theta}$ relative to the frame $o_0x_0y_0z_0$, then, relative to frame $o_1x_1y_1z_1$ we have

$$B = (R_1^0)^{-1} A R_1^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & s_\theta \\ 0 & -s_\theta & c_\theta \end{bmatrix}$$

In other words, B is a rotation about the z_0-axis but expressed relative to the frame $o_1x_1y_1z_1$. This notion will be useful below and in later sections.

2.4 Composition of Rotations

In this section we discuss the composition of rotations. It is important for subsequent chapters that the reader understand the material in this section thoroughly before moving on.

2.4.1 Rotation with Respect to the Current Frame

Recall that the matrix R_1^0 in Equation (2.9) represents a rotational transformation between the frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$. Suppose we now add a

third coordinate frame $o_2x_2y_2z_2$ related to the frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ by rotational transformations. A given point p can then be represented by coordinates specified with respect to any of these three frames: p^0, p^1, and p^2. The relationship among these representations of p is

$$p^0 = R_1^0 p^1 \tag{2.13}$$
$$p^1 = R_2^1 p^2 \tag{2.14}$$
$$p^0 = R_2^0 p^2 \tag{2.15}$$

where each R_j^i is a rotation matrix. Substituting Equation (2.14) into Equation (2.13) gives

$$p^0 = R_1^0 R_2^1 p^2 \tag{2.16}$$

Note that R_1^0 and R_2^0 represent rotations relative to the frame $o_0x_0y_0z_0$ while R_2^1 represents a rotation relative to the frame $o_1x_1y_1z_1$. Comparing Equations (2.15) and (2.16) we can immediately infer

$$R_2^0 = R_1^0 R_2^1 \tag{2.17}$$

Equation (2.17) is the composition law for rotational transformations. It states that, in order to transform the coordinates of a point p from its representation p^2 in the frame $o_2x_2y_2z_2$ to its representation p^0 in the frame $o_0x_0y_0z_0$, we may first transform to its coordinates p^1 in the frame $o_1x_1y_1z_1$ using R_2^1 and then transform p^1 to p^0 using R_1^0.

We may also interpret Equation (2.17) as follows. Suppose that initially all three of the coordinate frames coincide. We first rotate the frame $o_1x_1y_1z_1$ relative to $o_0x_0y_0z_0$ according to the transformation R_1^0. Then, with the frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ coincident, we rotate $o_2x_2y_2z_2$ relative to $o_1x_1y_1z_1$ according to the transformation R_2^1. The resulting frame, $o_2x_2y_2z_2$ has orientation with respect to $o_0x_0y_0z_0$ given by $R_1^0R_2^1$. We call the frame relative to which the rotation occurs the **current frame**.

Example 2.5. *Suppose a rotation matrix R represents a rotation of angle ϕ about the current y-axis followed by a rotation of angle θ about the current z-axis as shown in Figure 2.8. Then the matrix R is given by*

$$\begin{aligned} R &= R_{y,\phi}R_{z,\theta} \qquad (2.18)\\ &= \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta & -c_\phi s_\theta & s_\phi \\ s_\theta & c_\theta & 0 \\ -s_\phi c_\theta & s_\phi s_\theta & c_\phi \end{bmatrix} \end{aligned}$$

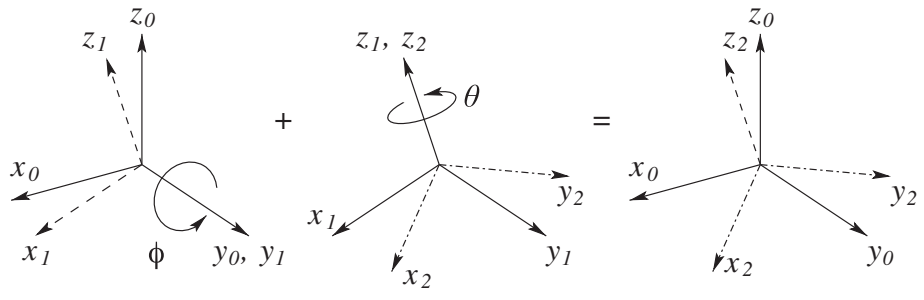


Figure 2.8: Composition of rotations about current axes.

It is important to remember that the order in which a sequence of rotations is performed, and consequently the order in which the rotation matrices are multiplied together, is crucial. The reason is that rotation, unlike position, is not a vector quantity and so rotational transformations do not commute in general.

Example 2.6. *Suppose that the above rotations are performed in the reverse order, that is, first a rotation about the current z-axis followed by a rotation about the current y-axis. Then the resulting rotation matrix is given by*

$$\begin{aligned} R' &= R_{z,\theta} R_{y,\phi} \\ &= \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} \\ &= \begin{bmatrix} c_\theta c_\phi & -s_\theta & c_\theta s_\phi \\ s_\theta c_\phi & c_\theta & s_\theta s_\phi \\ -s_\phi & 0 & c_\phi \end{bmatrix} \end{aligned} \tag{2.19}$$

Comparing Equations (2.18) and (2.19) we see that $R \neq R'$.

2.4.2 Rotation with Respect to the Fixed Frame

Many times it is desired to perform a sequence of rotations, each about a given fixed coordinate frame, rather than about successive current frames. For example we may wish to perform a rotation about x_0 followed by a rotation about y_0 (and not y_1!). We will refer to $o_0x_0y_0z_0$ as the **fixed frame**. In this case the composition law given by Equation (2.17) is not valid. It turns out that the correct composition law in this case is simply to multiply the successive rotation matrices **in the reverse order** from that given by Equation (2.17). Note that the rotations themselves are not

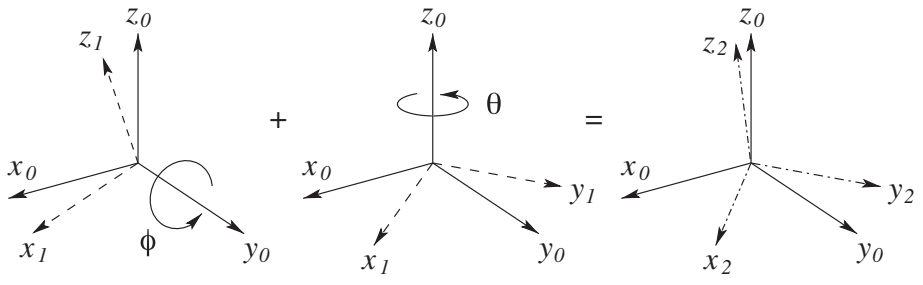


Figure 2.9: Composition of rotations about fixed axes.

performed in reverse order. Rather they are performed about the fixed frame instead of about the current frame.

To see this, suppose we have two frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ related by the rotational transformation R_1^0. If $R \in SO(3)$ represents a rotation relative to $o_0x_0y_0z_0$, we know from Section 2.3 that the representation for R in the **current** frame $o_1x_1y_1z_1$ is given by $(R_1^0)^{-1}RR_1^0$. Therefore, applying the composition law for rotations about the current axis yields

$$R_2^0 = R_1^0\left[(R_1^0)^{-1}RR_1^0\right] = RR_1^0 \tag{2.20}$$

Thus, when a rotation R is performed with respect to the world coordinate frame, the current rotation matrix is **premultiplied** by R to obtain the desired rotation matrix.

Example 2.7 (Rotations about Fixed Axes)**.** *Referring to Figure 2.9, suppose that a rotation matrix R represents a rotation of angle ϕ about y_0 followed by a rotation of angle θ about the fixed z_0. The second rotation about the fixed axis is given by $R_{y,-\phi}R_{z,\theta}R_{y,\phi}$, which is the basic rotation about the z-axis expressed relative to the frame $o_1x_1y_1z_1$ using a similarity transformation. Therefore, the composition rule for rotational transformations gives us*

$$R = R_{y,\phi}\left[R_{y,-\phi}R_{z,\theta}R_{y,\phi}\right] = R_{z,\theta}R_{y,\phi} \tag{2.21}$$

It is not necessary to remember the above derivation, only to note by comparing Equation (2.21) with Equation (2.18) that we obtain the same basic rotation matrices, but in the reverse order.

2.4.3 Rules for Composition of Rotations

We can summarize the rule of composition of rotational transformations by the following recipe. Given a fixed frame $o_0x_0y_0z_0$ and a current frame

$o_1x_1y_1z_1$, together with rotation matrix R_1^0 relating them, if a third frame $o_2x_2y_2z_2$ is obtained by a rotation R performed relative to the **current frame** then **postmultiply** R_1^0 by $R = R_2^1$ to obtain

$$R_2^0 = R_1^0 R_2^1 \tag{2.22}$$

If the second rotation is to be performed relative to the **fixed frame** then it is both confusing and inappropriate to use the notation R_2^1 to represent this rotation. Therefore, if we represent the rotation by R, we **premultiply** R_1^0 by R to obtain

$$R_2^0 = RR_1^0 \tag{2.23}$$

In each case R_2^0 represents the transformation between the frames $o_0x_0y_0z_0$ and $o_2x_2y_2z_2$. The frame $o_2x_2y_2z_2$ that results from Equation (2.22) will be different from that resulting from Equation (2.23).

Using the above rule for composition of rotations, it is an easy matter to determine the result of multiple sequential rotational transformations.

Example 2.8. *Suppose R is defined by the following sequence of basic rotations in the order specified:*

1. *A rotation of θ about the current x-axis*
2. *A rotation of ϕ about the current z-axis*
3. *A rotation of α about the fixed z-axis*
4. *A rotation of β about the current y-axis*
5. *A rotation of δ about the fixed x-axis*

In order to determine the cumulative effect of these rotations we simply begin with the first rotation $R_{x,\theta}$ and pre- or postmultiply as the case may be to obtain

$$R = R_{x,\delta}R_{z,\alpha}R_{x,\theta}R_{z,\phi}R_{y,\beta} \tag{2.24}$$

2.5 Parameterizations of Rotations

The nine elements r_{ij} in a general rotational transformation $R \in SO(3)$ are not independent quantities. Indeed, a rigid body possesses at most three rotational degrees of freedom, and thus at most three quantities are

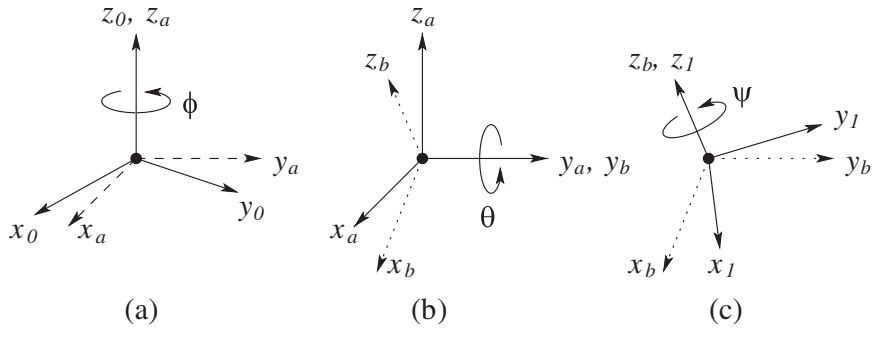


Figure 2.10: Euler angle representation.

required to specify its orientation. This can be easily seen by examining the constraints that govern the matrices in $SO(3)$:

$$\sum_i r_{ij}^2 = 1, \quad j \in \{1, 2, 3\} \tag{2.25}$$

$$r_{1i}r_{1j} + r_{2i}r_{2j} + r_{3i}r_{3j} = 0, \quad i \neq j \tag{2.26}$$

Equation (2.25) follows from the fact that the columns of a rotation matrix are unit vectors, and Equation (2.26) follows from the fact that columns of a rotation matrix are mutually orthogonal. Together, these constraints define six independent equations with nine unknowns, which implies that there are three free variables.

In this section we derive three ways in which an arbitrary rotation can be represented using only three independent quantities: the **Euler angle** representation, the **roll-pitch-yaw** representation, and the **axis-angle** representation.

2.5.1 Euler Angles

A common method of specifying a rotation matrix in terms of three independent quantities is to use the so-called **Euler angles**. Consider the fixed coordinate frame $o_0x_0y_0z_0$ and the rotated frame $o_1x_1y_1z_1$ shown in Figure 2.10. We can specify the orientation of the frame $o_1x_1y_1z_1$ relative to the frame $o_0x_0y_0z_0$ by three angles (ϕ, θ, ψ), known as Euler angles, and obtained by three successive rotations as follows. First rotate about the z-axis by the angle ϕ. Next rotate about the current y-axis by the angle θ. Finally rotate about the current z-axis by the angle ψ. In Figure 2.10, frame $o_ax_ay_az_a$ represents the new coordinate frame after the rotation by ϕ, frame $o_bx_by_bz_b$ represents the new coordinate frame after the rotation by

θ, and frame $o_1x_1y_1z_1$ represents the final frame, after the rotation by ψ. Frames $o_ax_ay_az_a$ and $o_bx_by_bz_b$ are shown in the figure only to help visualize the rotations.

In terms of the basic rotation matrices the resulting rotational transformation can be generated as the product

$$\begin{aligned} R_{ZYZ} &= R_{z,\phi}R_{y,\theta}R_{z,\psi} \\ &= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \end{aligned} \tag{2.27}$$

The matrix R_{ZYZ} in Equation (2.27) is called the ZYZ–**Euler angle transformation**.

The more important and more difficult problem is to determine for a particular $R = (r_{ij})$ the set of Euler angles ϕ, θ, and ψ, that satisfy

$$R = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{2.28}$$

for a matrix $R \in SO(3)$. This problem will be important later when we address the inverse kinematics problem for manipulators in Chapter 5.

To find a solution for this problem we break it down into two cases. First, suppose that not both of r_{13}, r_{23} are zero. Then from Equation (2.27) we deduce that $s_\theta \neq 0$, and hence that not both of r_{31}, r_{32} are zero. If not both r_{13} and r_{23} are zero, then $r_{33} \neq \pm 1$, and we have $c_\theta = r_{33}$, $s_\theta = \pm\sqrt{1 - r_{33}^2}$ so

$$\theta = \text{Atan2}\left(r_{33}, \sqrt{1 - r_{33}^2}\right) \tag{2.29}$$

or

$$\theta = \text{Atan2}\left(r_{33}, -\sqrt{1 - r_{33}^2}\right) \tag{2.30}$$

where the function Atan2 is the **two-argument arctangent function** defined in Appendix A.

If we choose the value for θ given by Equation (2.29), then $s_\theta > 0$, and

$$\phi = \text{Atan2}(r_{13}, r_{23}) \tag{2.31}$$

$$\psi = \text{Atan2}(-r_{31}, r_{32}) \tag{2.32}$$

If we choose the value for θ given by Equation (2.30), then $s_\theta < 0$, and

$$\begin{aligned} \phi &= \text{Atan2}(-r_{13}, -r_{23}) && (2.33) \\ \psi &= \text{Atan2}(r_{31}, -r_{32}) && (2.34) \end{aligned}$$

Thus, there are two solutions depending on the sign chosen for θ.

If $r_{13} = r_{23} = 0$, then the fact that R is orthogonal implies that $r_{33} = \pm 1$, and that $r_{31} = r_{32} = 0$. Thus, R has the form

$$R = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \quad (2.35)$$

If $r_{33} = 1$, then $c_\theta = 1$ and $s_\theta = 0$, so that $\theta = 0$. In this case, Equation (2.27) becomes

$$\begin{bmatrix} c_\phi c_\psi - s_\phi s_\psi & -c_\phi s_\psi - s_\phi c_\psi & 0 \\ s_\phi c_\psi + c_\phi s_\psi & -s_\phi s_\psi + c_\phi c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\phi+\psi} & -s_{\phi+\psi} & 0 \\ s_{\phi+\psi} & c_{\phi+\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus, the sum $\phi + \psi$ can be determined as

$$\phi + \psi = \text{Atan2}(r_{11}, r_{21}) = \text{Atan2}(r_{11}, -r_{12}) \quad (2.36)$$

Since only the sum $\phi + \psi$ can be determined in this case, there are infinitely many solutions. In this case, we may take $\phi = 0$ by convention. If $r_{33} = -1$, then $c_\theta = -1$ and $s_\theta = 0$, so that $\theta = \pi$. In this case Equation (2.27) becomes

$$\begin{bmatrix} -c_{\phi-\psi} & -s_{\phi-\psi} & 0 \\ s_{\phi-\psi} & c_{\phi-\psi} & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (2.37)$$

The solution is thus

$$\phi - \psi = \text{Atan2}(-r_{11}, -r_{12}) \quad (2.38)$$

As before there are infinitely many solutions.

2.5.2 Roll, Pitch, Yaw Angles

A rotation matrix R can also be described as a product of successive rotations about the principal coordinate axes x_0, y_0, and z_0 taken in a specific

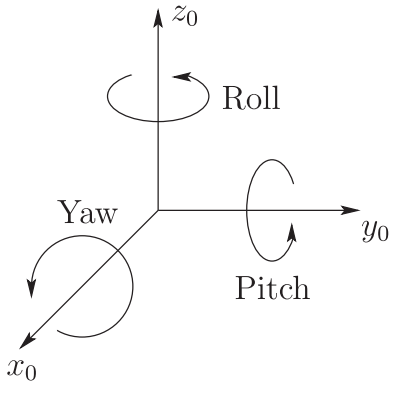


Figure 2.11: Roll, pitch, and yaw angles.

order. These rotations define the **roll**, **pitch**, and **yaw** angles, which we shall also denote ϕ, θ, ψ, and which are shown in Figure 2.11.

We specify the order of rotation as $x - y - z$, in other words, first a yaw about x_0 through an angle ψ, then pitch about the y_0 by an angle θ, and finally roll about the z_0 by an angle ϕ.[2] Since the successive rotations are relative to the fixed frame, the resulting transformation matrix is given by

$$\begin{aligned} R &= R_{z,\phi} R_{y,\theta} R_{x,\psi} \\ &= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \end{aligned} \tag{2.39}$$

Of course, instead of yaw-pitch-roll relative to the fixed frames we could also interpret the above transformation as roll-pitch-yaw, in that order, each taken with respect to the current frame. The end result is the same matrix as in Equation (2.39).

The three angles ϕ, θ, and ψ can be obtained for a given rotation matrix using a method that is similar to that used to derive the Euler angles above.

[2]It should be noted that other conventions exist for naming the roll, pitch, and yaw angles.

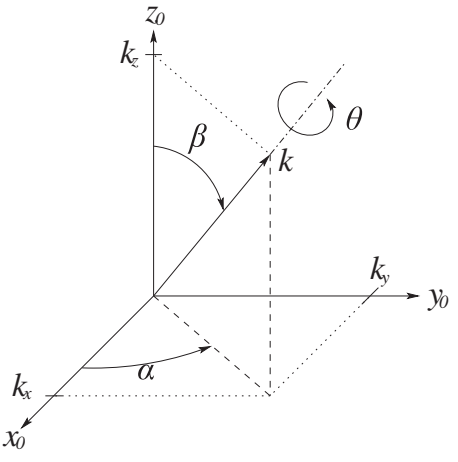


Figure 2.12: Rotation about an arbitrary axis.

2.5.3 Axis-Angle Representation

Rotations are not always performed about the principal coordinate axes. We are often interested in a rotation about an arbitrary axis in space. This provides both a convenient way to describe rotations, and an alternative parameterization for rotation matrices. Let $k = (k_x, k_y, k_z)$, expressed in the frame $o_0x_0y_0z_0$, be a unit vector defining an axis. We wish to derive the rotation matrix $R_{k,\theta}$ representing a rotation of θ about this axis.

There are several ways in which the matrix $R_{k,\theta}$ can be derived. One approach is to note that the rotational transformation $R = R_{z,\alpha}R_{y,\beta}$ will bring the world z-axis into alignment with the vector k. Therefore, a rotation about the axis k can be computed using a similarity transformation as

$$R_{k,\theta} = RR_{z,\theta}R^{-1} \tag{2.40}$$

$$= R_{z,\alpha}R_{y,\beta}R_{z,\theta}R_{y,-\beta}R_{z,-\alpha} \tag{2.41}$$

From Figure 2.12 we see that

$$\sin\alpha = \frac{k_y}{\sqrt{k_x^2 + k_y^2}}, \quad \cos\alpha = \frac{k_x}{\sqrt{k_x^2 + k_y^2}} \tag{2.42}$$

$$\sin\beta = \sqrt{k_x^2 + k_y^2}, \quad \cos\beta = k_z \tag{2.43}$$

Note that the final two equations follow from the fact that k is a unit vector. Substituting Equations (2.42) and (2.43) into Equation (2.41), we obtain

after some lengthy calculation (Problem 2–17)

$$R_{k,\theta} = \left[\begin{array}{c|c|c} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{array}\right] \tag{2.44}$$

where $v_\theta = \text{ vers } \theta = 1 - c_\theta$.

In fact, any rotation matrix $R \in S0(3)$ can be represented by a single rotation about a suitable axis in space by a suitable angle,

$$R \;=\; R_{k,\theta} \tag{2.45}$$

where k is a unit vector defining the axis of rotation, and θ is the angle of rotation about k. The pair (k, θ) is called the **axis-angle representation** of R. Given an arbitrary rotation matrix R with components r_{ij}, the equivalent angle θ and equivalent axis k are given by the expressions

$$\theta \;=\; \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

and

$$k \;=\; \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{2.46}$$

These equations can be obtained by direct manipulation of the entries of the matrix given in Equation (2.44). The axis-angle representation is not unique since a rotation of $-\theta$ about $-k$ is the same as a rotation of θ about k, that is,

$$R_{k,\theta} \;=\; R_{-k,-\theta} \tag{2.47}$$

If $\theta = 0$ then R is the identity matrix and the axis of rotation is undefined.

Example 2.9. *Suppose R is generated by a rotation of 90° about z_0 followed by a rotation of 30° about y_0 followed by a rotation of 60° about x_0. Then*

$$\begin{aligned} R &= R_{x,60} R_{y,30} R_{z,90} \qquad (2.48) \\ &= \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{4} & -\frac{3}{4} \\ \frac{\sqrt{3}}{2} & \frac{1}{4} & \frac{\sqrt{3}}{4} \end{bmatrix} \end{aligned}$$

We see that $Tr(R) = 0$ *and hence the equivalent angle is given by Equation (2.46) as*

$$\theta = \cos^{-1}\left(-\frac{1}{2}\right) = 120^\circ \tag{2.49}$$

The equivalent axis is given from Equation (2.46) as

$$k = \left(\frac{1}{\sqrt{3}}, \frac{1}{2\sqrt{3}} - \frac{1}{2}, \frac{1}{2\sqrt{3}} + \frac{1}{2}\right) \tag{2.50}$$

The above axis-angle representation characterizes a given rotation by four quantities, namely the three components of the equivalent axis k and the equivalent angle θ. However, since the equivalent axis k is given as a unit vector only two of its components are independent. The third is constrained by the condition that k is of unit length. Therefore, only three independent quantities are required in this representation of a rotation R. We can represent the equivalent axis-angle by a single vector r as

$$r = (r_x, r_y, r_z) = (\theta k_x, \theta k_y, \theta k_z) \tag{2.51}$$

Note, since k is a unit vector, that the length of the vector r is the equivalent angle θ and the direction of r is the equivalent axis k.

One should be careful to note that the representation in Equation (2.51) does not mean that two axis-angle representations may be combined using standard rules of vector algebra, as doing so would imply that rotations commute which, as we have seen, is not true in general.

2.5.4 Exponential Coordinates

In this section we introduce the so-called **exponential coordinates** and give an alternate description of the axis-angle transformation (2.44). We showed above in Section 2.5.3 that any rotation matrix $R \in SO(3)$ can be expressed as an axis-angle matrix $R_{k,\theta}$ using Equation (2.44). The components of the vector $k\theta \in \mathbb{R}^3$ are called **exponential coordinates** of R.

To see why this terminology is used, we first recall from Appendix B the definition of $so(3)$ as the set of 3×3 **skew-symmetric** matrices S satisfying

$$S^T + S = 0 \tag{2.52}$$

For $k = (k_x, k_y, k_z) \in \mathbb{R}^3$ let $S(k)$ be the skew-symmetric matrix

$$S(k) = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \tag{2.53}$$

and let $e^{S(k)\theta}$ be the matrix exponential as defined in Appendix B

$$e^{S(k)\theta} = I + S(k)\theta + \frac{1}{2}S^2(k)\theta^2 + \frac{1}{3!}S^3(k)\theta^3 + \cdots \tag{2.54}$$

Then we have the following proposition, which gives an important relationship between $SO(3)$ and $so(3)$.

Proposition 2.1. *The matrix $e^{S(k)\theta}$ is an element of $SO(3)$ for any $S(k) \in so(3)$ and, conversely, every element of $SO(3)$ can be expressed as the exponential of an element of $so(3)$.*

Proof: To show that the matrix $e^{S(k)\theta}$ is in $SO(3)$ we need to show that $e^{S(k)\theta}$ is an orthogonal matrix with determinant equal to $+1$. To show this we rely on the following properties that hold for any $n \times n$ matrices A and B

1. $e^{A^T} = (e^A)^T$
2. If the $n \times n$ matrices A and B commute, i.e., $AB = BA$, then $e^A e^B = e^{(A+B)}$
3. The determinant $\det(e^A) = e^{tr(A)}$, where $tr(A)$ is the trace of A.

The first two properties above can be shown by direct calculation using the series expansion (2.54) for e^A. The third property follows from the **Jacobi Identity** (Appendix B). Now, since $S^T = -S$, if S is skew-symmetric, then S and S^T clearly commute. Therefore, with $S = S(k\theta) \in so(3)$, we have

$$e^S(e^S)^T = e^S e^{S^T} = e^{S+S^T} = e^0 = I \tag{2.55}$$

which shows that $e^{S(k\theta)}$ is an orthogonal matrix. Also

$$\det(e^S) = e^{tr(S)} = 1 \tag{2.56}$$

since the trace of a skew-symmetric matrix is zero. Thus $e^{S(k\theta)} \in SO(3)$ for $S(k\theta) \in so(3)$.

The converse, namely, that every element of $SO(3)$ is the exponential of an element of $so(3)$, follows from the axis-angle representation of R and **Rodrigues' formula**, which we derive next.

Rodrigues' Formula

Given the skew-symmetric matrix $S(k)$ it is easy to show that $S^3(k) = -S(k)$, from which it follows that $S^4(k) = -S^2(k)$, etc. Thus the series

expansion for $e^{S(k)\theta}$ reduces to

$$\begin{aligned} e^{S(k)\theta} &= I + S(k)\theta + \frac{1}{2}S^2(k)\theta^2 + \frac{1}{3!}S^3(k)\theta^3 + \cdots \\ &= I + S(k)(\theta - \frac{1}{3!}\theta^3 + \cdots) - S^2(k)(\frac{1}{2}\theta^2 - \frac{1}{4!}\theta^4 + \cdots) \\ &= I + \sin(\theta)S(k) + (1 - \cos(\theta))S^2(k) \end{aligned}$$

the latter equality following from the series expansion of the sine and cosine functions. The expression

$$e^{S(k)\theta} = I + \sin(\theta)S(k) + (1 - \cos(\theta))S^2(k) \tag{2.57}$$

is known as **Rodrigues' formula**. It can be shown by direct calculation that the angle-axis representation for $R_{k,\theta}$ given by Equation (2.44) and Rodrigues' formula in Equation (2.57) are identical.

Remark 2.1. *The above results show that the matrix exponential function defines a one-to-one mapping from so*(3) *onto SO*(3)*. Mathematically, so*(3) *is a* **Lie algebra** *and SO*(3) *is a* **Lie group***.*

2.6 Rigid Motions

We have now seen how to represent both positions and orientations. We combine these two concepts in this section to define a **rigid motion** and, in the next section, we derive an efficient matrix representation for rigid motions using the notion of homogeneous transformation.

Definition 2.2. *A rigid motion is an ordered pair* (d, R) *where* $d \in \mathbb{R}^3$ *and* $R \in SO(3)$*. The group of all rigid motions is known as the* **special Euclidean group** *and is denoted by* $SE(3)$*. We see then that* $SE(3) = \mathbb{R}^3 \times SO(3)$*.*

A rigid motion is a pure translation together with a pure rotation.[3] Let R_1^0 be the rotation matrix that specifies the orientation of frame $o_1x_1y_1z_1$ with respect to $o_0x_0y_0z_0$, and d be the vector from the origin of frame $o_0x_0y_0z_0$ to the origin of frame $o_1x_1y_1z_1$. Suppose the point p is rigidly attached to coordinate frame $o_1x_1y_1z_1$, with local coordinates p^1. We can express the coordinates of p with respect to frame $o_0x_0y_0z_0$ using

$$p^0 = R_1^0 p^1 + d^0 \tag{2.58}$$

[3]The definition of rigid motion is sometimes broadened to include **reflections**, which correspond to $\det R = -1$. We will always assume in this text that $\det R = +1$ so that $R \in SO(3)$.

Now consider three coordinate frames $o_0x_0y_0z_0$, $o_1x_1y_1z_1$, and $o_2x_2y_2z_2$. Let d_1 be the vector from the origin of $o_0x_0y_0z_0$ to the origin of $o_1x_1y_1z_1$ and d_2 be the vector from the origin of $o_1x_1y_1z_1$ to the origin of $o_2x_2y_2z_2$. If the point p is attached to frame $o_2x_2y_2z_2$ with local coordinates p^2, we can compute its coordinates relative to frame $o_0x_0y_0z_0$ using

$$p^1 = R_2^1 p^2 + d_2^1 \tag{2.59}$$

and

$$p^0 = R_1^0 p^1 + d_1^0 \tag{2.60}$$

The composition of these two equations defines a third rigid motion, which we can describe by substituting the expression for p^1 from Equation (2.59) into Equation (2.60)

$$p^0 = R_1^0 R_2^1 p^2 + R_1^0 d_2^1 + d_1^0 \tag{2.61}$$

Since the relationship between p^0 and p^2 is also a rigid motion, we can equally describe it as

$$p^0 = R_2^0 p^2 + d_2^0 \tag{2.62}$$

Comparing Equations (2.61) and (2.62) we have the relationships

$$R_2^0 = R_1^0 R_2^1 \tag{2.63}$$
$$d_2^0 = d_1^0 + R_1^0 d_2^1 \tag{2.64}$$

Equation (2.63) shows that the orientation transformations can simply be multiplied together and Equation (2.64) shows that the vector from the origin o_0 to the origin o_2 has coordinates given by the sum of d_1^0 (the vector from o_0 to o_1 expressed with respect to $o_0x_0y_0z_0$) and $R_1^0 d_2^1$ (the vector from o_1 to o_2, expressed in the orientation of the coordinate frame $o_0x_0y_0z_0$).

2.6.1 Homogeneous Transformations

One can easily see that the calculation leading to Equation (2.61) would quickly become intractable if a long sequence of rigid motions were considered. In this section we show how rigid motions can be represented in matrix form so that composition of rigid motions can be reduced to matrix multiplication as was the case for composition of rotations.

In fact, a comparison of Equations (2.63) and (2.64) with the matrix identity

$$\begin{bmatrix} R_1^0 & d_1^0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2^1 & d_2^1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1^0 R_2^1 & R_1^0 d_2^1 + d_1^0 \\ 0 & 1 \end{bmatrix} \tag{2.65}$$

where 0 denotes the row vector $(0, 0, 0)$, shows that the rigid motions can be represented by the set of matrices of the form

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, \quad R \in SO(3), \quad d \in \mathbb{R}^3 \tag{2.66}$$

Transformation matrices of the form given in Equation (2.66) are called **homogeneous transformations**. A homogeneous transformation is therefore nothing more than a matrix representation of a rigid motion and we will use $SE(3)$ interchangeably to represent both the set of rigid motions and the set of all 4×4 matrices H of the form given in Equation (2.66).

Using the fact that R is orthogonal it is an easy exercise to show that the inverse transformation H^{-1} is given by

$$H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix} \tag{2.67}$$

In order to represent the transformation given in Equation (2.58) by a matrix multiplication, we must augment the vectors p^0 and p^1 by the addition of a fourth component of 1 as follows,

$$P^0 = \begin{bmatrix} p^0 \\ 1 \end{bmatrix} \tag{2.68}$$

$$P^1 = \begin{bmatrix} p^1 \\ 1 \end{bmatrix} \tag{2.69}$$

The vectors P^0 and P^1 are known as **homogeneous representations** of the vectors p^0 and p^1, respectively. It can now be seen directly that the transformation given in Equation (2.58) is equivalent to the (homogeneous) matrix equation

$$P^0 = H_1^0 P^1 \tag{2.70}$$

A set of **basic homogeneous transformations** generating $SE(3)$ is given by

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Rot}_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.71}$$

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Rot}_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.72}$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Rot}_{z,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.73}$$

for translation and rotation about the x, y, z-axes, respectively.

The most general homogeneous transformation that we will consider may be written now as

$$H_1^0 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.74}$$

In the above equation $n = (n_x, n_y, n_z)$ is a vector representing the direction of x_1 in the $o_0x_0y_0z_0$ frame, $s = (s_x, s_y, s_z)$ represents the direction of y_1, and $a = (a_x, a_y, a_z)$ represents the direction of z_1. The vector $d = (d_x, d_y, d_z)$ represents the vector from the origin o_0 to the origin o_1 expressed in the frame $o_0x_0y_0z_0$. The rationale behind the choice of letters n, s, and a is explained in Chapter 3.

The same interpretation regarding composition and ordering of transformations holds for 4×4 homogeneous transformations as for 3×3 rotations. Given a homogeneous transformation H_1^0 relating two frames, if a second rigid motion, represented by $H \in SE(3)$ is performed relative to the current frame, then

$$H_2^0 = H_1^0 H$$

whereas if the second rigid motion is performed relative to the fixed frame, then

$$H_2^0 = H H_1^0$$

Example 2.10. *The homogeneous transformation matrix H that represents a rotation by angle α about the current x-axis followed by a translation of b units along the current x-axis, followed by a translation of d units along the current z-axis, followed by a rotation by angle θ about the current z-axis, is given by*

$$H = Rot_{x,\alpha}\, Trans_{x,b}\, Trans_{z,d} Rot_{z,\theta}$$

$$= \begin{bmatrix} c_\theta & -s_\theta & 0 & b \\ c_\alpha s_\theta & c_\alpha c_\theta & -s_\alpha & -d s_\alpha \\ s_\alpha s_\theta & s_\alpha c_\theta & c_\alpha & d c_\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.6.2 Exponential Coordinates for General Rigid Motions

Just as we represented rotation matrices as exponentials of skew-symmetric matrices, we can also represent homogeneous transformations as exponentials using so-called **twists**.

Definition 2.3. *Let v and k be vectors in $\mathbb{R}^3$ with k a unit vector. A* **twist** *ξ defined by k and v is the 4×4 matrix*

$$\begin{bmatrix} S(k) & v \\ 0 & 0 \end{bmatrix} \tag{2.75}$$

We define $se(3)$ as

$$se(3) = \{(v, S(k)) \mid v \in \mathbb{R}^3, S(k) \in so(3)\} \tag{2.76}$$

$se(3)$ is the vector space of twists, and a similar argument as before in Section 2.5.4 can be used to show that, given any twist $\xi \in se(3)$ and angle $\theta \in \mathbb{R}$, the matrix exponential of $\xi\theta$ is an element of $SE(3)$ and, conversely, every homogeneous transformation (rigid motion) in $SE(3)$ can be expressed as the exponential of a twist. We omit the details here.

2.7 Chapter Summary

In this chapter, we have seen how matrices in $SE(n)$ can be used to represent the relative position and orientation of two coordinate frames for $n = 2, 3$. We have adopted a notional convention in which a superscript is used to indicate a reference frame. Thus, the notation p^0 represents the coordinates of the point p relative to frame 0.

The relative orientation of two coordinate frames can be specified by a rotation matrix, $R \in SO(n)$, with $n = 2, 3$. In two dimensions, the orientation of frame 1 with respect to frame 0 is given by

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

in which θ is the angle between the two coordinate frames. In the three-dimensional case, the rotation matrix is given by

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

In each case, the columns of the rotation matrix are obtained by projecting an axis of the target frame (in this case, frame 1) onto the coordinate axes of the reference frame (in this case, frame 0).

The set of $n \times n$ rotation matrices is known as the special orthogonal group of order n, and is denoted by $SO(n)$. An important property of these matrices is that $R^{-1} = R^T$ for any $R \in SO(n)$.

Rotation matrices can be used to perform coordinate transformations between frames that differ only in orientation. We derived rules for the composition of rotational transformations as

$$R_2^0 = R_1^0 R$$

for the case where the second transformation, R, is performed relative to the current frame and

$$R_2^0 = R R_1^0$$

for the case where the second transformation, R, is performed relative to the fixed frame.

In the three-dimensional case, a rotation matrix can be parameterized using three angles. A common convention is to use the Euler angles (ϕ, θ, ψ), which correspond to successive rotations about the z, y, and z-axes. The corresponding rotation matrix is given by

$$R(\phi, \theta, \psi) = R_{z,\phi} R_{y,\theta} R_{z,\psi}$$

Roll, pitch, and yaw angles are similar, except that the successive rotations are performed with respect to the fixed, world frame instead of being performed with respect to the current frame.

Homogeneous transformations combine rotation and translation. In the three-dimensional case, a homogeneous transformation has the form

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, R \in SO(3),\ d \in \mathbb{R}^3$$

The set of all such matrices comprises the set $SE(3)$, and these matrices can be used to perform coordinate transformations, analogous to rotational transformations using rotation matrices.

Homogeneous transformation matrices can be used to perform coordinate transformations between frames that differ in orientation and translation. We derived rules for the composition of rotational transformations as

$$H_2^0 = H_1^0 H$$

for the case where the second transformation, H, is performed relative to the current frame and

$$H_2^0 = H H_1^0$$

for the case where the second transformation, H, is performed relative to the fixed frame.

We also defined the vector spaces

$$\begin{aligned} so(3) &= \{S \in \mathbb{R}^{3\times 3} \mid S^T = -S\} \\ se(3) &= \{(v, S(k)) \mid v \in \mathbb{R}^3, S(k) \in so(3)\} \end{aligned}$$

and showed that elements of $SO(3)$ and $SE(3)$ can be expressed as matrix exponentials of elements of $so(3)$ and $se(3)$. Formally, $SO(3)$ and $SE(3)$ are **Lie groups** and $so(3)$ and $se(3)$ are their associated **Lie algebras**.

Problems

2–1 Using the fact that $v_1 \cdot v_2 = v_1^T v_2$, show that the dot product of two free vectors does not depend on the choice of frames in which their coordinates are defined.

2–2 Show that the length of a free vector is not changed by rotation, that is, that $\|v\| = \|Rv\|$.

2–3 Show that the distance between points is not changed by rotation, that is, $\|p_1 - p_2\| = \|Rp_1 - Rp_2\|$.

2–4 If a matrix R satisfies $R^T R = I$, show that the column vectors of R are of unit length and mutually perpendicular.

2–5 If a matrix R satisfies $R^T R = I$, then
a) Show that $\det R = \pm 1$
b) Show that $\det R = +1$ if we restrict ourselves to right-handed coordinate frames.

2–6 Verify Equations (2.4)–(2.6).

2–7 A **group** is a set X together with an operation $*$ defined on that set such that

- $x_1 * x_2 \in X$ for all $x_1, x_2 \in X$
- $(x_1 * x_2) * x_3 = x_1 * (x_2 * x_3)$

- There exists an element $I \in X$ such that $I * x = x * I = x$ for all $x \in X$
- For every $x \in X$, there exists some element $y \in X$ such that $x * y = y * x = I$

Show that SO(n) with the operation of matrix multiplication is a group.

2–8 Derive Equations (2.7) and (2.8).

2–9 Suppose A is a 2×2 rotation matrix. In other words $A^T A = I$ and $\det A = 1$. Show that there exists a unique θ such that A is of the form

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2–10 Consider the following sequence of rotations:

1. Rotate by ϕ about the world x-axis.
2. Rotate by θ about the current z-axis.
3. Rotate by ψ about the world y-axis.

Write the matrix product that will give the resulting rotation matrix (do not perform the matrix multiplication).

2–11 Consider the following sequence of rotations:

1. Rotate by ϕ about the world x-axis.
2. Rotate by θ about the world z-axis.
3. Rotate by ψ about the current x-axis.

Write the matrix product that will give the resulting rotation matrix (do not perform the matrix multiplication).

2–12 Consider the following sequence of rotations:

1. Rotate by ϕ about the world x-axis.
2. Rotate by θ about the current z-axis.
3. Rotate by ψ about the current x-axis.
4. Rotate by α about the world z-axis.

Write the matrix product that will give the resulting rotation matrix (do not perform the matrix multiplication).

2–13 Consider the following sequence of rotations:

1. Rotate by ϕ about the world x-axis.
2. Rotate by θ about the world z-axis.
3. Rotate by ψ about the current x-axis.
4. Rotate by α about the world z-axis.

Write the matrix product that will give the resulting rotation matrix (do not perform the matrix multiplication).

2–14 If the coordinate frame $o_1x_1y_1z_1$ is obtained from the coordinate frame $o_0x_0y_0z_0$ by a rotation of $\frac{\pi}{2}$ about the x-axis followed by a rotation of $\frac{\pi}{2}$ about the fixed y-axis, find the rotation matrix R representing the composite transformation. Sketch the initial and final frames.

2–15 Suppose that three coordinate frames $o_1x_1y_1z_1$, $o_2x_2y_2z_2$, and $o_3x_3y_3z_3$ are given, and suppose

$$R_2^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}, \; R_3^1 = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Find the matrix R_3^2.

2–16 Derive equations for the roll, pitch, and yaw angles corresponding to the rotation matrix $R = (r_{ij})$.

2–17 Verify Equation (2.44).

2–18 Verify Equation (2.46).

2–19 If R is a rotation matrix show that $+1$ is an eigenvalue of R. Let k be a unit eigenvector corresponding to the eigenvalue $+1$. Give a physical interpretation of k.

2–20 Let $k = \frac{1}{\sqrt{3}}[1, 1, 1]^T$, $\theta = 90°$. Find $R_{k,\theta}$.

2–21 Show by direct calculation that $R_{k,\theta}$ given by Equation (2.44) is equal to R given by Equation (2.48) if θ and k are given by Equations (2.49) and (2.50), respectively.

2–22 Compute the rotation matrix given by the product

$$R_{x,\theta}R_{y,\phi}R_{z,\pi}R_{y,-\phi}R_{x,-\theta}$$

2–23 Suppose R represents a rotation of 90° about y_0 followed by a rotation of 45° about z_1. Find the equivalent axis-angle to represent R. Sketch the initial and final frames and the equivalent axis vector k.

2–24 Find the rotation matrix corresponding to the Euler angles $\phi = \frac{\pi}{2}$, $\theta = 0$, and $\psi = \frac{\pi}{4}$. What is the direction of the x_1 axis relative to the base frame?

2–25 Unit magnitude complex numbers $a + ib$ with $a^2 + b^2 = 1$ can be used to represent orientation in the plane. In particular, for the complex number $a + ib$, we can define the angle $\theta = \text{Atan2}(a, b)$. Show that multiplication of two complex numbers corresponds to addition of the corresponding angles.

2–26 Show that complex numbers together with the operation of complex multiplication define a group. What is the identity for the group? What is the inverse for $a + ib$?

2–27 Complex numbers can be generalized by defining three independent square roots for -1 that obey the multiplication rules

$$\begin{aligned} -1 &= i^2 = j^2 = k^2, \\ i &= jk = -kj, \\ j &= ki = -ik, \\ k &= ij = -ji \end{aligned}$$

Using these, we define a **quaternion** by $Q = q_0 + iq_1 + jq_2 + kq_3$, which is typically represented by the 4-tuple (q_0, q_1, q_2, q_3). A rotation by θ about the unit vector $n = (n_x, n_y, n_z)$ can be represented by the unit quaternion $Q = \left(\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}\right)$. Show that such a quaternion has unit norm, that is, $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$.

2–28 Using $Q = \left(\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}\right)$, and the results from Section 2.5.3, determine the rotation matrix R that corresponds to the rotation represented by the quaternion (q_0, q_1, q_2, q_3).

2–29 Determine the quaternion Q that represents the same rotation as given by the rotation matrix R.

2–30 The quaternion $Q = (q_0, q_1, q_2, q_3)$ can be thought of as having a scalar component q_0 and a vector component $q = (q_1, q_2, q_3)$. Show that the product of two quaternions, $Z = XY$ is given by

$$\begin{aligned} z_0 &= x_0 y_0 - x^T y \\ z &= x_0 y + y_0 x + x \times y, \end{aligned}$$

Hint: Perform the multiplication $(x_0 + ix_1 + jx_2 + kx_3)(y_0 + iy_1 + jy_2 + ky_3)$ and simplify the result.

2–31 Show that $Q_I = (1, 0, 0, 0)$ is the identity element for unit quaternion multiplication, that is, $QQ_I = Q_I Q = Q$ for any unit quaternion Q.

2–32 The conjugate Q^* of the quaternion Q is defined as

$$Q^* = (q_0, -q_1, -q_2, -q_3)$$

Show that Q^* is the inverse of Q, that is, $Q^*Q = QQ^* = (1, 0, 0, 0)$.

2–33 Let v be a vector whose coordinates are given by (v_x, v_y, v_z). If the quaternion Q represents a rotation, show that the new, rotated coordinates of v are given by $Q(0, v_x, v_y, v_z)Q^*$, in which $(0, v_x, v_y, v_z)$ is a quaternion with zero as its real component.

2–34 Let the point p be rigidly attached to the end effector coordinate frame with local coordinates (x, y, z). If Q specifies the orientation of the end effector frame with respect to the base frame, and T is the vector from the base frame to the origin of the end effector frame, show that the coordinates of p with respect to the base frame are given by

$$Q(0, x, y, z)Q^* + T \tag{2.77}$$

in which $(0, x, y, z)$ is a quaternion with zero as its real component.

2–35 Verify Equation (2.67).

2–36 Compute the homogeneous transformation representing a translation of 3 units along the x-axis followed by a rotation of $\frac{\pi}{2}$ about the current z-axis followed by a translation of 1 unit along the fixed y-axis. Sketch the frame. What are the coordinates of the origin o_1 with respect to the original frame in each case?

2–37 Consider the diagram of Figure 2.13. Find the homogeneous transformations H_1^0, H_2^0, H_2^1 representing the transformations among the three frames shown. Show that $H_2^0 = H_1^0, H_2^1$.

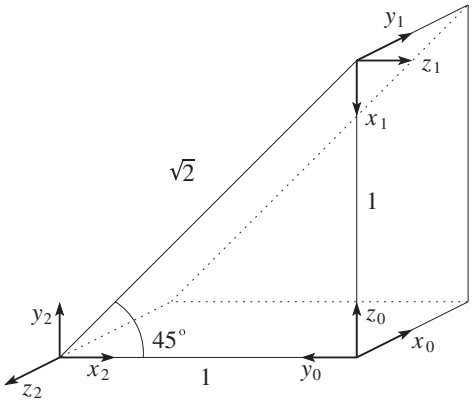


Figure 2.13: Diagram for Problem 2–37.

2–38 Consider the diagram of Figure 2.14. A robot is set up 1 meter from a table. The table top is 1 meter high and 1 meter square. A frame $o_1x_1y_1z_1$ is fixed to the edge of the table as shown. A cube measuring 20 cm on a side is placed in the center of the table with frame $o_2x_2y_2z_2$ established at the center of the cube as shown. A camera is situated directly above the center of the block 2 meters above the table top with frame $o_3x_3y_3z_3$ attached as shown. Find the homogeneous transformations relating each of these frames to the base frame $o_0x_0y_0z_0$. Find the homogeneous transformation relating the frame $o_2x_2y_2z_2$ to the camera frame $o_3x_3y_3z_3$.

2–39 In Problem 2–38, suppose that, after the camera is calibrated, it is rotated 90° about z_3. Recompute the above coordinate transformations.

2–40 If the block on the table is rotated 90° about z_2 and moved so that its center has coordinates $[0, .8, .1]^T$ relative to the frame $o_1x_1y_1z_1$, compute the homogeneous transformation relating the block frame to the camera frame; the block frame to the base frame.

2–41 Consult an astronomy book to learn the basic details of the Earth's rotation about the sun and about its own axis. Define for the Earth a local coordinate frame whose z-axis is the Earth's axis of rotation. Define $t = 0$ to be the exact moment of the summer solstice, and the global reference frame to be coincident with the Earth's frame at time $t = 0$. Give an expression $R(t)$ for the rotation matrix that represents the instantaneous orientation of the earth at time t. Determine as a function of time the homogeneous transformation that specifies the

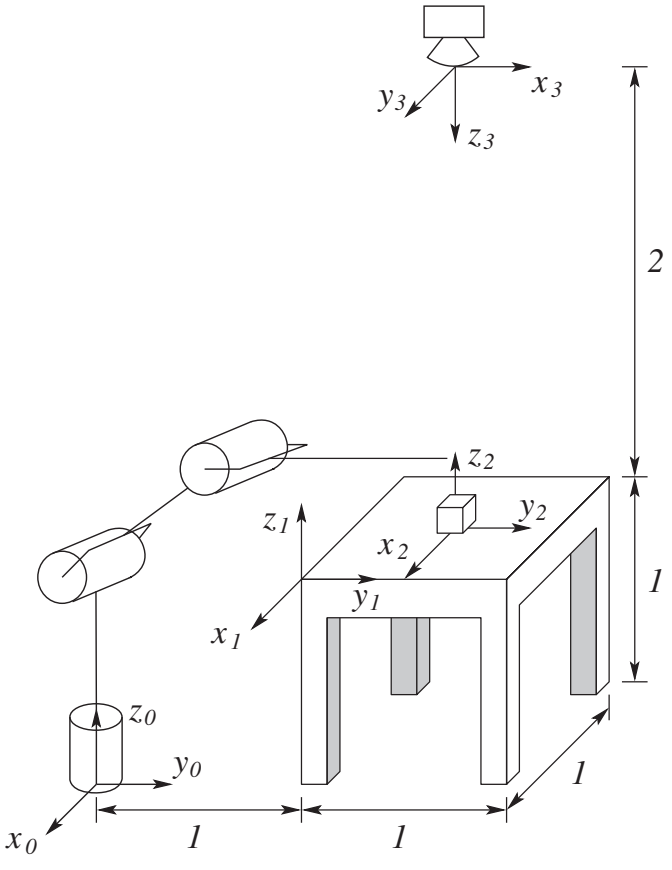


Figure 2.14: Diagram for Problem 2–38.

Earth's frame with respect to the global reference frame.

2–42 In general, multiplication of homogeneous transformation matrices is not commutative. Consider the matrix product

$$H = \mathrm{Rot}_{x,\alpha}\mathrm{Trans}_{x,b}\mathrm{Trans}_{z,d}\mathrm{Rot}_{z,\theta}$$

Determine which pairs of the four matrices on the right-hand side commute. Explain why these pairs commute. Find all permutations of these four matrices that yield the same homogeneous transformation matrix, H.

Notes and References

Rigid body motions and the groups $SO(n)$ and $SE(n)$ are often addressed in mathematics books on the topic of linear algebra. Standard texts for this material include [9], [30], and [49]. These topics are also often covered in applied mathematics texts for physics and engineering, such as [143], [155], and [182]. In addition to these, a detailed treatment of rigid body motion developed with the aid of exponential coordinates and Lie groups is given in [118].

Chapter 3

FORWARD KINEMATICS

The problem of manipulator kinematics is to describe the motion of a manipulator without consideration of the forces and torques causing the motion. The kinematic description is therefore a geometric one. In this chapter we consider the **forward kinematics** problem for serial link manipulators, which is to determine the position and orientation of the end effector given the values for the joint variables of the robot. This problem is easily solved by attaching coordinate frames to each link of the robot and expressing the relationships among these frames as homogeneous transformations. We use a systematic procedure, known as the **Denavit–Hartenberg** convention, to attach these coordinate frames to the robot. The position and orientation of the robot end effector is then reduced to a matrix multiplication of homogeneous transformations. We give examples of this procedure for several of the standard configurations that we introduced in Chapter 1.

In subsequent chapters we will consider the problems of **velocity kinematics** and **inverse kinematics**. The former problem is to determine the relation between the end-effector velocity and the joint velocities whereas the inverse kinematics problem is to determine the joint variables given the end-effector position and orientation.

3.1 Kinematic Chains

As described in Chapter 1, a robot manipulator is composed of a set of links connected together by joints. The joints can either be very simple, such as a revolute joint or a prismatic joint, or they can be more complex, such as a ball and socket joint (recall that a revolute joint is like a hinge that allows a relative rotation about a single axis, and a prismatic joint permits

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

a linear motion along a single axis, namely an extension or retraction). The difference between the two situations is that in the first instance the joint has only a single degree-of-freedom of motion: the angle of rotation in the case of a revolute joint, and the amount of linear displacement in the case of a prismatic joint. In contrast, a ball and socket joint has two degrees of freedom. In this book it is assumed throughout that all joints have only a single degree of freedom. This assumption does not involve any real loss of generality, since joints such as a ball and socket joint (two degrees of freedom) or a spherical wrist (three degrees of freedom) can always be thought of as a succession of single degree-of-freedom joints with links of length zero in between.

With the assumption that each joint has a single degree-of-freedom, the action of each joint can be described by a single real number: the angle of rotation in the case of a revolute joint or the displacement in the case of a prismatic joint.

A robot manipulator with n joints will have $n+1$ links, since each joint connects two links. We number the joints from 1 to n, and we number the links from 0 to n, starting from the base. By this convention, joint i connects link $i-1$ to link i. We will consider the location of joint i to be fixed with respect to link $i-1$. When joint i is actuated, link i moves. Therefore, link 0 (the first link or **base**) is fixed, and does not move when the joints are actuated. Of course, the robot manipulator could itself be mobile (e.g., it could be mounted on a mobile platform or on an autonomous vehicle), but we will not consider this case in the present chapter, since it can be handled easily by slightly extending the techniques presented here.

With the i^{th} joint, we associate a **joint variable**, denoted by q_i. In the case of a revolute joint, q_i is the angle of rotation, and in the case of a prismatic joint, q_i is the joint displacement:

$$q_i = \begin{cases} \theta_i & \text{if joint } i \text{ is revolute} \\ d_i & \text{if joint } i \text{ is prismatic} \end{cases} \tag{3.1}$$

To perform the kinematic analysis, we attach a coordinate frame rigidly to each link. In particular, we attach $o_i x_i y_i z_i$ to link i. This means that, whatever motion the robot executes, the coordinates of each point on link i are constant when expressed in the i^{th} coordinate frame. Furthermore, when joint i is actuated, link i and its attached frame, $o_i x_i y_i z_i$, experience a resulting motion. The frame $o_0 x_0 y_0 z_0$, which is attached to the robot base, is referred to as the **base frame**, **inertial frame** or **world frame**. Figure 3.1 illustrates the idea of attaching frames rigidly to links in the case of an elbow manipulator.

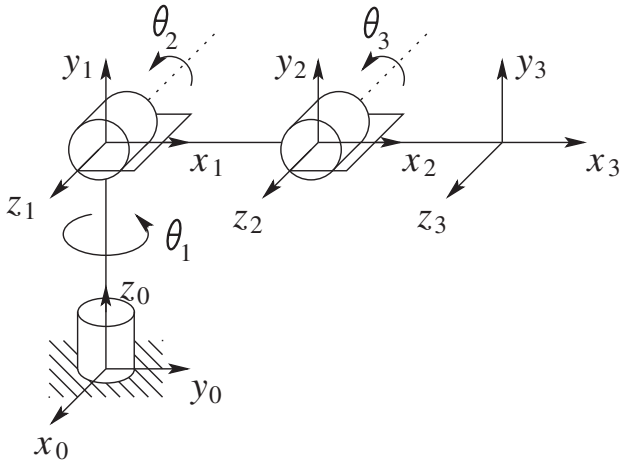


Figure 3.1: Coordinate frames attached to elbow manipulator.

Now, suppose A_i is the homogeneous transformation matrix that gives the position and orientation of $o_i x_i y_i z_i$ with respect to $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$. The matrix A_i is not constant, but varies as the configuration of the robot is changed. However, the assumption that all joints are either revolute or prismatic means that A_i is a function of only a single joint variable, namely q_i. In other words,

$$A_i \quad = \quad A_i(q_i) \tag{3.2}$$

The homogeneous transformation matrix that expresses the position and orientation of $o_j x_j y_j z_j$ with respect to $o_i x_i y_i z_i$ is called a **transformation matrix**, and is denoted by T_j^i. From Chapter 2 we see that

$$T_j^i = \begin{cases} A_{i+1} A_{i+2} \cdots A_{j-1} A_j & \text{if } i < j \\ I & \text{if } i = j \\ (T_i^j)^{-1} & \text{if } j > i \end{cases} \tag{3.3}$$

By the manner in which we have rigidly attached the various frames to the corresponding links, it follows that the position of any point on the end effector when expressed in the last frame n is a constant independent of the configuration of the robot. We denote the position and orientation of the end effector with respect to the inertial or base frame by a three-vector o_n^0 (which gives the coordinates of the origin of the end-effector frame with respect to the base frame) and the 3×3 rotation matrix R_n^0, and define the

homogeneous transformation matrix

$$H = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix} \tag{3.4}$$

Then the position and orientation of the end effector in the inertial frame are given by the product

$$H = T_n^0 = A_1(q_1) \cdots A_n(q_n) \tag{3.5}$$

Each homogeneous transformation A_i is of the form

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

Hence, for $i < j$

$$T_j^i = A_{i+1} \cdots A_j = \begin{bmatrix} R_j^i & o_j^i \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

The matrix R_j^i expresses the orientation of $o_j x_j y_j z_j$ relative to $o_i x_i y_i z_i$ and is given by the rotational parts of the A matrices as

$$R_j^i = R_{i+1}^i \cdots R_j^{j-1} \tag{3.8}$$

The coordinate vectors o_j^i are given recursively by the formula

$$o_j^i = o_{j-1}^i + R_{j-1}^i o_j^{j-1} \tag{3.9}$$

These expressions will be useful in Chapter 4 when we study Jacobian matrices.

In principle, that is all there is to forward kinematics; determine the functions $A_i(q_i)$, and multiply them together as needed. However, it is possible to achieve a considerable amount of streamlining and simplification by introducing further conventions, such as the Denavit–Hartenberg representation of a joint, and this is the objective of the next section.

3.2 The Denavit–Hartenberg Convention

In this section we develop a set of conventions that provide a systematic procedure for computing the forward kinematic equations. It is, of course, possible to carry out forward kinematics analysis even without respecting these conventions, as we did for the two-link planar manipulator example in

Chapter 1. However, the kinematic analysis of an n-link manipulator can be extremely complex and the conventions introduced below simplify the analysis considerably. Moreover, they give rise to a universal language with which engineers can communicate.

A commonly used convention for selecting frames of reference in robotic applications is the **Denavit–Hartenberg**, or **DH convention**. In this convention, each homogeneous transformation A_i is represented as a product of four basic transformations

$$
\begin{aligned}
A_i &= \mathrm{Rot}_{z,\theta_i}\mathrm{Trans}_{z,d_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i} \qquad (3.10)\\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

where the four quantities θ_i, a_i, d_i, α_i are parameters associated with link i and joint i. The four parameters a_i, α_i, d_i, and θ_i in Equation (3.10) are generally given the names **link length**, **link twist**, **link offset**, and **joint angle**, respectively. These names derive from specific aspects of the geometric relationship between two coordinate frames, as will become apparent below. Since the matrix A_i is a function of a single variable, three of the above four quantities are constant for a given link, while the fourth parameter, θ_i for a revolute joint and d_i for a prismatic joint, is the joint variable.

From Chapter 2 one can see that an arbitrary homogeneous transformation matrix can be characterized by six numbers, for example, three numbers to specify the fourth column of the matrix and three Euler angles to specify the upper left 3×3 rotation matrix. In the DH representation, in contrast, there are only four parameters. How is this possible? The answer is that, while frame i is required to be rigidly attached to link i, we have considerable freedom in choosing the origin and the coordinate axes of the frame. For example, it is not necessary that the origin o_i of frame i be placed at the physical end of link i. In fact, it is not even necessary that frame i be placed within the physical link. Frame i could lie in free space so long as

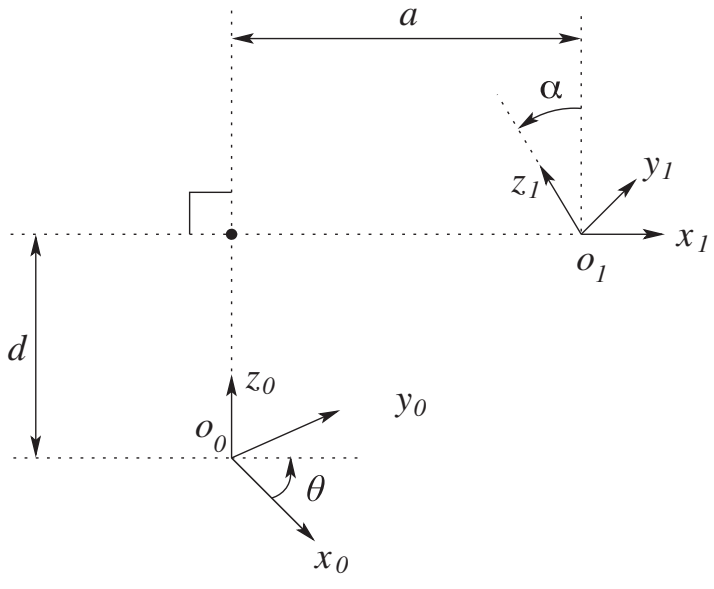


Figure 3.2: Coordinate frames satisfying assumptions DH1 and DH2.

frame i is rigidly attached to link i. By a clever choice of the origin and the coordinate axes, it is possible to cut down the number of parameters needed from six to four (or even fewer in some cases). In Section 3.2.1 we will show why, and under what conditions, this can be done, and in Section 3.2.2 we will show exactly how to make the coordinate frame assignments.

3.2.1 Existence and Uniqueness

Clearly it is not possible to represent any arbitrary homogeneous transformation using only four parameters. Therefore, we begin by determining just which homogeneous transformations can be expressed in the form given by Equation (3.10). Suppose we are given two frames, denoted by frames 0 and 1, respectively. Then there exists a unique homogeneous transformation matrix A that takes the coordinates from frame 1 into those of frame 0. Now, suppose the two frames have the following two additional features:

(DH1) The axis x_1 is perpendicular to the axis z_0.

(DH2) The axis x_1 intersects the axis z_0.

These two properties are illustrated in Figure 3.2. Under these

conditions, we claim that there exist unique numbers a, d, θ, α such that

$$A = \mathrm{Rot}_{z,\theta}\mathrm{Trans}_{z,d}\mathrm{Trans}_{x,a}\mathrm{Rot}_{x,\alpha} \tag{3.11}$$

Of course, since θ and α are angles, we really mean that they are unique to within a multiple of 2π. To show that the matrix A can be written in this form, write A as

$$A = \begin{bmatrix} R_1^0 & o_1^0 \\ 0 & 1 \end{bmatrix} \tag{3.12}$$

If (DH1) is satisfied, then x_1 is perpendicular to z_0 and we have $x_1 \cdot z_0 = 0$. Expressing this constraint with respect to $o_0x_0y_0z_0$, using the fact that the first column of R_1^0 is the representation of the unit vector x_1 with respect to frame 0, we obtain

$$\begin{aligned} 0 &= x_1^0 \cdot z_0^0 \\ &= [r_{11}, r_{21}, r_{31}] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = r_{31} \end{aligned}$$

Since $r_{31} = 0$, we now need only show that there exist unique angles θ and α such that

$$R_1^0 = R_{x,\theta}R_{x,\alpha} = \begin{bmatrix} c_\theta & -s_\theta c_\alpha & s_\theta s_\alpha \\ s_\theta & c_\theta c_\alpha & -c_\theta s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix} \tag{3.13}$$

The only information we have is that $r_{31} = 0$, but this is enough. First, since each row and column of R_1^0 must have unit length, $r_{31} = 0$ implies that

$$r_{11}^2 + r_{21}^2 = 1, \; r_{32}^2 + r_{33}^2 = 1$$

Hence, there exist unique θ and α such that

$$(r_{11}, r_{21}) = (c_\theta, s_\theta), \qquad (r_{33}, r_{32}) = (c_\alpha, s_\alpha)$$

So far, we have shown that

$$R_1^0 = \begin{bmatrix} c_\theta & r_{12} & r_{13} \\ s_\theta & r_{22} & r_{23} \\ 0 & s_\alpha & c_\alpha \end{bmatrix}$$

Again, using the fact that R_1^0 is a rotation matrix, we must have

$$\begin{aligned} r_{12}^2 + r_{13}^2 &= s_\theta^2 \\ r_{22}^2 + r_{23}^2 &= c_\theta^2 \end{aligned}$$

Since these equations must hold for all values of θ, in particular for $\theta = 0, \pi$, we must also have

$$\begin{aligned} r_{12}^2 + r_{13}^2 &= 1, \text{ when } \theta = \pi \\ r_{22}^2 + r_{23}^2 &= 1, \text{ when } \theta = 0 \end{aligned}$$

Putting these expressions together, we can write these variables as

$$\begin{aligned} r_{12}^2 + r_{13}^2 &= s_\theta^2 c_\alpha^2 \\ r_{22}^2 + r_{23}^2 &= c_\theta^2 s_\alpha^2 \end{aligned}$$

which are satisfied by taking r_{12}, r_{13}, r_{22}, and r_{23} as in Equation (3.13).

Next, assumption (DH2) means that the displacement between o_0 and o_1 can be expressed as a linear combination of the vectors z_0 and x_1. This can be written as $o_1 = o_0 + dz_0 + ax_1$. Again, we can express this relationship in the coordinates of $o_0x_0y_0z_0$, and we obtain

$$\begin{aligned} o_1^0 &= o_0^0 + dz_0^0 + ax_1^0 \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + d \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + a \begin{bmatrix} c_\theta \\ s_\theta \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} ac_\theta \\ as_\theta \\ d \end{bmatrix} \end{aligned}$$

Combining the above results, we obtain Equation (3.10) as claimed. Thus, we see that four parameters are sufficient to specify any homogeneous transformation that satisfies the constraints (DH1) and (DH2).

Now that we have established that each homogeneous transformation matrix satisfying conditions (DH1) and (DH2) above can be expressed as in Equation (3.10), we can give a physical interpretation to each of these four quantities. The parameter a is the distance between the axes z_0 and z_1, and is measured along the axis x_1. The angle α is the angle between the axes z_0 and z_1, measured in a plane normal to x_1. The positive sense for α is determined from z_0 to z_1 by the right hand rule as shown in Figure 3.3. The parameter d is the distance from the origin o_0 to the intersection of the

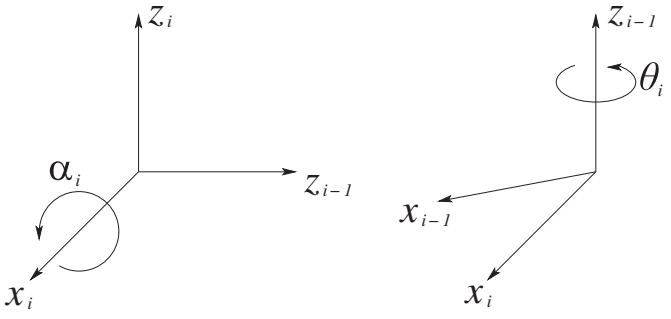


Figure 3.3: Positive sense for α_i and θ_i.

x_1 axis with z_0 measured along the z_0 axis. Finally, θ is the angle from x_0 to x_1 measured in a plane normal to z_0. These physical interpretations will prove useful in developing a procedure for assigning coordinate frames that satisfy the constraints (DH1) and (DH2), and we now turn our attention to developing such a procedure.

3.2.2 Assigning the Coordinate Frames

For a given robot manipulator, one can always choose the frames $0, \ldots, n$ in such a way that the above two conditions are satisfied. In certain circumstances, this will require placing the origin o_i of frame i in a location that may not be intuitively satisfying, but typically this will not be the case. In reading the material below, it is important to keep in mind that the choices of the various coordinate frames are not unique, even when constrained by the requirements above. Thus, it is possible that different engineers will derive differing, but equally correct, coordinate frame assignments for the links of the robot. It is very important to note, however, that the end result (i.e., the matrix T_n^0) will be the same, regardless of the assignment of intermediate DH frames (assuming that the coordinate frames for links 0 and n coincide). We begin by deriving the general procedure. We then discuss various common special cases for which it is possible to further simplify the homogeneous transformation matrix.

To start, note that the choice of z_i is arbitrary. In particular, from Equation (3.13), we see that by choosing α_i and θ_i appropriately, we can obtain any arbitrary direction for z_i. Thus, for our first step, we assign the axes $z_0, \ldots, z_{n-1}$ in an intuitively pleasing fashion. Specifically, we assign z_i to be the axis of actuation for joint $i+1$. Thus, z_0 is the axis of actuation for joint 1, z_1 is the axis of actuation for joint 2, etc. There are two cases to

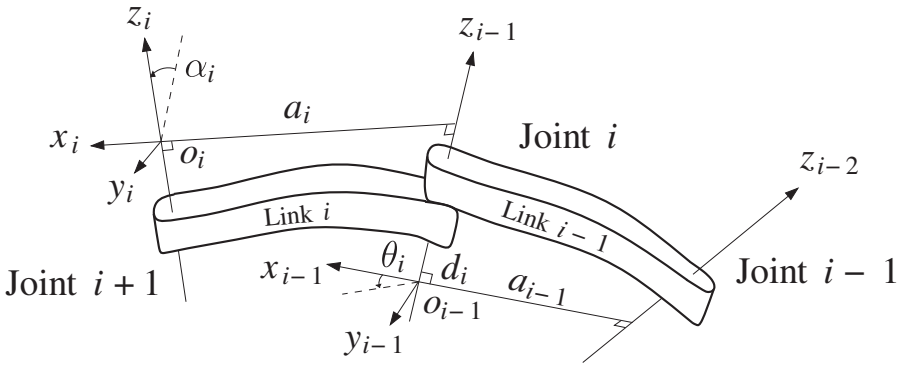


Figure 3.4: Denavit–Hartenberg frame assignment.

consider: (i) if joint $i+1$ is revolute, z_i is the axis of revolution of joint $i+1$; (ii) if joint $i+1$ is prismatic, z_i is the axis of translation of joint $i+1$. At first it may seem a bit confusing to associate z_i with joint $i+1$, but recall that this satisfies the convention that we established above, namely that when joint i is actuated, link i and its attached frame, $o_i x_i y_i z_i$, experience a resulting motion.

Once we have established the z-axes for the links, we establish the base frame. The choice of a base frame is nearly arbitrary. We may choose the origin o_0 of the base frame to be any point on z_0. We then choose x_0, y_0 in any convenient manner so long as the resulting frame is right-handed. This sets up frame 0.

Once frame 0 has been established, we begin an iterative process in which we define frame i using frame $i-1$, beginning with frame 1. Figure 3.4 will be useful for understanding the process that we now describe.

In order to set up frame i it is convenient to consider three cases: (i) the axes z_{i-1}, z_i are not coplanar, (ii) the axes z_{i-1}, z_i intersect, (iii) the axes z_{i-1}, z_i are parallel. Note that in both cases (ii) and (iii) the axes z_{i-1} and z_i are coplanar. This situation is in fact quite common, as we will see in Section 3.3. We now consider each of these three cases.

(i) z_{i-1} and z_i are not coplanar: If z_{i-1} and z_i are not coplanar, then there exists a unique shortest line segment from z_{i-1} to z_i, perpendicular to both z_{i-1} to z_i. This line segment defines x_i, and the point where it intersects z_i is the origin o_i. By construction, both conditions (DH1) and (DH2) are satisfied and the vector from o_{i-1} to o_i is a linear combination of z_{i-1} and x_i. The specification of frame i is completed by choosing the axis

y_i to form a right-handed frame. Since assumptions (DH1) and (DH2) are satisfied, the homogeneous transformation matrix A_i is of the form given in Equation (3.10).

(ii) z_{i-1} is parallel to z_i: If the axes z_{i-1} and z_i are parallel, then there are infinitely many common normals between them and condition (DH1) does not specify x_i completely. In this case we are free to choose the origin o_i at any convenient point on the z_i-axis. The axis x_i is then chosen either to be directed from o_i toward z_{i-1}, along the common normal, or as the opposite of this vector. A common method for choosing o_i is to choose the normal that passes through o_{i-1} as the x_i axis; o_i is then the point at which this normal intersects z_i. In this case, d_i would be equal to zero. Once x_i is fixed, y_i is determined, as usual by the right hand rule. Since the axes z_{i-1} and z_i are parallel, α_i will be zero in this case.

(iii) z_{i-1} intersects z_i: In this case x_i is chosen normal to the plane formed by z_i and z_{i-1}. The positive direction of x_i is arbitrary. Then we choose the origin o_i at the point of intersection of z_i and z_{i-1}. Note that in this case the parameter a_i will be zero.

This constructive procedure works for frames $0, \ldots, n-1$ in an n-link robot. To complete the construction it is necessary to specify frame n. The final coordinate system $o_n x_n y_n z_n$ is commonly referred to as the **end effector** or **tool frame** (see Figure 3.5). The origin o_n is most often placed symmetrically between the fingers of the gripper. The unit vectors along the x_n, y_n, and z_n axes are labeled as n, s, and a, respectively. The terminology arises from the fact that the direction a is the **approach** direction, in the sense that the gripper typically approaches an object along the a direction. Similarly the s direction is the **sliding** direction, the direction along which the fingers of the gripper slide to open and close, and n is the direction **normal** to the plane formed by a and s.

In most contemporary robots the final joint motion is a rotation of the end effector by θ_n and the final two joint axes, z_{n-1} and z_n, coincide. In this case, the transformation between the final two coordinate frames is a translation along z_{n-1} by a distance d_n followed (or preceded) by a rotation of θ_n about z_{n-1}. This is an important observation that will simplify the computation of the inverse kinematics in the next section.

Finally, note the following important fact. In all cases, whether the joint in question is revolute or prismatic, the quantities a_i and α_i are always constant for all i and are characteristic of the manipulator. If joint i is prismatic, then θ_i is also a constant, while d_i is the i^{th} joint variable. Similarly,

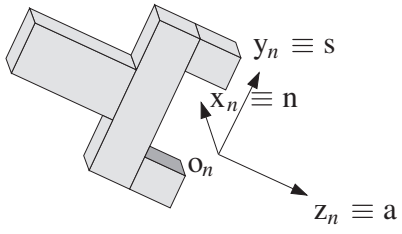


Figure 3.5: Tool frame assignment.

if joint i is revolute, then d_i is constant and θ_i is the i^{th} joint variable.

Summary of the DH Procedure

We may summarize the procedure based on the DH convention in the following algorithm for deriving the forward kinematics for any manipulator.

Step l: Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.

For $i = 1, \ldots, n-1$ perform Steps 3 to 5.

Step 3: Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish y_i to complete a right-handed frame.

Step 6: Establish the end-effector frame $o_n x_n y_n z_n$. Assuming the n^{th} joint is revolute, set $z_n = a$ parallel to z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the center of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as $s \times a$. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

Step 7: Create a table of DH parameters a_i, d_i, α_i, θ_i.

$a_i =$ distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i.

$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. If joint i is prismatic, d_i is variable.

$\alpha_i =$ the angle from z_{i-1} to z_i measured about x_i.

$\theta_i =$ the angle from x_{i-1} to x_i measured about z_{i-1}. If joint i is revolute, θ_i is variable.

Step 8: Form the homogeneous transformation matrices A_i by substituting the above parameters into Equation (3.10).

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

3.3 Examples

In the DH convention the only variable angle is θ, so we simplify notation by writing c_i for $\cos\theta_i$, etc. We also denote $\theta_1 + \theta_2$ by θ_{12}, and $\cos(\theta_1 + \theta_2)$ by c_{12}, and so on. In the following examples it is important to remember that the DH convention, while systematic, still allows considerable freedom in the choice of some of the manipulator parameters. This is particularly true in the case of parallel joint axes or when prismatic joints are involved.

3.3.1 Planar Elbow Manipulator

Consider the two-link planar arm of Figure 3.6. The joint axes z_0 and z_1 are normal to the page. We establish the base frame $o_0x_0y_0z_0$ as shown by choosing the origin at the point of intersection of the z_0 axis with the page and choosing the x_0 axis in the horizontal direction. Note that the direction of the x_0 is arbitrary. Once the base frame is established, the $o_1x_1y_1z_1$ frame is fixed as shown by the DH convention, where the origin o_1 has been located at the intersection of z_1 and the page. The final frame $o_2x_2y_2z_2$ is fixed by choosing the origin o_2 at the end of link 2 as shown. The DH parameters are shown in Table 3.1.

The A matrices are determined from Equation (3.10) as

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

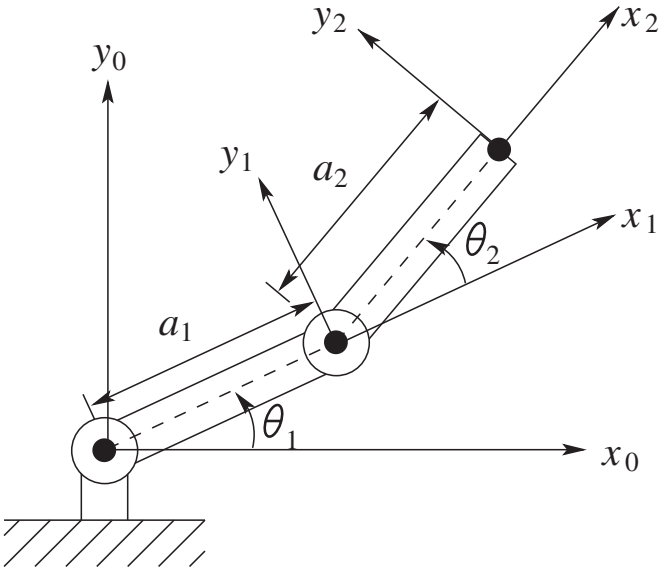


Figure 3.6: Two-link planar manipulator. The z-axes all point out of the page, and are not shown in the figure.

Table 3.1: DH parameters for 2-link planar manipulator. θ_1 and θ_2 are the joint variables.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

The T matrices are thus given by

$$
\begin{aligned}
T_1^0 &= A_1 \\
T_2^0 &= A_1 A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Notice that the first two entries of the last column of T_2^0 are the x and y components of the origin o_2 in the base frame; that is,

$$
\begin{aligned}
x &= a_1 c_1 + a_2 c_{12} \\
y &= a_1 s_1 + a_2 s_{12}
\end{aligned}
$$

Table 3.2: DH parameters for 3-link cylindrical manipulator. θ_1, d_2, and d_3 are the joint variables.

Link	a_i	α_i	d_i	θ_i
1	0	0	d_1	θ_1
2	0	-90	d_2	0
3	0	0	d_3	0

are the coordinates of the end effector in the base frame. The rotational part of T_2^0 gives the orientation of the frame $o_2x_2y_2z_2$ relative to the base frame.

3.3.2 Three-Link Cylindrical Robot

Consider now the three-link cylindrical robot represented symbolically by Figure 3.7. We establish o_0 as shown at joint 1. Note that the placement of the origin o_0 along z_0 and the direction of the x_0 axis are arbitrary. Our choice of o_0 is the most natural, but o_0 could just as well be placed at joint 1. Next, since z_0 and z_1 coincide, the origin o_1 is chosen at joint 1 as shown. The x_1 axis is parallel to x_0 when $\theta_1 = 0$ but, of course its direction will change since θ_1 is variable. Since z_2 and z_1 intersect, the origin o_2 is placed at this intersection. The direction of x_2 is chosen parallel to x_1 so that θ_2 is zero. Finally, the third frame is chosen at the end of link 3 as shown.

The DH parameters are shown in Table 3.2. The corresponding A and T matrices are

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = A_1A_2A_3 = \begin{bmatrix} c_1 & 0 & -s_1 & -s_1d_3 \\ s_1 & 0 & c_1 & c_1d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

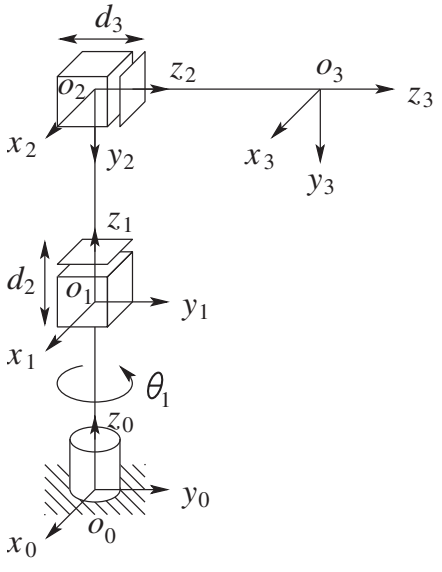


Figure 3.7: Three-link cylindrical manipulator.

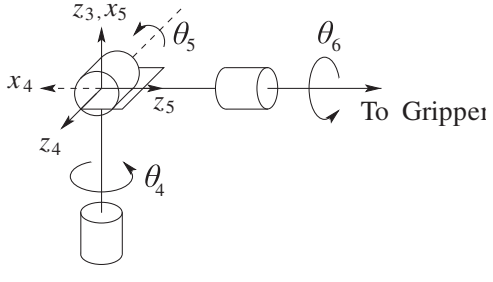


Figure 3.8: The spherical wrist frame assignment.

3.3.3 The Spherical Wrist

Figure 3.8 shows the spherical wrist, a three-link wrist mechanism for which the joint axes z_3, z_4, z_5 intersect at o. The point o is called the **wrist center**. The DH parameters are shown in Table 3.3. The Stanford Manipulator is an example of a manipulator that possesses a wrist of this type.

We show now that the final three joint variables, θ_4, θ_5, θ_6 are a set of Euler angles ϕ, θ, and ψ, with respect to the coordinate frame $o_3x_3y_3z_3$. To see this we need only compute the matrices A_4, A_5, and A_6 using Table 3.3

Table 3.3: DH parameters for the spherical wrist.

Link	a_i	α_i	d_i	θ_i
4	0	-90	0	θ_4
5	0	90	0	θ_5
6	0	0	d_6	θ_6

and Equation (3.10). This gives

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying these together yields

$$\begin{aligned} T_6^3 &= A_4 A_5 A_6 \\ &= \begin{bmatrix} R_6^3 & o_6^3 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & c_4s_5d_6 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & s_4s_5d_6 \\ -s_5c_6 & s_5s_6 & c_5 & c_5d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.15}$$

Comparing the rotational part R_6^3 of T_6^3 with the Euler angle transformation in Equation (2.27) shows that $\theta_4, \theta_5, \theta_6$ can indeed be identified as the Euler angles ϕ, θ, and ψ with respect to the coordinate frame $o_3x_3y_3z_3$.

3.3.4 Cylindrical Manipulator with Spherical Wrist

Suppose that we now attach a spherical wrist to the cylindrical manipulator of Example 3.3.2 as shown in Figure 3.9. Note that the axis of rotation of joint 4 is parallel to z_2 and thus coincides with the axis z_3 of Example 3.3.2. The implication of this is that we can immediately combine Equations (3.14) and (3.15) to derive the forward kinematic equations as

$$T_6^0 = T_3^0 T_6^3 \tag{3.16}$$

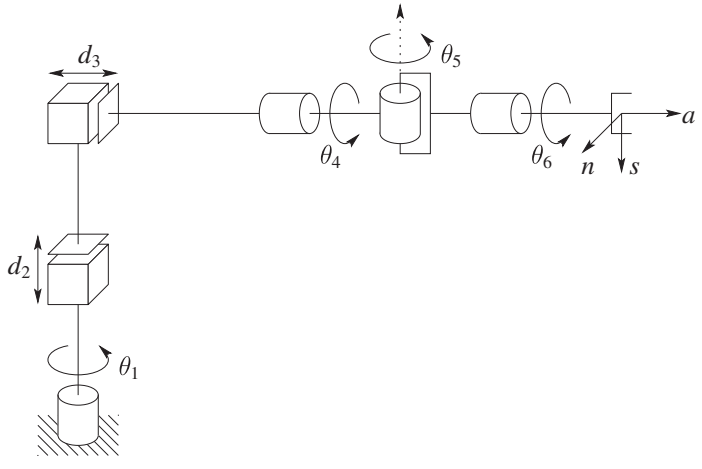


Figure 3.9: Cylindrical robot with spherical wrist.

with T_3^0 given by Equation (3.14) and T_6^3 given by Equation (3.15). Therefore, the forward kinematic equations of this manipulator are given by

$$T_6^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

in which

$$\begin{aligned}
r_{11} &= c_1c_4c_5c_6 - c_1s_4s_6 + s_1s_5c_6 \\
r_{21} &= s_1c_4c_5c_6 - s_1s_4s_6 - c_1s_5c_6 \\
r_{31} &= -s_4c_5c_6 - c_4s_6 \\
r_{12} &= -c_1c_4c_5s_6 - c_1s_4c_6 - s_1s_5c_6 \\
r_{22} &= -s_1c_4c_5s_6 - s_1s_4s_6 + c_1s_5c_6 \\
r_{32} &= s_4c_5c_6 - c_4c_6 \\
r_{13} &= c_1c_4s_5 - s_1c_5 \\
r_{23} &= s_1c_4s_5 + c_1c_5 \\
r_{33} &= -s_4s_5 \\
d_x &= c_1c_4s_5d_6 - s_1c_5d_6 - s_1d_3 \\
d_y &= s_1c_4s_5d_6 + c_1c_5d_6 + c_1d_3 \\
d_z &= -s_4s_5d_6 + d_1 + d_2
\end{aligned}$$

Notice how most of the complexity of the forward kinematics for this

Table 3.4: DH parameters for the Stanford Manipulator. $\theta_i, i = 1, 2, 4, 5, 6$ and d_3 are variable.

Link	d_i	a_i	α_i	θ_i
1	0	0	-90	θ_1
2	d_2	0	$+90$	θ_2
3	d_3	0	0	0
4	0	0	-90	θ_4
5	0	0	$+90$	θ_5
6	d_6	0	0	θ_6

manipulator results from the orientation of the end effector while the expression for the arm position from Equation (3.14) is fairly simple. The spherical wrist assumption not only simplifies the derivation of the forward kinematics here, but will also greatly simplify the inverse kinematics problem in Chapter 5.

3.3.5 Stanford Manipulator

Consider now the Stanford Manipulator shown in Figure 3.10. This manipulator is an example of a spherical (RRP) manipulator with a spherical wrist. This manipulator has an offset in the shoulder joint that slightly complicates both the forward and inverse kinematics problems.

We first establish the joint coordinate frames using the DH convention. The DH parameters are shown in Table 3.4.

It is straightforward to compute the matrices A_i as

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

$$A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

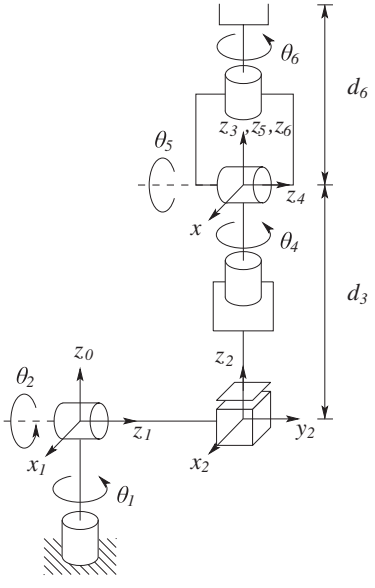


Figure 3.10: DH coordinate frame assignment for the Stanford manipulator.

T_6^0 is then given as

$$T_6^0 = A_1 \cdots A_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

in which

$$\begin{aligned}
r_{11} &= c_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] - s_1(s_4c_5c_6 + c_4s_6) \\
r_{21} &= s_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] + c_1(s_4c_5c_6 + c_4s_6) \\
r_{31} &= -s_2(c_4c_5c_6 - s_4s_6) - c_2s_5c_6 \\
r_{12} &= c_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] - s_1(-s_4c_5s_6 + c_4c_6) \\
r_{22} &= -s_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] + c_1(-s_4c_5s_6 + c_4c_6) \\
r_{32} &= s_2(c_4c_5s_6 + s_4c_6) + c_2s_5s_6 \\
r_{13} &= c_1(c_2c_4s_5 + s_2c_5) - s_1s_4s_5 \\
r_{23} &= s_1(c_2c_4s_5 + s_2c_5) + c_1s_4s_5 \\
r_{33} &= -s_2c_4s_5 + c_2c_5 \\
d_x &= c_1s_2d_3 - s_1d_2 + +d_6(c_1c_2c_4s_5 + c_1c_5s_2 - s_1s_4s_5) \\
d_y &= s_1s_2d_3 + c_1d_2 + d_6(c_1s_4s_5 + c_2c_4s_1s_5 + c_5s_1s_2) \\
d_z &= c_2d_3 + d_6(c_2c_5 - c_4s_2s_5)
\end{aligned}$$

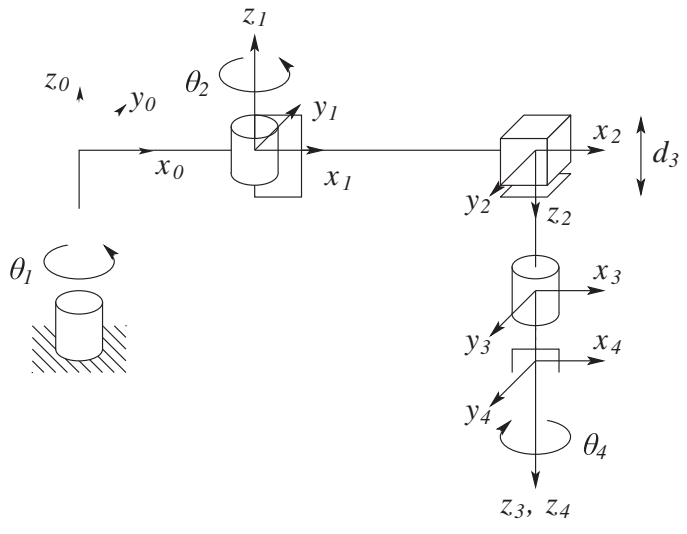


Figure 3.11: DH coordinate frame assignment for the SCARA manipulator.

Table 3.5: DH parameters for the SCARA manipulator. $\theta_1, i = 1, 2, 4$ and d_3 are variable.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	180	0	θ_2
3	0	0	d_3	0
4	0	0	d_4	θ_4

3.3.6 SCARA Manipulator

As another example of the general procedure, consider the SCARA manipulator of Figure 3.11. This manipulator consists of an RRP arm and a one degree-of-freedom wrist, whose motion is a roll about the vertical axis. The first step is to locate and label the joint axes as shown. Since all joint axes are parallel, we have some freedom in the placement of the origins. The origins are placed as shown for convenience. We establish the x_0 axis in the plane of the page as shown. This choice is completely arbitrary, but it does determine the **home position** of the manipulator, which is defined relative to the zero configuration of the manipulator, that is, the position of the manipulator when the joint variables are equal to zero. The DH parameters are given in Table 3.5 and the A matrices are as follows.

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} c_2 & s_2 & 0 & a_2c_2 \\ s_2 & -c_2 & 0 & a_2s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

The forward kinematic equations are therefore given by

$$\begin{aligned} T_4^0 &= A_1 \cdots A_4 \\ &= \begin{bmatrix} c_{12}c_4 + s_{12}s_4 & -c_{12}s_4 + s_{12}c_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -s_{12}s_4 - c_{12}c_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.24}$$

3.4 Chapter Summary

In this chapter we studied the relationships between joint variables q_i and the position and orientation of the end effector for a general serial-link manipulator. We began by introducing the Denavit–Hartenberg convention for assigning coordinate frames to the links of a serial manipulator. Using the DH convention, four parameters are associated to each link of the robot, namely, the link length, link twist, link offset, and joint angle. Once a table of DH parameters is generated, the forward kinematic equations are computed as a product of homogeneous transformations that relate successive DH coordinate frames. We also discussed the kinematics of the spherical wrist and showed that the wrist joint variables can be identified with a set of Euler angles relating end-effector frame and a frame attached to the wrist center, i.e., the point where the spherical wrist joint axes intersect.

Problems

3–1 Consider the three-link planar manipulator shown in Figure 3.12. Derive the forward kinematic equations using the DH convention.

3–2 Consider the two-link Cartesian manipulator of Figure 3.13. Derive the forward kinematic equations using the DH convention.

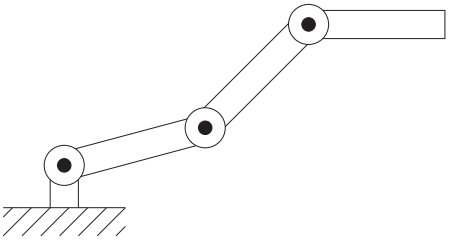


Figure 3.12: Three-link planar arm of Problem 3–1.

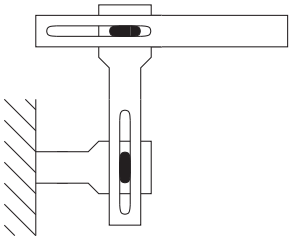


Figure 3.13: Two-link Cartesian robot of Problem 3–2.

3–3 Consider the two-link manipulator of Figure 3.14, which has joint 1 revolute and joint 2 prismatic. Derive the forward kinematic equations using the DH convention.

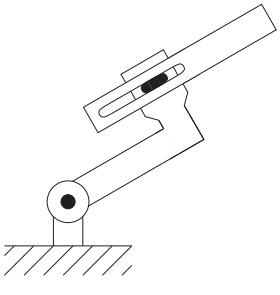


Figure 3.14: Two-link planar arm of Problem 3–3.

3–4 Consider the three-link planar manipulator of Figure 3.15. Derive the forward kinematic equations using the DH convention.

3–5 Consider the three-link articulated robot of Figure 3.16. Derive the forward kinematic equations using the DH convention.

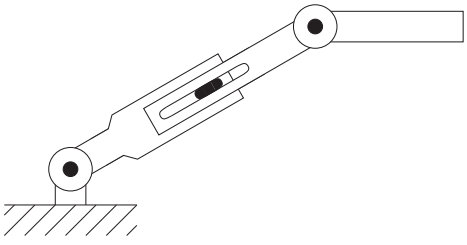


Figure 3.15: Three-link planar arm with prismatic joint of Problem 3–4.

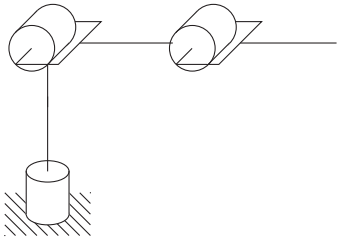


Figure 3.16: Three-link articulated robot.

3–6 Consider the three-link Cartesian manipulator of Figure 3.17. Derive the forward kinematic equations using the DH convention.

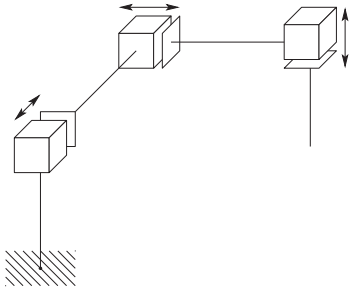


Figure 3.17: Three-link Cartesian robot.

3–7 Attach a spherical wrist to the three-link articulated manipulator of Problem 3–5 as shown in Figure 3.18. Derive the forward kinematic equations for this manipulator.

3–8 Attach a spherical wrist to the three-link Cartesian manipulator of Problem 3–6 as shown in Figure 3.19. Derive the forward kinematic equations for this manipulator.

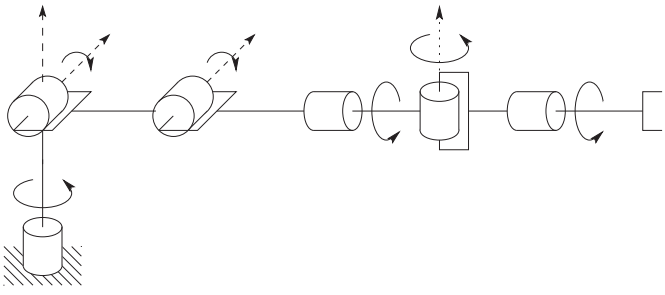


Figure 3.18: Elbow manipulator with spherical wrist.

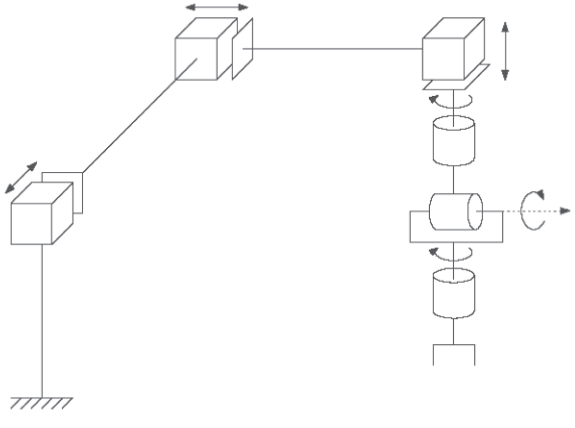


Figure 3.19: Cartesian manipulator with spherical wrist.

3–9 Consider the PUMA 260 manipulator shown in Figure 3.20. Derive the complete set of forward kinematic equations by establishing appropriate DH coordinate frames, constructing a table of DH parameters, forming the A matrices, etc.

Notes and References

The Denavit–Hartenberg convention for assigning coordinate frames was first described in [35] and [70]. Since then, many articles and books have been written on the topics of kinematics of robots and general mechanisms, including [25], [37], [135], [16], and [181].

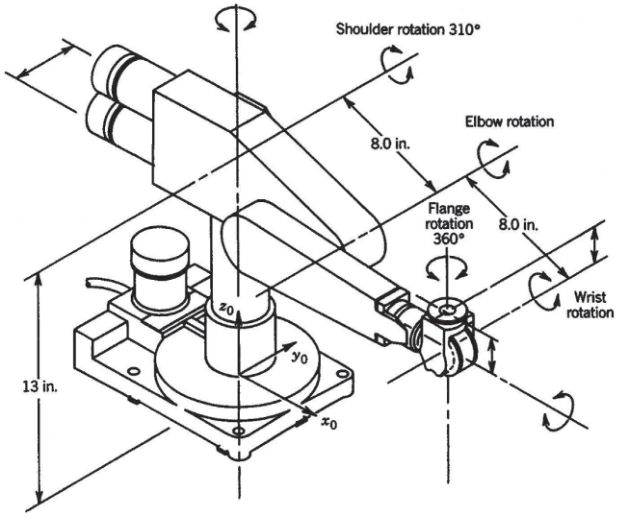


Figure 3.20: PUMA 260 manipulator.

An alternative method to the DH convention to derive the forward kinematics of robots is based on the so-called **product of exponentials**. [118, 105] This method uses the fact that any rigid body motion (rotation and translation) may be represented as a so-called **screw motion**, which is a rotation about a fixed axis in space using the exponential coordinates defined in Chapter 2. Homogeneous transformations are then given as (matrix) exponentials. The reader may consult these texts for details.

Chapter 4

VELOCITY KINEMATICS

In the previous chapter we derived the forward kinematic equations relating joint positions to end-effector positions and orientations. In this chapter we derive the velocity relationships, relating the linear and angular velocities of the end effector to the joint velocities.

Mathematically, the forward kinematic equations define a function from the configuration space of joint positions to the space of Cartesian positions and orientations. The velocity relationships are then determined by the **Jacobian** of this function. The Jacobian is a matrix that generalizes the notion of the ordinary derivative of a scalar function. The Jacobian is one of the most important quantities in the analysis and control of robot motion. It arises in virtually every aspect of robotic manipulation: in the planning and execution of smooth trajectories, in the determination of singular configurations, in the execution of coordinated anthropomorphic motion, in the derivation of the dynamic equations of motion, and in the transformation of forces and torques from the end effector to the manipulator joints.

We begin this chapter with an investigation of velocities and how to represent them. We first consider angular velocity about a fixed axis and then generalize this to rotation about an arbitrary, possibly moving axis with the aid of skew symmetric matrices. Equipped with this general representation of angular velocities, we are able to derive equations for both the angular velocity and the linear velocity for the origin of a moving frame.

We then proceed to the derivation of the manipulator Jacobian. For an n-link manipulator we first derive the Jacobian representing the instantaneous transformation between the n-vector of joint velocities and the six-dimensional vector consisting of the linear and angular velocities of the end effector. This Jacobian is then a $6 \times n$ matrix. The same approach is used to

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

determine the transformation between the joint velocities and the linear and angular velocity of any point on the manipulator. This will be important when we discuss the derivation of the dynamic equations of motion in Chapter 6. We then show how the end-effector velocity is related to the velocity of an attached tool. Following this, we describe the **analytic Jacobian**, which uses an alternative parameterization of end-effector velocity. We then discuss the notion of **singular configurations**. These are configurations in which the manipulator loses one or more degrees of freedom of motion. We show how the singular configurations are determined geometrically and give several examples. Following this, we briefly discuss the inverse problems of determining joint velocities and accelerations for specified end-effector velocities and accelerations. We then discuss how the manipulator Jacobian can be used to relate forces at the end effector to joint torques. We end the chapter by considering redundant manipulators. This includes discussions of the inverse velocity problem, singular value decomposition and manipulability.

4.1 Angular Velocity: The Fixed Axis Case

When a rigid body moves in a pure rotation about a fixed axis, every point of the body moves in a circle. The centers of these circles lie on the axis of rotation. As the body rotates, a perpendicular from any point of the body to the axis sweeps out an angle θ, and this angle is the same for every point of the body. If k is a unit vector in the direction of the axis of rotation, then the angular velocity is given by

$$\omega = \dot{\theta} k \tag{4.1}$$

in which $\dot{\theta}$ is the time derivative of θ.

Given the angular velocity of the body, one learns in introductory dynamics courses that the linear velocity of any point on the body is given by the cross product

$$v = \omega \times r \tag{4.2}$$

in which r is a vector from the origin (which in this case is assumed to lie on the axis of rotation) to the point. In fact, the computation of this velocity v is normally the goal in introductory dynamics courses, and therefore, the main role of an angular velocity is to induce linear velocities of points in a rigid body. In our applications, we are interested in describing the motion of a moving frame, including the motion of the origin of the frame through space and also the rotational motion of the frame's axes. Therefore, for our purposes, the angular velocity will hold equal status with linear velocity.

As in previous chapters, in order to specify the orientation of a rigid object, we attach a coordinate frame rigidly to the object, and then specify the orientation of the attached frame. Since every point on the object experiences the same angular velocity (each point sweeps out the same angle θ in a given time interval), and since each point of the body is in a fixed geometric relationship to the body-attached frame, we see that the angular velocity is a property of the attached coordinate frame itself. Angular velocity is not a property of individual points. Individual points may experience a **linear velocity** that is induced by an angular velocity, but it makes no sense to speak of a point itself rotating. Thus, in Equation (4.2) v corresponds to the linear velocity of a point, while ω corresponds to the angular velocity associated with a rotating coordinate frame.

In this fixed axis case, the problem of specifying angular displacements is really a planar problem, since each point traces out a circle, and since every circle lies in a plane. Therefore, it is tempting to use $\dot{\theta}$ to represent the angular velocity. However, as we have already seen in Chapter 2, this choice does not generalize to the three-dimensional case, either when the axis of rotation is not fixed, or when the angular velocity is the result of multiple rotations about distinct axes. For this reason, we will develop a more general representation for angular velocities. This is analogous to our development of rotation matrices in Chapter 2 to represent orientation in three dimensions. The key tool that we will need to develop this representation is the skew-symmetric matrix, which is the topic of the next section.

4.2 Skew-Symmetric Matrices

We defined **skew-symmetric** matrices and presented some of their properties in Appendix B, which we used in Chapter 2 when we discussed exponential coordinates and Rodrigues' formula. In this section we give a more detailed description of skew-symmetric matrices and their properties that will simplify many of the computations involved in computing relative velocity transformations among coordinate frames.

Recall that an $n \times n$ matrix S is **skew-symmetric** if and only if

$$S^T + S = 0 \tag{4.3}$$

and that we denote the set of all 3×3 skew-symmetric matrices by $so(3)$. If $S \in so(3)$ has components s_{ij}, $i, j = 1, 2, 3$ then Equation (4.3) is equivalent to the nine equations

$$s_{ij} + s_{ji} = 0 \qquad i, j = 1, 2, 3 \tag{4.4}$$

From Equation (4.4) we see that $s_{ii} = 0$; that is, the diagonal terms of S are zero and the off-diagonal terms $s_{ij}, \quad i \neq j$ satisfy $s_{ij} = -s_{ji}$. Thus, S contains only three independent entries and every 3×3 skew-symmetric matrix has the form

$$S = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix} \tag{4.5}$$

If $a = (a_x, a_y, a_z)$ is a 3-vector, we define the skew symmetric matrix $S(a)$ as

$$S(a) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

Example 4.1. *Denote by i, j, and k the three unit basis coordinate vectors,*

$$i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \quad j = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \quad k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The skew-symmetric matrices $S(i)$, $S(j)$, and $S(k)$ are given by

$$S(i) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \qquad S(j) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$S(k) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

4.2.1 Properties of Skew-Symmetric Matrices

Skew-symmetric matrices possess several properties that will prove useful for subsequent derivations.[1] Among these properties are

1. The operator S is linear, that is,

$$S(\alpha a + \beta b) = \alpha S(a) + \beta S(b) \tag{4.6}$$

for any vectors a and b belonging to $\mathbb{R}^3$ and scalars α and β.

[1] These properties are consequences of the fact that $so(3)$ is a *Lie algebra*, a vector space with a suitably defined product operation [15].

2. For any vectors a and p belonging to $\mathbb{R}^3$,

$$S(a)p = a \times p \tag{4.7}$$

where $a \times p$ denotes the vector cross product. Equation (4.7) can be verified by direct calculation.

3. For $R \in SO(3)$ and $a \in \mathbb{R}^3$

$$RS(a)R^T = S(Ra) \tag{4.8}$$

To show this, we use that fact that if $R \in SO(3)$ and a, b are vectors in $\mathbb{R}^3$

$$R(a \times b) = Ra \times Rb \tag{4.9}$$

This can be shown by direct calculation. Equation (4.9) is **not** true in general unless R is orthogonal. It says that if we first rotate the vectors a and b using the rotation transformation R and then form the cross product of the rotated vectors Ra and Rb, the result is the same as that obtained by first forming the cross product $a \times b$ and then rotating to obtain $R(a \times b)$. Equation (4.8) now follows easily from Equations (4.7) and (4.9) as follows. Let $b \in \mathbb{R}^3$ be an arbitrary vector. Then

$$\begin{aligned} RS(a)R^T b &= R(a \times R^T b) \\ &= (Ra) \times (RR^T b) \\ &= (Ra) \times b \\ &= S(Ra)b \end{aligned}$$

and the result follows. The left-hand side of Equation (4.8) represents a similarity transformation of the matrix $S(a)$. The equation says, therefore, that the matrix representation of $S(a)$ in a coordinate frame rotated by R is the same as the skew-symmetric matrix $S(Ra)$ corresponding to the vector a rotated by R.

4.2.2 The Derivative of a Rotation Matrix

Suppose now that a rotation matrix R is a function of the single variable θ. Hence, $R = R(\theta) \in SO(3)$ for every θ. Since R is orthogonal for all θ it follows that

$$R(\theta)R(\theta)^T = I \tag{4.10}$$

Differentiating both sides of Equation (4.10) with respect to θ using the product rule gives

$$\left[\frac{d}{d\theta}R\right] R(\theta)^T + R(\theta)\left[\frac{d}{d\theta}R^T\right] = 0 \tag{4.11}$$

Let us define the matrix S as

$$S = \left[\frac{d}{d\theta}R\right] R(\theta)^T \tag{4.12}$$

Then the transpose of S is

$$S^T = \left(\left[\frac{d}{d\theta}R\right] R(\theta)^T\right)^T = R(\theta)\left[\frac{d}{d\theta}R^T\right] \tag{4.13}$$

Equation (4.11) says that

$$S + S^T = 0 \tag{4.14}$$

In other words, the matrix S defined by Equation (4.12) is skew-symmetric. Multiplying both sides of Equation (4.12) on the right by R and using the fact that $R^T R = I$ yields

$$\frac{d}{d\theta}R = SR(\theta) \tag{4.15}$$

Equation (4.15) is very important. It says that computing the derivative of the rotation matrix R is equivalent to a matrix multiplication by a skew-symmetric matrix S. The most commonly encountered situation is the case where R is a basic rotation matrix or a product of basic rotation matrices.

Example 4.2. *If $R = R_{x,\theta}$, the basic rotation matrix given by Equation (2.7), then direct computation shows that*

$$S = \left[\frac{d}{d\theta}R\right] R^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\theta & -\cos\theta \\ 0 & \cos\theta & -\sin\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = S(i)$$

Thus, we have shown that

$$\frac{d}{d\theta}R_{x,\theta} = S(i)R_{x,\theta}$$

Similar computations show that

$$\frac{d}{d\theta}R_{y,\theta} = S(j)R_{y,\theta} \quad \text{and} \quad \frac{d}{d\theta}R_{z,\theta} = S(k)R_{z,\theta} \tag{4.16}$$

Example 4.3. *Let $R_{k,\theta}$ be a rotation about the axis defined by $k = (k_1, k_2, k_3)$ as in Equation (2.44). It is easy to show using Rodrigues' formula that*

$$\frac{d}{d\theta}R_{k,\theta} = S(k)R_{k,\theta} \tag{4.17}$$

4.3 Angular Velocity: The General Case

We now consider the general case of angular velocity about an arbitrary, possibly moving, axis. Suppose that a rotation matrix R is time-varying, so that $R = R(t) \in SO(3)$ for every $t \in \mathbb{R}$. Assuming that $R(t)$ is continuously differentiable as a function of t, an argument identical to the one in the previous section shows that the time derivative $\dot{R}(t)$ of $R(t)$ can be written as

$$\dot{R}(t) = S(\omega(t))R(t) \tag{4.18}$$

where the matrix $S(\omega(t))$ is skew symmetric. The vector $\omega(t)$ is the **angular velocity** of the rotating frame with respect to the fixed frame at time t. To see that ω is the angular velocity vector, consider a point p rigidly attached to a moving frame. The coordinates of p relative to the fixed frame are given by $p^0 = R_1^0 p^1$. Differentiating this expression we obtain

$$\begin{aligned} \frac{d}{dt}p^0 &= \dot{R}_1^0 p^1 \\ &= S(\omega)R_1^0 p^1 \\ &= \omega \times R_1^0 p^1 \\ &= \omega \times p^0 \end{aligned}$$

which shows that ω is indeed the traditional angular velocity vector.

Equation (4.18) shows the relationship between angular velocity and the derivative of a rotation matrix. In particular, if the instantaneous orientation of a frame $o_1x_1y_1z_1$ with respect to a frame $o_0x_0y_0z_0$ is given by R_1^0, then the angular velocity of frame $o_1x_1y_1z_1$ is directly related to the derivative of R_1^0 by Equation (4.18). When there is a possibility of ambiguity, we will use the notation $\omega_{i,j}$ to denote the angular velocity that corresponds to the derivative of the rotation matrix R_j^i. Since ω is a free vector, we can express it with respect to any coordinate system of our choosing. As usual we use a superscript to denote the reference frame. For example, $\omega_{1,2}^0$ would give the angular velocity that corresponds to the derivative of R_2^1, expressed in coordinates relative to frame $o_0x_0y_0z_0$. In cases where the angular velocities specify rotation relative to the base frame, we will

often simplify the subscript, for example, using ω_2 to represent the angular velocity that corresponds to the derivative of R_2^0.

Example 4.4. *Suppose that* $R(t) = R_{x,\theta(t)}$. *Then* $\dot{R}(t)$ *is computed using the chain rule as*

$$\dot{R} = \frac{dR}{dt} = \frac{dR}{d\theta}\frac{d\theta}{dt} = \dot{\theta}S(i)R(t) = S(\omega(t))R(t) \tag{4.19}$$

in which $\omega = i\dot{\theta}$ *is the angular velocity. Note,* $i = (1, 0, 0)$.

4.4 Addition of Angular Velocities

We are often interested in finding the resultant angular velocity due to the relative rotation of several coordinate frames. We now derive the expressions for the composition of angular velocities of two moving frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ relative to the fixed frame $o_0x_0y_0z_0$. For now, we assume that the three frames share a common origin. Let the relative orientations of the frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ be given by the rotation matrices $R_1^0(t)$ and $R_2^1(t)$ (both time-varying).

In the derivations that follow, we will use the notation $\omega_{i,j}$ to denote the angular velocity vector that corresponds to the time derivative of the rotation matrix R_j^i. Since we can express this vector relative to the coordinate frame of our choosing, we again use a superscript to define the reference frame. Thus, $\omega_{i,j}^k$ would denote the angular velocity vector corresponding to the derivative of R_j^i, expressed relative to frame k.

As in Chapter 2,

$$R_2^0(t) = R_1^0(t)R_2^1(t) \tag{4.20}$$

Taking derivatives of both sides of Equation (4.20) with respect to time yields

$$\dot{R}_2^0 = \dot{R}_1^0 R_2^1 + R_1^0 \dot{R}_2^1 \tag{4.21}$$

Using Equation (4.18), the term $\dot{R}_2^0$ on the left-hand side of Equation (4.21) can be written

$$\dot{R}_2^0 = S(\omega_{0,2}^0)R_2^0 \tag{4.22}$$

In this expression, $\omega_{0,2}^0$ denotes the total angular velocity experienced by frame $o_2x_2y_2z_2$. This angular velocity results from the combined rotations expressed by R_1^0 and R_2^1.

The first term on the right-hand side of Equation (4.21) is simply

$$\dot{R}_1^0 R_2^1 = S(\omega_{0,1}^0) R_1^0 R_2^1 = S(\omega_{0,1}^0) R_2^0 \tag{4.23}$$

In this equation, $\omega_{0,1}^0$ denotes the angular velocity of frame $o_1x_1y_1z_1$ that results from the changing R_1^0, and this angular velocity vector is expressed relative to the coordinate system $o_0x_0y_0z_0$.

Let us examine the second term on the right-hand side of Equation (4.21). Using Equation (4.8) we have

$$\begin{aligned} R_1^0 \dot{R}_2^1 &= R_1^0 S(\omega_{1,2}^1) R_2^1 \\ &= R_1^0 S(\omega_{1,2}^1) {R_1^0}^T R_1^0 R_2^1 = S(R_1^0 \omega_{1,2}^1) R_1^0 R_2^1 \\ &= S(R_1^0 \omega_{1,2}^1) R_2^0 \end{aligned} \tag{4.24}$$

In this equation, $\omega_{1,2}^1$ denotes the angular velocity of frame $o_2x_2y_2z_2$ that corresponds to the changing R_2^1, expressed relative to the coordinate system $o_1x_1y_1z_1$. Thus, the product $R_1^0\omega_{1,2}^1$ expresses this angular velocity relative to the coordinate system $o_0x_0y_0z_0$. In other words, $R_1^0\omega_{1,2}^1$ gives the coordinates of the free vector $\omega_{1,2}$ with respect to frame 0.

Now, combining the above expressions we have shown that

$$S(\omega_2^0) R_2^0 = \{S(\omega_{0,1}^0) + S(R_1^0 \omega_{1,2}^1)\} R_2^0 \tag{4.25}$$

Since $S(a) + S(b) = S(a + b)$, we see that

$$\omega_2^0 = \omega_{0,1}^0 + R_1^0 \omega_{1,2}^1 \tag{4.26}$$

In other words, the angular velocities can be added once they are expressed relative to the same coordinate frame, in this case $o_0x_0y_0z_0$.

The above reasoning can be extended to any number of coordinate systems. In particular, suppose that we are given

$$R_n^0 = R_1^0 R_2^1 \cdots R_n^{n-1} \tag{4.27}$$

Extending the above reasoning we obtain

$$\dot{R}_n^0 = S(\omega_{0,n}^0) R_n^0 \tag{4.28}$$

in which

$$\omega_{0,n}^0 = \omega_{0,1}^0 + R_1^0 \omega_{1,2}^1 + R_2^0 \omega_{2,3}^2 + R_3^0 \omega_{3,4}^3 + \cdots + R_{n-1}^0 \omega_{n-1,n}^{n-1} \tag{4.29}$$

$$= \omega_{0,1}^0 + \omega_{1,2}^0 + \omega_{2,3}^0 + \omega_{3,4}^0 + \cdots + \omega_{n-1,n}^0 \tag{4.30}$$

4.5 Linear Velocity of a Point Attached to a Moving Frame

We now consider the linear velocity of a point that is rigidly attached to a moving frame. Suppose the point p is rigidly attached to the frame $o_1x_1y_1z_1$, and that $o_1x_1y_1z_1$ is rotating relative to the frame $o_0x_0y_0z_0$. Then the coordinates of p with respect to the frame $o_0x_0y_0z_0$ are given by

$$p^0 = R_1^0(t)p^1 \tag{4.31}$$

The velocity $\dot{p}^0$ is then given by the product rule for differentiation as

$$\begin{aligned} \dot{p}^0 &= \dot{R}_1^0(t)p^1 + R_1^0(t)\dot{p}^1 \\ &= S(\omega^0)R_1^0(t)p^1 \\ &= S(\omega^0)p^0 \\ &= \omega^0 \times p^0 \end{aligned} \tag{4.32}$$

which is the familiar expression for the velocity in terms of the vector cross product. Note that Equation (4.32) follows from the fact that p is rigidly attached to frame $o_1x_1y_1z_1$, and therefore its coordinates relative to frame $o_1x_1y_1z_1$ do not change, giving $\dot{p}^1 = 0$.

Now, suppose that the motion of the frame $o_1x_1y_1z_1$ relative to $o_0x_0y_0z_0$ is more general. Suppose that the homogeneous transformation relating the two frames is time-dependent, so that

$$H_1^0(t) = \begin{bmatrix} R_1^0(t) & o_1^0(t) \\ 0 & 1 \end{bmatrix} \tag{4.33}$$

For simplicity we omit the argument t and the subscripts and superscripts on R_1^0 and o_1^0, and write

$$p^0 = Rp^1 + o \tag{4.34}$$

Differentiating the above expression using the product rule gives

$$\begin{aligned} \dot{p}^0 &= \dot{R}p^1 + \dot{o} \\ &= S(\omega)Rp^1 + \dot{o} \\ &= \omega \times r + v \end{aligned} \tag{4.35}$$

where $r = Rp^1$ is the vector from o_1 to p expressed in the orientation of the frame $o_0x_0y_0z_0$, and v is the rate at which the origin o_1 is moving.

If the point p is moving relative to the frame $o_1x_1y_1z_1$, then we must add to the term v the term $R(t)\dot{p}^1$, which is the rate of change of the coordinates p^1 expressed in the frame $o_0x_0y_0z_0$.

4.6 Derivation of the Jacobian

Consider an n-link manipulator with joint variables $q_1, \ldots, q_n$. Let

$$T_n^0(q) = \begin{bmatrix} R_n^0(q) & o_n^0(q) \\ 0 & 1 \end{bmatrix} \tag{4.36}$$

denote the transformation from the end-effector frame to the base frame, where $q = (q_1, \ldots, q_n)$ is the vector of joint variables. As the robot moves about, both the joint variables q_i and the end-effector position o_n^0 and orientation R_n^0 will be functions of time. The objective of this section is to relate the linear and angular velocity of the end effector to the vector of joint velocities $\dot{q}(t)$. Let

$$S(\omega_n^0) = \dot{R}_n^0 (R_n^0)^T \tag{4.37}$$

define the angular velocity vector ω_n^0 of the end effector, and let

$$v_n^0 = \dot{o}_n^0 \tag{4.38}$$

denote the linear velocity of the end effector. We seek expressions of the form

$$v_n^0 = J_v \dot{q} \tag{4.39}$$
$$\omega_n^0 = J_\omega \dot{q} \tag{4.40}$$

where J_v and J_ω are $3 \times n$ matrices. We may write Equations (4.39) and (4.40) together as

$$\xi = J\dot{q} \tag{4.41}$$

in which ξ and J are given by

$$\xi = \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} \quad \text{and} \quad J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{4.42}$$

The vector ξ is sometimes called a body velocity. Note that this velocity vector is not the derivative of a position variable, since the angular velocity vector is not the derivative of any particular time-varying quantity. The matrix J is called the **manipulator Jacobian** or **Jacobian** for short. Note that J is a $6 \times n$ matrix where n is the number of links. We next derive a simple expression for the Jacobian of any manipulator.

4.6.1 Angular Velocity

Recall from Equation (4.29) that angular velocities can be added as free vectors, provided that they are expressed relative to a common coordinate frame. Thus, we can determine the angular velocity of the end effector relative to the base by expressing the angular velocity contributed by each joint in the orientation of the base frame and then summing these.

If the i^{th} joint is revolute, then the i^{th} joint variable q_i equals θ_i and the axis of rotation is z_{i-1}. Slightly abusing notation, let ω_i^{i-1} represent the angular velocity of link i that is imparted by the rotation of joint i, expressed relative to frame $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$. This angular velocity is expressed in the frame $i-1$ by

$$\omega_i^{i-1} \quad = \quad \dot{q}_i z_{i-1}^{i-1} \quad = \quad \dot{q}_i k \tag{4.43}$$

in which, as above, k is the unit coordinate vector $(0, 0, 1)$.

If the i^{th} joint is prismatic, then the motion of frame i relative to frame $i-1$ is a translation and

$$\omega_i^{i-1} \quad = \quad 0 \tag{4.44}$$

Thus, if joint i is prismatic, the angular velocity of the end effector does not depend on q_i, which now equals d_i.

Therefore, the overall angular velocity of the end effector, ω_n^0, in the base frame is determined by Equation (4.29) as

$$\omega_n^0 \quad = \quad \rho_1 \dot{q}_1 k + \rho_2 \dot{q}_2 R_1^0 k + \cdots + \rho_n \dot{q}_n R_{n-1}^0 k = \sum_{i=1}^{n} \rho_i \dot{q}_i z_{i-1}^0 \tag{4.45}$$

in which ρ_i is equal to 1 if joint i is revolute and 0 if joint i is prismatic, since

$$z_{i-1}^0 \quad = \quad R_{i-1}^0 k \tag{4.46}$$

Of course, $z_0^0 = k = [0, 0, 1]^T$.

The lower half of the Jacobian J_ω, in Equation (4.42) is thus given as

$$J_\omega \quad = \quad [\rho_1 z_0 \cdots \rho_n z_{n-1}] \tag{4.47}$$

Note that, in this equation, we have omitted the superscripts for the unit vectors along the z-axes, since these are all referenced to the world frame. In the remainder of the chapter we will follow this convention when there is no ambiguity concerning the reference frame.

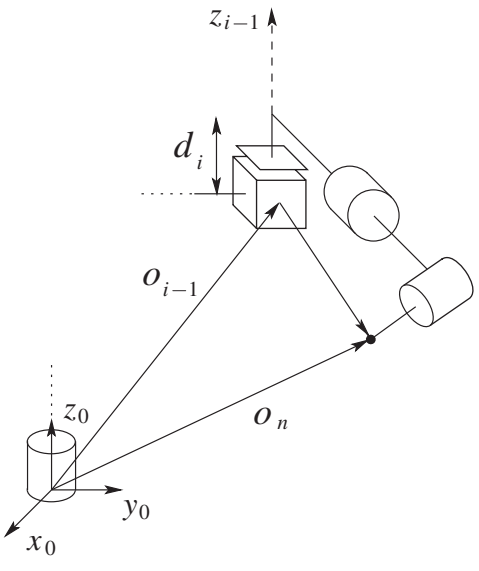


Figure 4.1: Motion of the end effector due to prismatic joint i.

4.6.2 Linear Velocity

The linear velocity of the end effector is just $\dot{o}_n^0$. By the chain rule for differentiation

$$\dot{o}_n^0 = \sum_{i=1}^{n} \frac{\partial o_n^0}{\partial q_i} \dot{q}_i \tag{4.48}$$

Thus, we see that the i^{th} column of J_v, which we denote as J_{v_i} is given by

$$J_{v_i} = \frac{\partial o_n^0}{\partial q_i} \tag{4.49}$$

Furthermore, this expression is just the linear velocity of the end effector that would result if $\dot{q}_i$ were equal to one and the other $\dot{q}_j$ were zero. In other words, the i^{th} column of the Jacobian can be generated by holding all joints but the i^{th} joint fixed and actuating the i^{th} at unit velocity. This observation leads to a simple and intuitive derivation of the linear velocity Jacobian as we now show. We now consider the two cases of prismatic and revolute joints separately.

Case 1: Prismatic Joints

Figure 4.1 illustrates the case when all joints are fixed except a single prismatic joint. Since joint i is prismatic, it imparts a pure translation to the

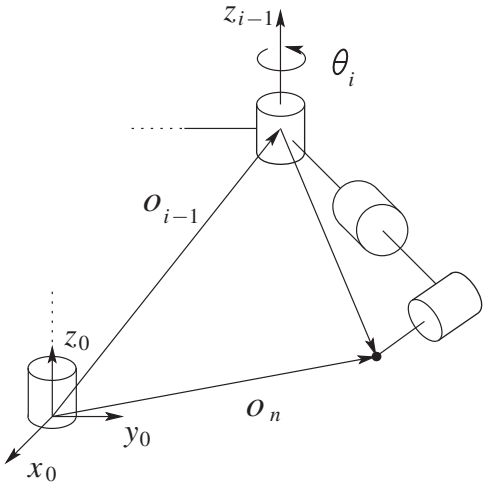


Figure 4.2: Motion of the end effector due to revolute joint i.

end effector. The direction of translation is parallel to the axis z_{i-1} and the magnitude of the translation is $\dot{d}_i$, where d_i is the DH joint variable. Thus, in the orientation of the base frame we have

$$\dot{o_n^0} = \dot{d}_i R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \dot{d}_i z_{i-1}^0 \tag{4.50}$$

in which d_i is the joint variable for prismatic joint i. Thus, for the case of prismatic joints, after dropping the superscripts we have

$$J_{v_i} = z_{i-1} \tag{4.51}$$

Case 2: Revolute Joints

Figure 4.2 illustrates the case when all joints are fixed except a single revolute joint. Since joint i is revolute, we have $q_i = \theta_i$. Referring to Figure 4.2 and assuming that all joints are fixed except joint i, we see that the linear velocity of the end effector is simply of the form $\omega \times r$, where

$$\omega = \dot{\theta}_i z_{i-1} \tag{4.52}$$

and

$$r = o_n - o_{i-1} \tag{4.53}$$

Thus, putting these together and expressing the coordinates relative to the base frame, for a revolute joint we obtain

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1}) \tag{4.54}$$

in which we have omitted the zero superscripts following our convention.

4.6.3 Combining the Linear and Angular Velocity Jacobians

As we have seen in the preceding section, the upper half of the Jacobian J_v is given as

$$J_v = [J_{v_1} \cdots J_{v_n}] \tag{4.55}$$

in which the i^{th} column J_{v_i} is

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases} \tag{4.56}$$

The lower half of the Jacobian is given as

$$J_\omega = [J_{\omega_1} \cdots J_{\omega_n}] \tag{4.57}$$

in which the i^{th} column J_{ω_i} is

$$J_{\omega_i} = \begin{cases} z_{i-1} & \text{for revolute joint } i \\ 0 & \text{for prismatic joint } i \end{cases} \tag{4.58}$$

The above formulas make the determination of the Jacobian of any manipulator simple since all of the quantities needed are available once the forward kinematics are worked out. Indeed, the only quantities needed to compute the Jacobian are the unit vectors z_i and the coordinates of the origins $o_1, \ldots, o_n$. A moment's reflection shows that the coordinates for z_i with respect to the base frame are given by the first three elements in the third column of T_i^0 while o_i is given by the first three elements of the fourth column of T_i^0. Thus, only the third and fourth columns of the T matrices are needed in order to evaluate the Jacobian according to the above formulas.

The above procedure works not only for computing the velocity of the end effector but also for computing the velocity of any point on the manipulator. This will be important in Chapter 6 when we will need to compute the velocity of the center of mass of the various links in order to derive the dynamic equations of motion.

Example 4.5 (Two-Link Planar Manipulator)**.** *Consider the two-link planar manipulator of Example 3.3.1. Since both joints are revolute, the Jacobian matrix, which in this case is* 6×2*, is of the form*

$$J(q) = \begin{bmatrix} z_0 \times (o_2 - o_0) & z_1 \times (o_2 - o_1) \\ z_0 & z_1 \end{bmatrix} \tag{4.59}$$

The various quantities above are easily seen to be

$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix} \quad o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix} \tag{4.60}$$

$$z_0 = z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.61}$$

Performing the required calculations then yields

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{4.62}$$

It is easy to see how the above Jacobian compares with Equation (1.1) derived in Chapter 1. The first two rows of Equation (4.62) are exactly the 2×2 *Jacobian of Chapter 1 and give the linear velocity of the origin* o_2 *relative to the base. The third row in Equation (4.62) is the linear velocity in the direction of* z_0*, which is of course always zero in this case. The last three rows represent the angular velocity of the final frame, which is simply a rotation about the vertical axis at the rate* $\dot{\theta}_1 + \dot{\theta}_2$*.*

Example 4.6 (Jacobian for an Arbitrary Point on a Link)**.** *Consider the three-link planar manipulator of Figure 4.3. Suppose we wish to compute the linear velocity* v *and the angular velocity* ω *of the center of link 2 as shown. In this case we have that*

$$J(q) = \begin{bmatrix} z_0 \times (o_c - o_0) & z_1 \times (o_c - o_1) & 0 \\ z_0 & z_1 & 0 \end{bmatrix} \tag{4.63}$$

which is merely the usual Jacobian with o_c *in place of* o_n*. Note that the third column of the Jacobian is zero, since the velocity of the second link*

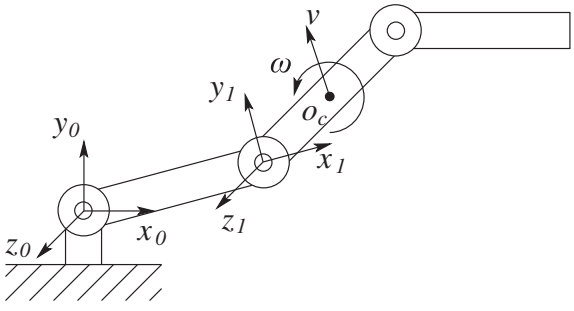


Figure 4.3: Finding the velocity of link 2 of a 3-link planar robot.

is unaffected by motion of the third link.[2] *Note that in this case the vector* o_c *must be computed as it is not given directly by the* T *matrices (Problem 4–11).*

Example 4.7 (Stanford Manipulator)**.** *Consider the Stanford Manipulator of Example 3.3.5 with its associated Denavit–Hartenberg coordinate frames. Note that joint 3 is prismatic and that* $o_3 = o_4 = o_5$ *as a consequence of the spherical wrist and the frame assignment. Denoting this common origin by* o *we see that the columns of the Jacobian have the form*

$$J_i = \begin{bmatrix} z_{i-1} \times (o_6 - o_{i-1}) \\ z_{i-1} \end{bmatrix} \quad i = 1, 2$$

$$J_3 = \begin{bmatrix} z_2 \\ 0 \end{bmatrix}$$

$$J_i = \begin{bmatrix} z_{i-1} \times (o_6 - o) \\ z_{i-1} \end{bmatrix} \quad i = 4, 5, 6$$

Now, using the A *matrices given by Equations (3.18)–(3.20) and the* T *matrices formed as products of the* A *matrices, these quantities are easily computed as follows. First,* o_j *is given by the first three entries of the last column of* $T_j^0 = A_1 \cdots A_j$*, with* $o_0 = (0, 0, 0) = o_1$*. The vector* z_j *is given as* $z_j = R_j^0 k$ *where* R_j^0 *is the rotational part of* T_j^0*. Thus, it is only necessary to compute the matrices* T_j^0 *to calculate the Jacobian. Carrying out these calculations one obtains the following expressions for the Stanford*

[2]Note that we are treating only kinematic effects here. Reaction forces on link 2 due to the motion of link 3 will influence the motion of link 2. These dynamic effects are treated by the methods of Chapter 6.

Manipulator:

$$o_6 = \begin{bmatrix} c_1 s_2 d_3 - s_1 d_2 + d_6(c_1 c_2 c_4 s_5 + c_1 c_5 s_2 - s_1 s_4 s_5) \\ s_1 s_2 d_3 - c_1 d_2 + d_6(c_1 s_4 s_5 + c_2 c_4 s_1 s_5 + c_5 s_1 s_2) \\ c_2 d_3 + d_6(c_2 c_5 - c_4 s_2 s_5) \end{bmatrix} \quad (4.64)$$

$$o_3 = \begin{bmatrix} c_1 s_2 d_3 - s_1 d_2 \\ s_1 s_2 d_3 + c_1 d_2 \\ c_2 d_3 \end{bmatrix} \quad (4.65)$$

The various z-axes are given as

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad z_1 = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \quad (4.66)$$

$$z_2 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix}, \qquad z_3 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \quad (4.67)$$

$$z_4 = \begin{bmatrix} -c_1 c_2 s_4 - s_1 c_4 \\ -s_1 c_2 s_4 + c_1 c_4 \\ s_2 s_4 \end{bmatrix} \quad (4.68)$$

$$z_5 = \begin{bmatrix} c_1 c_2 c_4 s_5 - s_1 s_4 s_5 + c_1 s_2 c_5 \\ s_1 c_2 c_4 s_5 + c_1 s_4 s_5 + s_1 s_2 c_5 \\ -s_2 c_4 s_5 + c_2 c_5 \end{bmatrix} \quad (4.69)$$

The Jacobian of the Stanford Manipulator is now given by combining these expressions according to the given formulas (Problem 4–17).

Example 4.8 (SCARA Manipulator)**.** *We will now derive the Jacobian of the SCARA manipulator of Example 3.3.6. This Jacobian is a* 6×4 *matrix since the SCARA has only four degrees of freedom. As before we need only compute the matrices* $T_j^0 = A_1 \dots A_j$*, where the A matrices are given by Equations (3.22) and (3.23).*

Since joints 1, 2, and 4 are revolute and joint 3 is prismatic, and since $o_4 - o_3$ *is parallel to* z_3 *(and thus,* $z_3 \times (o_4 - o_3) = 0$*), the Jacobian is of the form*

$$J = \begin{bmatrix} z_0 \times (o_4 - o_0) & z_1 \times (o_4 - o_1) & z_2 & 0 \\ z_0 & z_1 & 0 & z_3 \end{bmatrix} \quad (4.70)$$

The origins of the DH frames are given by

$$o_1 = \begin{bmatrix} a_1c_1 \\ a_1s_1 \\ 0 \end{bmatrix}, \qquad o_2 = \begin{bmatrix} a_1c_1 + a_2c_{12} \\ a_1s_1 + a_2s_{12} \\ 0 \end{bmatrix} \tag{4.71}$$

$$o_4 = \begin{bmatrix} a_1c_1 + a_2c_{12} \\ a_1s_2 + a_2s_{12} \\ d_3 - d_4 \end{bmatrix} \tag{4.72}$$

Similarly $z_0 = z_1 = k$, and $z_2 = z_3 = -k$. Therefore the Jacobian of the SCARA Manipulator is

$$J = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \tag{4.73}$$

4.7 The Tool Velocity

Many tasks require that a tool be attached to the end effector. In such cases, it is necessary to relate the velocity of the tool frame to the velocity of the end-effector frame. Suppose that the tool is rigidly attached to the end effector, and the fixed spatial relationship between the end effector and the tool frame is given by the constant homogeneous transformation matrix

$$T^6_{\text{tool}} = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{4.74}$$

We will assume that the end effector velocity is given and expressed in coordinates relative to the end-effector frame, that is, we are given ξ^6_6. In this section we will derive the velocity of the tool expressed in coordinates relative to the tool frame, that is, we will derive $\xi^{\text{tool}}_{\text{tool}}$.

Since the two frames are rigidly attached, the angular velocity of the tool frame is the same as the angular velocity of the end-effector frame. To see this, simply compute the angular velocities of each frame by taking the derivatives of the appropriate rotation matrices. Since R is constant and $R^0_{\text{tool}} = R^0_6 R$, we have

$$\begin{aligned} & \dot{R}^0_{\text{tool}} = \dot{R}^0_6 R \\ \Longrightarrow \quad & S(\omega^0_{\text{tool}}) R^0_{\text{tool}} = S(\omega^0_6) R^0_6 R \\ \Longrightarrow \quad & S(\omega^0_{\text{tool}}) = S(\omega^0_6) \end{aligned}$$

Thus, $\omega_{\text{tool}} = \omega_6$, and to obtain the tool angular velocity relative to the tool frame we apply a rotational transformation

$$\omega_{\text{tool}}^{\text{tool}} = \omega_6^{\text{tool}} = R^T \omega_6^6 \tag{4.75}$$

If the end-effector frame is moving with body velocity $\xi = (v_6, \omega_6)$, then the linear velocity of the origin of the tool frame, which is rigidly attached to the end-effector frame, is given by

$$v_{\text{tool}} = v_6 + \omega_6 \times r \tag{4.76}$$

where r is the vector from the origin of the end-effector frame to the origin of the tool frame. From Equation (4.74), we see that d gives the coordinates of the origin of the tool frame with respect to the end-effector frame, and therefore we can express r in coordinates relative to the tool frame as $r^{\text{tool}} = R^T d$. Thus, we write $\omega_6 \times r$ in coordinates with respect to the tool frame as

$$\begin{aligned} \omega_6^{\text{tool}} \times r^{\text{tool}} &= R^T \omega_6^6 \times (R^T d) \\ &= -R^T d \times R^T \omega_6^6 \\ &= -S(R^T d) R^T \omega_6^6 \\ &= -R^T S(d) R R^T \omega_6^6 \\ &= -R^T S(d) \omega_6^6 \end{aligned} \tag{4.77}$$

To express the free vector v_6^6 in coordinates relative to the tool frame, we apply the rotational transformation

$$v_6^{\text{tool}} = R^T v_6^6 \tag{4.78}$$

Combining Equations (4.76), (4.77), and (4.78) to obtain the linear velocity of the tool frame and using Equation (4.75) for the angular velocity of the tool frame, we have

$$\begin{aligned} v_{\text{tool}}^{\text{tool}} &= R^T v_6^6 - R^T S(d) \omega_6^6 \\ \omega_{\text{tool}}^{\text{tool}} &= R^T \omega_6^6 \end{aligned}$$

which can be written as the matrix equation

$$\xi_{\text{tool}}^{\text{tool}} = \begin{bmatrix} R^T & -R^T S(d) \\ 0_{3\times 3} & R^T \end{bmatrix} \xi_6^6 \tag{4.79}$$

In many cases, it is useful to solve the inverse problem: compute the required end effector-velocity to produce a desired tool velocity. Since

$$\begin{bmatrix} R & S(d)R \\ 0_{3\times 3} & R \end{bmatrix} = \begin{bmatrix} R^T & -R^T S(d) \\ 0_{3\times 3} & R^T \end{bmatrix}^{-1} \tag{4.80}$$

(Problem 4–18) we can solve Equation (4.79) for ξ_6^6, obtaining

$$\xi_6^6 = \begin{bmatrix} R & S(d)R \\ 0_{3\times 3} & R \end{bmatrix} \xi_{\text{tool}}^{\text{tool}}$$

This gives the general expression for transforming velocities between two rigidly attached moving frames

$$\xi_A^A = \begin{bmatrix} R_B^A & S(d_B^A)R_B^A \\ 0_{3\times 3} & R_B^A \end{bmatrix} \xi_B^B \tag{4.81}$$

4.8 The Analytical Jacobian

The Jacobian matrix derived above is sometimes called the **geometric Jacobian** to distinguish it from the **analytical Jacobian**, denoted $J_a(q)$, which is based on a minimal representation for the orientation of the end-effector frame. Let

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix} \tag{4.82}$$

denote the end-effector pose, where $d(q)$ is the usual vector from the origin of the base frame to the origin of the end-effector frame and α denotes a minimal representation for the orientation of the end-effector frame relative to the base frame. For example, let $\alpha = (\phi, \theta, \psi)$ be a vector of Euler angles as defined in Chapter 2. Then we seek an expression of the form

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q} \tag{4.83}$$

to define the analytical Jacobian.

It can be shown (Problem 4–7) that, if $R = R_{z,\phi}R_{y,\theta}R_{z,\psi}$ is the Euler angle transformation, then

$$\dot{R} = S(\omega)R \tag{4.84}$$

in which ω defining the angular velocity is given by

$$\omega = \begin{bmatrix} c_\psi s_\theta \dot{\phi} - s_\psi \dot{\theta} \\ s_\psi s_\theta \dot{\phi} + c_\psi \dot{\theta} \\ \dot{\psi} + c_\theta \dot{\phi} \end{bmatrix} \tag{4.85}$$

$$= \begin{bmatrix} c_\psi s_\theta & -s_\psi & 0 \\ s_\psi s_\theta & c_\psi & 0 \\ c_\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = B(\alpha)\dot{\alpha} \tag{4.86}$$

The components of ω are called **nutation**, **spin**, and **precession**, respectively. Combining the above relationship with the previous definition of the Jacobian

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ \omega \end{bmatrix} = J(q)\dot{q} \tag{4.87}$$

yields

$$J(q)\dot{q} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ B(\alpha)\dot{\alpha} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & B(\alpha) \end{bmatrix} \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & B(\alpha) \end{bmatrix} J_a(q)\dot{q}$$

Thus, the analytical Jacobian, $J_a(q)$, may be computed from the geometric Jacobian as

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B^{-1}(\alpha) \end{bmatrix} J(q) \tag{4.88}$$

provided $\det B(\alpha) \neq 0$.

In the next section we discuss the notion of Jacobian singularities, which are configurations where the Jacobian loses rank. Singularities of the matrix $B(\alpha)$ are called **representational singularities**. It can easily be shown (Problem 4–19) that $B(\alpha)$ is invertible provided $s_\theta \neq 0$. This means that the singularities of the analytical Jacobian include the singularities of the geometric Jacobian, J, as defined in the next section, together with the representational singularities.

4.9 Singularities

The $6 \times n$ Jacobian $J(q)$ defines a mapping

$$\xi \;=\; J(q)\dot{q} \tag{4.89}$$

between the vector $\dot{q}$ of joint velocities and the vector $\xi = (v, \omega)$ of end-effector velocities. This implies that all possible end-effector velocities are linear combinations of the columns of the Jacobian matrix,

$$\xi = J_1\dot{q}_1 + J_2\dot{q}_2 + \cdots + J_n\dot{q}_n$$

For example, for the two-link planar arm, the Jacobian matrix given in Equation (4.62) has two columns. It is easy to see that the linear velocity of the end effector must lie in the xy-plane, since neither column has a nonzero entry for the third row. Since $\xi \in \mathbb{R}^6$, it is necessary that J have six linearly independent columns for the end effector to be able to achieve any arbitrary velocity (see Appendix B).

The rank of a matrix is the number of linearly independent columns (or rows) in the matrix. Thus, when $\operatorname{rank} J = 6$, the end effector can execute any arbitrary velocity. For a matrix $J \in \mathbb{R}^{6\times n}$, it is always the case that $\operatorname{rank} J \leq \min(6, n)$. For example, for the two-link planar arm, we always have $\operatorname{rank} J \leq 2$, while for an anthropomorphic arm with spherical wrist we always have $\operatorname{rank} J \leq 6$.

The rank of a matrix is not necessarily constant. Indeed, the rank of the manipulator Jacobian matrix will depend on the configuration q. Configurations for which $\operatorname{rank} J(q)$ is less than its maximum value are called **singularities** or **singular configurations**.

Identifying manipulator singularities is important for several reasons.

- Singularities represent configurations from which certain directions of motion may be unattainable.
- At singularities, bounded end-effector velocities may correspond to unbounded joint velocities.
- At singularities, bounded joint torques may correspond to unbounded end-effector forces and torques. (We will see this in Chapter 10.)
- Singularities often correspond to points on the boundary of the manipulator workspace, that is, to points of maximum reach of the manipulator.
- Singularities correspond to points in the manipulator workspace that may be unreachable under small perturbations of the link parameters, such as length, offset, etc.

There are a number of methods that can be used to determine the singularities of the Jacobian. In this chapter, we will exploit the fact that a square matrix is singular when its determinant is equal to zero. In general, it is difficult to solve the nonlinear equation $\det J(q) = 0$. Therefore, we now introduce the method of decoupling singularities, which is applicable whenever, for example, the manipulator is equipped with a spherical wrist.

4.9.1 Decoupling of Singularities

We saw in Chapter 3 that a set of forward kinematic equations can be derived for any manipulator by attaching a coordinate frame rigidly to each link in any manner that we choose, computing a set of homogeneous transformations relating the coordinate frames, and multiplying them together as needed. The DH convention is merely a systematic way to do this. Although

the resulting equations are dependent on the coordinate frames chosen, the manipulator configurations themselves are geometric quantities, independent of the frames used to describe them. Recognizing this fact allows us to decouple the determination of singular configurations, for those manipulators with spherical wrists, into two simpler problems. The first is to determine so-called **arm singularities**, that is, singularities resulting from motion of the arm, which consists of the first three or more links, while the second is to determine the **wrist singularities** resulting from motion of the spherical wrist.

Consider the case that $n = 6$, that is, the manipulator consists of a 3-DOF arm with a 3-DOF spherical wrist. In this case the Jacobian is a 6×6 matrix and a configuration q is singular if and only if

$$\det J(q) \quad = \quad 0 \tag{4.90}$$

If we now partition the Jacobian J into 3×3 blocks as

$$J = [J_P \mid J_O] \quad = \quad \left[\begin{array}{c|c} J_{11} & J_{12} \\ \hline J_{21} & J_{22} \end{array} \right] \tag{4.91}$$

then, since the final three joints are always revolute

$$J_O \quad = \quad \begin{bmatrix} z_3 \times (o_6 - o_3) & z_4 \times (o_6 - o_4) & z_5 \times (o_6 - o_5) \\ z_3 & z_4 & z_5 \end{bmatrix} \tag{4.92}$$

Since the wrist axes intersect at a common point o, if we choose the coordinate frames so that $o_3 = o_4 = o_5 = o_6 = o$, then J_O becomes

$$J_O \quad = \quad \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix} \tag{4.93}$$

In this case the Jacobian matrix has the block triangular form

$$J \quad = \quad \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix} \tag{4.94}$$

with determinant

$$\det J \quad = \quad \det J_{11} \det J_{22} \tag{4.95}$$

where J_{11} and J_{22} are each 3×3 matrices. J_{11} has i^{th} column $z_{i-1} \times (o - o_{i-1})$ if joint i is revolute, and z_{i-1} if joint i is prismatic, while

$$J_{22} \quad = \quad [z_3 \; z_4 \; z_5] \tag{4.96}$$

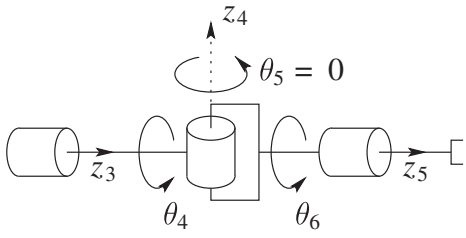


Figure 4.4: Spherical wrist singularity.

Therefore the set of singular configurations of the manipulator is the union of the set of arm configurations satisfying $\det J_{11} = 0$ and the set of wrist configurations satisfying $\det J_{22} = 0$. *Note that this form of the Jacobian does not necessarily give the correct relation between the velocity of the end effector and the joint velocities. It is intended only to simplify the determination of singularities.*

4.9.2 Wrist Singularities

We can now see from Equation (4.96) that a spherical wrist is in a singular configuration whenever the vectors z_3, z_4, and z_5 are linearly dependent. Referring to Figure 4.4 we see that this happens when the joint axes z_3 and z_5 are collinear, that is, when $\theta_5 = 0$ or π. These are the only singularities of the spherical wrist, and they are unavoidable without imposing mechanical limits on the wrist design to restrict its motion in such a way that z_3 and z_5 are prevented from lining up. In fact, when any two revolute-joint axes are collinear a singularity results, since an equal and opposite rotation about the axes results in no net motion of the end effector.

4.9.3 Arm Singularities

To investigate arm singularities we need only to compute $\det J_{11}$, which is done using Equation (4.56) but with the wrist center o in place of o_n. In the remainder of this section, we will determine the singularities for three common arms, the elbow manipulator, the spherical manipulator and the SCARA manipulator.

Example 4.9 (Elbow Manipulator Singularities)**.** *Consider the three-link articulated manipulator with coordinate frames attached as shown in Figure 4.5. It is left as an exercise (Problem 4–12) to show that*

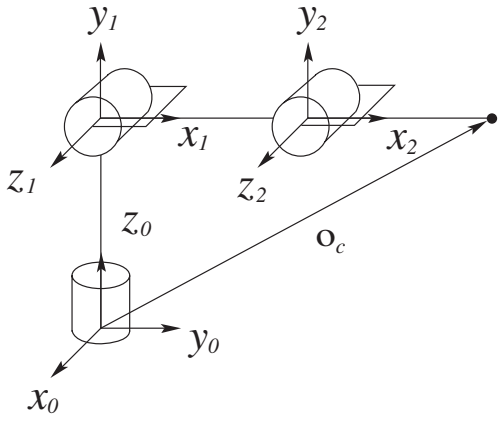


Figure 4.5: Elbow manipulator.

$$J_{11} = \begin{bmatrix} -a_2 s_1 c_2 - a_3 s_1 c_{23} & -a_2 s_2 c_1 - a_3 s_{23} c_1 & -a_3 c_1 s_{23} \\ a_2 c_1 c_2 + a_3 c_1 c_{23} & -a_2 s_1 s_2 - a_3 s_1 s_{23} & -a_3 s_1 s_{23} \\ 0 & a_2 c_2 + a_3 c_{23} & a_3 c_{23} \end{bmatrix} \tag{4.97}$$

and that the determinant of J_{11} *is*

$$\det J_{11} \;=\; -a_2 a_3 s_3 (a_2 c_2 + a_3 c_{23}) \tag{4.98}$$

We see from Equation (4.98) that the elbow manipulator is in a singular configuration when

$$s_3 \;=\; 0, \quad \textit{that is, } \theta_3 = 0 \textit{ or } \pi \tag{4.99}$$

and whenever

$$a_2 c_2 + a_3 c_{23} \;=\; 0 \tag{4.100}$$

The situation of Equation (4.99) is shown in Figure 4.6 and arises when the elbow is fully extended or fully retracted as shown. The second situation of Equation (4.100) is shown in Figure 4.7. This configuration occurs when the wrist center intersects the axis of the base rotation, z_0*. As we saw in Chapter 5, there are infinitely many singular configurations and infinitely many solutions to the inverse position kinematics when the wrist center is along this axis.*

For an elbow manipulator with an offset, as shown in Figure 4.8, the wrist center cannot intersect z_0*, which corroborates our earlier statement that points reachable at singular configurations may not be reachable under arbitrarily small perturbations of the manipulator parameters, in this case an offset in either the elbow or the shoulder.*

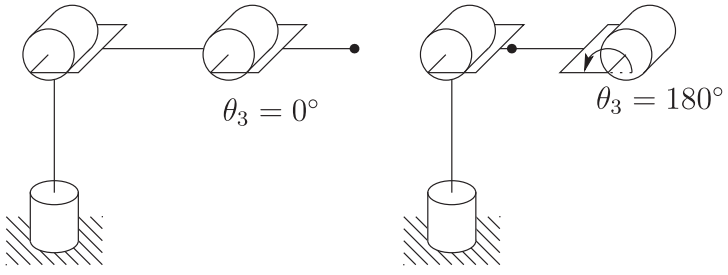


Figure 4.6: Elbow singularities of the elbow manipulator.

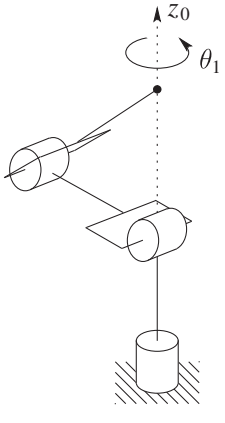


Figure 4.7: Singularity of the elbow manipulator with no offsets.

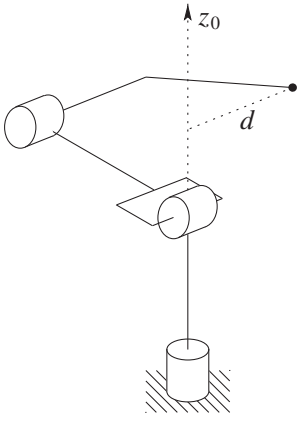


Figure 4.8: Elbow manipulator with an offset at the elbow.

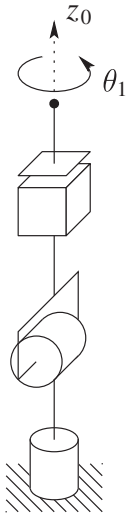


Figure 4.9: Singularity of spherical manipulator with no offsets.

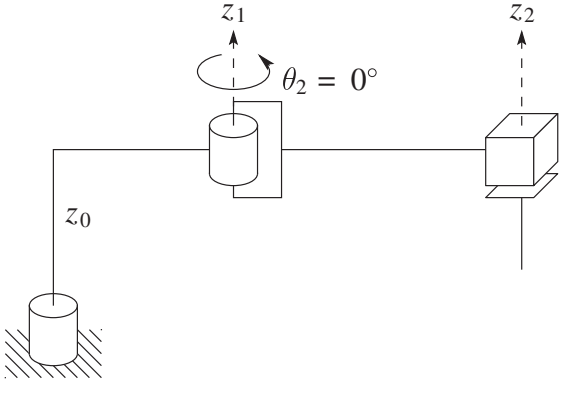


Figure 4.10: SCARA manipulator singularity.

Example 4.10 (Spherical Manipulator)**.** *Consider the spherical arm of Figure 4.9. This manipulator is in a singular configuration when the wrist center intersects z_0 as shown since, as before, any rotation about the base leaves this point fixed.*

Example 4.11 (SCARA Manipulator)**.** *We have already derived the complete Jacobian for the SCARA manipulator. This Jacobian is simple enough to be used directly rather than deriving the modified Jacobian as we have done above. Referring to Figure 4.10 we can see geometrically that the only*

singularity of the SCARA arm is when the elbow is fully extended or fully retracted. Indeed, since the portion of the Jacobian of the SCARA governing arm singularities is given as

$$J_{11} = \begin{bmatrix} \alpha_1 & \alpha_3 & 0 \\ \alpha_2 & \alpha_4 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.101}$$

where

$$\begin{aligned} \alpha_1 &= -a_1 s_1 - a_2 s_{12} \\ \alpha_2 &= a_1 c_1 + a_2 c_{12} \\ \alpha_3 &= -a_1 s_{12} \\ \alpha_4 &= a_1 c_{12} \end{aligned} \tag{4.102}$$

we see that the rank of J_{11} will be less than three when $\alpha_1\alpha_4 - \alpha_2\alpha_3 = 0$. It is easy to compute this quantity and show that it is equivalent to (Problem 4–14)

$$s_2 = 0, \quad \textit{which implies} \quad \theta_2 = 0, \pi \tag{4.103}$$

Note the similarities between this case and the singularities for the elbow manipulator shown in Figure 4.6. In each case, the relevant portion of the arm is merely a two-link planar manipulator. As can be seen from Equation (4.62), the Jacobian for the two-link planar manipulator loses rank when $\theta_2 = 0$ or π.

4.10 Static Force/Torque Relationships

Interaction of the manipulator with the environment produces forces and moments at the end effector or tool. These, in turn, produce torques at the joints of the robot.[3] In this section we discuss the role of the manipulator Jacobian in the quantitative relationship between the end-effector forces and joint torques. This relationship is important for the development of path planning methods in Chapter 7, the derivation of the dynamic equations in Chapter 6 and in the design of force control algorithms in Chapter 10. We can state the main result concisely as follows.

Let $F = (F_x, F_y, F_z, n_x, n_y, n_z)$ represent the vector of forces and moments at the end effector. Let τ denote the corresponding vector of joint torques. Then F and τ are related by

$$\tau = J^T(q)F \tag{4.104}$$

[3]Here, we consider the case of revolute joints. If the joints are prismatic, forces and moments at the end effector produce forces at the joints.

where $J^T(q)$ is the transpose of the manipulator Jacobian.

An easy way to derive this relationship is through the so-called **principle of virtual work**. We will defer a detailed discussion of the principle of virtual work until Chapter 6, where we will introduce the notions of generalized coordinates, holonomic constraints, and virtual constraints in more detail. However, we can give a somewhat informal justification in this section as follows. Let δX and δq represent infinitesimal displacements in the task space and joint space, respectively. These displacements are called **virtual displacements** if they are consistent with any constraints imposed on the system. For example, if the end effector is in contact with a rigid wall, then the virtual displacements in position are tangent to the wall. These virtual displacements are related through the manipulator Jacobian $J(q)$ according to

$$\delta X = J(q)\delta q \tag{4.105}$$

The virtual work δw of the system is

$$\delta w = F^T \delta X - \tau^T \delta q \tag{4.106}$$

Substituting Equation (4.105) into Equation (4.106) yields

$$\delta w = (F^T J - \tau^T)\delta q \tag{4.107}$$

The principle of virtual work says, in effect, that the quantity given by Equation (4.107) is equal to zero if the manipulator is in equilibrium. This leads to the relationship given by Equation (4.104). In other words, the end-effector forces are related to the joint torques by the **transpose** of the manipulator Jacobian.

Example 4.12. *Consider the two-link planar manipulator of Figure 4.11, with a force F applied at the end of link two as shown. The Jacobian of this manipulator is given by Equation (4.62). The resulting joint torques $\tau = (\tau_1, \tau_2)$ are then given as*

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} & 0 & 0 & 0 & 1 \\ -a_2 s_{12} & a_2 c_{12} & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \\ n_x \\ n_y \\ n_z \end{bmatrix} \tag{4.108}$$

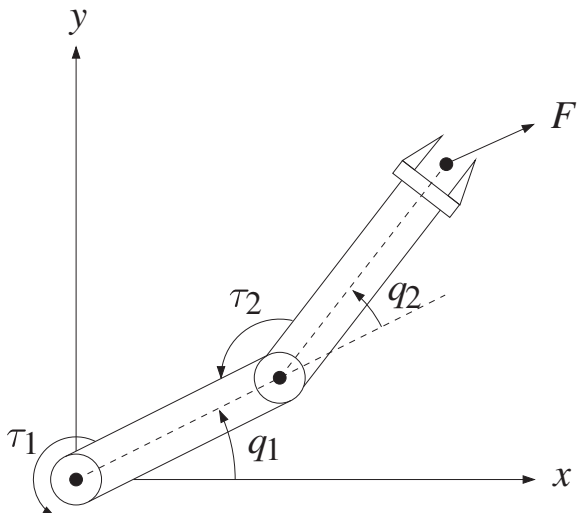


Figure 4.11: Two-link planar robot.

4.11 Inverse Velocity and Acceleration

The Jacobian relationship

$$\xi = J\dot{q} \tag{4.109}$$

specifies the end-effector velocity that will result when the joints move with velocity $\dot{q}$. The inverse velocity problem is the problem of finding the joint velocities $\dot{q}$ that produce the desired end-effector velocity. It is perhaps a bit surprising that the inverse velocity relationship is conceptually simpler than the inverse position relationship. When the Jacobian is square and nonsingular, this problem can be solved by simply inverting the Jacobian matrix to give

$$\dot{q} = J^{-1}\xi \tag{4.110}$$

For manipulators that do not have exactly six joints, the Jacobian cannot be inverted. In this case there will be a solution to Equation (4.109) if and only if ξ lies in the range space of the Jacobian. This can be determined by the following simple rank test. A vector ξ belongs to the range of J if and only if

$$\operatorname{rank} J(q) \quad = \quad \operatorname{rank} [J(q) \mid \xi] \tag{4.111}$$

In other words, Equation (4.109) may be solved for $\dot{q} \in \mathbb{R}^n$ provided that the rank of the **augmented matrix** $[J(q) \mid \xi]$ is the same as the rank of the Jacobian $J(q)$. This is a standard result from linear algebra, and several

algorithms exist, such as Gaussian elimination, for solving such systems of linear equations.

For the case when $n > 6$ we can solve for $\dot{q}$ using the right pseudoinverse of J. We leave it as an exercise to show (Problem 4–20) that a solution to Equation (4.109) is given by

$$\dot{q} = J^{+}\xi + (I - J^{+}J)b \tag{4.112}$$

in which $b \in \mathbb{R}^n$ is an arbitrary vector.

In general, for $m < n$, $(I - J^{+}J) \neq 0$, and all vectors of the form $(I - J^{+}J)b$ lie in the null space of J. This means that, if $\dot{q}'$ is a joint velocity vector such that $\dot{q}' = (I - J^{+}J)b$, then when the joints move with velocity $\dot{q}'$, the end effector will remain fixed since $J\dot{q}' = 0$. Thus, if $\dot{q}$ is a solution to Equation (4.109), then so is $\dot{q} + \dot{q}'$ with $\dot{q}' = (I - J^{+}J)b$, for any value of b. If the goal is to minimize the resulting joint velocities, we choose $b = 0$ (Problem 4–20).

We can construct the right pseudoinverse of J using its singular value decomposition (see Appendix B).

We can apply a similar approach when the analytical Jacobian is used in place of the manipulator Jacobian. Recall from Equation (4.83) that the joint velocities and the end-effector velocities are related by the analytical Jacobian as

$$\dot{X} \;=\; J_a(q)\dot{q} \tag{4.113}$$

Thus, the inverse velocity problem becomes one of solving the linear system given by Equation (4.113), which can be accomplished as above for the manipulator Jacobian.

Inverse Acceleration

Differentiating Equation (4.113) yields an expression for the acceleration

$$\ddot{X} \;=\; J_a(q)\ddot{q} + \left(\frac{d}{dt}J_a(q)\right)\dot{q} \tag{4.114}$$

Thus, given a vector $\ddot{X}$ of end-effector accelerations, the instantaneous joint acceleration vector $\ddot{q}$ is given as a solution of

$$J_a(q)\ddot{q} = \ddot{X} - \left(\frac{d}{dt}J_a(q)\right)\dot{q}$$

For 6-DOF manipulators the inverse velocity and acceleration equations can therefore be written as

$$\dot{q} = J_a(q)^{-1}\dot{X} \tag{4.115}$$

and

$$\ddot{q} = J_a(q)^{-1}\left[\ddot{X} - \left(\frac{d}{dt}J_a(q)\right)\dot{q}\right] \tag{4.116}$$

provided $\det J_a(q) \neq 0$.

4.12 Manipulability

For a specific value of q, the Jacobian relationship defines the linear system given by $\xi = J\dot{q}$. We can think of J as scaling the input $\dot{q}$ to produce the output ξ. It is often useful to characterize quantitatively the effects of this scaling. Often, in systems with a single input and a single output, this kind of characterization is given in terms of the so-called impulse response of a system, which essentially characterizes how the system responds to a unit input. In this multidimensional case, the analogous concept is to characterize the output in terms of an input that has unit norm. Consider the set of all robot joint velocities $\dot{q}$ such that

$$\|\dot{q}\|^2 = \dot{q}_1^2 + \dot{q}_2^2 + \cdots + \dot{q}_n^2 \leq 1 \tag{4.117}$$

If we use the minimum norm solution $\dot{q} = J^+\xi$, we obtain

$$\begin{aligned} \|\dot{q}\|^2 &= \dot{q}^T\dot{q} \\ &= (J^+\xi)^T J^+\xi \\ &= \xi^T(JJ^T)^{-1}\xi \end{aligned} \tag{4.118}$$

The derivation of this is left as an exercise (Problem 4–21). Equation (4.118) gives us a quantitative characterization of the scaling effected by the Jacobian. In particular, if the manipulator Jacobian is full rank, that is, if rank $J = m$, then Equation (4.118) defines an m-dimensional ellipsoid that is known as the **manipulability ellipsoid**. If the input (joint velocity) vector has unit norm, then the output (end-effector velocity) will lie within the ellipsoid given by Equation (4.118). We can more easily see that Equation (4.118) defines an ellipsoid by replacing the Jacobian by its SVD $J = U\Sigma V^T$ (see Appendix B) to obtain

$$\xi^T(JJ^T)^{-1}\xi = (U^T\xi)^T\Sigma_m^{-2}(U^T\xi) \tag{4.119}$$

in which

$$\Sigma_m^{-2} = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_m^{-2} \end{bmatrix}$$

and $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_m \geq 0$. The derivation of Equation (4.119) is left as an exercise (Problem 4–22). If we make the substitution $w = U^T \xi$, then Equation (4.119) can be written as

$$w^T \Sigma_m^{-2} w = \sum \frac{w_i^2}{\sigma_i^2} \leq 1 \tag{4.120}$$

and it is clear that this is the equation for an axis-aligned ellipse in a new coordinate system that is obtained by rotation according to the orthogonal matrix U^T. In the original coordinate system, the axes of the ellipsoid are given by the vectors $\sigma_i u_i$. The volume of the ellipsoid is given by

$$\text{volume} = K \sigma_1 \sigma_2 \cdots \sigma_m$$

in which K is a constant that depends only on the dimension m of the ellipsoid. The manipulability measure is given by

$$\mu = \sigma_1 \sigma_2 \cdots \sigma_m \tag{4.121}$$

Note that the constant K is not included in the definition of manipulability, since it is fixed once the task has been defined, that is, once the dimension of the task space has been fixed.

Now, consider the special case when the robot is not redundant, that is, $J \in \mathbb{R}^{m \times m}$. Recall that the determinant of a product is equal to the product of the determinants, and that a matrix and its transpose have the same determinant. Thus, we have

$$\det\, JJ^T \quad = \quad \lambda_1^2 \lambda_2^2 \cdots \lambda_m^2 \tag{4.122}$$

in which $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ are the eigenvalues of J. This leads to

$$\mu = \sqrt{\det\, JJ^T} = |\lambda_1 \lambda_2 \cdots \lambda_m| = |\det\, J| \tag{4.123}$$

The manipulability μ has the following properties.

- In general, $\mu = 0$ holds if and only if $\text{rank}(J) < m$, that is, when J is not full rank.

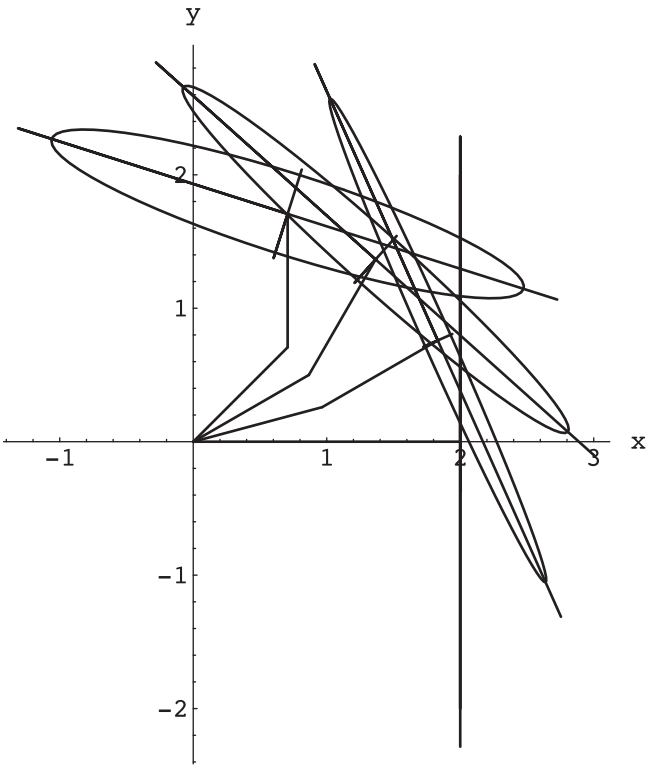


Figure 4.12: Manipulability ellipsoids are shown for several configurations of the two-link arm.

- Suppose that there is some error in the measured velocity $\Delta\xi$. We can bound the corresponding error in the computed joint velocity $\Delta\dot{q}$ by

$$(\sigma_1)^{-1} \leq \frac{\|\Delta\dot{q}\|}{\|\Delta\xi\|} \leq (\sigma_m)^{-1} \tag{4.124}$$

Example 4.13 (Two-link Planar Arm)**.** *Consider the two-link planar arm and the task of positioning in the plane. The Jacobian is given by*

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \tag{4.125}$$

and the manipulability is given by

$$\mu = |det\ J| = a_1 a_2 |s_2|$$

Manipulability ellipsoids for several configurations of the two-link arm are shown in Figure 4.12.

We can use manipulability to determine the optimal configurations in which to perform certain tasks. In some cases it is desirable to perform a task in the configuration for which the end effector has the maximum manipulability. For the two-link arm the maximum manipulability is obtained for $\theta_2 = \pm\pi/2$.

Manipulability can also be used to aid in the design of manipulators. For example, suppose that we wish to design a two-link planar arm whose total link length a_1+a_2 *is fixed. What values should be chosen for* a_1 *and* a_2*? If we design the robot to maximize the maximum manipulability, then we need to maximize* $\mu = a_1 a_2 |s_2|$. *We have already seen that the maximum is obtained when* $\theta_2 = \pm\pi/2$, *so we need only find* a_1 *and* a_2 *to maximize the product* $a_1 a_2$. *This is achieved when* $a_1 = a_2$. *Thus, to maximize manipulability, the link lengths should be chosen to be equal.*

4.13 Chapter Summary

A moving coordinate frame has both a linear and an angular velocity. Linear velocity is associated to a moving point, while angular velocity is associated to a rotating frame. Thus, the linear velocity of a moving frame is merely the velocity of its origin. The angular velocity for a moving frame is related to the time derivative of the rotation matrix that describes the instantaneous orientation of the frame. In particular, if $R(t) \in SO(3)$, then

$$\dot{R}(t) = S(\omega(t))R(t) \tag{4.126}$$

and the vector $\omega(t)$ is the instantaneous angular velocity of the frame. The operator S gives a skew-symmetric matrix

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{4.127}$$

The manipulator Jacobian relates the vector of joint velocities to the body velocity $\xi = (v^T, \omega)$ of the end effector

$$\xi = J\dot{q} \tag{4.128}$$

This relationship can be written as two equations, one for linear velocity and one for angular velocity,

$$v = J_v \dot{q} \tag{4.129}$$

$$\omega = J_\omega \dot{q} \tag{4.130}$$

The i^{th} column of the Jacobian matrix corresponds to the i^{th} joint of the robot manipulator, and takes one of two forms depending on whether the i^{th} joint is prismatic or revolute

$$J_i = \begin{cases} \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{if joint i is revolute} \\ \\ \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{if joint i is prismatic} \end{cases} \tag{4.131}$$

It is often the case that a tool is attached to the end effector. When two frames are rigidly attached, their velocities are related by

$$\xi_A^A = \begin{bmatrix} R_B^A & S(d_B^A)R_B^A \\ 0_{3\times 3} & R_B^A \end{bmatrix} \xi_B^B$$

and this relationship allows us to compute the required end effector velocity to achieve a desired tool velocity.

For a given parameterization of orientation, for example, Euler angles, the analytical Jacobian relates joint velocities to the time derivative of the pose parameters

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix} \qquad \dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}$$

in which $d(q)$ is the usual vector from the origin of the base frame to the origin of the end-effector frame and α denotes a parameterization of the rotation matrix that specifies the orientation of the end-effector frame relative to the base frame. For the Euler angle parameterization, the analytical Jacobian is given by

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B(\alpha)^{-1} \end{bmatrix} J(q) \tag{4.132}$$

in which

$$B(\alpha) = \begin{bmatrix} c_\psi s_\theta & -s_\psi & 0 \\ s_\psi s_\theta & c_\psi & 0 \\ c_\theta & 0 & 1 \end{bmatrix}$$

A configuration at which the Jacobian loses rank, that is, a configuration q such that $\operatorname{rank} J \leq \max_q \operatorname{rank} J(q)$, is called a singularity. For a manipulator with a spherical wrist, the set of singular configurations includes singularities of the wrist (which are merely the singularities in the

Euler angle parameterization) and singularities in the arm. The latter can be found by solving

$$\det J_{11} = 0$$

with J_{11} the upper left 3×3 block of the manipulator Jacobian.

The Jacobian matrix can also be used to relate forces F applied at the end-effector frame to the induced joint torques τ

$$\tau = J^T(q)F$$

For nonsingular configurations, the Jacobian relationship can be used to find the joint velocities $\dot{q}$ necessary to achieve a desired end-effector velocity ξ. The minimum norm solution is given by

$$\dot{q} = J^+\xi$$

in which $J^+ = J^T(JJ^T)^{-1}$ is the right pseudoinverse of J.

Manipulability is defined by $\mu = \sigma_1\sigma_2\cdots\sigma_m$ in which σ_i are the singular values for the manipulator Jacobian. The manipulability can be used to characterize the range of possible end-effector velocities for a given configuration q.

Problems

4–1 Verify Equation (4.6) by direct calculation.

4–2 Verify Equation (4.7) by direct calculation.

4–3 Prove the assertion given in Equation (4.9) that $R(a \times b) = Ra \times Rb$, for $R \in S0(3)$.

4–4 Verify Equation (4.16) by direct calculation.

4–5 Suppose that $a = (1, -1, 2)$ and that $R = R_{x,90}$. Show by direct calculation that $RS(a)R^T = S(Ra)$.

4–6 Given $R = R_{x,\theta}R_{y,\phi}$, compute $\frac{\partial R}{\partial \phi}$. Evaluate $\frac{\partial R}{\partial \phi}$ at $\theta = \frac{\pi}{2}$, $\phi = \frac{\pi}{2}$.

4–7 Given the Euler angle transformation

$$R = R_{z,\phi}R_{y,\theta}R_{z,\psi}$$

show that $\frac{d}{dt}R = S(\omega)R$ where

$$\omega = \{c_\psi s_\theta \dot{\phi} - s_\psi \dot{\theta}\}i + \{s_\psi s_\theta \dot{\phi} + c_\psi \dot{\theta}\}j + \{\dot{\psi} + c_\theta \dot{\phi}\}k$$

The components of i, j, k, respectively, are called the **nutation**, **spin**, and **precession**.

4–8 Repeat Problem 4–7 for the Roll-Pitch-Yaw transformation. In other words, find an explicit expression for ω such that $\frac{d}{dt}R = S(\omega)R$, where R is given by Equation (2.39).

4–9 Section 2.5.1 described only the Z-Y-Z Euler angles. List all possible sets of Euler angles. Is it possible to have Z-Z-Y Euler angles? Why or why not?

4–10 Two frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ are related by the homogeneous transformation

$$H = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A particle has velocity $v_1(t) = (3, 1, 0)$ relative to frame $o_1x_1y_1z_1$. What is the velocity of the particle in frame $o_0x_0y_0z_0$?

4–11 For the three-link planar manipulator of Example 4.6, compute the vector o_c and derive the manipulator Jacobian matrix.

4–12 Compute the Jacobian J_{11} for the 3-link elbow manipulator of Example 4.9 and show that it agrees with Equation (4.97). Show that the determinant of this matrix agrees with Equation (4.98).

4–13 Compute the Jacobian J_{11} for the three-link spherical manipulator of Example 4.10.

4–14 Use Equation (4.101) to show that the singularities of the SCARA manipulator are given by Equation (4.103).

4–15 Find the 6×3 Jacobian for the three links of the cylindrical manipulator of Figure 3.7. Find the singular configurations for this arm.

4–16 Repeat Problem 4–15 for the Cartesian manipulator of Figure 3.17.

4–17 Complete the derivation of the Jacobian for the Stanford Manipulator from Example 4.7.

4–18 Verify Equation (4.80) by direct computation.

4–19 Show that $B(\alpha)$ given by Equation (4.86) is invertible provided $s_\theta \neq 0$.

4–20 Suppose that $\dot{q}$ is a solution to Equation (4.109) for $m < n$.

1. Show that $\dot{q} + (I - J^+ J)b$ is also a solution to Equation (4.109) for any $b \in \mathbb{R}^n$.
2. Show that $b = 0$ gives the solution that minimizes the resulting joint velocities.

4–21 Verify Equation (4.118).

4–22 Verify Equation (4.119).

Notes and References

Angular velocity is fundamentally related to the derivative of a rotation matrix, and therefore to the Lie algebra $so(3)$. This relationship, and more generally the geometry of $so(n)$, is explored in differential geometry texts, as well as in more advanced robotics texts such as [118].

The concept of a Jacobian matrix as a linear mapping from the tangent space of one manifold to the tangent space of a second manifold is also dealt with in differential geometry texts, and even in advanced calculus texts when one or both of the manifolds is a Euclidean space. The use of the geometric Jacobian matrix of Equation (4.41) to map joint velocities (which lie in the tangent space to the configuration space) to a velocity $\xi = (v, \omega)$ is not commonly found in mathematics texts (note that ξ is not itself the derivative of any quantity). However, most robotics texts include some description of the geometric Jacobian, including [134], [145], and [118].

Since the relationship between the end-effector velocity and the joint velocities is defined by a linear map, the inverse velocity problem is a special case of the more general problem of solving linear systems, a problem that is the subject of linear algebra. Algorithms for solving this problem can be found in a variety of texts, including [138] and [56].

The manipulability measure discussed in Section 4.12 is due to Yoshikawa [187].

Chapter 5

INVERSE KINEMATICS

In Chapter 3 we showed how to determine the end effector's position and orientation in terms of the joint variables. This chapter is concerned with the inverse problem, that of finding the joint variables in terms of the end effector's position and orientation. This is the problem of **inverse kinematics**, and it is, in general, more difficult than the forward kinematics problem.

We begin by formulating the general inverse kinematics problem. Following this, we describe the principle of kinematic decoupling and how it can be used to simplify the inverse kinematics of most modern manipulators that are equipped with spherical wrists. Using kinematic decoupling, we can consider the position and orientation problems independently. We describe a geometric approach for solving the positioning problem, while we exploit the Euler angle parameterization to solve the orientation problem.

We also discuss numerical solution of the inverse kinematics using methods based on both the Jacobian inverse and the Jacobian transpose. The Jacobian inverse method is similar to a Newton–Raphson search whereas the Jacobian transpose method is a gradient search method.

5.1 The General Inverse Kinematics Problem

The general problem of inverse kinematics can be stated as follows. Given a 4×4 homogeneous transformation

$$H = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{5.1}$$

find a solution, or possibly multiple solutions, of the equation

$$T_n^0(q_1, \ldots, q_n) = H \tag{5.2}$$

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

where

$$T_n^0(q_1, \ldots, q_n) \quad = \quad A_1(q_1) \cdots A_n(q_n) \tag{5.3}$$

specifies the forward kinematics of an n-degree-of-freedom manipulator. Here the homogeneous transformation H represents the desired position and orientation of the end effector, and our task is to find the values for the joint variables $q_1, \ldots, q_n$ so that $T_n^0(q_1, \ldots, q_n) = H$.

Equation (5.2) yields sixteen equations to be solved for the n unknown variables, $q_1, \ldots, q_n$. However, since the bottom row of both T_n^0 and H are (0,0,0,1), four of these sixteen equations are trivial and so we can express the remaining twelve nonlinear equations as

$$T_{ij}(q_1, \ldots, q_n) = h_{ij}, \qquad \text{for } i = 1, 2, 3, \quad j = 1, 2, 3, 4 \tag{5.4}$$

Example 5.1 (The Stanford Manipulator)**.** *Recall the Stanford Manipulator of Section 3.3.5. Suppose that the desired position and orientation of the final frame are given by*

$$H \quad = \quad \begin{bmatrix} 0 & 1 & 0 & -0.154 \\ 0 & 0 & 1 & 0.763 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

To find the corresponding joint variables θ_1, θ_2, d_3, θ_4, θ_5, and θ_6 we must solve the following simultaneous set of nonlinear trigonometric equations:

$$\begin{aligned}
c_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] - s_1(s_4c_5c_6 + c_4s_6) &= 0 \\
s_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] + c_1(s_4c_5c_6 + c_4s_6) &= 0 \\
-s_2(c_4c_5c_6 - s_4s_6) - c_2s_5c_6 &= 1 \\
c_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] - s_1(-s_4c_5s_6 + c_4c_6) &= 1 \\
s_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] + c_1(-s_4c_5s_6 + c_4c_6) &= 0 \\
s_2(c_4c_5s_6 + s_4c_6) + c_2s_5s_6 &= 0 \\
c_1(c_2c_4s_5 + s_2c_5) - s_1s_4s_5 &= 0 \\
s_1(c_2c_4s_5 + s_2c_5) + c_1s_4s_5 &= 1 \\
-s_2c_4s_5 + c_2c_5 &= 0 \\
c_1s_2d_3 - s_1d_2 + d_6(c_1c_2c_4s_5 + c_1c_5s_2 - s_1s_4s_5) &= -0.154 \\
s_1s_2d_3 + c_1d_2 + d_6(c_1s_4s_5 + c_2c_4s_1s_5 + c_5s_1s_2) &= 0.763 \\
c_2d_3 + d_6(c_2c_5 - c_4s_2s_5) &= 0
\end{aligned}$$

If the values of the nonzero DH parameters are $d_2 = 0.154$ *and* $d_6 = 0.263$*, one solution to this set of equations is given by:*

$$\theta_1 = \pi/2, \quad \theta_2 = \pi/2, \quad d_3 = 0.5, \quad \theta_4 = \pi/2, \quad \theta_5 = 0, \quad \theta_6 = \pi/2.$$

Even though we have not yet seen how one might derive this solution, it is not difficult to verify that it satisfies the forward kinematics equations for the Stanford Arm.

The equations in the preceding example are, of course, much too difficult to solve directly in closed form. This is the case for most robot arms. Therefore, we need to develop efficient and systematic techniques that exploit the particular kinematic structure of the manipulator. Whereas the forward kinematics problem always has a unique solution that can be obtained simply by evaluating the forward equations, the inverse kinematics problem may or may not have a solution. Even if a solution exists, it may or may not be unique. Furthermore, because these forward kinematic equations are in general complicated nonlinear functions of the joint variables, the solutions may be difficult to obtain even when they exist.

The inverse kinematics problem may be solved numerically or in closed form. Finding a closed-form solution means finding an explicit relationship

$$q_k \quad = \quad f_k(h_{11}, \ldots, h_{34}), \qquad k = 1, \ldots, n \tag{5.6}$$

We first discuss the computation of closed-form solutions. In Section 5.5 we discuss numerical algorithms to find inverse kinematics solutions.

The practical question of the existence of solutions to the inverse kinematics problem depends on engineering as well as mathematical considerations. For example, the motion of the revolute joints may be restricted to less than a full 360 degrees of rotation so that not all mathematical solutions of the kinematic equations will correspond to physically realizable configurations of the manipulator. We will assume that the given position and orientation is such that at least one solution of Equation (5.2) exists. Once a solution to the mathematical equations is identified, it must be further checked to see whether or not it satisfies all constraints on the ranges of possible joint motions.

5.2 Kinematic Decoupling

Although the general problem of inverse kinematics is quite difficult, it turns out that for manipulators having six joints with three consecutive three joint axes intersecting at a point (such as the Stanford Manipulator above), it is

possible to decouple the inverse kinematics problem into two simpler problems, known respectively as **inverse position kinematics**, and **inverse orientation kinematics**. To put it another way, for a six-DOF manipulator with a spherical wrist, the inverse kinematics problem may be separated into two simpler problems, namely, first finding the position of the intersection of the wrist axes, hereafter called the **wrist center**, and then finding the orientation of the wrist.

For concreteness let us suppose that there are exactly six degrees of freedom and that the last three joint axes intersect at a point o_c. We express Equation (5.2) as two sets of equations representing the rotational and positional equations

$$R_6^0(q_1, \ldots, q_6) = R \tag{5.7}$$

$$o_6^0(q_1, \ldots, q_6) = o \tag{5.8}$$

where o and R are the desired position and orientation of the tool frame, expressed with respect to the world coordinate system. Thus, we are given o and R, and the inverse kinematics problem is to solve for $q_1, \ldots, q_6$.

The assumption of a spherical wrist means that the axes z_3, z_4, and z_5 intersect at o_c and hence the origins o_4 and o_5 assigned by the DH convention will always be at the wrist center o_c. Often o_3 will also be at o_c, but this is not necessary for our subsequent development. The important point of this assumption for the inverse kinematics is that motion of the final three joints about these axes will not change the position of o_c, and thus the position of the wrist center is a function of only the first three joint variables.

The origin of the tool frame (whose desired coordinates are given by o) is simply obtained by a translation of distance d_6 along z_5 from o_c (see Table 3.3). In our case, z_5 and z_6 are the same axis, and the third column of R expresses the direction of z_6 with respect to the base frame. Therefore, we have

$$o = o_c^0 + d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.9}$$

Thus, in order to have the end effector of the robot at the point with coordinates given by o and with the orientation of the end effector given by $R = (r_{ij})$, it is necessary and sufficient that the wrist center o_c have coordinates given by

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.10}$$

and that the orientation of the frame $o_6x_6y_6z_6$ with respect to the base be given by R. If the components of the end effector's position o are denoted o_x, o_y, o_z and the components of the wrist center o_c^0 are denoted x_c, y_c, z_c then Equation (5.10) gives the relationship

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix} \tag{5.11}$$

Using Equation (5.11) we may find the values of the first three joint variables. This determines the orientation transformation R_3^0 which depends only on these first three joint variables. We can now determine the orientation of the end effector relative to the frame $o_3x_3y_3z_3$ from the expression

$$R = R_3^0 R_6^3 \tag{5.12}$$

as

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R \tag{5.13}$$

As we shall see in Section 5.4, the final three joint angles can then be found as a set of Euler angles corresponding to R_6^3. Note that the right-hand side of Equation (5.13) is completely known since R is given and R_3^0 can be calculated once the first three joint variables are known. The idea of kinematic decoupling is illustrated in Figure 5.1.

5.3 Inverse Position: A Geometric Approach

For the most common kinematic arrangements that we consider, we can use a geometric approach to find the variables q_1, q_2, q_3 corresponding to o_c^0 given by Equation (5.10). Since most six-DOF manipulator designs are kinematically simple, usually consisting of one of the five basic configurations of Chapter 1 with a spherical wrist, the geometric approach is simple and effective. Indeed, it is partly due to the difficulty of the general inverse kinematics problem that manipulator designs have evolved to their present state.

In general, the complexity of the inverse kinematics problem increases with the number of nonzero DH parameters. For most manipulators, many of the a_i, d_i are zero, the α_i are 0 or $\pm\pi/2$, etc. In these cases especially, a geometric approach is the simplest and most natural. The general idea of the geometric approach is to solve for joint variable q_i by projecting the manipulator onto the $x_{i-1} - y_{i-1}$ plane and solving a simple trigonometry

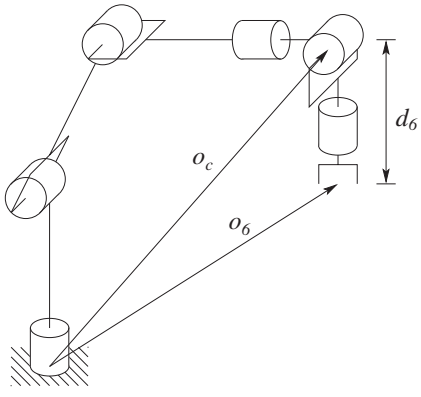


Figure 5.1: Kinematic decoupling in the case of a spherical wrist. The vector o_c is the position of the wrist center point and o_6 is the position of the end effector, both with respect to the base frame. The wrist center point coordinates do not depend on the wrist orientation variables θ_4, θ_5, and θ_6.

problem. For example, to solve for θ_1, we project the arm onto the $x_0 - y_0$ plane and use trigonometry to find θ_1. We will illustrate this method with two important examples: the spherical (RRP) and the articulated (RRR) arms.

5.3.1 Spherical Configuration

We first solve the inverse position kinematics for a three degree of freedom spherical manipulator shown in Figure 5.2, with the components of $o_c = o_c^0$ denoted by x_c, y_c, z_c. Projecting o_c onto the $x_0 - y_0$ plane, we see that

$$\theta_1 = \text{Atan2}(x_c, y_c) \tag{5.14}$$

in which Atan2 denotes the two-argument arctangent function defined in Appendix A. Note that a second valid solution for θ_1 is

$$\theta_1 = \pi + \text{Atan2}(x_c, y_c) \tag{5.15}$$

Of course, this will, in turn, lead to a different solution for θ_2.

These solutions for θ_1, are valid unless $x_c = y_c = 0$. In this case, Equation (5.14) is undefined and the manipulator is in a singular configuration, in which the wrist center o_c intersects z_0 as shown in Figure 5.3. In this configuration any value of θ_1 leaves o_c fixed. There are thus infinitely many solutions for θ_1 when o_c intersects z_0.

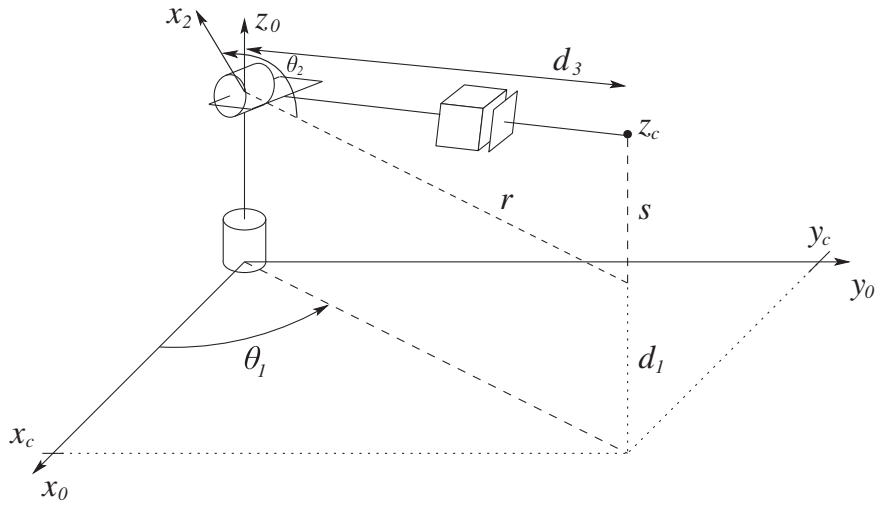


Figure 5.2: First three joints of a spherical manipulator.

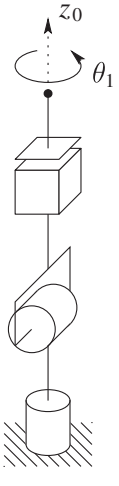


Figure 5.3: Singular configuration for a spherical manipulator in which the wrist center lies on the z_0 axis.

The angle θ_2 is given from Figure 5.2 as

$$\theta_2 = \text{Atan2}(r, s) + \frac{\pi}{2} \tag{5.16}$$

where $r^2 = x_c^2 + y_c^2$ and $x = z_c - d_1$.

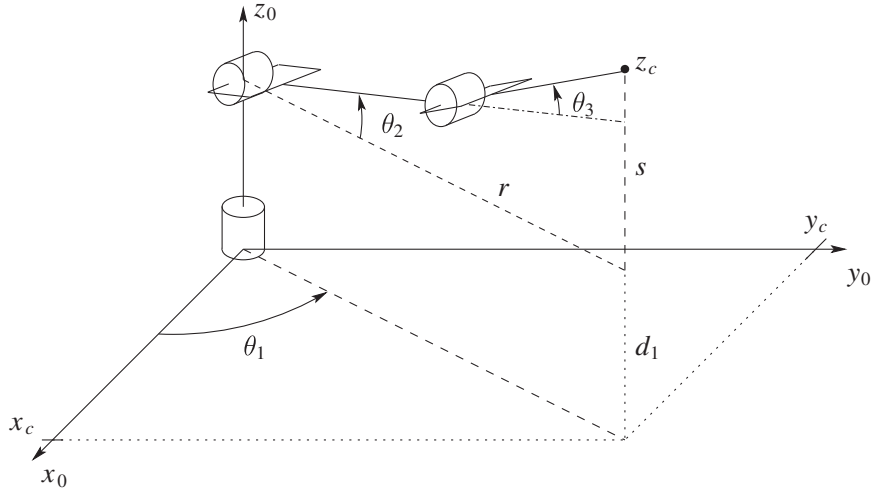


Figure 5.4: First three joints of an elbow manipulator.

The linear distance d_3 is found as

$$d_3 = \sqrt{r^2 + s^2} \quad = \quad \sqrt{x_c^2 + y_c^2 + (z_c - d_1)^2} \tag{5.17}$$

The negative square root solution for d_3 is disregarded and thus in this case we obtain two solutions to the inverse position kinematics as long as the wrist center does not intersect z_0.

5.3.2 Articulated Configuration

We next consider the elbow manipulator shown in Figure 5.4, As in the case of the spherical manipulator, the first joint variable is the base rotation and there are two possible solutions

$$\theta_1 = \text{Atan2}(x_c, y_c) \tag{5.18}$$

$$\theta_1 = \pi + \text{Atan2}(x_c, y_c) \tag{5.19}$$

provided x_c and y_c are not both zero.

If both x_c and y_c are zero, as shown in Figure 5.5, the configuration is singular as before and θ_1 may take on any value.

If there is an offset $d \neq 0$ as shown in Figure 5.6 then the wrist center cannot intersect z_0. In this case, depending on how the DH parameters have been assigned, we will have $d_2 = d$ or $d_3 = d$, and there will, in general, be only two solutions for θ_1.

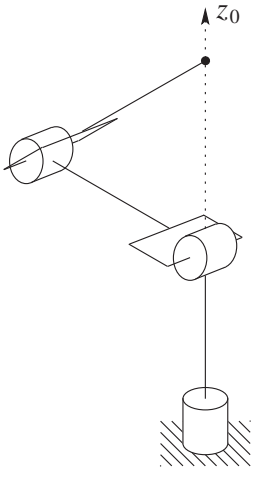


Figure 5.5: Singular configuration for an elbow manipulator in which the wrist center lies on the z_0 axis.

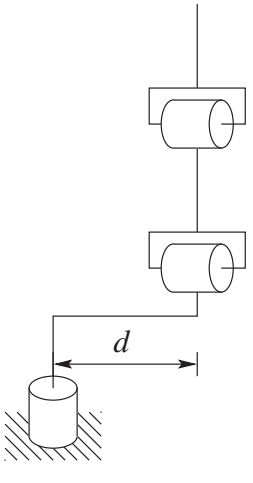


Figure 5.6: Elbow manipulator with shoulder offset.

These correspond to the so-called **left arm** and **right arm** configurations as shown in Figures 5.7.

For the left arm configuration in Figure 5.7 we see geometrically that

$$\theta_1 \quad = \quad \phi - \alpha \tag{5.20}$$

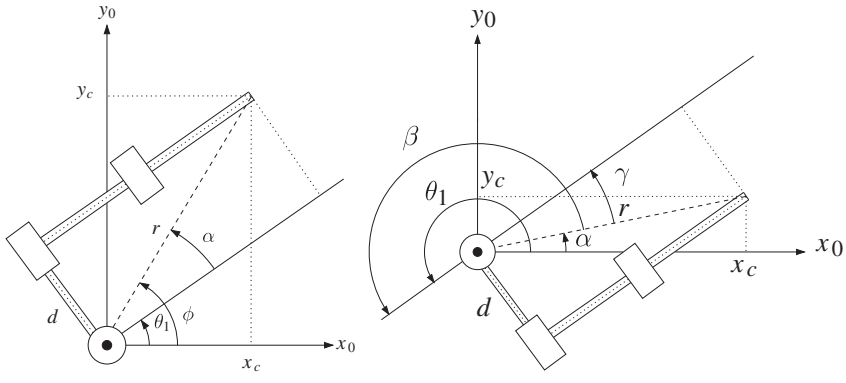


Figure 5.7: Left arm (left) and right arm (right) configurations for an elbow manipulator with an offset.

in which

$$\begin{aligned} \phi &= \text{Atan2}(x_c, y_c) && (5.21) \\ \alpha &= \text{Atan2}\left(\sqrt{r^2 - d^2}, d\right) && (5.22) \\ &= \text{Atan2}\left(\sqrt{x_c^2 + y_c^2 - d^2}, d\right) \end{aligned}$$

The second solution, given by the right arm configuration in Figure 5.7 is given by

$$\theta_1 = \text{Atan2}(x_c, y_c) + \text{Atan2}\left(-\sqrt{r^2 - d^2}, -d\right) \quad (5.23)$$

To see this, note that

$$\begin{aligned} \theta_1 &= \alpha + \beta \\ \alpha &= \text{Atan2}(x_c, y_c) \\ \beta &= \gamma + \pi \\ \gamma &= \text{Atan2}\left(\sqrt{r^2 - d^2}, d\right) \end{aligned}$$

which together imply that

$$\beta = \text{Atan2}\left(-\sqrt{r^2 - d^2}, -d\right)$$

since $\cos(\theta + \pi) = -\cos(\theta)$ and $\sin(\theta + \pi) = -\sin(\theta)$.

To find the angles θ_2, θ_3 for the elbow manipulator given θ_1, we consider the plane formed by the second and third links as shown in Figure 5.8. Since

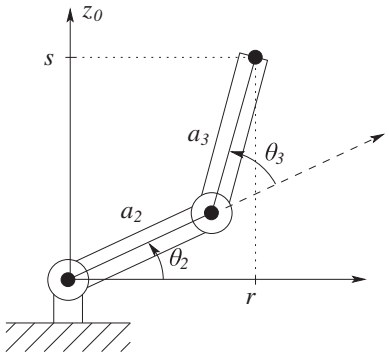


Figure 5.8: Projecting onto the plane formed by links 2 and 3.

the motion of second and third links is planar, the solution is analogous to that of the two-link manipulator of Chapter 1. As in our previous derivation (cf. Equations (1.10) and (1.11)), we can apply the law of cosines to obtain

$$\begin{aligned} \cos\theta_3 &= \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2a_3} \\ &= \frac{x_c^2 + y_c^2 - d^2 + (z_c - d_1)^2 - a_2^2 - a_3^2}{2a_2a_3} := D \end{aligned} \tag{5.24}$$

since $r^2 = x_c^2 + y_c^2 - d^2$ and $s = z_c - d_1$. Hence, θ_3 is given by

$$\theta_3 = \text{Atan2}\left(D, \pm\sqrt{1 - D^2}\right) \tag{5.25}$$

The two solutions for θ_3 correspond to the elbow down position and elbow up position, respectively. Similarly θ_2 is given as

$$\begin{aligned} \theta_2 &= \text{Atan2}(r, s) - \text{Atan2}(a_2 + a_3c_3, a_3s_3) \\ &= \text{Atan2}\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c - d_1\right) - \text{Atan2}(a_2 + a_3c_3, a_3s_3) \end{aligned} \tag{5.26}$$

An example of an elbow manipulator with offsets is the PUMA shown in Figure 5.9. There are four solutions to the inverse position kinematics as shown. These correspond to the situations left arm–elbow up, left arm–elbow down, right arm–elbow up and right arm–elbow down. We will see that there are two solutions for the wrist orientation thus giving a total of eight solutions of the inverse kinematics for the PUMA manipulator.

5.4 Inverse Orientation

In the previous section we used a geometric approach to solve the inverse position problem. This gives the values of the first three joint variables

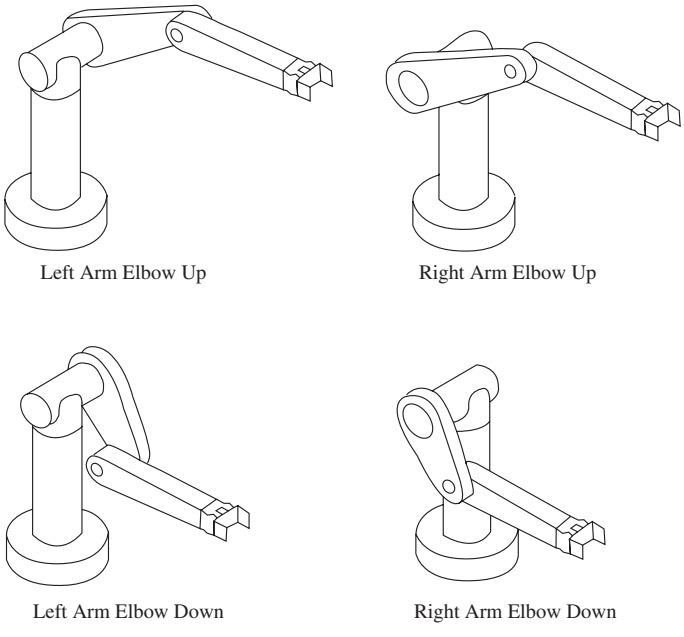


Figure 5.9: Four solutions of the inverse position kinematics for the PUMA manipulator.

corresponding to a given position of the wrist center. The inverse orientation problem is now one of finding the values of the final three joint variables corresponding to a given orientation with respect to the frame $o_3x_3y_3z_3$. For a spherical wrist, this can be interpreted as the problem of finding a set of Euler angles corresponding to a given rotation matrix R. Recall that Equation (3.15) shows that the rotation matrix obtained for the spherical wrist has the same form as the rotation matrix for the Euler transformation given in Equation (2.27). Therefore, we can use the method developed in Section 2.5.1 to solve for the three joint angles of the spherical wrist. In particular, we solve for the three Euler angles, ϕ, θ, ψ, using Equations (2.29)–(2.34), and then use the mapping

$$\theta_4 = \phi, \theta_5 = \theta, \theta_6 = \psi$$

Example 5.2 (Articulated Manipulator with Spherical Wrist)**.** *The DH parameters for the frame assignment shown in Figure 5.4 are summarized in Table 5.1. Multiplying the corresponding A_i matrices gives the matrix R_3^0*

Table 5.1: DH parameters for the articulated manipulator of Figure 5.4.

Link	a_i	α_i	d_i	θ_i
1	0	90	d_1	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3

for the articulated or elbow manipulator as

$$R_3^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 \\ s_1c_{23} & -s_1s_{23} & -c_1 \\ s_{23} & c_{23} & 0 \end{bmatrix} \tag{5.27}$$

The matrix R_6^3 is the upper left 3×3 submatrix of the matrix product $A_4A_5A_6$ given by

$$R_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix} \tag{5.28}$$

The equation to be solved for the final three variables is therefore

$$R_6^3 = (R_3^0)^T R \tag{5.29}$$

and the Euler angle solution can be applied to this equation. For example, the three equations given by the third column in the above matrix equation are given by

$$c_4s_5 = c_1c_{23}r_{13} + s_1c_{23}r_{23} + s_{23}r_{33} \tag{5.30}$$

$$s_4s_5 = -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33} \tag{5.31}$$

$$c_5 = s_1r_{13} - c_1r_{23} \tag{5.32}$$

Hence, if both of Equations (5.30) and (5.31) are not zero, we obtain θ_5 from Equations (2.29) and (2.30) as

$$\theta_5 = \text{Atan2}\left(s_1r_{13} - c_1r_{23}, \pm\sqrt{1 - (s_1r_{13} - c_1r_{23})^2}\right) \tag{5.33}$$

If the positive square root is chosen in Equation (5.33), then θ_4 and θ_6 are given by Equations (2.31) and (2.32), respectively, as

$$\theta_4 = \text{Atan2}(c_1c_{23}r_{13} + s_1c_{23}r_{23} + s_{23}r_{33}, -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33}) \tag{5.34}$$

$$\theta_6 = \text{Atan2}(-s_1r_{11} + c_1r_{21}, s_1r_{12} - c_1r_{22}) \tag{5.35}$$

The other solutions are obtained analogously. If $s_5 = 0$, then joint axes z_3 and z_5 are collinear. This is a singular configuration and only the sum $\theta_4 + \theta_6$ can be determined. One solution is to choose θ_4 arbitrarily and then determine θ_6 using Equation (2.36) or (2.38).

Example 5.3 (Elbow Manipulator — Complete Solution)**.** *To summarize the geometric approach for solving the inverse kinematics equations, we give here one solution to the inverse kinematics of the six degree-of-freedom elbow manipulator shown in Figure 5.4, which has no joint offsets and a spherical wrist.*

Given

$$o = \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{5.36}$$

then with

$$x_c = o_x - d_6 r_{13} \tag{5.37}$$
$$y_c = o_y - d_6 r_{23} \tag{5.38}$$
$$z_c = o_z - d_6 r_{33} \tag{5.39}$$

a set of DH joint variables is given by

$$\theta_1 = \text{Atan2}(x_c, y_c) \tag{5.40}$$
$$\theta_2 = \text{Atan2}\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c - d_1\right) - \text{Atan2}(a_2 + a_3 c_3, a_3 s_3) \tag{5.41}$$
$$\theta_3 = \text{Atan2}\left(D, \pm\sqrt{1 - D^2}\right), \quad \text{with } D = \frac{x_c^2 + y_c^2 - d^2 + (z_c - d_1)^2 - a_2^2 - a_3^2}{2a_2 a_3} \tag{5.42}$$
$$\theta_4 = \text{Atan2}(c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33}, -c_1 s_{23} r_{13} - s_1 s_{23} r_{23} + c_{23} r_{33}) \tag{5.43}$$
$$\theta_5 = \text{Atan2}\left(s_1 r_{13} - c_1 r_{23}, \pm\sqrt{1 - (s_1 r_{13} - c_1 r_{23})^2}\right) \tag{5.44}$$
$$\theta_6 = \text{Atan2}(-s_1 r_{11} + c_1 r_{21}, s_1 r_{12} - c_1 r_{22}) \tag{5.45}$$

The other possible solutions are left as an exercise (Problem 5–10).

Example 5.4 (SCARA Manipulator)**.** *As another example, consider the SCARA manipulator illustrated in Figure 5.10, with forward kinematics defined by T_4^0 from Equation (3.24). The inverse kinematics solution is then*

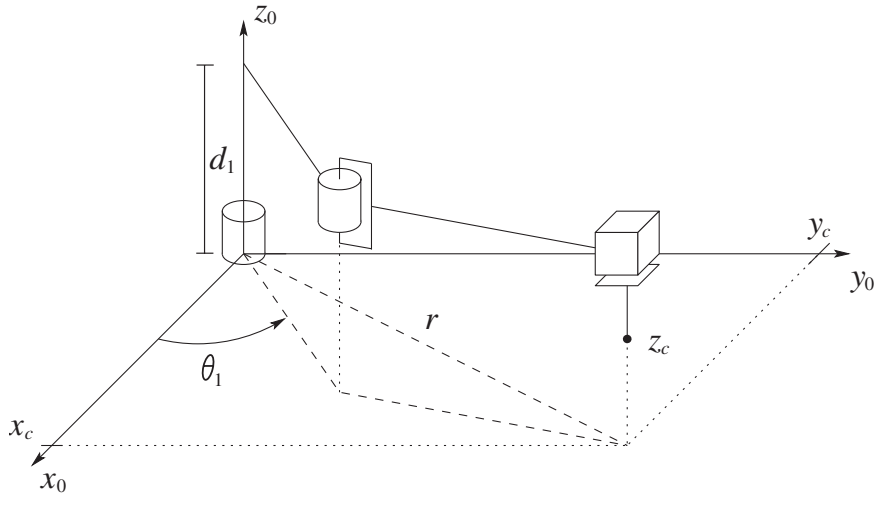


Figure 5.10: First three joints of a SCARA manipulator.

given as the set of solutions of the equation

$$\begin{aligned} T_4^0 &= \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_{12}c_4 + s_{12}s_4 & s_{12}c_4 - c_{12}s_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -c_{12}c_4 - s_{12}s_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5.46}$$

We first note that, since the SCARA has only four degrees of freedom, not every possible H from $SE(3)$ allows a solution of Equation (5.46). In fact we can easily see that there is no solution of Equation (5.46) unless R is of the form

$$R = \begin{bmatrix} c_\alpha & s_\alpha & 0 \\ s_\alpha & -c_\alpha & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5.47}$$

and if this is the case, the sum $\theta_1 + \theta_2 - \theta_4$ is determined by

$$\theta_1 + \theta_2 - \theta_4 = \alpha = \text{Atan2}(r_{11}, r_{12}) \tag{5.48}$$

Projecting the manipulator configuration onto the $x_0 - y_0$ plane yields

the geometry shown in Figure 5.10. Using the law of cosines

$$c_2 = \frac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{5.49}$$

$$\tag{5.50}$$

and

$$\theta_2 = \text{Atan2}\left(c_2, \pm\sqrt{1 - c_2^2}\right) \tag{5.51}$$

The value for θ_1 is then obtained as

$$\theta_1 = \text{Atan2}(o_x, o_y) - \text{Atan2}(a_1 + a_2c_2, a_2s_2) \tag{5.52}$$

We may now determine θ_4 from Equation (5.48) as

$$\begin{aligned} \theta_4 &= \theta_1 + \theta_2 - \alpha \\ &= \theta_1 + \theta_2 - \text{Atan2}(r_{11}, r_{12}) \end{aligned} \tag{5.53}$$

Finally d_3 is given as

$$d_3 = d_1 - o_z - d_4 \tag{5.54}$$

5.5 Numerical Inverse Kinematics

For the manipulators considered in the previous sections we derived closed-form solutions for the inverse kinematics. In this section, we consider iterative, numerical algorithms for the computation of the inverse kinematics. Numerical methods are increasingly popular due to the availability of high-performance computation and the advent of open-source software. In addition, in cases where closed-form solutions do not exist, or if the manipulator is redundant, a recourse to numerical methods may be the better option.

Let $x^d \in \mathbb{R}^m$ be a vector of Cartesian coordinates. For example, x^d could represent the wrist center point ($m = 3$) or the end-effector position and orientation using a minimal representation for the end-effector orientation ($m = 6$). The forward kinematics for an n-link manipulator, in this case, is a function $f : \mathbb{R}^n \to \mathbb{R}^m$. If we set

$$G(q) = x^d - f(q) \tag{5.55}$$

then a solution to the inverse kinematics is a configuration q^d satisfying $G(q^d) = x^d - f(q^d) = 0$. Below we will give details of the most common algorithms to iteratively solve for q^d given x^d; the first based on the Jacobian inverse, which is similar to the Newton–Raphson method for root finding, and the second is based on the Jacobian transpose and is derived as a gradient search algorithm.

Jacobian Inverse Method

With x^d given as a desired robot configuration, we expand the forward kinematics function $f(q)$ in a Taylor series about a configuration q^d, where $x^d = f(q^d)$ to obtain

$$f(q) = f(q^d) + J(q^d)(q - q^d) + h.o.t. \tag{5.56}$$

where we take $J = J_a(q)$ as the analytic Jacobian in Equation (4.83). Neglecting the higher order terms (h.o.t.), we have

$$q^d - q = J^{-1}(q)(x^d - f(q)) \tag{5.57}$$

assuming that the Jacobian is square and invertible. To find a solution for q^d, we begin with an initial guess, q_0, and form a sequence of successive estimates, $q_0, q_1, q_2, \ldots$, as

$$q_k = q_{k-1} + \alpha_k J^{-1}(q_{k-1})(x^d - f(q_{k-1})),\ k = 1, 2, \ldots \tag{5.58}$$

Note that we have introduced a **step size**, $\alpha_k > 0$, into the equation, which can be adjusted to aid convergence. The step size α_k may be chosen as a constant or as a function of k, a scalar or as a diagonal matrix, the last in order to scale each component of the configuration separately.

Remark 5.1. *Since Equation (5.58) is based on a first-order approximation of the inverse kinematics, only local convergence can be expected. Also, since there are generally multiple solutions to the inverse kinematics, the particular configuration that results from running the algorithm is dependent on the initial guess.*

Figure (5.11) shows the results obtained for a two-link, planar RR manipulator. The algorithm converges to within 10^{-4} of the exact solution after 10 iterations with the given parameters.

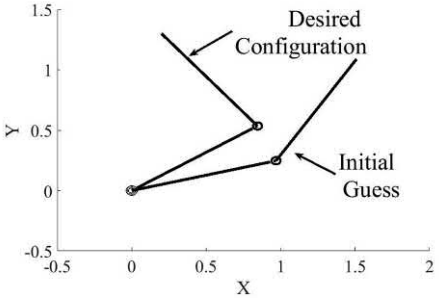
If the Jacobian is not square or not invertible, then one may use the pseudoinverse $J^\dagger = J_a^\dagger$ in place of J_a^{-1}. For $m \leq n$, we defined the right pseudoinverse in Appendix B as $J^\dagger = J^T(JJ^T)^{-1}$. In this case, we can define the update rule for q_k as

$$q_k = q_{k-1} + \alpha_k J^\dagger(q_{k-1})(f(q^d) - f(q_{k-1})) \tag{5.59}$$

Jacobian Transpose Method

The second method that we outline is based on the Jacobian transpose $J^T(q)$ instead of the Jacobian inverse. To begin, we define an optimization problem

$$\min_q F(q) = \min_q \frac{1}{2}(f(q) - x^d)^T(f(q) - x^d) \tag{5.60}$$



Iteration	θ_1	θ_2
1	−0.33284	2.6711
2	0.80552	2.1025
3	0.46906	1.9316
4	0.53554	1.7697
5	0.55729	1.7227
6	0.56308	1.7104
7	0.56455	1.7073
8	0.56492	1.7065
9	0.56501	1.7063
10	0.56503	1.7062

Figure 5.11: Inverse kinematics solution using the Jacobian inverse. Desired end-effector coordinates are $x^d = (0.2, 1.3)$. The joint variables corresponding to x^d are $\theta_1 = 0.5650$, $\theta_2 = 1.7062$. The initial guess is $\theta_1 = 0.25$, $\theta_2 = 0.75$. The step size α was chosen as 0.75.

where, as above, x^d is the desired configuration and $f(q)$ is the forward kinematics map. The gradient of the above cost function $F(q)$ is given by

$$\nabla F(q) = J^T(q)(f(q) - x^d) \tag{5.61}$$

A **gradient descent** algorithm to minimize $F(q)$ (see Appendix D) is then

$$q_k = q_{k-1} - \alpha_k \nabla F(q_{k-1}) = q_{k-1} - \alpha_k J^T(q_{k-1})(f(q_{k-1}) - x^d) \tag{5.62}$$

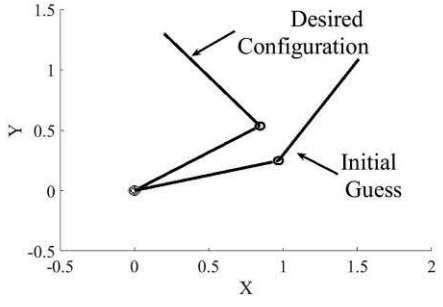
where, again, $\alpha_k > 0$ is the step size.

Remark 5.2. *An advantage of this method is that the Jacobian transpose is easier to compute than the Jacobian inverse and does not suffer from configuration singularities. In general, however, the convergence, in terms of number of iterations, may be slower with this method.*

Figure 5.12 shows the response for the two-link RR manipulator for the same desired configuration and initial guess as in Figure 5.11. In this case, the algorithm converges after 30 iterations.

5.6 Chapter Summary

This DH convention defines the forward kinematics equations for a manipulator, i.e., the mapping from joint variables to end effector position and orientation. To control a manipulator, it is necessary to solve the inverse



Iteration	θ_1	θ_2
1	1.8362	1.3412
2	0.4667	1.1025
3	1.1215	1.6233
4	0.45264	1.415
5	0.83519	1.7273
26	0.56522	1.7063
27	0.56492	1.7061
28	0.56514	1.7063
29	0.56498	1.7062
30	0.5650	1.7062

Figure 5.12: Inverse kinematics solution using the Jacobian transpose. Desired end-effector coordinates are $x^d = (0.2, 1.3)$. The joint variables corresponding to x^d are $\theta_1 = 0.5650$, $\theta_2 = 1.7062$. The initial guess is $\theta_1 = 0.25$, $\theta_2 = 0.75$. The step size α was chosen as 0.75.

problem, i.e., given a position and orientation for the end effector, solve for the corresponding set of joint variables.

In this chapter we considered the special case of manipulators for which kinematic decoupling can be used (e.g., a manipulator with a spherical wrist). For this class of manipulators the determination of the inverse kinematics can be summarized by the following algorithm.

Step 1: Find q_1, q_2, q_3 such that the wrist center o_c has coordinates given by

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.63}$$

Step 2: Using the joint variables determined in Step 1, evaluate R_3^0.

Step 3: Find a set of Euler angles corresponding to the rotation matrix

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R \tag{5.64}$$

We demonstrated a geometric approach for Step 1. In particular, to solve for joint variable q_i, we project the manipulator (including the wrist center) onto the $x_{i-1} - y_{i-1}$ plane and use trigonometry to find q_i.

We also discuss numerical methods for computing the inverse kinematics using the **inverse Jacobian** and the **Jacobian transpose**. The Jacobian

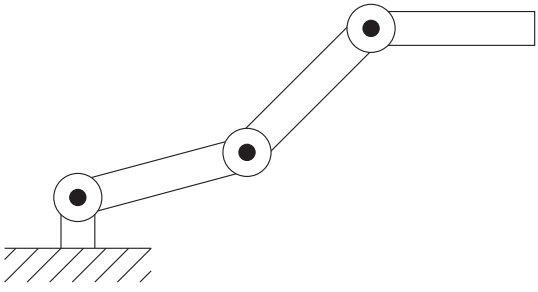


Figure 5.13: Three-link planar robot with revolute joints.

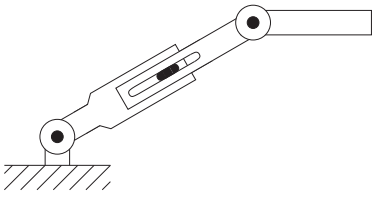


Figure 5.14: Three-link planar robot with prismatic joint.

inverse kinematics algorithm takes the form

$$q_k = q_{k-1} + \alpha_k J^{-1}(q_{k-1})(x^d - f(q_{k-1})),\ k = 1, 2, \ldots \tag{5.65}$$

The Jacobian transpose inverse kinematics algorithm takes the form

$$q_k = q_{k-1} - \alpha_k J^T(q_{k-1})(f(q_{k-1}) - x^d),\ k = 1, 2, \ldots \tag{5.66}$$

Problems

5–1 Given a desired position of the end effector, how many solutions are there to the inverse kinematics of the three-link planar arm shown in Figure 5.13? If the orientation of the end effector is also specified, how many solutions are there? Use any method in this chapter to characterize the inverse kinematics.

5–2 Repeat Problem 5–1 for the three-link planar arm with prismatic joint of Figure 5.14.

5–3 Solve the inverse position kinematics for the cylindrical manipulator of Figure 5.15.

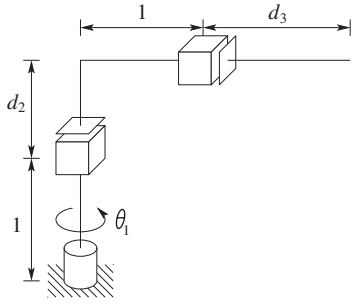


Figure 5.15: Cylindrical configuration.

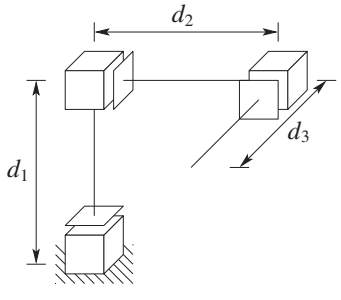


Figure 5.16: Cartesian configuration.

5–4 Solve the inverse position kinematics for the Cartesian manipulator of Figure 5.16.

5–5 Add a spherical wrist to the three-link cylindrical arm of Problem 5–3 and write the complete inverse kinematics solution.

5–6 Add a spherical wrist to the Cartesian manipulator of Problem 5–4 and write the complete inverse kinematics solution.

5–7 Write a computer program to compute the inverse kinematic equations for the elbow manipulator using Equations (5.40)–(5.45). Include procedures for identifying singular configurations and choosing a particular solution when the configuration is not singular. Test your routine for various special cases, including singular configurations.

5–8 The Stanford manipulator of Example 3.3.5 has a spherical wrist. Given a desired position o and orientation R of the end effector,

1. Compute the desired coordinates of the wrist center o_c^0.

2. Solve the inverse position kinematics, that is, find values of the first three joint variables that will place the wrist center at o_c. Is the solution unique? How many solutions did you find?
3. Compute the rotation matrix R_3^0. Solve the inverse orientation problem for this manipulator by finding a set of Euler angles corresponding to R_6^3 given by Equation (5.28).

5–9 Repeat Problem 5–8 for the PUMA 260 manipulator of Problem 5–9, which also has a spherical wrist. How many total solutions did you find?

5–10 Find all other solutions to the inverse kinematics of the elbow manipulator of Section 5.3.2.

5–11 Modify the solutions θ_1 and θ_2 for the spherical manipulator given by Equations (5.15) and (5.16) for the case of a shoulder offset.

Notes and References

The problem of inverse kinematics is classical and predates robotics by several hundred years. Consequently there is an extensive literature dealing with inverse kinematics within and without the robotics communities. An early fundamental result in robotics was the work of Pieper [137], who showed that any 6 DOF robot with at least three consecutive joint axes intersecting at a common point has a closed-form solution to the inverse kinematics. This property of robot kinematics is, in fact, the main reason for the importance of the spherical wrist. The approach that we use here of partitioning the inverse kinematics into inverse position and inverse orientation in the case of a spherical wrist is due to Hollerbach and Gideon [65]. We showed that the PUMA robot has 8 distinct inverse kinematic solutions. Primrose [139] gave an explicit upper bound of 16 for the number of inverse kinematic solutions for a general 6 DOF manipulator. This is a hard bound as shown by Manseur and Doty [108], who gave an example of a robot with 16 solutions. Although we do not treat the kinematics of parallel robots in this text, it turns out that the inverse kinematics problem for parallel robots is easier than it is for serial-link robots [68]. Numerical solutions to the inverse kinematics problem is treated in several sources, for example, [176] and [54]. Other good references for inverse kinematics of manipulators are, for example, [7], [92], [93], [135], [148], [119], and [105].

Part II

DYNAMICS AND MOTION PLANNING

Chapter 6

DYNAMICS

This chapter deals with the dynamics of robot manipulators. Whereas the kinematic equations describe the motion of the robot without consideration of the forces and torques producing the motion, the dynamic equations explicitly describe the relationship between force and motion. The equations of motion are important to consider in the design of robots, in simulation and animation of robot motion, and in the design of control algorithms. We introduce the so-called **Euler–Lagrange equations**, which describe the evolution of a mechanical system subject to **holonomic constraints** (this term is defined later on). To motivate the Euler–Lagrange approach we begin with a simple derivation of these equations from Newton's second law for a one-degree-of-freedom system. We then derive the Euler–Lagrange equations from the **principle of virtual work** in the general case.

In order to determine the Euler-Lagrange equations in a specific situation, one has to form the **Lagrangian** of the system, which is the difference between the **kinetic energy** and the **potential energy**; we show how to do this in several commonly encountered situations. We then derive the dynamic equations of several example robotic manipulators, including a two-link Cartesian robot, a two-link planar robot, and a two-link robot with remotely driven joints.

We also discuss several important properties of the Euler–Lagrange equations that can be exploited to design and analyze feedback control algorithms. Among these are explicit bounds on the inertia matrix, linearity in the inertia parameters, and the skew symmetry and passivity properties.

This chapter is concluded with a derivation of an alternate formulation of the dynamical equations of a robot, known as the **Newton–Euler formulation**, which is a recursive formulation of the dynamic equations that is often used for numerical calculation and simulation.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

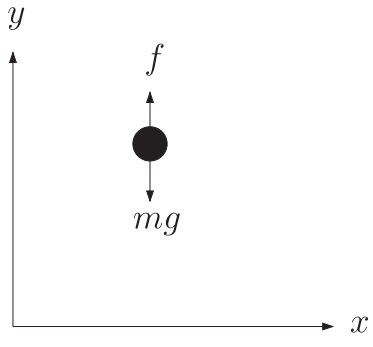


Figure 6.1: A particle of constant mass m constrained to move vertically constitutes a one-degree-of-freedom system. The gravitational force mg acts downward and an external force f acts upward.

6.1 The Euler–Lagrange Equations

In this section we present a general set of differential equations that describe the time evolution of mechanical systems subjected to holonomic constraints when the constraint forces satisfy the principle of virtual work. These are called the **Euler-Lagrange equations** of motion. Note that there are at least two distinct ways of deriving these equations. The method presented here is based on the method of virtual work, but it is also possible to derive the same equations using Hamilton's principle of least action.

6.1.1 Motivation

To motivate the subsequent discussion, we show first how the Euler–Lagrange equations can be derived from Newton's second law for the one-degree-of-freedom system shown in Figure 6.1.
By Newton's second law, the equation of motion of the particle is

$$m\ddot{y} \;=\; f - mg \tag{6.1}$$

Notice that the left-hand side of Equation (6.1) can be written as

$$m\ddot{y} = \frac{d}{dt}(m\dot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}\left(\frac{1}{2}m\dot{y}^2\right) = \frac{d}{dt}\frac{\partial \mathcal{K}}{\partial \dot{y}} \tag{6.2}$$

where $\mathcal{K} = \frac{1}{2}m\dot{y}^2$ is the **kinetic energy**. We use the partial derivative notation in the above expression to be consistent with systems considered

later when the kinetic energy will be a function of several variables. Likewise we can express the gravitational force in Equation (6.1) as

$$mg \;=\; \frac{\partial}{\partial y}(mgy) \;=\; \frac{\partial \mathcal{P}}{\partial y} \tag{6.3}$$

where $\mathcal{P} = mgy$ is the **potential energy due to gravity**. If we define

$$\mathcal{L} \;=\; \mathcal{K} - \mathcal{P} \;=\; \frac{1}{2}m\dot{y}^2 - mgy \tag{6.4}$$

and note that

$$\frac{\partial \mathcal{L}}{\partial \dot{y}} \;=\; \frac{\partial \mathcal{K}}{\partial \dot{y}} \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial y} \;=\; -\frac{\partial \mathcal{P}}{\partial y}$$

then we can write Equation (6.1) as

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{y}} - \frac{\partial \mathcal{L}}{\partial y} \;=\; f \tag{6.5}$$

The function $\mathcal{L}$, which is the difference of the kinetic and potential energy, is called the **Lagrangian** of the system, and Equation (6.5) is called the **Euler–Lagrange equation**.

The general procedure that we discuss below is, of course, the reverse of the above; namely, one first writes the kinetic and potential energies of a system in terms of a set of so-called **generalized coordinates** $(q_1, \ldots, q_n)$, where n is the number of degrees of freedom of the system, and then computes the equations of motion of the n-DOF system according to

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} \;=\; \tau_k \;;\; k = 1, \ldots, n \tag{6.6}$$

where τ_k is the (generalized) force associated with q_k. In the above single DOF example, the variable y serves as the generalized coordinate. Application of the Euler–Lagrange equations leads to a set of coupled second-order ordinary differential equations and provides a formulation of the dynamic equations of motion for serial-link robots equivalent to those derived using Newton's second law. However, as we shall see, the Lagrangian approach is advantageous for complex systems such as multi-link robots.

Example 6.1 (Single-Link Manipulator)**.** *Consider the single-link robot arm shown in Figure 6.2, consisting of a rigid link coupled through a gear train to a DC motor. Let θ_ℓ and θ_m denote the angles of the link and motor shaft, respectively. Then, $\theta_m = r\theta_\ell$ where $r : 1$ is the gear ratio. The*

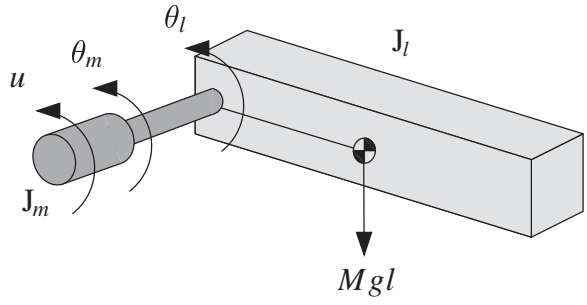


Figure 6.2: Single-link robot. The motor shaft is coupled to the axis of rotation of the link through a gear train which amplifies the motor torque and reduces the motor speed.

algebraic relation between the link and motor shaft angles means that the system has only one degree of freedom and we can therefore use as generalized coordinate either θ_m or θ_ℓ.

If we choose as generalized coordinate $q = \theta_\ell$, the kinetic energy of the system is given by

$$\begin{aligned} K &= \frac{1}{2} J_m \dot{\theta}_m^2 + \frac{1}{2} J_\ell \dot{\theta}_\ell^2 \\ &= \frac{1}{2}(r^2 J_m + J_\ell)\dot{q}^2 \end{aligned} \tag{6.7}$$

where J_m, J_ℓ are the rotational inertias of the motor and link, respectively. The potential energy is given as

$$P = Mg\ell(1 - \cos q) \tag{6.8}$$

where M is the total mass of the link and ℓ is the distance from the joint axis to the link center of mass. Defining $I = r^2 J_m + J_\ell$, the Lagrangian $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{2} I \dot{q}^2 - Mg\ell(1 - \cos q) \tag{6.9}$$

Substituting this expression into the Equation (6.6) with $n = 1$ and generalized coordinate θ_ℓ yields the equation of motion

$$I\ddot{q} + Mg\ell \sin q = \tau_\ell \tag{6.10}$$

The generalized force τ_ℓ represents those external forces and torques that are not derivable from a potential function. For this example, τ_ℓ consists of

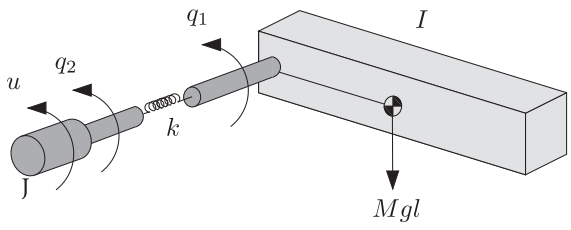


Figure 6.3: Single-link, flexible-joint robot. The joint elasticity arises from flexibility in the shaft or gears.

the input motor torque $u = r\tau_m$, reflected to the link, and (nonconservative) damping torques $B_m\dot{\theta}_m$ and $B_\ell\dot{\theta}_\ell$. Reflecting the motor damping to the link yields

$$\tau_\ell \quad = \quad u - B\dot{q}$$

where $B = rB_m + B_\ell$. Therefore, the complete expression for the dynamics of this system is

$$I\ddot{q} + B\dot{q} + Mg\ell \sin q \quad = \quad u \tag{6.11}$$

Example 6.2 (Single-Link Manipulator with Elastic Joint)**.** *Next, consider a single-link manipulator including the transmission flexibility as shown in Figure 6.3.*

In this case the motor angle $q_1 = \theta_\ell$ and the link angle $q_2 = \theta_m$ are independent variables and so the system possesses two degrees of freedom. Thus, two generalized coordinates are required to specify the configuration of the system.

The kinetic energy of this system is

$$K = \frac{1}{2}J_\ell \dot{q}_1^2 + \frac{1}{2}J_m \dot{q}_2^2 \tag{6.12}$$

The potential energy includes the spring potential energy in addition to the gravitational potential energy,

$$P \quad = \quad Mg\ell(1 - \cos q_1) + \frac{1}{2}k(q_1 - q_2)^2 \tag{6.13}$$

Forming the Lagrangian $\mathcal{L} = K - P$ and ignoring damping for simplicity, the equations of motion are found from the Euler-Lagrange equations as

$$\begin{aligned} J_\ell \ddot{q}_1 + Mg\ell \sin(q_1) + k(q_1 - q_2) &= 0 \\ J_m \ddot{q}_2 + k(q_2 - q_1) &= u \end{aligned} \tag{6.14}$$

The details are left as an exercise (Problem 6–1).

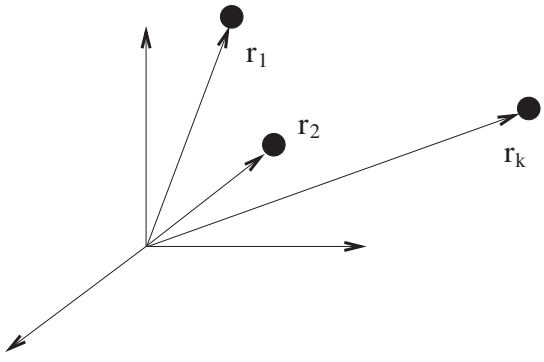


Figure 6.4: An unconstrained system of k particles has $3k$ degrees of freedom. If the particles are constrained, the number of degrees of freedom is reduced.

6.1.2 Holonomic Constraints and Virtual Work

Now, consider a system of k particles with corresponding position vectors $r_1, \ldots, r_k$ as shown in Figure 6.4.

If these particles are free to move about without any restrictions, then it is quite an easy matter to describe their motion, by noting that the mass times acceleration of each particle equals the external force applied to it. However, if the motion of the particles is constrained in some fashion, then one must take into account not only the externally applied forces, but also the so-called **constraint forces**, that is, the forces needed to make the constraints hold. As a simple illustration of this, suppose the system consists of two particles joined by a massless rigid wire of length ℓ. Then the two coordinates r_1 and r_2 must satisfy the constraint

$$\|r_1 - r_2\| = \ell \quad \text{or} \quad (r_1 - r_2)^T(r_1 - r_2) = \ell^2 \tag{6.15}$$

If one applies some external forces to each particle, then the particles experience not only these external forces but also the force exerted by the wire, which is along the direction $r_2 - r_1$ and of appropriate magnitude. Therefore, in order to analyze the motion of the two particles, we could follow one of two options. We could compute, under each set of external forces, what the corresponding constraint force must be in order that the equation above continues to hold. Alternatively, we can search for a method of analysis that does not require us to know the constraint force. Clearly, the second alternative is preferable, since it is generally quite an involved task to compute the constraint forces. This section is aimed at achieving this latter objective.

First, it is necessary to introduce some terminology. A constraint on the k coordinates $r_1, \ldots, r_k$ is called **holonomic** if it is an equality constraint of the form

$$g_i(r_1, \ldots, r_k) = 0, \qquad i = 1, \ldots, \ell \tag{6.16}$$

The constraint given in Equation (6.15) imposed by connecting two particles by a massless rigid wire is an example of a holonomic constraint. By differentiating Equation (6.16) we have an expression of the form

$$\sum_{j=1}^{k} \frac{\partial g_i}{\partial r_j} dr_j = 0 \tag{6.17}$$

A constraint of the form

$$\sum_{j=1}^{k} \omega_j dr_j = 0 \tag{6.18}$$

is called **nonholonomic** if it cannot be integrated to an equality constraint of the form (6.16). It is interesting to note that, while the method of deriving the equations of motion using the principle of virtual work remains valid for nonholonomic systems, methods based on variational principles, such as Hamilton's principle, can no longer be applied to derive the equations of motion. We will discuss systems subject to nonholonomic constraints in Chapter 14.

If a system is subjected to ℓ holonomic constraints, then one can think in terms of the constrained system having ℓ fewer degrees of freedom than the unconstrained system. In this case, it may be possible to express the coordinates of the k particles in terms of n **generalized coordinates** $q_1, \ldots, q_n$. In other words, we assume that the coordinates of the various particles, subjected to the set of constraints given by Equation (6.16), can be expressed in the form

$$r_i = r_i(q_1, \ldots, q_n), \qquad i = 1, \ldots, k \tag{6.19}$$

where $q_1, \ldots, q_n$ are all independent. In fact, the idea of generalized coordinates can be used even when there are infinitely many particles. For example, a physical rigid object such as a bar contains an infinity of particles; but since the distance between each pair of particles is fixed throughout the motion of the bar, six coordinates are sufficient to specify completely the coordinates of any particle in the bar. In particular, one could use three position coordinates to specify the location of the center of mass of the bar,

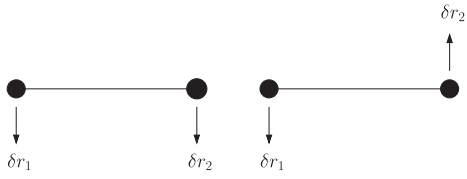


Figure 6.5: Examples of virtual displacements for a rigid bar. These infinitesimal motions do not change the distance between the endpoints and are thus compatible with the assumption that the bar is rigid.

and three Euler angles to specify the orientation of the body. Typically, generalized coordinates are positions, angles, etc. In fact, in Chapter 3 we chose to denote the joint variables by the symbols $q_1, \ldots, q_n$ precisely because these joint variables form a set of generalized coordinates for an n-link robot manipulator.

One can now speak of **virtual displacements**, which are any set of infinitesimal displacements, $\delta r_1, \ldots, \delta r_k$, that are consistent with the constraints. For example, consider once again the constraint (6.15) and suppose r_1, r_2 are perturbed to $r_1 + \delta r_1$, $r_2 + \delta r_2$, respectively. Then, in order that the perturbed coordinates continue to satisfy the constraint, the length of the bar must not change and so we must have

$$(r_1 + \delta r_1 - r_2 - \delta r_2)^T (r_1 + \delta r_1 - r_2 - \delta r_2) \quad = \quad \ell^2 \tag{6.20}$$

Now, let us expand the above product and take advantage of the fact that the original coordinates r_1, r_2 satisfy the constraint given by Equation (6.15). If we neglect quadratic terms in $\delta r_1, \delta r_2$ we obtain after some algebra (Problem 6–2)

$$(r_1 - r_2)^T (\delta r_1 - \delta r_2) \quad = \quad 0 \tag{6.21}$$

Thus, any infinitesimal perturbations in the positions of the two particles must satisfy the above equation in order that the perturbed positions continue to satisfy the constraint Equation (6.15). Any pair of infinitesimal vectors $\delta r_1, \delta r_2$ that satisfy Equation (6.21) constitutes a set of virtual displacements for this problem. Figure 6.5 shows some representative virtual displacements for a rigid bar.

Now, the reason for using generalized coordinates is to avoid dealing with complicated relationships such as Equation (6.21) above. If Equation (6.19) holds, then one can see that the set of all virtual displacements is precisely

$$\delta r_i \quad = \quad \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \delta q_j, \quad i = 1, \ldots, k \tag{6.22}$$

where the virtual displacements $\delta q_1, \ldots, \delta q_n$ of the generalized coordinates are unconstrained (that is what makes them generalized coordinates).

Next, we begin a discussion of constrained systems in equilibrium. Suppose each particle is in equilibrium. Then the net force on each particle is zero, which in turn implies that the work done by each set of virtual displacements is zero. Hence, the sum of the work done by any set of virtual displacements is also zero; that is,

$$\sum_{i=1}^{k} F_i^T \delta r_i \quad = \quad 0 \tag{6.23}$$

where F_i is the total force on particle i. As mentioned earlier, the force F_i is the sum of two quantities, namely (i) the externally applied force f_i, and (ii) the constraint force f_i^a. Now, suppose that the total work done by the constraint forces corresponding to any set of virtual displacements is zero, that is,

$$\sum_{i=1}^{k} f_i^{aT} \delta r_i \quad = \quad 0 \tag{6.24}$$

This will be true whenever the constraint force between a pair of particles is directed along the radial vector connecting the two particles (see the discussion in the next paragraph). Substituting Equation (6.24) into Equation (6.23) results in

$$\sum_{i=1}^{k} f_i^T \delta r_i \quad = \quad 0 \tag{6.25}$$

The beauty of this equation is that it does not involve the unknown constraint forces, but only the known external forces. This equation expresses the **principle of virtual work**, which can be stated in words as: *The work done by external forces corresponding to any set of virtual displacements is zero.*

Note that the principle is not universally applicable; it requires that Equation (6.24) hold, that is, that the constraint forces do no work. Thus, if the principle of virtual work applies, one can analyze the dynamics of a system *without* having to evaluate the constraint forces.

It is easy to verify that the principle of virtual work applies whenever the constraint force between a pair of particles acts along the vector connecting the position coordinates of the two particles. In particular, when the constraints are of the form (6.15), the principle applies. To see this, consider

once again a single constraint of the form (6.15). In this case, the constraint force, if any, must be exerted by the rigid massless wire, and therefore must be directed along the radial vector connecting the two particles. In other words, the force exerted on the first particle by the wire must be of the form

$$f_1^a = c(r_1 - r_2) \tag{6.26}$$

for some constant c (which could change as the particles move about). By the law of action and reaction, the force exerted on the second particle by the wire must be just the negative of the above, that is,

$$f_2^a = -c(r_1 - r_2) \tag{6.27}$$

Now, the work done by the constraint forces corresponding to a set of virtual displacements is

$$f_1^{aT}\delta r_1 + f_2^{aT}\delta r_2 = c(r_1 - r_2)^T(\delta r_1 - \delta r_2) \tag{6.28}$$

But, Equation (6.21) shows that the above expression must be zero for any set of virtual displacements. The same reasoning can be applied if the system consists of several particles that are pairwise connected by rigid massless wires of fixed lengths, in which case the system is subjected to several constraints of the form (6.15). Now, the requirement that the motion of a body be rigid can be equivalently expressed as the requirement that the distance between any pair of points on the body remain constant as the body moves, that is, as an infinity of constraints of the form (6.15). Thus, the principle of virtual work applies whenever rigidity is the only constraint on the motion. There are indeed situations when this principle does not apply, such as in the presence of magnetic fields. However, in all situations encountered in this book, we can safely assume that the principle of virtual work is valid.

6.1.3 D'Alembert's Principle

In Equation (6.25), the virtual displacements δr_i are not independent, so we cannot conclude from this equation that each coefficient F_i individually equals zero. In order to apply such reasoning, we must transform to generalized coordinates. Before doing this, we consider systems that are not necessarily in equilibrium. For such systems, **D'Alembert's principle** states that, if one introduces a fictitious additional force $-\dot{p}_i$ on each particle, where p_i is the momentum of particle i, then each particle will be in equilibrium. Thus, if one modifies Equation (6.23) by replacing F_i by

$F_i - \dot{p}_i$, then the resulting equation is valid for arbitrary systems. One can then remove the constraint forces as before using the principle of virtual work. This results in the equation

$$\sum_{i=1}^{k} f_i^T \delta r_i - \sum_{i=1}^{k} \dot{p}_i^T \delta r_i = 0 \tag{6.29}$$

The above equation does not mean that each coefficient of δr_i is zero since the virtual constraints δr_i are not independent. The remainder of this derivation is aimed at expressing the above equation in terms of the generalized coordinates, which are independent. For this purpose, we express each δr_i in terms of the corresponding virtual displacements of generalized coordinates, as is done in Equation (6.22). Then, the virtual work done by the forces f_i is given by

$$\sum_{i=1}^{k} f_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} f_i^T \frac{\partial r_i}{\partial q_j} \delta q_j = \sum_{j=1}^{n} \psi_j \delta q_j \tag{6.30}$$

where

$$\psi_j = \sum_{i=1}^{k} f_i^T \frac{\partial r_i}{\partial q_j} \tag{6.31}$$

is called the j^{th} **generalized force**. Note that ψ_j need not have dimensions of force, just as q_j need not have dimensions of length; however, $\psi_j \delta q_j$ must always have dimensions of work.

Now, let us study the second summation in Equation (6.29). Since $p_i = m_i \dot{r}_i$, it follows that

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{i=1}^{k} m_i \ddot{r}_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \delta q_j \tag{6.32}$$

Next, using the product rule of differentiation, we have

$$\frac{d}{dt}\left[m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] = m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} + m_i \dot{r}_i^T \frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right] \tag{6.33}$$

Rearranging the above and summing over all $i = 1, \ldots, n$ yields

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] - m_i \dot{r}_i^T \frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right] \right\} \tag{6.34}$$

Now, differentiating Equation (6.19) using the chain rule gives

$$v_i = \dot{r}_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \dot{q}_j \tag{6.35}$$

Observe from the above equation that

$$\frac{\partial v_i}{\partial \dot{q}_j} = \frac{\partial r_i}{\partial q_j} \tag{6.36}$$

Next,

$$\frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j}\right] = \sum_{\ell=1}^{n} \frac{\partial^2 r_i}{\partial q_j \partial q_\ell} \dot{q}_\ell = \frac{\partial}{\partial q_j} \sum_{\ell=1}^{n} \frac{\partial r_i}{\partial q_\ell} \dot{q}_\ell = \frac{\partial v_i}{\partial q_j} \tag{6.37}$$

where the last equality follows from Equation (6.35). Substituting from Equation (6.36) and Equation (6.37) into Equation (6.34) and noting that $\dot{r}_i = v_i$ gives

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i v_i^T \frac{\partial v_i}{\partial \dot{q}_j} \right] - m_i v_i^T \frac{\partial v_i}{\partial q_j} \right\} \tag{6.38}$$

If we define the **kinetic energy** K to be the quantity

$$K = \sum_{i=1}^{k} \frac{1}{2} m_i v_i^T v_i \tag{6.39}$$

then Equation (6.38) can be compactly expressed as

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} \tag{6.40}$$

Now, substituting from Equation (6.40) into Equation (6.32) shows that the second summation in Equation (6.29) is

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} \right\} \delta q_j \tag{6.41}$$

Finally, combining Equations (6.29), (6.30), and (6.41) gives

$$\sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} - \psi_j \right\} \delta q_j = 0 \tag{6.42}$$

Now, since the virtual displacements δq_j are independent, we can conclude that each coefficient in Equation (6.42) is zero, that is,

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} = \psi_j, \quad j = 1, \ldots, n \tag{6.43}$$

If the generalized force ψ_j is the sum of an externally applied generalized force and another one due to a potential field, then a further modification is possible. Suppose there exist functions τ_j and a potential energy function $P(q)$ such that

$$\psi_j = -\frac{\partial P}{\partial q_j} + \tau_j \tag{6.44}$$

Then Equation (6.43) can be written in the form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j \tag{6.45}$$

where $\mathcal{L} = K - P$ is the Lagrangian and we have recovered the **Euler–Lagrange equations of motion** as in Equation (6.6).

6.2 Kinetic and Potential Energy

In the previous section, we showed that the Euler–Lagrange equations can be used to derive the dynamical equations in a straightforward manner, provided one is able to express the kinetic and potential energies of the system in terms of a set of generalized coordinates. In order for this result to be useful in a practical context, it is important that one be able to compute these terms readily for an n-link robotic manipulator. In this section we derive formulas for the kinetic energy and potential energy of a robot with rigid links using the Denavit–Hartenberg joint variables as generalized coordinates.

To begin we note that the kinetic energy of a rigid object is the sum of two terms, the translational kinetic energy obtained by concentrating the entire mass of the object at the center of mass, and the rotational kinetic energy of the body about the center of mass. Referring to Figure 6.6 we attach a coordinate frame at the center of mass (called the **body-attached frame**) as shown.

The kinetic energy of the rigid body is then given as

$$\mathcal{K} = \frac{1}{2} m v^T v + \frac{1}{2} \omega^T \mathcal{I} \omega \tag{6.46}$$

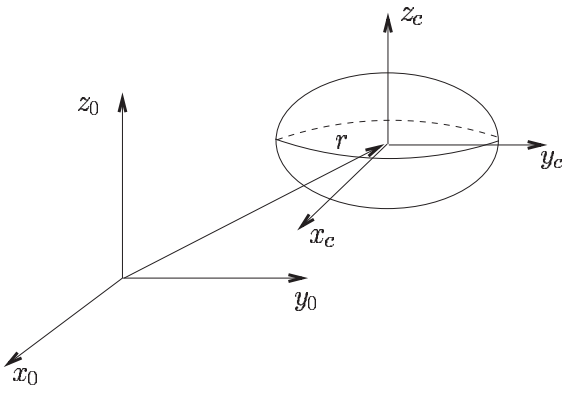


Figure 6.6: A general rigid body has six degrees of freedom. The kinetic energy consists of kinetic energy of rotation and kinetic energy of translation.

where m is the total mass of the object, v and ω are the linear and angular velocity vectors, respectively, and $\mathcal{I}$ is a symmetric 3×3 matrix called the **inertia tensor**.

6.2.1 The Inertia Tensor

It is understood that the above linear and angular velocity vectors, v and ω, respectively, are expressed in the inertial frame. In this case, we know that ω is found from the skew-symmetric matrix

$$S(\omega) = \dot{R}R^T \tag{6.47}$$

where R is the orientation transformation from the body-attached frame and the inertial frame. It is therefore necessary to express the inertia tensor, $\mathcal{I}$, also in the inertial frame in order to compute the triple product $\omega^T \mathcal{I} \omega$. The inertia tensor relative to the inertial reference frame will depend on the configuration of the object. If we denote as I the inertia tensor expressed instead in the body-attached frame, then the two matrices are related via a similarity transformation according to

$$\mathcal{I} = RIR^T \tag{6.48}$$

This is an important observation because the inertia matrix expressed in the body-attached frame is a constant matrix independent of the motion of the object and easily computed.

We next show how to compute this matrix explicitly. Let the mass density of the object be represented as a function of position, $\rho(x, y, z)$.

Then the inertia tensor in the body attached frame is computed as

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{6.49}$$

where

$$\begin{aligned} I_{xx} &= \int\int\int (y^2+z^2)\rho(x,y,z)dx\ dy\ dz \\ I_{yy} &= \int\int\int (x^2+z^2)\rho(x,y,z)dx\ dy\ dz \\ I_{zz} &= \int\int\int (x^2+y^2)\rho(x,y,z)dx\ dy\ dz \end{aligned}$$

and

$$\begin{aligned} I_{xy} = I_{yx} &= -\int\int\int xy\rho(x,y,z)dx\ dy\ dz \\ I_{xz} = I_{zx} &= -\int\int\int xz\rho(x,y,z)dx\ dy\ dz \\ I_{yz} = I_{zy} &= -\int\int\int yz\rho(x,y,z)dx\ dy\ dz \end{aligned}$$

The integrals in the above expression are computed over the region of space occupied by the rigid body. The diagonal elements of the inertia tensor, I_{xx}, I_{yy}, I_{zz}, are called the **principal moments of inertia** about the x, y, and z-axes, respectively. The off-diagonal terms I_{xy}, I_{xz}, etc., are called the **cross products of inertia**. If the mass distribution of the body is symmetric with respect to the body-attached frame, then the cross products of inertia are identically zero.

Example 6.3 (Uniform Rectangular Solid)**.** *Consider the rectangular solid of length a width b and height c shown in Figure 6.7 and suppose that the density is constant, $\rho(x,y,z) = \rho$. If the body frame is attached at the geometric center of the object, then by symmetry, the cross products of inertia are all zero and it is a simple exercise to compute*

$$\begin{aligned} I_{xx} &= \int_{-c/2}^{c/2}\int_{-b/2}^{b/2}\int_{-a/2}^{a/2} (y^2+z^2)\rho(x,y,z)dx\ dy\ dz \\ &= \rho\frac{abc}{12}(b^2+c^2) = \frac{m}{12}(b^2+c^2) \end{aligned}$$

since $\rho abc = m$, the total mass. Likewise, a similar calculation shows that

$$I_{yy} = \frac{m}{12}(a^2+c^2) \quad ; \quad I_{zz} = \frac{m}{12}(a^2+b^2)$$

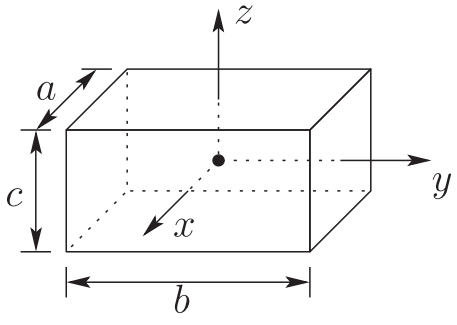


Figure 6.7: A rectangular solid with uniform mass density and coordinate frame attached at the geometric center of the solid.

6.2.2 Kinetic Energy for an n-Link Robot

Now, consider a manipulator with n links. We have seen in Chapter 4 that the linear and angular velocities of any point on any link can be expressed in terms of the Jacobian matrix and the derivatives of the joint variables. Since, in our case, the joint variables are indeed generalized coordinates, it follows that, for appropriate Jacobian matrices J_{v_i} and J_{ω_i}, of dimension $3 \times n$, we have

$$v_i = J_{v_i}(q)\dot{q}, \qquad \omega_i = J_{\omega_i}(q)\dot{q} \tag{6.50}$$

Now, suppose the mass of link i is m_i and that the inertia matrix of link i, evaluated around a coordinate frame parallel to frame i but whose origin is at the center of mass, equals I_i. Then from Equations (6.46) and (6.50) it follows that the overall kinetic energy of the manipulator equals

$$K = \frac{1}{2}\dot{q}^T \left[\sum_{i=1}^{n} \{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \} \right] \dot{q} \tag{6.51}$$

$$= \frac{1}{2}\dot{q}^T D(q)\dot{q} \tag{6.52}$$

where

$$D(q) = \left[\sum_{i=1}^{n} \{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \} \right] \tag{6.53}$$

is an $n \times n$ configuration dependent matrix called the **inertia matrix**. In Section 6.4 we will compute this matrix for several commonly occurring

manipulator configurations. The inertia matrix is **symmetric** and **positive definite** for any manipulator. Symmetry of $D(q)$ is easily seen from Equation (6.53). Positive definiteness can be inferred from the fact that the kinetic energy is always nonnegative and is zero if and only if all of the joint velocities are zero. The formal proof is left as an exercise (Problem 6–6).

6.2.3 Potential Energy for an n-Link Robot

Now, consider the potential energy term. In the case of rigid dynamics, the only source of potential energy is gravity. The potential energy of the i^{th} link can be computed by assuming that the mass of the entire object is concentrated at its center of mass and is given by

$$P_i = m_i g^T r_{ci} \tag{6.54}$$

where g is the vector giving the direction of gravity in the inertial frame and the vector r_{ci} gives the coordinates of the center of mass of link i. The total potential energy of the n-link robot is therefore

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g^T r_{ci} \tag{6.55}$$

In the case that the robot contains elasticity, for example if the joints are flexible, then the potential energy will include terms containing the energy stored in the elastic elements. Note that the potential energy is a function only of the generalized coordinates and not their derivatives.

6.3 Equations of Motion

In this section we specialize the Euler–Lagrange equations derived in Section 6.1 to the case when two conditions hold. First, the kinetic energy is a quadratic function of the vector $\dot{q}$ of the form

$$K = \frac{1}{2}\dot{q}^T D(q)\dot{q} = \frac{1}{2}\sum_{i,j} d_{ij}(q)\dot{q}_i\dot{q}_j \tag{6.56}$$

where $d_{i,j}$ are the entries of the $n \times n$ inertia matrix $D(q)$, which is symmetric and positive definite for each $q \in \mathbb{R}^n$, and second, the potential energy $P = P(q)$ is independent of $\dot{q}$. We have already remarked that robotic manipulators satisfy these conditions.

The Euler–Lagrange equations for such a system can be derived as follows. Using Equation (6.56) we can write the Lagrangian as

$$L = K - P = \frac{1}{2}\sum_{i,j} d_{ij}(q)\dot{q}_i\dot{q}_j - P(q) \tag{6.57}$$

The partial derivatives of the Lagrangian with respect to the k^{th} joint velocity is given by

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}\dot{q}_j \tag{6.58}$$

and therefore

$$\begin{aligned}\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} &= \sum_j d_{kj}\ddot{q}_j + \sum_j \frac{d}{dt} d_{kj}\dot{q}_j \\ &= \sum_j d_{kj}\ddot{q}_j + \sum_{i,j} \frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\dot{q}_j \end{aligned} \tag{6.59}$$

Similarly the partial derivative of the Lagrangian with respect to the k^{th} joint position is given by

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i,j} \frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial P}{\partial q_k} \tag{6.60}$$

Thus, for each $k = 1, \ldots, n$, the Euler–Lagrange equations can be written

$$\sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j + \frac{\partial P}{\partial q_k} = \tau_k \tag{6.61}$$

By interchanging the order of summation and taking advantage of symmetry, one can show (Problem 6–7) that

$$\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i}\right\}\dot{q}_i\dot{q}_j = \frac{1}{2}\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j}\right\}\dot{q}_i\dot{q}_j \tag{6.62}$$

Hence

$$\begin{aligned}\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j &= \sum_{i,j}\frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j \\ &= \sum_{i,j} c_{ijk}\dot{q}_i\dot{q}_j \end{aligned}$$

where we define

$$c_{ijk} := \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\} \tag{6.63}$$

The terms c_{ijk} in Equation (6.63) are known as **Christoffel symbols** (of the first kind). Note that, for a fixed k, we have $c_{ijk} = c_{jik}$, which reduces the effort involved in computing these symbols by a factor of about one half. Finally, if we define

$$g_k = \frac{\partial P}{\partial q_k} \tag{6.64}$$

then we can write the Euler–Lagrange equations as

$$\sum_{j=1}^{n} d_{kj}(q)\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k, \qquad k = 1, \ldots, n \tag{6.65}$$

In the above equations, there are three types of terms. The first type involves the second derivative of the generalized coordinates. The second type involves quadratic terms in the first derivatives of q, where the coefficients may depend on q. These latter terms are further classified into those involving a product of the type $\dot{q}_i^2$ and those involving a product of the type $\dot{q}_i\dot{q}_j$ where $i \neq j$. Terms of the type $\dot{q}_i^2$ are called **centrifugal**, while terms of the type $\dot{q}_i\dot{q}_j$ are called **Coriolis** terms. The third type of terms are those involving only q but not its derivatives. This third type arises from differentiating the potential energy. It is common to write Equation (6.65) in matrix form as

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{6.66}$$

where the $(k, j)^{th}$ element of the matrix $C(q, \dot{q})$ is defined as

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i = \sum_{i=1}^{n} \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i \tag{6.67}$$

and the gravity vector $g(q)$ is given by

$$g(q) = (g_1(q), \ldots, g_n(q)) \tag{6.68}$$

In summary, the development in this section is very general and applies to any mechanical system whose kinetic energy is of the form (6.56) and whose potential energy is independent of $\dot{q}$. In the next section we apply this discussion to study specific robot configurations.

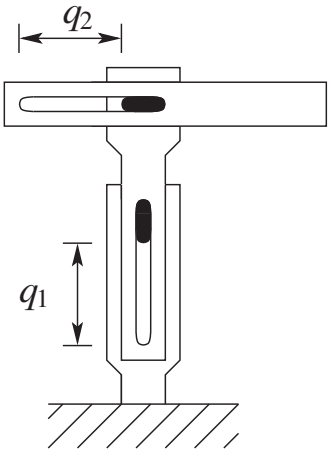


Figure 6.8: Two-link planar Cartesian robot. The orthogonal joint axes and linear joint motion of the Cartesian robot result in simple kinematics and dynamics.

6.4 Some Common Configurations

In this section we apply the above method of analysis to several manipulator configurations and derive the corresponding equations of motion. The configurations are progressively more complex, beginning with a two-link Cartesian manipulator and ending with a five-bar linkage mechanism that has a particularly simple inertia matrix.

Two-Link Cartesian Manipulator

Consider the manipulator shown in Figure 6.8 consisting of two links and two prismatic joints. Denote the masses of the two links by m_1 and m_2, respectively, and denote the displacement of the two prismatic joints by q_1 and q_2, respectively. It is easy to see, as mentioned in Section 6.1, that these two quantities serve as generalized coordinates for the manipulator. Since the generalized coordinates have dimensions of distance, the corresponding generalized forces have units of force. In fact, they are just the forces applied at each joint. Let us denote these by f_i, $i = 1, 2$.

Since we are using the joint variables as the generalized coordinates, we know that the kinetic energy is of the form (6.56) and that the potential energy is only a function of q_1 and q_2. Hence, we can use the formulas in Section 6.3 to obtain the dynamical equations. Also, since both joints are prismatic, the angular velocity Jacobian is zero and the kinetic energy of

each link consists solely of the translational term.

It follows that the velocity of the center of mass of link 1 is given by

$$v_{c1} = J_{v_{c1}}\dot{q} \tag{6.69}$$

where

$$J_{v_{c1}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \qquad \dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{6.70}$$

Similarly,

$$v_{c2} = J_{v_{c2}}\dot{q} \tag{6.71}$$

where

$$J_{v_{c2}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{6.72}$$

Hence, the kinetic energy is given by

$$K = \frac{1}{2}\dot{q}^T \left\{ m_1 J_{v_c}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}} \right\} \dot{q} \tag{6.73}$$

Comparing with Equation (6.56), we see that the inertia matrix D is given simply by

$$D = \begin{bmatrix} m_1 + m_2 & 0 \\ 0 & m_2 \end{bmatrix} \tag{6.74}$$

Next, the potential energy of link 1 is $m_1 g q_1$, while that of link 2 is $m_2 g q_1$, where g is the acceleration due to gravity. Hence, the overall potential energy is

$$P = g(m_1 + m_2)q_1 \tag{6.75}$$

Now, we are ready to write down the equations of motion. Since the inertia matrix is constant, all Christoffel symbols are zero. Furthermore, the components g_k of the gravity vector are given by

$$g_1 = \frac{\partial P}{\partial q_1} = g(m_1 + m_2), \qquad g_2 = \frac{\partial P}{\partial q_2} = 0 \tag{6.76}$$

Substituting into Equation (6.65) gives the dynamical equations as

$$\begin{array}{rcl} (m_1 + m_2)\ddot{q}_1 + g(m_1 + m_2) & = & f_1 \\ m_2\ddot{q}_2 & = & f_2 \end{array} \tag{6.77}$$

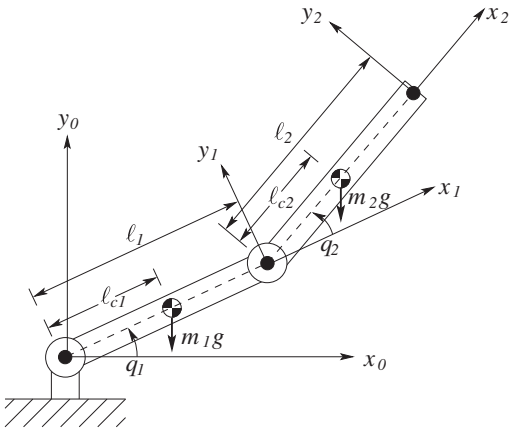


Figure 6.9: Two-link revolute joint arm. The rotational joint motion introduces dynamic coupling between the joints.

Planar Elbow Manipulator

Now, consider the planar manipulator with two revolute joints shown in Figure 6.9. Let us fix notation as follows: For $i = 1, 2$, q_i denotes the joint angle, which also serves as a generalized coordinate; m_i denotes the mass of link i; ℓ_i denotes the length of link i; ℓ_{ci} denotes the distance from the previous joint to the center of mass of link i; and I_i denotes the moment of inertia of link i about an axis coming out of the page, passing through the center of mass of link i.

We will use the Denavit–Hartenberg joint variables as generalized coordinates, which will allow us to make effective use of the Jacobian expressions in Chapter 4 in computing the kinetic energy. First,

$$v_{c1} = J_{v_{c1}} \dot{q} \tag{6.78}$$

where,

$$J_{v_{c1}} = \begin{bmatrix} -\ell_c \sin q_1 & 0 \\ \ell_{c1} \cos q_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{6.79}$$

Similarly,

$$v_{c2} = J_{v_{c2}} \dot{q} \tag{6.80}$$

where

$$J_{v_{c2}} = \begin{bmatrix} -\ell_1 \sin q_1 - \ell_{c2} \sin(q_1 + q_2) & -\ell_{c2} \sin(q_1 + q_2) \\ \ell_1 \cos q_1 + \ell_{c2} \cos(q_1 + q_2) & \ell_{c2} \cos(q_1 + q_2) \\ 0 & 0 \end{bmatrix} \quad (6.81)$$

Hence, the translational part of the kinetic energy is

$$\frac{1}{2} m_1 v_{c1}^T v_{c1} + \frac{1}{2} m_2 v_{c2}^T v_{c2} = \frac{1}{2} \dot{q} \left\{ m_1 J_{v_{c1}}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}} \right\} \dot{q} \quad (6.82)$$

Next, we consider the angular velocity terms. Because of the particularly simple nature of this manipulator, many of the potential difficulties do not arise. First, it is clear that

$$\omega_1 = \dot{q}_1 k, \qquad \omega_2 = (\dot{q}_1 + \dot{q}_2) k \quad (6.83)$$

when expressed in the base inertial frame. Moreover, since ω_i is aligned with the z-axes of each joint coordinate frame, the rotational kinetic energy reduces simply to $\frac{1}{2} I_i \omega_i^2$, where I_i is the moment of inertia about an axis through the center of mass of link i parallel to the z_i-axis. Hence, the rotational kinetic energy of the overall system in terms of the generalized coordinates is

$$\frac{1}{2} \dot{q}^T \left\{ I_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + I_2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right\} \dot{q} \quad (6.84)$$

Now, we are ready to form the inertia matrix $D(q)$. For this purpose, we merely have to add the two matrices in Equation (6.82) and Equation (6.84), respectively. Thus

$$D(q) = m_1 J_{v_{c1}}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}} + \begin{bmatrix} I_1 + I_2 & I_2 \\ I_2 & I_2 \end{bmatrix} \quad (6.85)$$

Carrying out the above multiplications and using the standard trigonometric identities $\cos^2 \theta + \sin^2 \theta = 1$, $\cos \alpha \cos \beta + \sin \alpha \sin \beta = \cos(\alpha - \beta)$ leads to

$$\begin{aligned} d_{11} &= m_1 \ell_{c1}^2 + m_2 (\ell_1^2 + \ell_{c2}^2 + 2 \ell_1 \ell_{c2} \cos q_2) + I_1 + I_2 \\ d_{12} &= d_{21} = m_2 (\ell_{c2}^2 + \ell_1 \ell_{c2} \cos q_2) + I_2 \\ d_{22} &= m_2 \ell_{c2}^2 + I_2 \end{aligned} \quad (6.86)$$

Now, we can compute the Christoffel symbols using Equation (6.63). This

gives

$$\begin{aligned}
c_{111} &= \frac{1}{2}\frac{\partial d_{11}}{\partial q_1} = 0 \\
c_{121} &= c_{211} = \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = -m_2\ell_1\ell_{c2}\sin q_2 = h \\
c_{221} &= \frac{\partial d_{12}}{\partial q_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = h \\
c_{112} &= \frac{\partial d_{21}}{\partial q_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = -h \\
c_{122} &= c_{212} = \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = 0 \\
c_{222} &= \frac{1}{2}\frac{\partial d_{22}}{\partial q_2} = 0
\end{aligned}$$

Next, the potential energy of the manipulator is just the sum of those of the two links. For each link, the potential energy is just its mass multiplied by the gravitational acceleration and the height of its center of mass. Thus

$$\begin{aligned}
P_1 &= m_1 g \ell_{c1}\sin q_1 \\
P_2 &= m_2 g(\ell_1 \sin q_1 + \ell_{c2}\sin(q_1+q_2))
\end{aligned}$$

and so the total potential energy is

$$P = P_1 + P_2 = (m_1\ell_{c1} + m_2\ell_1)g\sin q_1 + m_2\ell_{c2}g\sin(q_1+q_2) \tag{6.87}$$

Therefore, the functions g_k defined in Equation (6.64) become

$$g_1 = \frac{\partial P}{\partial q_1} = (m_1\ell_{c1} + m_2\ell_1)g\cos q_1 + m_2\ell_{c2}g\cos(q_1+q_2) \tag{6.88}$$

$$g_2 = \frac{\partial P}{\partial q_2} = m_2\ell_{c2}g\cos(q_1+q_2) \tag{6.89}$$

Finally, we can write down the dynamical equations of the system as in Equation (6.65). Substituting for the various quantities in this equation and omitting zero terms leads to

$$\begin{aligned}
d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{121}\dot{q}_1\dot{q}_2 + c_{211}\dot{q}_2\dot{q}_1 + c_{221}\dot{q}_2^2 + g_1 &= \tau_1 \\
d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{112}\dot{q}_1^2 + g_2 &= \tau_2
\end{aligned} \tag{6.90}$$

In this case, the matrix $C(q,\dot{q})$ is given as

$$C = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix} \tag{6.91}$$

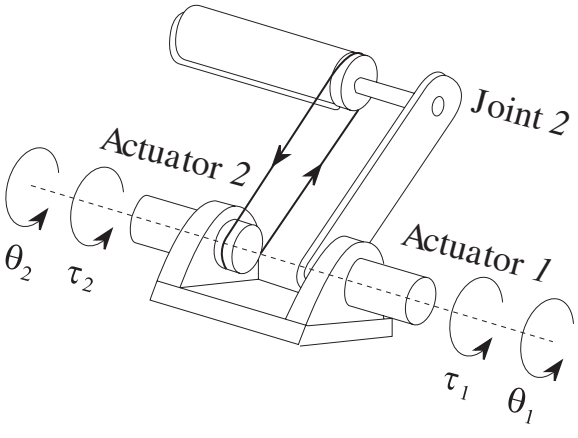


Figure 6.10: Two-link revolute joint arm with remotely driven link. Because of the remote drive the motor shaft angles are not proportional to the joint angles.

Planar Elbow Manipulator with Remotely Driven Link

Now, we illustrate the use of Lagrangian equations in a situation where the generalized coordinates are not the joint variables defined in earlier chapters. Consider again the planar elbow manipulator, but suppose now that both joints are driven by motors mounted at the base. The first joint is turned directly by one of the motors, while the other is turned via a gearing mechanism or a timing belt (see Figure 6.10).

In this case, one should choose the generalized coordinates as shown in Figure 6.11, because the angle p_2 is determined by driving motor number 2, and is not affected by the angle p_1. We will derive the dynamical equations for this configuration and show that some simplifications will result.

Since p_1 and p_2 are not the joint angles used earlier, we cannot use the velocity Jacobians derived in Chapter 4 in order to find the kinetic energy of each link. Instead, we have to carry out the analysis directly. It is easy to see that

$$v_{c1} = \begin{bmatrix} -\ell_{c1}\sin p_1 & 0 \\ \ell_{c1}\cos p_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} \tag{6.92}$$

$$v_{c2} = \begin{bmatrix} -\ell_1\sin p_1 & -\ell_{c2}\sin p_2 \\ \ell_1\cos p_1 & \ell_{c2}\cos p_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} \tag{6.93}$$

$$\omega_1 = \dot{p}_1 k, \qquad \omega_2 = \dot{p}_2 k \tag{6.94}$$

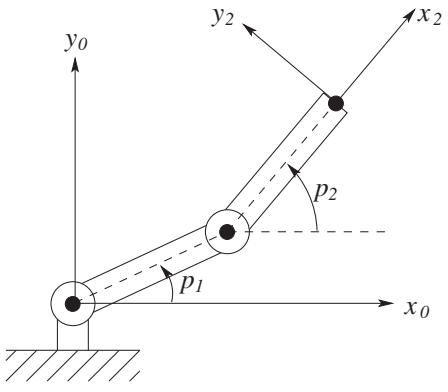


Figure 6.11: Generalized coordinates for the robot of Figure 6.10.

Hence, the kinetic energy of the manipulator equals

$$K = \frac{1}{2}\dot{p}^T D(p)\dot{p} \tag{6.95}$$

where

$$D(p) = \begin{bmatrix} m_1\ell_{c1}^2 + m_2\ell_1^2 + I_1 & m_2\ell_1\ell_{c2}\cos(p_2 - p_1) \\ m_2\ell_1\ell_{c2}\cos(p_2 - p_1) & m_2\ell_{c2}^2 + I_2 \end{bmatrix} \tag{6.96}$$

Computing the Christoffel symbols as in Equation (6.63) gives

$$\begin{aligned} c_{111} &= \frac{1}{2}\frac{\partial d_{11}}{\partial p_1} = 0 \\ c_{121} &= c_{211} = \frac{1}{2}\frac{\partial d_{11}}{\partial p_2} = 0 \\ c_{221} &= \frac{\partial d_{12}}{\partial p_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial p_1} = -m_2\ell_1\ell_{c2}\sin(p_2 - p_1) \\ c_{112} &= \frac{\partial d_{21}}{\partial p_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial p_2} = m_2\ell_1\ell_{c2}\sin(p_2 - p_1) \\ c_{212} &= = c_{122} = \frac{1}{2}\frac{\partial d_{22}}{\partial p_1} = 0 \\ c_{222} &= \frac{1}{2}\frac{\partial d_{22}}{\partial p_2} = 0 \end{aligned} \tag{6.97}$$

Next, the potential energy of the manipulator, in terms of p_1 and p_2, equals

$$P = m_1 g\ell_{c1}\sin p_1 + m_2 g(\ell_1 \sin p_1 + \ell_{c2}\sin p_2) \tag{6.98}$$

Hence, the gravitational generalized forces are

$$\begin{aligned} g_1 &= (m_1\ell_{c1} + m_2\ell_1)g\cos p_1 \\ g_2 &= m_2\ell_{c2}g\cos p_2 \end{aligned}$$

Finally, the equations of motion are

$$\begin{aligned} d_{11}\ddot{p}_1 + d_{12}\ddot{p}_2 + c_{221}\dot{p}_2^2 + g_1 &= \tau_1 \\ d_{21}\ddot{p}_1 + d_{22}\ddot{p}_2 + c_{112}\dot{p}_1^2 + g_2 &= \tau_2 \end{aligned} \tag{6.99}$$

Comparing Equation (6.99) and Equation (6.90), we see that by driving the second joint remotely from the base we have eliminated the Coriolis forces, but we still have the centrifugal forces coupling the two joints.

Five-Bar Linkage

Now, consider the manipulator shown in Figure 6.12. We will show that, if the parameters of the manipulator satisfy a simple relationship, then the equations of the manipulator are decoupled, so that each quantity q_1 and q_2 can be controlled independently of the other. The mechanism in Figure 6.12 is called a **five-bar linkage**. Clearly, there are only four bars in the figure, but in the theory of mechanisms it is a convention to count the ground as an additional linkage, which explains the terminology. It is assumed that the lengths of links 1 and 3 are the same, and that the two lengths marked ℓ_2 are the same; in this way the closed path in the figure is in fact a parallelogram, which greatly simplifies the computations. Notice, however, that the quantities ℓ_{c1} and ℓ_{c3} need not be equal. For example, even though links 1 and 3 have the same length, they need not have the same mass distribution. It is clear from the figure that, even though there are four links being moved, there are in fact only two degrees of freedom, identified as q_1 and q_2. Thus, in contrast to the earlier mechanisms studied in this book, this one is a closed kinematic chain (though of a particularly simple kind). As a result, we cannot use the earlier results on Jacobian matrices, and instead have to start from scratch. As a first step we write down the coordinates of the centers of mass of the various links as a function of the

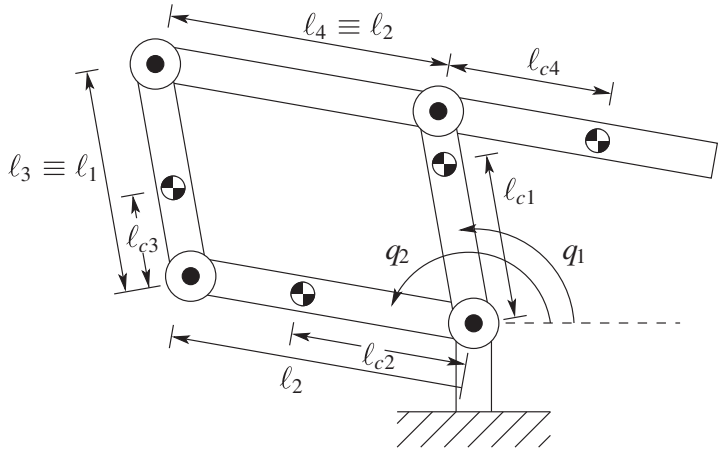


Figure 6.12: Five-bar linkage.

generalized coordinates. This gives

$$\begin{bmatrix} x_{c1} \\ y_{c1} \end{bmatrix} = \begin{bmatrix} \ell_{c1}\cos q_1 \\ \ell_{c1}\sin q_1 \end{bmatrix} \tag{6.100}$$

$$\begin{bmatrix} x_{c2} \\ y_{c2} \end{bmatrix} = \begin{bmatrix} \ell_{c2}\cos q_2 \\ \ell_{c2}\sin q_2 \end{bmatrix} \tag{6.101}$$

$$\begin{bmatrix} x_{c3} \\ y_{c3} \end{bmatrix} = \begin{bmatrix} \ell_2\cos q_2 \\ \ell_2\sin q_2 \end{bmatrix} + \begin{bmatrix} \ell_{c3}\cos q_1 \\ \ell_{c3}\sin q_1 \end{bmatrix} \tag{6.102}$$

$$\begin{aligned} \begin{bmatrix} x_{c4} \\ y_{c4} \end{bmatrix} &= \begin{bmatrix} \ell_1\cos q_1 \\ \ell_1\sin q_1 \end{bmatrix} + \begin{bmatrix} \ell_{c4}\cos(q_2-\pi) \\ \ell_{c4}\sin(q_2-\pi) \end{bmatrix} \\ &= \begin{bmatrix} \ell_1\cos q_1 \\ \ell_1\sin q_1 \end{bmatrix} - \begin{bmatrix} \ell_{c4}\cos q_2 \\ \ell_{c4}\sin q_2 \end{bmatrix} \end{aligned} \tag{6.103}$$

Next, with the aid of these expressions, we can write down the velocities of the various centers of mass as a function of $\dot{q}_1$ and $\dot{q}_2$. For convenience we drop the third row of each of the following Jacobian matrices as it is always

zero. The result is

$$\begin{aligned} v_{c1} &= \begin{bmatrix} -\ell_{c1}\sin q_1 & 0 \\ \ell_{c1}\cos q_1 & 0 \end{bmatrix}\dot{q} \\ v_{c2} &= \begin{bmatrix} 0 & -\ell_{c2}\sin q_2 \\ 0 & \ell_{c2}\cos q_2 \end{bmatrix}\dot{q} \\ v_{c3} &= \begin{bmatrix} -\ell_{c3}\sin q_1 & -\ell_2\sin q_2 \\ \ell_{c3}\cos q_1 & \ell_2\cos q_2 \end{bmatrix}\dot{q} \\ v_{c4} &= \begin{bmatrix} -\ell_1\sin q_1 & \ell_{c4}\sin q_2 \\ \ell_1\cos q_1 & \ell_{c4}\cos q_2 \end{bmatrix}\dot{q} \end{aligned} \tag{6.104}$$

Let us define the velocity Jacobians $J_{v_{ci}}$, $i \in \{1, \ldots, 4\}$ in the obvious fashion, that is, as the four matrices appearing in the above equations. Next, it is clear that the angular velocities of the four links are simply given by

$$\omega_1 = \omega_3 = \dot{q}_1 k, \omega_2 = \omega_4 = \dot{q}_2 k \tag{6.105}$$

Thus, the inertia matrix is given by

$$D(q) = \sum_{i=1}^{4} m_i J_{vc}^T J_{vc} + \begin{bmatrix} I_1 + I_3 & 0 \\ 0 & I_2 + I_4 \end{bmatrix} \tag{6.106}$$

If we now substitute from Equation (6.104) into the above equation and use the standard trigonometric identities, we are left with

$$\begin{aligned} d_{11}(q) &= m_1\ell_{c1}^2 + m_3\ell_{c3}^2 + m_4\ell_1^2 + I_1 + I_3 \\ d_{12}(q) &= d_{21}(q) = (m_3\ell_2\ell_{c3} - m_4\ell_1\ell_{c4})\cos(q_2 - q_1) \\ d_{22}(q) &= m_2\ell_{c2}^2 + m_3\ell_2^2 + m_4\ell_{c4}^2 + I_2 + I_4 \end{aligned} \tag{6.107}$$

Now, we note from the above expressions that if

$$m_3\ell_2\ell_{c3} = m_4\ell_1\ell_{c4} \tag{6.108}$$

then d_{12} and d_{21} are zero, that is, the inertia matrix is diagonal and constant. As a consequence the dynamical equations will contain neither Coriolis nor centrifugal terms.

Turning now to the potential energy, we have that

$$\begin{aligned} P &= g\textstyle\sum_{i=1}^{4} y_{ci} \\ &= g\sin q_1(m_1\ell_{c1} + m_3\ell_{c3} + m_4\ell_1) \\ &+ g\sin q_2(m_2\ell_{c2} + m_3\ell_2 - m_4\ell_{c4}) \end{aligned} \tag{6.109}$$

Hence,

$$\begin{aligned} g_1 &= g\cos q_1(m_1\ell_{c1} + m_3\ell_{c3} + m_4\ell_1) \\ g_2 &= g\cos q_2(m_2\ell_{c2} + m_3\ell_2 - m_4\ell_{c4}) \end{aligned} \tag{6.110}$$

Notice that g_1 depends only on q_1 but not on q_2 and similarly that g_2 depends only on q_2 but not on q_1. Hence, if the relationship (6.108) is satisfied, then the rather complex-looking manipulator in Figure 6.12 is described by the decoupled set of equations

$$d_{11}\ddot{q}_1 + g_1(q_1) = \tau_1, \qquad d_{22}\ddot{q}_2 + g_2(q_2) = \tau_2 \tag{6.111}$$

This discussion helps to explain the popularity of the parallelogram configuration in industrial robots. If the relationship (6.108) is satisfied, then one can adjust the two angles q_1 and q_2 independently, without worrying about interactions between the two angles. Compare this with the situation in the case of the planar elbow manipulators discussed earlier in this section.

6.5 Properties of Robot Dynamic Equations

The equations of motion for an n-link robot can be quite formidable especially if the robot contains one or more revolute joints. Fortunately, these equations contain some important structural properties that can be exploited to good advantage for developing control algorithms. We will see this in subsequent chapters. Here we will discuss some of these properties, the most important of which are the so-called **skew symmetry** property and the related **passivity** property, and the **linearity-in-the-parameters** property. For revolute joint robots, the inertia matrix also satisfies global bounds that are useful for control design.

6.5.1 Skew Symmetry and Passivity

The **skew symmetry** property refers to an important relationship between the inertia matrix $D(q)$ and the matrix $C(q,\dot{q})$ appearing in Equation (6.66).

Proposition 6.1 (The Skew Symmetry Property)**.** *Let $D(q)$ be the inertia matrix for an n-link robot and define $C(q,\dot{q})$ in terms of the elements of $D(q)$ according to Equation (6.67). Then the matrix $N(q,\dot{q}) = \dot{D}(q) - 2C(q,\dot{q})$ is skew symmetric, that is, the components n_{jk} of N satisfy $n_{jk} = -n_{kj}$.*

Proof: Given the inertia matrix $D(q)$, the $(k,j)^{th}$ component of $\dot{D}(q)$ is given by the chain rule as

$$\dot{d}_{kj} = \sum_{i=1}^{n} \frac{\partial d_{kj}}{\partial q_i}\dot{q}_i \tag{6.112}$$

Therefore, the $(k,j)^{th}$ component of $N = \dot{D} - 2C$ is given by

$$\begin{aligned} n_{kj} &= \dot{d}_{kj} - 2c_{kj} \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{kj}}{\partial q_i} - \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \right] \dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{ij}}{\partial q_k} - \frac{\partial d_{ki}}{\partial q_j} \right] \dot{q}_i \end{aligned} \tag{6.113}$$

Since the inertia matrix $D(q)$ is symmetric, that is, $d_{ij} = d_{ji}$, it follows from Equation (6.113) by interchanging the indices k and j that

$$n_{jk} = -n_{kj} \tag{6.114}$$

which completes the proof.

It is important to note that, in order for $N = \dot{D} - 2C$ to be skew-symmetric, one must define C according to Equation (6.67). This will be important in later chapters when we discuss robust and adaptive control algorithms.

Related to the skew symmetry property is the so-called **passivity property** which, in the present context, means that there exists a constant, $\beta \geq 0$, such that

$$\int_0^T \dot{q}^T(t)\tau(t)dt \geq -\beta, \quad \forall\, T > 0 \tag{6.115}$$

The term $\dot{q}^T\tau$ has units of power. Thus, the expression $\int_0^T \dot{q}^T(t)\tau(t)dt$ is the energy produced by the system over the time interval $[0,T]$. Passivity means that the amount of energy dissipated by the system has a lower bound given by $-\beta$. The word passivity comes from circuit theory where a passive system according to the above definition is one that can be built from passive components (resistors, capacitors, inductors). Likewise a passive mechanical system can be built from masses, springs, and dampers.

To prove the passivity property, let H be the total energy of the system, that is, the sum of the kinetic and potential energies,

$$H = \frac{1}{2}\dot{q}^T D(q)\dot{q} + P(q) \tag{6.116}$$

The derivative $\dot{H}$ satisfies

$$\begin{aligned} \dot{H} &= \dot{q}^T D(q)\ddot{q} + \tfrac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T \frac{\partial P}{\partial q} \\ &= \dot{q}^T \{\tau - C(q,\dot{q})\dot{q} - g(q)\} + \tfrac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T \frac{\partial P}{\partial q} \end{aligned} \tag{6.117}$$

where we have substituted for $D(q)\ddot{q}$ using the equations of motion. Collecting terms and using the fact that $g(q) = \frac{\partial P}{\partial q}$ yields

$$\begin{aligned} \dot{H} &= \dot{q}^T\tau + \frac{1}{2}\dot{q}^T\{\dot{D}(q) - 2C(q,\dot{q})\}\dot{q} \\ &= \dot{q}^T\tau \end{aligned} \tag{6.118}$$

the latter equality following from the skew symmetry property. Integrating both sides of Equation (6.118) with respect to time gives,

$$\int_0^T \dot{q}^T(t)\tau(t)dt = H(T) - H(0) \geq -H(0) \tag{6.119}$$

since the total energy $H(T)$ is nonnegative, and the passivity property therefore follows with $\beta = H(0)$.

6.5.2 Bounds on the Inertia Matrix

We have remarked previously that the inertia matrix for an n-link rigid robot is symmetric and positive definite. For a fixed value of the generalized coordinate q, let $0 < \lambda_1(q) \leq \cdots \leq \lambda_n(q)$ denote the n eigenvalues of $D(q)$. These eigenvalues are positive as a consequence of the positive definiteness of $D(q)$. As a result, it can easily be shown that

$$\lambda_1(q)I_{n\times n} \leq D(q) \leq \lambda_n(q)I_{n\times n} \tag{6.120}$$

where $I_{n\times n}$ denotes the $n \times n$ identity matrix. The above inequalities are interpreted in the standard sense of matrix inequalities, namely, if A and B are $n \times n$ matrices, then $B < A$ means that the matrix $A - B$ is positive definite and $B \leq A$ means that $A - B$ is positive semi-definite.

If all of the joints are revolute, then the inertia matrix contains only terms involving sine and cosine functions and, hence, is bounded as a function of the generalized coordinates. As a result, one can find constants λ_m and λ_M that provide uniform (independent of q) bounds in the inertia matrix

$$\lambda_m I_{n\times n} \leq D(q) \leq \lambda_M I_{n\times n} \tag{6.121}$$

6.5.3 Linearity in the Parameters

The robot equations of motion are defined in terms of certain parameters, such as link masses, moments of inertia, etc., that must be determined

for each particular robot in order, for example, to simulate the equations or to tune controllers. The complexity of the dynamic equations makes the determination of these parameters a difficult task. Fortunately, the equations of motion are linear in these inertia parameters in the following sense. There exists an $n \times \ell$ function, $Y(q, \dot{q}, \ddot{q})$ and an ℓ-dimensional vector Θ such that the Euler–Lagrange equations can be written as

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = Y(q, \dot{q}, \ddot{q})\Theta \tag{6.122}$$

The function $Y(q, \dot{q}, \ddot{q})$ is called the **regressor** and $\Theta \in \mathbb{R}^\ell$ is the **parameter vector**. The dimension of the parameter space, that is, the number of parameters needed to write the dynamics in this way, is not unique. In general, a given rigid body is described by ten parameters, namely, the total mass, the six independent entries of the inertia tensor, and the three coordinates of the center of mass. An n-link robot then has a maximum of $10n$ dynamics parameters. However, since the link motions are constrained and coupled by the joint interconnections, there are actually fewer than $10n$ independent parameters. Finding a minimal set of parameters that can parameterize the dynamic equations is, however, difficult in general.

Consider the two-link, revolute-joint, planar robot from Section 6.4. If we group the inertia terms appearing in Equation (6.86) as

$$\Theta_1 = m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2) + I_1 + I_2 \tag{6.123}$$
$$\Theta_2 = m_2\ell_1\ell_{c2} \tag{6.124}$$
$$\Theta_3 = m_2\ell_{c2}^2 + I_2 \tag{6.125}$$

then we can write the inertia matrix elements as

$$d_{11} = \Theta_1 + 2\Theta_2\cos(q_2) \tag{6.126}$$
$$d_{12} = d_{21} = \Theta_3 + \Theta_2\cos(q_2) \tag{6.127}$$
$$d_{22} = \Theta_3 \tag{6.128}$$

No additional parameters are required in the Christoffel symbols as these are functions of the elements of the inertia matrix. However, the gravitational torques generally require additional parameters. Setting

$$\Theta_4 = m_1\ell_{c1} + m_2\ell_1 \tag{6.129}$$
$$\Theta_5 = m_2\ell_{c2} \tag{6.130}$$

we can write the gravitational terms g_1 and g_2 as

$$g_1 = \Theta_4 g\cos(q_1) + \Theta_5 g\cos(q_1 + q_2) \tag{6.131}$$
$$g_2 = \Theta_5 g\cos(q_1 + q_2) \tag{6.132}$$

Substituting these into the equations of motion it is straightforward to write the dynamics in the form (6.122) where

$$Y(q,\dot{q},\ddot{q}) = \begin{bmatrix} \ddot{q}_1 & \cos(q_2)(2\ddot{q}_1+\ddot{q}_2)-\sin(q_2)(\dot{q}_1^2+2\dot{q}_1\dot{q}_2) & \ddot{q}_2 & g\cos(q_1) & g\cos(q_1+q_2) \\ 0 & \cos(q_2)\ddot{q}_1+\sin(q_2)\dot{q}_1^2 & \ddot{q}_1+\ddot{q}_2 & 0 & g\cos(q_1+q_2) \end{bmatrix}$$

and the parameter vector Θ is given by

$$\Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \\ \Theta_4 \\ \Theta_5 \end{bmatrix} = \begin{bmatrix} m_1\ell_{c1}^2 + m_2(\ell_1^2+\ell_{c2}^2) + I_1 + I_2 \\ m_2\ell_1\ell_{c2} \\ m_2\ell_{c2}^2 + I_2 \\ m_1\ell_{c1} + m_2\ell_1 \\ m_2\ell_{c2} \end{bmatrix} \tag{6.133}$$

Thus, we have parameterized the dynamics using a five dimensional parameter space. Note that in the absence of gravity only three parameters are needed.

6.6 Newton–Euler Formulation

In this section, we present a method for analyzing the dynamics of robot manipulators known as the **Newton–Euler formulation**. This method leads to exactly the same final answers as the Lagrangian formulation presented in earlier sections, but the route taken is quite different. In particular, in the Lagrangian formulation we treat the manipulator as a whole and perform the analysis using a Lagrangian function (the difference between the kinetic energy and the potential energy). In contrast, in the Newton–Euler formulation we treat each link of the robot in turn, and write down the equations describing its linear motion and its angular motion. Of course, since each link is coupled to other links, these equations that describe each link contain coupling forces and torques that appear also in the equations that describe neighboring links. By doing a so-called forward-backward recursion, we are able to determine all of these coupling terms and eventually to arrive at a description of the manipulator as a whole. Thus we see that the philosophy of the Newton–Euler formulation is quite different from that of the Lagrangian formulation.

At this stage the reader can justly ask whether there is a need for another formulation. Historically, both formulations were evolved in parallel, and each was perceived as having certain advantages. For instance, it was believed at one time that the Newton–Euler formulation is better suited to

recursive computation than the Lagrangian formulation. However, the current situation is that both of the formulations are equivalent in almost all respects. At present the main reason for having another method of analysis at our disposal is that it might provide different insights.

In any mechanical system one can identify a set of generalized coordinates (which we introduced in Section 6.1 and labeled q) and corresponding generalized forces (also introduced in Section 6.1 and labeled τ). Analyzing the dynamics of a system means finding the relationship between q and τ. At this stage we must distinguish between two aspects: First, we might be interested in obtaining **closed-form equations** that describe the time evolution of the generalized coordinates, such as Equation (6.90). Second, we might be interested in knowing what generalized forces need to be applied in order to realize a particular time evolution of the generalized coordinates. The distinction is that in the latter case we only want to know what time dependent function $\tau(\cdot)$ produces a particular trajectory $q(\cdot)$ and may not care to know the general functional relationship between the two. It is perhaps fair to say that in the former type of analysis, the Lagrangian formulation is superior while in the latter case the Newton–Euler formulation is superior. Looking ahead to topics beyond the scope of the book, if one wishes to study more advanced mechanical phenomena such as elastic deformations of the links (i.e., if one no longer assumes rigidity of the links), then the Lagrangian formulation is superior.

In this section we present the general equations that describe the Newton–Euler formulation. In the next section we illustrate the method by applying it to the planar elbow manipulator studied in Section 6.4 and show that the resulting equations are the same as Equation (6.90).

The facts of Newtonian mechanics that are pertinent to the present discussion can be stated as follows:

1. Every action has an equal and opposite reaction. Thus, if body 1 applies a force f and torque τ to body 2, then body 2 applies a force of $-f$ and torque of $-\tau$ to body 1.

2. The rate of change of the linear momentum equals the total force applied to the body.

3. The rate of change of the angular momentum equals the total torque applied to the body.

Applying the second fact to the linear motion of a body yields the relationship

$$\frac{d(mv)}{dt} = f \tag{6.134}$$

where m is the mass of the body, v is the velocity of the center of mass with respect to an inertial frame, and f is the sum of external forces applied to the body. Since in robotic applications the mass is constant as a function of time, Equation (6.134) can be simplified to the familiar relationship

$$ma = f \tag{6.135}$$

where $a = \dot{v}$ is the acceleration of the center of mass.

Applying the third fact to the angular motion of a body gives

$$\frac{d(I_0\omega_0)}{dt} = \tau_0 \tag{6.136}$$

where I_0 is the moment of inertia of the body about an inertial frame whose origin is at the center of mass, ω_0 is the angular velocity of the body, and τ_0 is the sum of torques applied to the body. Now there is an essential difference between linear motion and angular motion. Whereas the mass of a body is constant in most applications, its moment of inertia with respect an inertial frame may or may not be constant. To see this, suppose we attach a frame rigidly to the body, and let I denote the inertia matrix of the body with respect to this frame. Then I remains the same irrespective of whatever motion the body executes. However, the matrix I_0 is given by

$$I_0 = RIR^T \tag{6.137}$$

where R is the rotation matrix that transforms coordinates from the body attached-frame to the inertial frame. Thus there is no reason to expect that I_0 is constant as a function of time.

One possible way of overcoming this difficulty is to write the angular motion equation in terms of a frame rigidly attached to the body. This leads to

$$I\dot{\omega} + \omega \times (I\omega) = \tau \tag{6.138}$$

where I is the (constant) inertia matrix of the body with respect to the body-attached frame, ω is the angular velocity, but expressed in the body-attached frame, and τ is the total torque on the body, again expressed in the body-attached frame. Let us now give a derivation of Equation (6.138) to demonstrate clearly where the term $\omega \times (I\omega)$ comes from; note that this term is called the **gyroscopic term**.

Let R denote the orientation of the frame rigidly attached to the body w.r.t. the inertial frame; note that it could be a function of time. Then

Equation (6.137) gives the relation between I and I_0. Now by the definition of the angular velocity, we know that

$$\dot{R}R^T = S(\omega_0) \tag{6.139}$$

In other words, the angular velocity of the body, expressed in an inertial frame, is given by Equation (6.139). Of course, the same vector, expressed in the body-attached frame, is given by

$$\omega_0 = R\omega, \quad \omega = R^T\omega_0 \tag{6.140}$$

Hence the angular momentum, expressed in the inertial frame, is

$$h = RIR^TR\omega = RI\omega \tag{6.141}$$

Differentiating and noting that I is constant gives an expression for the rate of change of the angular momentum, expressed as a vector in the inertial frame:

$$\dot{h} = \dot{R}I\omega + RI\dot{\omega} \tag{6.142}$$

Now

$$S(\omega_0) = \dot{R}R^T, \quad \dot{R} = S(\omega)R \tag{6.143}$$

Hence, with respect to the inertial frame,

$$\dot{h} = S(\omega_0)RI\omega + RI\dot{\omega} \tag{6.144}$$

With respect to the frame rigidly attached to the body, the rate of change of the angular momentum is

$$\begin{aligned} R^T\dot{h} &= R^TS(\omega_0)RI\omega + I\dot{\omega} \\ &= S(R^T\omega_0)I\omega + I\dot{\omega} \\ &= S(\omega)I\omega + I\dot{\omega} = \omega \times (I\omega) + I\dot{\omega} \end{aligned} \tag{6.145}$$

This establishes Equation (6.138). Of course, if we wish, we can write the same equation in terms of vectors expressed in an inertial frame. But we will see shortly that there is an advantage to writing the force and moment equations with respect to a frame attached to link i, namely that a great many vectors in fact reduce to constant vectors, thus leading to significant simplifications in the equations.

Now we derive the Newton–Euler formulation of the equations of motion of an n-link manipulator. For this purpose, we first choose frames $0, \ldots, n$,

where frame 0 is an inertial frame, and frame i is rigidly attached to link i for $i \geq 1$. We also introduce several vectors, which are all expressed in frame i. The first set of vectors pertains to the velocities and accelerations of various parts of the manipulator.

$$\begin{aligned} a_{c,i} &= \text{the acceleration of the center of mass of link } i \\ a_{e,i} &= \text{the acceleration of the end of link } i \text{ (i.e., joint } i+1) \\ \omega_i &= \text{the angular velocity of frame } i \text{ w.r.t. frame 0} \\ \alpha_i &= \text{the angular acceleration of frame } i \text{ w.r.t. frame 0} \end{aligned}$$

The next several vectors pertain to forces and torques.

$$\begin{aligned} g_i &= \text{the acceleration due to gravity (expressed in frame } i \text{)} \\ f_i &= \text{the force exerted by link } i-1 \text{ on link } i \\ \tau_i &= \text{the torque exerted by link } i-1 \text{ on link } i \\ R_{i+1}^i &= \text{the rotation matrix from frame } i+1 \text{ to frame } i \end{aligned}$$

The final set of vectors pertain to physical features of the manipulator. Note that each of the following vectors is constant as a function of q. In other words, each of the vectors listed here is independent of the configuration of the manipulator.

$$\begin{aligned} m_i &= \text{the mass of link } i \\ I_i &= \text{the inertia matrix of link } i \text{ about a frame parallel} \\ &\quad \text{to frame } i \text{ whose origin is at the center of mass of link } i \\ r_{i,ci} &= \text{the vector from joint } i \text{ to the center of mass of link } i \\ r_{i+1,ci} &= \text{the vector from joint } i+1 \text{ to the center of mass of link } i \\ r_{i,i+1} &= \text{the vector from joint } i \text{ to joint } i+1 \end{aligned}$$

Now consider the free body diagram shown in Figure 6.13; this shows link i together with all forces and torques acting on it. Let us analyze each of the forces and torques shown in the figure. First, f_i is the force applied by link $i-1$ to link i. Next, by the law of action and reaction, link $i+1$ applies a force of $-f_{i+1}$ to link i, but this vector is expressed in frame $i+1$ according to our convention. In order to express the same vector in frame i, it is necessary to multiply it by the rotation matrix R_i^{i+1}. Similar explanations apply to the torques τ_i and $-R_i^{i+1}\tau_{i+1}$. The force $m_i g_i$ is the gravitational force. Since all vectors in Figure 6.13 are expressed in frame i, the gravity vector g_i is in general a function of i.

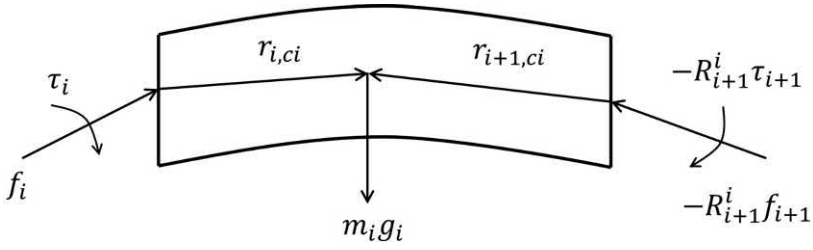


Figure 6.13: Forces and moments on link i

Writing down the force balance equation for link i gives

$$f_i - R^i_{i+1} f_{i+1} + m_i g_i \quad = \quad m_i a_{c,i} \tag{6.146}$$

Next we write down the moment balance equation for link i. For this purpose, it is important to note two things: First, the moment exerted by a force f about a point is given by $f \times r$, where r is the radial vector from the point where the force is applied to the point about which we are computing the moment. Second, in the moment equation below, the vector $m_i g_i$ does not appear, since it is applied directly at the center of mass. Thus we have

$$\begin{aligned} &\tau_i - R_i^{i+1}\tau_{i+1} + f_i \times r_{i,ci} - (R_i^{i+1} f_{i+1}) \times r_{i+1,ci} \\ &= I_i \alpha_i + \omega_i \times (I_i \omega_i) \end{aligned} \tag{6.147}$$

Now we present the heart of the Newton–Euler formulation, which consists of finding the vectors $f_1, \ldots, f_n$ and $\tau_1, \ldots, \tau_n$ corresponding to a given set of vectors $q, \dot{q}, \ddot{q}$. In other words, we find the forces and torques in the manipulator that correspond to a given set of generalized coordinates and first two derivatives. This information can be used to perform either type of analysis, as described above. That is, we can either use the equations below to find the f and τ corresponding to a **particular trajectory** $q(\cdot)$, or else to obtain closed-form dynamical equations. The general idea is as follows: Given $q, \dot{q}, \ddot{q}$, suppose we are somehow able to determine all of the velocities and accelerations of various parts of the manipulator, that is, all of the quantities $a_{c,i}$, ω_i, and α_i. Then we can solve Equations (6.146) and (6.147) recursively to find all the forces and torques, as follows: First, set $f_{n+1} = 0$ and $\tau_{n+1} = 0$. This expresses the fact that there is no link $n+1$. Then we can solve Equation (6.146) to obtain

$$f_i \quad = \quad R^i_{i+1} f_{i+1} + m_i a_{c,i} - m_i g_i \tag{6.148}$$

By successively substituting $i = n, n-1, \ldots, 1$ we find all forces. Similarly,

we can solve Equation (6.147) to obtain

$$\tau_i = R^i_{i+1}\tau_{i+1} - f_i \times r_{i,ci} + (R^i_{i+1}f_{i+1}) \times r_{i+1,ci} + I_i\alpha_i + \omega_i \times (I_i\omega_i) \tag{6.149}$$

By successively substituting $i = n,\ n-1, \ldots, 1$ we find all torques. Note that the above iteration is running in the direction of decreasing i.

Thus, the solution is complete once we find an easily computed relation between $q, \dot{q}, \ddot{q}$ and $a_{c,i}, \omega_i$, and α_i. This can be obtained by a recursive procedure in the direction of increasing i. This procedure is given below, for the case of revolute joints; the corresponding relationships for prismatic joints are actually easier to derive.

In order to distinguish between quantities expressed with respect to frame i and the base frame, we use a superscript (0) to denote the latter. Thus, for example, ω_i denotes the angular velocity of frame i expressed in frame i, while $\omega_i^{(0)}$ denotes the same quantity expressed in an inertial frame.

Now we have that

$$\omega_i^{(0)} = \omega_{i-1}^{(0)} + z_{i-1}\dot{q}_i \tag{6.150}$$

This merely expresses the fact that the angular velocity of frame i equals that of frame $i-1$ plus the added rotation from joint i. To get a relation between ω_i and ω_{i-1}, we need only to express the above equation in frame i rather than in the base frame, taking care to account for the fact that ω_i and ω_{i-1} are expressed in different frames. This leads to

$$\omega_i = (R_i^{i-1})^T\omega_{i-1} + b_i\dot{q}_i \tag{6.151}$$

where

$$b_i = (R_i^0)^T z_{i-1} \tag{6.152}$$

is the axis of rotation of joint i expressed in frame i.

Next let us work on the angular acceleration α_i. It is vitally important to note here that

$$\alpha_i = (R_i^0)^T\dot{\omega}_i^{(0)} \tag{6.153}$$

In other words, α_i is the derivative of the angular velocity of frame i, expressed in frame i. It is not true that $\alpha_i = \dot{\omega}_i$! We will encounter a similar situation with the velocity and acceleration of the center of mass. Now we see directly from Equation (6.150) that

$$\dot{\omega}_i^{(0)} = \dot{\omega}_{i-1}^{(0)} + z_{i-1}\ddot{q}_i + \omega_i^{(0)} \times z_{i-1}\dot{q}_i \tag{6.154}$$

Expressing the same equation in frame i gives

$$\alpha_i = (R_i^{i-1})^T \alpha_{i-1} + b_i \ddot{q}_i + \omega_i \times b_i \dot{q}_i \tag{6.155}$$

Now we come to the linear velocity and acceleration terms. Note that, in contrast to the angular velocity, the linear velocity does not appear anywhere in the dynamic equations; however, an expression for the linear velocity is needed before we can derive an expression for the linear acceleration. From Section 4.5, we get that the velocity of the center of mass of link i is given by

$$v_{c,i}^{(0)} = v_{e,i-1}^{(0)} + \omega_i^{(0)} \times r_{i,ci}^{(0)} \tag{6.156}$$

To obtain an expression for the acceleration, note that the vector $r_{i,ci}^{(0)}$ is constant in frame i. Thus

$$a_{c,i}^{(0)} = a_{e,i-1}^{(0)} + \dot{\omega}_i^{(0)} \times r_{i,ci}^{(0)} + \omega_i^{(0)} \times (\omega_i^{(0)} \times r_{i,ci}^{(0)}) \tag{6.157}$$

Now

$$a_{c,i} = (R_i^0)^T a_{c,i}^{(0)} \tag{6.158}$$

Let us carry out the multiplication and use the familiar property

$$R(a \times b) = (Ra) \times (Rb) \tag{6.159}$$

We also have to account for the fact that $a_{e,i-1}$ is expressed in frame $i-1$ and transform it to frame i. This gives

$$a_{c,i} = (R_i^{i-1})^T a_{e,i-1} + \dot{\omega}_i \times r_{i,ci} + \omega_i \times (\omega_i \times r_{i,ci}) \tag{6.160}$$

Now to find the acceleration of the end of link i, we can use Equation (6.160) with $r_{i,i+1}$ replacing $r_{i,ci}$. Thus

$$a_{e,i} = (R_i^{i-1})^T a_{e,i-1} + \dot{\omega}_i \times r_{i,i+1} + \omega_i \times (\omega_i \times r_{i,i+1}) \tag{6.161}$$

Now the recursive formulation is complete. We can now state the Newton–Euler formulation as follows.

1. Start with the initial conditions

$$\omega_0 = 0,\ \alpha_0 = 0,\ a_{c,0} = 0,\ a_{e,0} = 0 \tag{6.162}$$

and solve Equations (6.151), (6.155), (6.161), and (6.160) (in that order) to compute ω_i, α_i, and $a_{c,i}$ for i increasing from 1 to n.

2. Start with the terminal conditions

$$f_{n+1} = 0, \qquad \tau_{n+1} = 0 \tag{6.163}$$

and use Equations (6.148) and (6.149) to compute f_i and τ_i for i decreasing from n to 1.

6.6.1 Planar Elbow Manipulator Revisited

In this section we apply the recursive Newton–Euler formulation derived in Section 6.6 to analyze the dynamics of the planar elbow manipulator of Figure 6.9, and show that the Newton–Euler method leads to the same equations as the Lagrangian method, namely Equation (6.90).

We begin with the forward recursion to express the various velocities and accelerations in terms of q_1, q_2, and their derivatives. Note that, in this simple case, it is quite easy to see that

$$\omega_1 = \dot{q}_1 k,\ \alpha_1 = \ddot{q}_1 k,\ \omega_2 = (\dot{q}_1 + \dot{q}_2)k,\ \alpha_2 = (\ddot{q}_1 + \ddot{q}_2)k \quad (6.164)$$

so that there is no need to use Equations (6.151) and (6.155). Also, the vectors that are independent of the configuration are as follows:

$$r_{1,c1} = \ell_{c1} i,\ r_{2,c1} = (\ell_{c1} - \ell_1)i,\ r_{1,2} = \ell_1 i \quad (6.165)$$

$$r_{2,c2} = \ell_{c2} i,\ r_{3,c2} = (\ell_{c2} - \ell_2)i,\ r_{2,3} = \ell_2 i \quad (6.166)$$

where i here denotes the unit vector $(1, 0, 0)$ and not the index i in the iteration.

Forward Recursion Link 1

Using Equation (6.160) with index $i = 1$ and noting that $a_{e,0} = 0$ gives

$$\begin{aligned} a_{c,1} &= \ddot{q}_1 k \times \ell_{c1} i + \dot{q}_1 k \times (\dot{q}_1 k \times \ell_{c1} i) \\ &= \ell_{c1} \ddot{q}_1 j - \ell_{c1} \dot{q}_1^2 i = \begin{bmatrix} -\ell_{c1}\dot{q}_1^2 \\ \ell_c \ddot{q}_1 \\ 0 \end{bmatrix} \end{aligned} \quad (6.167)$$

where, again i and j the standard unit vectors in the x and y directions, respectively. Notice how simple this computation is when we do it with respect to frame 1, as opposed to the same computation in frame 0. Finally, we have

$$g_1 = -(R_1^0)^T g j = g \begin{bmatrix} -\sin q_1 \\ -\cos q_1 \end{bmatrix} \quad (6.168)$$

where g is the acceleration due to gravity. At this stage we can economize a bit by not displaying the third components of these accelerations, since they are all equal to zero. Similarly, the third component of all forces will be zero while the first two components of all torques will be zero. To complete the

computations for link 1, we compute the acceleration of end of link 1. This is obtained from Equation (6.167) by replacing ℓ_{c1} by ℓ_1. Thus

$$a_{e,1} = \begin{bmatrix} -\ell_1 \dot{q}_1^2 \\ \ell_1 \ddot{q}_1 \end{bmatrix} \tag{6.169}$$

Forward Recursion: Link 2

Once again we use Equation (6.160) and substitute for ω_2 from Equation (6.164). This yields

$$\begin{aligned} a_{c,2} &= (R_2^1)^T a_{e,1} + [(\ddot{q}_1 + \ddot{q}_2)k] \times \ell_{c2} i \\ &\quad + (\dot{q}_1 + \dot{q}_2)k \times [(\dot{q}_1 + \dot{q}_2)k \times \ell_{c2} i] \end{aligned} \tag{6.170}$$

The only quantity in the above equation which is configuration dependent is the first one. This can be computed as

$$\begin{aligned} (R_2^1)^T a_{e,1} &= \begin{bmatrix} \cos q_2 & \sin q_2 \\ -\sin q_2 & \cos q_2 \end{bmatrix} \begin{bmatrix} -\ell_1 \dot{q}_1^2 \\ \ell_1 \ddot{q}_1 \end{bmatrix} \\ &= \begin{bmatrix} -\ell_1 \dot{q}_1^2 \cos q_2 + \ell_1 \ddot{q}_1 \sin q_2 \\ \ell_1 \dot{q}_1^2 \sin q_2 + \ell_1 \ddot{q}_1 \cos q_2 \end{bmatrix} \end{aligned} \tag{6.171}$$

Substituting into Equation (6.170) gives

$$a_{c,2} = \begin{bmatrix} -\ell_1 \dot{q}_1^2 \cos q_2 + \ell_1 \ddot{q}_1 \sin q_2 - \ell_{c2}(\dot{q}_1 + \dot{q}_2)^2 \\ \ell_1 \dot{q}_1^2 \sin q_2 + \ell_1 \ddot{q}_1 \cos q_2 - \ell_{c2}(\ddot{q}_1 + \ddot{q}_2) \end{bmatrix} \tag{6.172}$$

The gravitational vector is

$$g_2 = g \begin{bmatrix} \sin(q_1 + q_2) \\ -\cos(q_1 + q_2) \\ 0 \end{bmatrix} \tag{6.173}$$

Since there are only two links, there is no need to compute $a_{e,2}$. Hence the forward recursions are complete at this point.

Backward Recursion: Link 2

Now we carry out the backward recursion to compute the forces and joint torques. Note that, in this instance, the joint torques are the externally applied quantities, and our ultimate objective is to derive dynamical equations involving the joint torques. First we apply Equation (6.148) with index $i = 2$ and note that $f_3 = 0$. This results in

$$f_2 = m_2 a_{c,2} - m_2 g_2 \tag{6.174}$$

$$\tau_2 = I_2 \alpha_2 + \omega_2 \times (I_2 \omega_2) - f_2 \times \ell_{c2} i \tag{6.175}$$

Now we can substitute for ω_2, α_2 from Equation (6.164), and for $a_{c,2}$ from Equation (6.172). We also note that the gyroscopic term equals zero, since both ω_2 and $I_2\omega_2$ are aligned with k. Now the cross product $f_2 \times \ell_{c2} i$ is clearly aligned with k, the unit vector in the z direction, and its magnitude is just the second component of f_2. The final result is

$$\begin{aligned} \tau_2 &= I_2(\ddot{q}_1 + \ddot{q}_2)k + [m_2\ell_1\ell_{c2}\sin q_2\dot{q}_1^2 + m_2\ell_1\ell_{c2}\cos q_2\ddot{q}_1 \\ &\quad + m_2\ell_{c2}^2(\ddot{q}_1 + \ddot{q}_2) + m)2\ell_{c2}g\cos(q_1 + q_2)]k \end{aligned} \tag{6.176}$$

Since $\tau_2 = \tau_2 k$, we see that the above equation is the same as the second equation in (6.90).

Backward Recursion: Link 1

To complete the derivation, we apply Equations (6.148) and (6.149) with index $i = 1$. First, the force equation is

$$f_1 = m_1 a_{c,1} + R_2^1 f_2 - m_1 g_1 \tag{6.177}$$

and the torque equation is

$$\begin{aligned} \tau_1 &= R_2^1\tau_2 - f_1 \times \ell_{c,1} i - (R_2^1 f_2) \times (\ell_1 - \ell_{c1})i \\ &\quad + I_1\alpha_1 + \omega_1 \times (I_1\omega_1) \end{aligned} \tag{6.178}$$

Now we can simplify things a bit. First, $R_2^1\tau_2 = \tau_2$, since the rotation matrix does not affect the third components of the vectors. Second, the gyroscopic term is again equal to zero. Finally, when we substitute for f_1 from Equation (6.177) into Equation (6.178), a little algebra gives

$$\begin{aligned} \tau_1 &= \tau_2 - m_1 a_{c,1} \times \ell_{c1} i + m_1 g_1 \times \ell_{c1} i - (R_2^1 f_2) \\ &\quad \times \ell_1 i + I_1 i + I_1\alpha_1 \end{aligned} \tag{6.179}$$

Once again, all these products are quite straightforward, and the only difficult calculation is that of $R_2^1 f_2$. The final result is:

$$\begin{aligned} \tau_1 &= \tau_2 + m_1\ell_{c1}^2 + m_1\ell_{c1}g\cos q_1 + m_2\ell_1 g\cos q_1 + I_1\ddot{q}_1 \\ &\quad + m_2\ell_1^2\ddot{q}_1 - m_1\ell_1\ell_{c2}(\dot{q}_1 + \dot{q}_2)^2\sin q_2 + m_2\ell_1\ell_{c2}(\ddot{q}_1 + \ddot{q}_2)\cos q_2 \end{aligned} \tag{6.180}$$

If we now substitute for τ_1 from Equation (6.176) and collect terms, we will get the first equation in (6.90); the details are routine and are left to the reader.

6.7 Chapter Summary

In this chapter we treated the dynamics of n-link rigid robots in detail. We derived the Euler–Lagrange equations from D'Alembert's principle and the principle of virtual work. These equations take the form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k \ ; \ k = 1, \ldots n$$

where n is the number of degrees of freedom and $\mathcal{L} = \mathcal{K} - \mathcal{P}$ is the Lagrangian function; the difference of the Kinetic and Potential energies, which are written in terms of a set of generalized coordinates $(q_1, \ldots, q_n)$. The terms τ_k are generalized forces acting on the system.

We derived computable formulas for the kinetic and potential energies; the kinetic energy $\mathcal{K}$ is given as

$$\begin{aligned} K &= \frac{1}{2}\dot{q}^T \left[\sum_{i=1}^{n} \{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \} \right] \dot{q} \\ &= \frac{1}{2}\dot{q}^T D(q) \dot{q} \end{aligned}$$

where

$$D(q) = \left[\sum_{i=1}^{n} \{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \} \right]$$

is the $n \times n$ **inertia matrix** of the manipulator. The matrices I_i in the above formula are the link inertia tensors. The inertia tensor is computed in a body-attached frame as

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

where

$$\begin{aligned} I_{xx} &= \int\int\int (y^2 + z^2)\rho(x,y,z)dx\ dy\ dz \\ I_{yy} &= \int\int\int (x^2 + z^2)\rho(x,y,z)dx\ dy\ dz \\ I_{zz} &= \int\int\int (x^2 + y^2)\rho(x,y,z)dx\ dy\ dz \end{aligned}$$

and

$$\begin{aligned} I_{xy} = I_{yx} &= -\int\int\int xy\rho(x,y,z)dx\ dy\ dz \\ I_{xz} = I_{zx} &= -\int\int\int xz\rho(x,y,z)dx\ dy\ dz \\ I_{yz} = I_{zy} &= -\int\int\int yz\rho(x,y,z)dx\ dy\ dz \end{aligned}$$

are the principal moments of inertia and cross products of inertia, respectively, where the integration is taken over the region of space occupied by the body.

The formula for the potential energy of the i^{th} link is

$$P_i = m_i g^T r_{ci}$$

where g is gravity vector expressed in the inertial frame and the vector r_{ci} gives the coordinates of the center of mass of link i in the inertial frame. The total potential energy of the n-link robot is therefore

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g^T r_{ci}$$

We then derived a special form of the Euler–Lagrange equations using the above expressions for the kinetic and potential energies as

$$\sum_{j=1}^{n} d_{kj}(q)\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k, \qquad k = 1,\ldots,n$$

where the terms

$$g_k = \frac{\partial P}{\partial q_k}$$

are gravitational generalized forces

$$c_{ijk} := \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}$$

and the terms c_{ijk} are Christoffel symbols of the first kind.

In vector-matrix form the Euler–Lagrange equations become

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

where the $(k,j)^{th}$ element of the matrix $C(q,\dot{q})$ is defined as

$$\begin{aligned} c_{kj} &= \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \\ &= \sum_{i=1}^{n} \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i \end{aligned}$$

and the gravity vector $g(q)$ is given by

$$g(q) = [g_1(q), \ldots, g_n(q)]^T$$

Next, we derived some important properties of the Euler–Lagrange equations, namely, the properties of **skew symmetry**, **passivity**, and **linearity in the parameters**. The skew symmetry property states that the matrix $N(q,\dot{q}) = \dot{D}(q) - 2C(q,\dot{q})$. The passivity property states that there exists a constant $\beta > 0$ such that

$$\int_0^T \dot{q}^T(t)\tau(t)dt \geq -\beta, \quad \forall\ T > 0$$

The linearity-in-the-parameters-property states that there exists an $n \times \ell$ function, $Y(q,\dot{q},\ddot{q})$, called the regressor, and an ℓ-dimensional vector Θ, called the parameter vector such that the Euler–Lagrange equations can be written

$$D(q) + C(q,\dot{q})\dot{q} + g(q) = Y(q,\dot{q},\ddot{q})\Theta = \tau$$

We also derived bounds on the inertia matrix for an n-link manipulator as

$$\lambda_1(q)I_{n\times n} \leq D(q) \leq \lambda_n(q)I_{n\times n}$$

In case the robot contains only revolute joints, the functions λ_1 and λ_n can be chosen as positive constants.

Finally, we discussed the recursive Newton–Euler formulation of robot dynamics. The Newton–Euler formulation is equivalent to the Euler–Lagrange method but offers some advantages from the standpoint of online computation.

Problems

6–1 Complete the derivation of the dynamic equations for the single-link, flexible-joint robot in Example 6.2.

6–2 Verify Equation (6.21) by direct calculation, neglecting quadratic terms in δr_1 and δr_2.

6–3 Consider a rigid body undergoing a pure rotation with no external forces acting on it. The kinetic energy is then given as

$$K = \frac{1}{2}(I_{xx}\omega_x^2 + I_{yy}\omega_y^2 + I_{zz}\omega_z^2)$$

with respect to a coordinate frame located at the center of mass and whose coordinate axes are principal axes. Take as generalized coordinates the Euler angles ϕ, θ, ψ and show that the Euler–Lagrange equations of motion of the rotating body are

$$\begin{aligned} I_{xx}\dot{\omega}_x + (I_{zz} - I_{yy})\omega_y\omega_z &= 0 \\ I_{yy}\dot{\omega}_y + (I_{xx} - I_{zz})\omega_z\omega_x &= 0 \\ I_{zz}\dot{\omega}_z + (I_{yy} - I_{xx})\omega_x\omega_y &= 0 \end{aligned}$$

6–4 Find the moments of inertia and cross products of inertia of a uniform rectangular solid of sides a, b, c with respect to a coordinate system with origin at the one corner and axes along the edges of the solid.

6–5 Given the inertia matrix $D(q)$ defined by Equation (6.86) show that $\det D(q) \neq 0$ for all q.

6–6 Show that the inertia matrix $D(q)$ for an n-link robot is always positive definite.

6–7 Verify the expression (6.62) that was used to derive the Christoffel symbols.

6–8 Consider a 3-link Cartesian manipulator,

(a) Compute the inertia tensor J_i for each link $i = 1, 2, 3$ assuming that the links are uniform rectangular solids of length 1, width $\frac{1}{4}$, and height $\frac{1}{4}$, and mass 1.

(b) Compute the 3×3 inertia matrix $D(q)$ for this manipulator.

(c) Show that the Christoffel symbols c_{ijk} are all zero for this robot. Interpret the meaning of this for the dynamic equations of motion.

(d) Derive the equations of motion in matrix form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \quad = \quad u$$

6–9 Derive the Euler–Lagrange equations for the planar RP robot in Figure 3.14.

6–10 Derive the Euler–Lagrange equations for the planar RPR robot in Figure 5.14.

6–11 Derive the Euler–Lagrange equations of motion for the three-link RRR robot of Figure 5.13. Explore the use of symbolic software, such as Maple or Mathematica, for this problem. See, for example, the **Robotica** package [126].

6–12 For each of the robots above, define a parameter vector, Θ, compute the regressor, $Y(q,\dot{q},\ddot{q})$ and express the equations of motion as

$$Y(q,\dot{q},\ddot{q})\Theta = \tau \tag{6.181}$$

6–13 Recall for a particle with kinetic energy $K = \frac{1}{2}m\dot{x}^2$, the **momentum** is defined as

$$p \quad = \quad m\dot{x} = \frac{dK}{d\dot{x}}$$

Therefore, for a mechanical system with generalized coordinates $q_1, \ldots, q_n$, we define the **generalized momentum** p_k as

$$p_k \quad = \quad \frac{\partial L}{\partial \dot{q}_k}$$

where L is the Lagrangian of the system. With $K = \frac{1}{2}\dot{q}^T D(q)\dot{q}$ and $L = K - V$ prove that

$$\sum_{k=1}^{n} \dot{q}_k p_k \quad = \quad 2K$$

6–14 There is another formulation of the equations of motion of a mechanical system that is useful, the so-called **Hamiltonian** formulation. Define the Hamiltonian function H by

$$H \quad = \quad \sum_{k-1}^{n} \dot{q}_k p_k - L$$

(a) Show that $H = K + V$.

(b) Using the Euler–Lagrange equations, derive Hamilton's equations

$$\begin{aligned} \dot{q}_k &= \frac{\partial H}{\partial p_k} \\ \dot{p}_k &= -\frac{\partial H}{\partial q_k} + \tau_k \end{aligned}$$

where τ_k is the input generalized force.

(c) For two-link manipulator of Figure 6.9 compute Hamiltonian equations in matrix form. Note that Hamilton's equations are a system of first-order differential equations as opposed to a second-order system given by Lagrange's equations.

6–15 Given the Hamiltonian H for a rigid robot, show that

$$\frac{dH}{dt} = \dot{q}^T \tau$$

where τ is the external force applied at the joints. What are the units of $\frac{dH}{dt}$?

Notes and References

A general reference for dynamics is [55]. More advanced treatments of dynamics can be found in [2] and [112]. The Lagrangian and recursive Newton–Euler formulations of the dynamic equations are given in [64]. These two approaches are shown to be equivalent in [152]. A detailed discussion of holonomic and nonholonomic constraints is found in [85]. The same reference also treats both the Lagrangian and Hamiltonian formulations of dynamics in detail. The properties of skew symmetry and passivity are discussed in [153], [86], and [131]. Parametrization of robot dynamics in terms of a minimal set of inertia parameters is treated in [51]. Identification of manipulator inertia parameters is discussed in [50].

Chapter 7

PATH AND TRAJECTORY PLANNING

In previous chapters we studied the geometry of robot arms, developing solutions for both the forward and inverse kinematics problems. The solutions to these problems depend only on the intrinsic geometry of the robot, and they do not reflect any constraints imposed by the workspace in which the robot operates. In particular, they do not take into account the possibility of collision between the robot and objects in the workspace or collision between the robot and the boundaries of its workspace (e.g., walls, floor, closed doors). In this chapter we address the problem of planning collision-free paths for robots. We will assume that the initial and final configurations of the robot are specified and that the problem is to find a collision-free path connecting these configurations.

The description of this problem is deceptively simple, yet the path planning problem is among the most difficult problems in computer science. For example, the computation time required by the best known complete[1] path planning algorithm grows exponentially with the number of internal degrees of freedom of the robot. For this reason, complete algorithms are used in practice only for simple robots with few degrees of freedom, such as mobile robots that translate in the plane. For robots that have more than a few degrees of freedom or are capable of rotational motion, path planning problems are often treated as search problems. The algorithms that are used for such problems are guided by heuristic strategies, and not guaranteed to find solutions for all problems. Nevertheless, they are quite effective in a wide

[1] An algorithm is said to be complete if it finds a solution whenever one exists, and signals failure in finite time when a solution does not exist.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

range of practical applications, are fairly easy to implement, and require only moderate computation time for most problems.

Path planning provides a geometric description of robot motion, but it does not consider any dynamic aspects of the motion. For example, for a manipulator arm, what should be the joint velocities and accelerations while traversing the path? For a car-like robot, what should be the acceleration profile along the path? These kinds of questions are addressed by a trajectory planner. The trajectory planner computes a function $q(t)$ that completely specifies the desired motion of the robot as it traverses the path.

We begin in Section 7.1 by revisiting the notion of **configuration space** that was first introduced in Chapter 1. We give a brief description of the geometry of the configuration space and describe how obstacles in the workspace can be mapped into the configuration space. Then, in Section 7.2, we consider the problem of planning the motion of polygon-shaped robots that translate in a planar workspace that contains polygonal objects. While this is a very special case, it has wide applicability to problems that involve mobile robots, such as the one shown in Figure 1.3. In Section 7.3 we present a first method that can be applied to general path planning problems, the so-called **artificial potential field** method. Potential field methods explore the configuration space by following the gradient of a function that rewards progress toward the goal while penalizing collision with objects in the workspace. It is generally not possible to construct such a function without introducing local minima, at which gradient descent algorithms will terminate without finding a solution. **Randomization** was introduced as a method to deal with this problem. In Section 7.4, we describe two methods that randomly generate a set of sample configurations, and use these as vertices in a graph that represents a set of free paths in the configuration space. Finally, since each of these methods generates a sequence of configurations, we describe how polynomial splines can be used to generate smooth trajectories from a sequence of configurations in Section 7.5.

7.1 The Configuration Space

In Chapter 3 we saw that the forward kinematic map can be used to determine the position and orientation of the end-effector frame given the vector of joint variables. Furthermore, the A matrices can be used to infer the position and orientation of any link of the robot. Since each link of the robot is assumed to be a rigid body, the A matrices can be used to infer the position of any point on the robot, given the values of the joint variables. Likewise, for a mobile robotic platform, if we rigidly attach a coordinate frame to the

robot, we can infer the position of any point on the mobile platform once we know the position and orientation of the body-attached frame. In the path planning literature a complete specification of the location of every point on the robot is referred to as a **configuration**, and the set of all possible configurations is referred to as the **configuration space**. In this section, we formalize the notion of a configuration space, including how it can be represented mathematically, how the presence of objects in the workspace impose constraints on the set of valid configurations, and the representation of paths in the configuration space.

7.1.1 Representing the Configuration Space

We denote by $\mathcal{Q}$ a representation of the configuration space. For example, as we saw in Chapter 2, for any rigid object we can specify the location of every point on the object by rigidly attaching a coordinate frame to the object and then specifying the position and orientation of this frame. Thus, for a rigid object moving in the plane we could represent a configuration by the triple $q = (x, y, \theta)$, and the configuration space could be represented by $\mathcal{Q} = \mathbb{R}^2 \times SO(2)$. Alternatively, we could choose to represent the orientation of the object as a point on the unit circle, S^1. In this case, the configuration space would be represented by $\mathcal{Q} = \mathbb{R}^2 \times S^1$.

For manipulator arms, the vector of joint variables often provides a convenient representation of a configuration. For a one-link revolute arm the configuration space is merely the set of orientations of the link, and thus $\mathcal{Q} = S^1$, where S^1 represents the unit circle. We can locally parameterize $\mathcal{Q}$ by a single parameter, the joint angle θ_1, exactly as was done in Chapter 3 using the DH convention. For the two-link planar arm we have $\mathcal{Q} = S^1 \times S^1 = T^2$, in which T^2 represents the torus, and we can represent a configuration by $q = (\theta_1, \theta_2)$. For a Cartesian arm, we have $\mathcal{Q} = \mathbb{R}^3$ and we can represent a configuration by $q = (d_1, d_2, d_3)$.

For mobile robots, we typically treat the robot as a single rigid object, ignoring the motion of individual components such as rotating wheels or propellers. For ground vehicles, the configuration space is typically represented as $\mathcal{Q} = SE(2)$, or, in cases where the orientation of the vehicle is not relevant, as $\mathcal{Q} = \mathbb{R}^2$. The latter case is specifically addressed below, in Section 7.2. For robots that move in three-dimensional space, such as air vehicles or underwater robots, we typically define the configuration space as $\mathcal{Q} = SE(3)$.

7.1.2 Configuration Space Obstacles

A collision occurs when any point on the robot contacts either an object in the workspace or the boundary of the workspace (e.g., walls or the floor), both of which are regarded as obstacles for the purposes of planning collision-free paths. Self-collision is also possible, for example if the end effector attached to a manipulator arm comes into contact with the base link, but we will not consider the case of self-collision here. To describe collisions we introduce some additional notation. We define the robot's workspace as the Cartesian space in which the robot moves, denoted by $\mathcal{W}$. In general, we consider $\mathcal{W} = \mathbb{R}^2$ for mobile robots that move in the plane, and $\mathcal{W} = \mathbb{R}^3$ for robots that move in three-dimensional space (e.g., non-planar robot arms and mobile robots such as air vehicles). We denote the subset of the workspace that is occupied by the robot at configuration q by $\mathcal{A}(q)$, and by $\mathcal{O}_i$ the subset of the workspace occupied by the i^{th} obstacle. To plan a collision free path we must ensure that the robot never reaches a configuration q that causes it to make contact with an obstacle. The set of configurations for which the robot collides with an obstacle is referred to as the **configuration space obstacle** and it is defined by

$$\mathcal{QO} = \{q \in \mathcal{Q} \mid \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$$

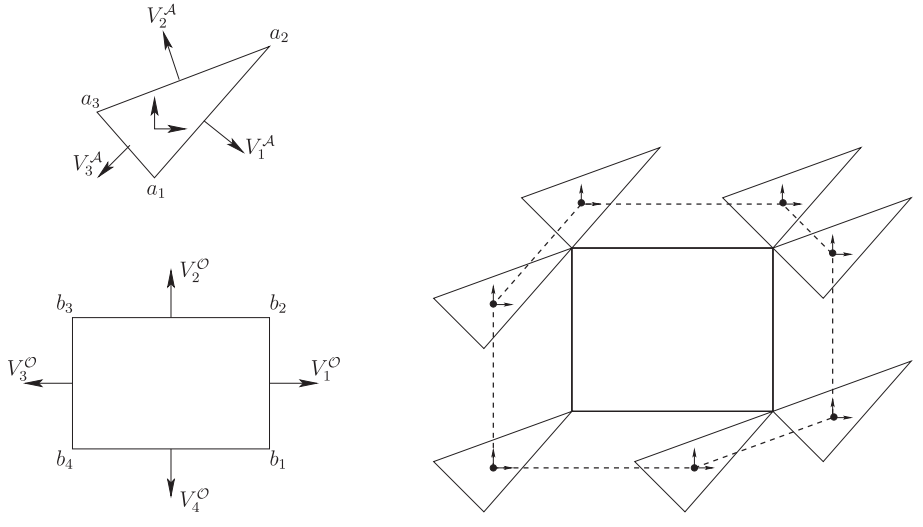
in which $\mathcal{O} = \bigcup \mathcal{O}_i$. The set of collision-free configurations, referred to as the free configuration space, is then simply the set difference

$$\mathcal{Q}_{\text{free}} = \mathcal{Q} \setminus \mathcal{QO}$$

Example 7.1 (A Rigid Body that Translates in the Plane)**.** *Consider a triangle-shaped mobile robot who possible motions include only translation in the plane as in Figure 7.1. For this case, the robot's configuration space is $\mathcal{Q} = \mathbb{R}^2$, so it is particularly easy to visualize both the configuration space and the configuration space obstacle region.*

If there is only one obstacle in the workspace and both the robot and the obstacle are convex polygons, it is a simple matter to compute the configuration space obstacle region $\mathcal{QO}$. Let $V_i^{\mathcal{A}}$ denote the vector that is normal to the i^{th} edge of the robot and $V_i^{\mathcal{O}}$ denote the vector that is normal to the i^{th} edge of the obstacle. Define a_i to be the vector from the origin of the robot's coordinate frame to the i^{th} vertex of the end effector, and b_j to be the vector from the origin of the world coordinate frame to the j^{th} vertex of the obstacle, as shown in Figure 7.1(a). The vertices of $\mathcal{QO}$ can be determined as follows.

- *For each pair $V_j^{\mathcal{O}}$ and $V_{j-1}^{\mathcal{O}}$, if $V_i^{\mathcal{A}}$ points between $-V_j^{\mathcal{O}}$ and $-V_{j-1}^{\mathcal{O}}$, then add to $\mathcal{QO}$ the vertices $b_j - a_i$ and $b_j - a_{i+1}$.*



(a) (b)

Figure 7.1: (a) The robot is a triangle-shaped rigid object in a two-dimensional workspace that contains a single rectangular obstacle. (b) The boundary of the configuration space obstacle $\mathcal{QO}$ (shown as a dashed line) can be obtained by computing the convex hull of the configurations at which the robot makes vertex-to-vertex contact with the single convex obstacle.

- *For each pair $V_i^{\mathcal{A}}$ and $V_{i-1}^{\mathcal{A}}$, if $V_j^{\mathcal{O}}$ points between $-V_i^{\mathcal{A}}$ and $-V_{i-1}^{\mathcal{A}}$, then add to $\mathcal{QO}$ the vertices $b_j - a_i$ and $b_{j+1} - a_i$.*

This is illustrated in Figure 7.1(b). Note that this algorithm essentially places the robot at all positions where vertex-to-vertex contact between robot and obstacle are possible. The origin of the robot's local coordinate frame at each such configuration defines a vertex of $\mathcal{QO}$. The polygon defined by these vertices is $\mathcal{QO}$.

If there are multiple convex obstacles $\mathcal{O}_i$, then the configuration space obstacle region is merely the union of the obstacle regions $\mathcal{QO}_i$ for the individual obstacles. For a nonconvex obstacle, the configuration space obstacle region can be computed by first decomposing the nonconvex obstacle into convex pieces $\mathcal{O}_i$ computing the configuration space obstacle region $\mathcal{QO}_i$ for each piece, and finally, computing the union of the $\mathcal{QO}_i$.

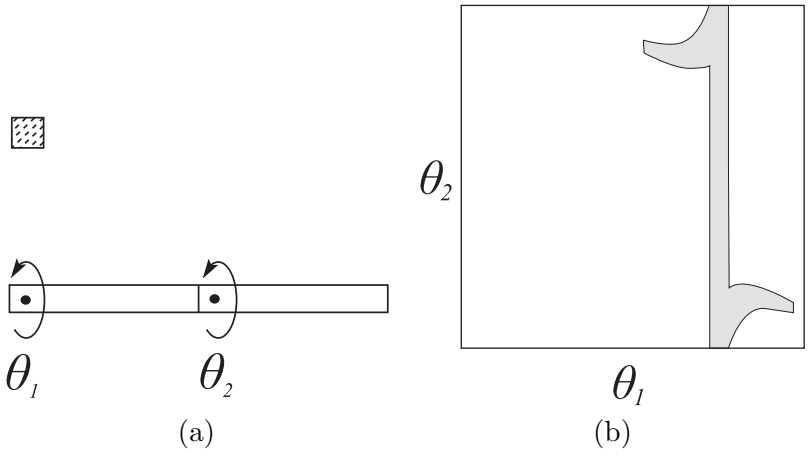


Figure 7.2: (a) The robot is a two-link planar arm and the workspace contains a single, small polygonal obstacle. (b) The corresponding configuration space obstacle region contains all configurations $q = (\theta_1, \theta_2)$ such that the arm at configuration q intersects the obstacle.

Example 7.2 (A Two-Link Planar Arm)**.** *The computation of $\mathcal{QO}$ is more difficult for robots with revolute joints. Consider a two-link planar arm in a workspace containing a single obstacle as shown in Figure 7.2(a). The configuration space obstacle region is illustrated in Figure 7.2(b).*

For values of θ_1 very near $\pi/2$, the first link of the arm collides with the obstacle. When the first link is near the obstacle (θ_1 near $\pi/2$), for some values of θ_2 the second link of the arm collides with the obstacle. The region $\mathcal{QO}$ shown in Figure 7.2(b) was computed using a discrete grid on the configuration space. For each cell in the grid, a collision test was performed, and the cell was shaded when a collision occurred. This is only an approximate representation of $\mathcal{QO}$; however, for robots with revolute joints, exact representations are very expensive to compute, and therefore such approximate representations are often used for robot arms with a few degrees of freedom.

Computing $\mathcal{QO}$ for the two-dimensional case of $\mathcal{Q} = \mathbb{R}^2$ and polygonal obstacles is straightforward, but, as can be seen from the two-link planar arm example, computing $\mathcal{QO}$ becomes difficult for even moderately complex configuration spaces. In the general case (for example, articulated arms or rigid bodies that can both translate and rotate), the problem of computing a representation of the configuration space obstacle region is intractable. One of the reasons for this complexity is that the size of the representation of the configuration space tends to grow exponentially with the number of

degrees of freedom. This is easy to understand intuitively by considering the number of n-dimensional unit cubes needed to fill a space of size k. For the one-dimensional case k unit intervals will cover the space. For the two-dimensional case k^2 squares are required. For the three-dimensional case k^3 cubes are required, and so on. Therefore, in this chapter we will develop methods that avoid the construction of an explicit representation of $\mathcal{QO}$ or of $\mathcal{Q}_{\text{free}}$.

7.1.3 Paths in the Configuration Space

The path planning problem is to find a path from an initial configuration q_{s} to a final configuration q_{f}, such that the robot does not collide with any obstacle as it traverses the path. More formally, a collision-free path from q_{s} to q_{f} is a continuous map, $\gamma : [0, 1] \to \mathcal{Q}_{\text{free}}$, with $\gamma(0) = q_{\text{s}}$ and $\gamma(1) = q_{\text{f}}$. We will develop path planning methods that compute a sequence of discrete configurations (set points) in the configuration space. In Section 7.5 we will show how smooth trajectories can be generated from such a sequence of set points.

7.2 Path Planning for $\mathcal{Q} = \mathbb{R}^2$

Before considering the general path planning problem, we first consider the special case when $\mathcal{Q} = \mathbb{R}^2$, and in particular, situations in which both the robot and obstacles can be represented as polygons in the plane, and in which the robot can move in any arbitrary direction (e.g., there are no constraints on the directions of motion such as would exist for car-like mobile robots, which cannot move in the direction perpendicular to the car's heading). Furthermore, we assume that the robot's workspace, and thus also its configuration space, is bounded. This is often an acceptable approximation for situations in which mobile robots navigate in workspaces such as warehouses, factory floors, office buildings, etc.

In this case, as we have seen in Example 7.1, it is a simple matter to construct an explicit representation of the configuration space obstacle region (and thus, of the free configuration space). In this section, we present three algorithms that can be used to construct collision-paths, given as input an explicit, polygonal representation of the configuration space obstacle regions. All three algorithms build graph data structures to represent the set of possible paths, and thus, once these representations have been constructed, the path planning problem is reduced to a graph search problem, for which many algorithms exist.

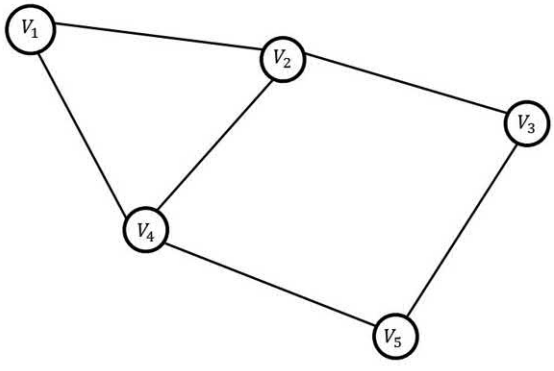


Figure 7.3: A graph with five vertices and six edges.

A graph is defined as $G = (V, E)$, in which V is a set of vertices, and E is a set of edges, each of which corresponds to a pair of vertices. An example is shown in Figure 7.3, in which the graph has vertices and edges

$$\begin{aligned} V &= \{v_1, v_2, ...v_5\} \\ E &= \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_1, v_4), (v_4, v_5), (v_3, v_5)\} \end{aligned}$$

Vertices are said to be adjacent if they are connected by an edge. A path from vertex v_i to vertex v_j is a sequence of adjacent vertices, beginning at v_i and ending at v_j, or equivalently, the set of edges defined by the pairwise adjacent vertices in this path. For example, the edges $(v_1, v_4), (v_4, v_2), (v_2, v_3), (v_3, v_5)$ define a path from v_1 to v_5.

7.2.1 The Visibility Graph

A visibility graph is a graph whose vertices can "see" each other, that is, edges in the graph correspond to paths that do not intersect the interior of any obstacle; the 'line of sight" between adjacent vertices is unoccluded. For path planning in a polygonal configuration space, the set of vertices V includes (a) the start and goal configurations q_{s} and q_{f}, and (b) every vertex of the configuration space obstacles. The set of edges, E, includes all edges (v_i, v_j) such that the line segment joining v_i to v_j lies entirely in the free configuration space, or such that (v_i, v_j) corresponds to an edge of a polygonal configuration space obstacle. Figure 7.4(a) shows a configuration space containing polygonal obstacle regions, and Figure 7.4(b) shows the corresponding visibility graph.

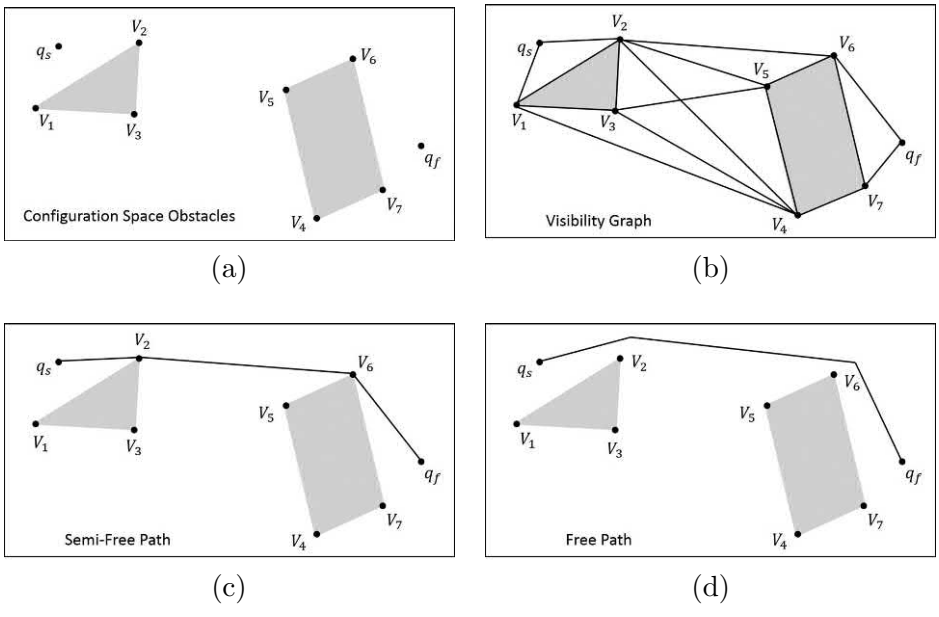


Figure 7.4: This figure illustrates the construction of a free path using the visibility graph. (a) An environment that contains polygonal obstacles, with start and goal configurations q_{s} and q_{f}. (b) The visibility graph. (c) A semi-free path from q_{s} to q_{f}. (d) A free path from q_{s} to q_{f}, obtained by a small perturbation of the path vertices that contact obstacle vertices for the semi-free path.

Any path in the visibility graph corresponds to a semi-free path in the configuration space, where by **semi-free** we mean that the path lies in the closure of the free configuration space (i.e., either in the free configuration space, or in the boundary of the configuration space obstacle region). However, it is always possible to deform a semi-free path into a **free** path by a small perturbation of the semi-free path. As an example, Figure 7.4(c) shows a semi-free path from q_{s} to q_{f} (note that the path contacts the boundary of the configuration space obstacles at the obstacle vertices v_2 and v_6), and Figure 7.4(d) shows a corresponding free path.

It can be shown (Problem 7–7) that the visibility graph contains the shortest semi-free paths for any q_{s} and q_{f}, provided these q_{s} and q_{f} are included in the construction of the visibility graph. It should be noted that such shortest paths may not be desirable in practice, since they bring the robot arbitrarily near to obstacles in the workspace, which could easily lead to collision if there is any uncertainty in the robot's location or in the map of the environment. For this reason, it is often preferable to plan paths that

maximize the clearance between the robot and any obstacles. The so-called **generalized Voronoi diagram**, discussed next, can be used to construct such paths.

7.2.2 The Generalized Voronoi Diagram

Consider a set of discrete points in the plane, $P = \{p_1, \ldots, p_n\}$. For each point p_i, we define its Voronoi cell to be the set of points in the plane that are closer to p_i than to any other $p_j \in P$,

$$\text{Vor}(p_i) = \{x \in \mathbb{R}^2 \mid \|x - p_i\| \leq \|x - p_j\|, \text{ for all } j \neq i\}$$

Two Voronoi regions are adjacent if $\text{Vor}(p_i) \cap \text{Vor}(p_j) \neq \emptyset$, in which case the intersection is a straight-line segment, referred to as a Voronoi edge, defined as

$$E_{ij} = \{x \in \mathbb{R}^2 \mid \|x - p_i\| = \|x - p_j\| \leq \|x - p_k\|, \text{for all } k \neq i, j\}$$

The Voronoi edge E_{ij} comprises the set of points in the plane that are minimally equidistant to the points p_i and p_j.

If we consider the points p_i as obstacles, then a collision-free path is merely a path that does not include any of the p_i. Define the **clearance** of a path γ to be the minimum distance between the path and any point $p \in P$, i.e.,

$$\rho(\gamma) = \min_{t \in [0,1]} \min_{p \in P} \|\gamma(t) - p\|$$

If q_s and q_f lie in Voronoi edges, then the maximum clearance path γ^* connecting q_s and q_f is contained completely in the set of Voronoi edges.

All of these concepts can be generalized from the case of discrete points to the case of polygons in the plane. A polygon consists of a set of vertices that are connected by edges. Together, these vertices and edges are sometimes referred to as the **features** that define the polygon. Thus, a polygon feature f is either a vertex, v or an edge, $e = (v, v')$. Now, let P be the set of features corresponding to the polygonal configuration space obstacles. The distance from a point x to a feature f of a polygon is defined as

$$d(x, f) = \begin{cases} \|x - v\| & : \quad f = v \\ \min_{\alpha \in [0,1]} \|x - (v - \alpha(v - v'))\| & : \quad f = e = (v, v') \end{cases}$$

in which $\|x - v\|$ is the Euclidean distance between the point x and the vertex v, and the expression $\min_{\alpha \in [0,1]} \|x - (v - \alpha(v - v'))\|$ gives the minimum distance from the point x to the edge $e = (v, v')$.

We define the Voronoi cell of feature f to be the set of points in the plane that are closer to f than to any other feature $f' \in P$,

$$\text{Vor}(f) = \{x \in \mathbb{R}^2 \mid d(x, f) \leq d(x, f'), \text{ for } f \neq f' \in P\}$$

We can also generalize the concept of Voronoi edges to be the set of points minimally equidistant to two features,

$$E_{ij} = \{x \in \mathbb{R}^2 \mid d(x, f_i) = d(x, f_j) \leq d(x, f_k), \text{for all } k \neq i, j\}$$

Since the set of features includes only points and line segments, the set of Voronoi edges will include only line segments that are minimally equidistant to two vertices or two edges, and sections of parabolas that are minimally equidistant to a vertex and an edge. We define the **generalized Voronoi diagram** as the graph $G = (V, E)$ whose edges are these Voronoi edges, and whose vertices are points at which multiple Voronoi edges intersect.

For the case of polygonal obstacles, we define the clearance of a path γ to be the minimum distance between the path and any feature $f \in P$, i.e.,

$$\rho(\gamma) = \min_{t \in [0,1]} \min_{f \in P} d(\gamma(t), f)$$

and, as above, if q_s and q_f lie in Voronoi edges, then the maximum clearance path γ^* connecting q_s and q_f is contained completely in the set of Voronoi edges.

For the case when q_s and q_f do not lie in Voronoi edges, it is necessary to also construct collision-free paths from each of q_s and q_f to Voronoi edges. This is actually quite simple to do. If, for example, q_s lies in the Voronoi region for a particular feature, f, a straight-line path from q_s that follows the gradient $\nabla d(q_\text{s}, f)$ will arrive to a Voronoi edge before reaching any other feature f' (Problem 7–8). For the case of an edge feature, this gradient is given by

$$\nabla d(q_\text{s}, f) = \frac{q_\text{s} - q^*}{\|q_\text{s} - q^*\|}$$

in which q^* is the closest point in the edge to q_s.

The path planning problem can now be solved as follows:

1. Find the straight-line path from q_s to the nearest Voronoi edge. Let ${q_\text{s}}'$ denote the point at which this path intersects the Voronoi edge.

2. Find the straight-line path from q_f to the nearest Voronoi edge. Let ${q_\text{f}}'$ denote the point at which this path intersects the Voronoi edge.

3. Find a path in the generalized Voronoi diagram from ${q_\text{s}}'$ to ${q_\text{f}}'$.

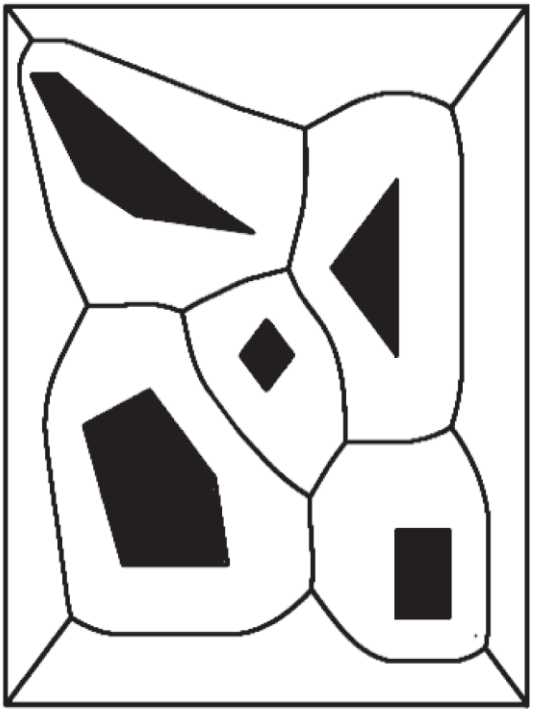


Figure 7.5: A polygonal configuration space containing five obstacles, and its generalized Voronoi diagram.

Efficient algorithms exist for computing the exact generalized Voronoi diagram, but in practice numerical, grid-based algorithms are often used, since (a) they are much easier to implement (b) the error induced by resorting to a discrete grid is typically small relative to the clearance achieved by paths that lie in the generalized Voronoi diagram, and (c) grid-based representations are feasible for two-dimensional space. Figure 7.5 shows a polygonal configuration space and the corresponding generalized Voronoi diagram.

7.2.3 Trapezoidal Decompositions

The visibility graph and the generalized Voronoi diagram are both graphs whose edges correspond directly to paths in the configuration space. For the visibility graph, edges correspond to portions of specific semi-free paths, while for the generalized Voronoi diagram, each Voronoi edge corresponds to a portion of a specific free path. Neither of these representations explicitly captures any information about the geometry of the free configuration space.

In contrast, a **spatial decomposition** of the free configuration space is a collection of regions $\{R_1, \ldots, R_m\}$ such that $\bigcup R_i = \mathcal{Q}_{\text{free}}$ and $\text{int}(R_i) \cap \text{int}(R_j) = \emptyset$ for all $i \neq j$, where $\text{int}(R)$ denotes the interior of region R. The regions R_i explicitly define the geometry of the free configuration space. A **convex spatial decomposition** has the additional property that each R_i is convex. Convex regions have the appealing property that for any $q_i, q_j \in R$, the line segment connecting them lies completely within R, that is, $q_i - \alpha(q_i - q_j) \in R$, for all $\alpha \in [0, 1]$. Therefore, constructing a free path between two configurations in the same convex subset of $\mathcal{Q}_{\text{free}}$ is trivial. A **trapezoidal decomposition** is a special case from the class convex spatial decomposition that applies to the case of polygons in the plane.

Figure 7.6(a) shows trapezoidal decomposition for the free configuration space for the case of polygonal obstacles. Each region in the decomposition is a trapezoid consisting of polygon edges and vertical line segments incident upon vertices of the polygons[2] A popular algorithm for constructing a trapezoidal decomposition proceeds by sweeping a vertical line across the configuration space, stopping at each vertex of the configuration space obstacle region, and making appropriate updates to the decomposition. Figure 7.6(b) shows one step in this process. At this stop of the sweep line, ℓ, regions R_3 and R_4 are "closed" and region R_5 is "opened."

The **connectivity graph** for a trapezoidal decomposition encodes adjacency relationships between the regions of the decomposition. The vertices of the connectivity graph correspond to the regions, and an edge exists between R_i and R_j if the two regions are adjacent (i.e., if $R_i \cap R_j \neq \emptyset$). Figure 7.6(c) shows the connectivity graph for the trapezoidal decomposition shown in Figure 7.6(a). For a given q_{s} and q_{f}, the path planning problem can be solved as follows.

1. Determine the region R_i that contains the initial configuration q_{s}.
2. Determine the region R_j that contains the final configuration q_{f}.
3. Find a path in the connectivity graph from R_i to R_j.
4. Construct a piecewise linear path from q_{s} and q_{f} that passes successively through the cells found in Step 3, crossing from one region to the next at the midpoint of the boundary of the two regions.

[2]Note that triangles are considered as trapezoids that have one side of length zero, and rectangles are special cases of trapezoids for which opposite sides are parallel.

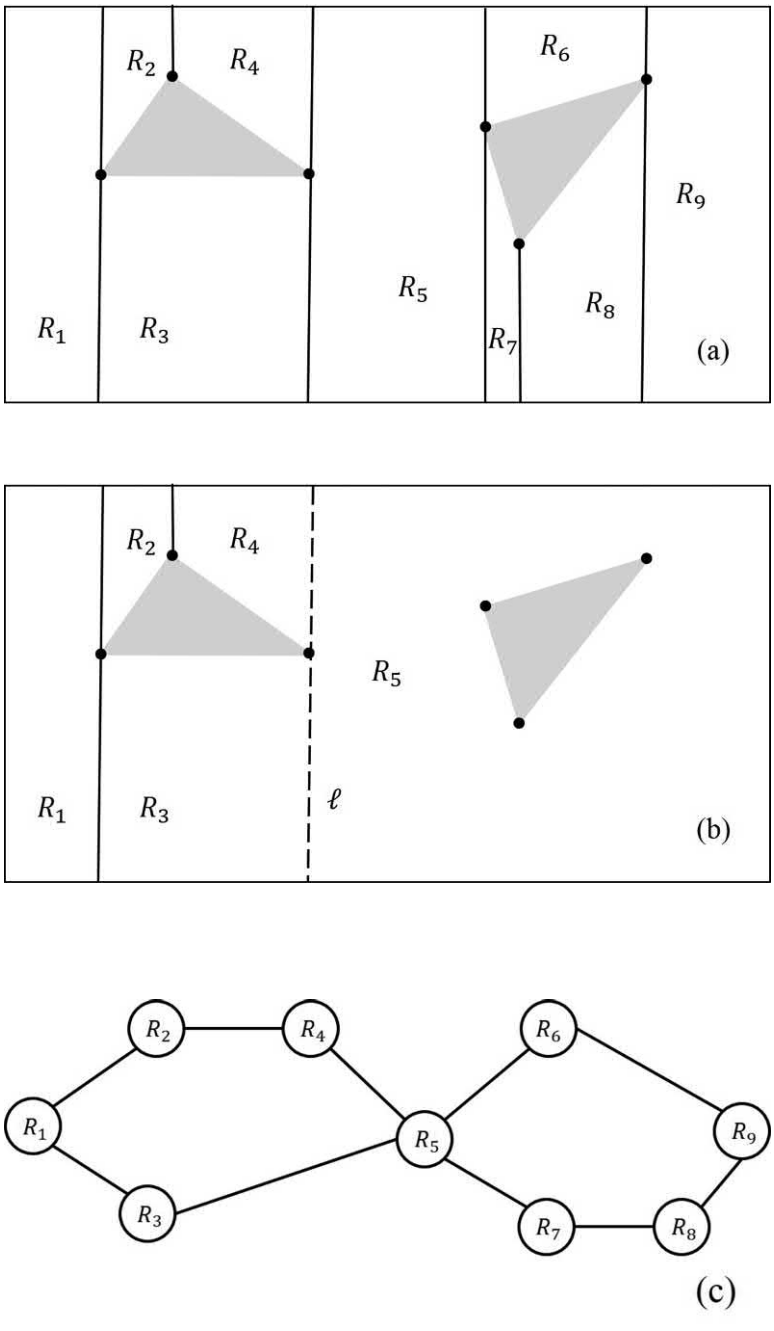


Figure 7.6: A trapezoidal decomposition for the free configuration space for the case of polygonal obstacles.

7.3 Artificial Potential Fields

The methods described in Section 7.2 are applicable only for very simple configuration spaces. For more complex configuration spaces (e.g., $SE(2)$ for a car-like robot, $SE(3)$ for an air vehicle, or the n-torus for an n-link arm) it is typically not feasible to build an explicit representation of $\mathcal{QO}$ or of $\mathcal{Q}_{\text{free}}$. An alternative is to develop a search algorithm that incrementally explores $\mathcal{Q}_{\text{free}}$ while searching for a path. One popular strategy for exploring $\mathcal{Q}_{\text{free}}$ uses an **artificial potential field** to guide the search.

The basic idea behind the potential field approach is to treat the robot as a point particle in the configuration space under the influence of an artificial potential field U. The field U is constructed so that the robot is attracted to the final configuration q_{f} while being repelled from the boundaries of $\mathcal{QO}$. If possible, U is constructed so that there is a single global minimum of U at q_{f} and there are no local minima. Unfortunately, it is typically difficult or even impossible to construct such a field.

In most potential field path planners, the field U is constructed as an additive field consisting of one component that attracts the robot to q_{f} and a second component that repels the robot from the boundary of $\mathcal{QO}$

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

Given this formulation path planning can be treated as an optimization problem, namely the problem of finding the global minimum of U starting from initial configuration q_{s}, and gradient descent methods are often used to find a solution. In physics, a conservative force field can be written as the negative gradient of a potential function. Thus, we may interpret gradient descent as being analogous to a particle moving under the influence of the force $F = -\nabla U$. Below we frequently refer to attractive and repulsive forces in order to exploit this natural intuition when defining potential fields.

We develop the potential field method for path planning in two stages. First, in Section 7.3.1, we develop the method for the special case of $\mathcal{Q} = \mathbb{R}^n$. By doing so, we can easily define U in terms of Euclidean distances, without dealing with the problem of defining metrics on arbitrary configuration spaces. This allows a straightforward initial development of the concepts and algorithms. In Section 7.3.2 we expand the approach to arbitrary configuration spaces. Rather than explicitly defining potential fields on these configuration spaces, we will define a collection of so-called **workspace potential fields**, and then show how the gradient of the corresponding configuration space potential can be computed using the Jacobian of the forward kinematic map. For both cases, we describe specific gradient descent algorithms for path planning.

7.3.1 Artificial Potential Fields for $\mathcal{Q} = \mathbb{R}^n$

For the case of $\mathcal{Q} = \mathbb{R}^n$, we will define an attractive potential in terms of the Euclidean distance to the goal, and a repulsive potential in terms of the Euclidean distance to the nearest obstacle boundary. We then describe how gradient descent methods can be used to find a collision-free path. Because gradient descent often fails for situations in which the potential field has multiple local minima, we end with a discussion of how randomization can be used to escape local minima of the potential function.

The Attractive Field

To attract the robot to its goal configuration, we will define an attractive potential field U_{att}. At the goal configuration $q = q_{\text{f}}$. There are several criteria that the potential field U_{att} should satisfy. First, U_{att} should be monotonically increasing with the distance to the goal configuration. The simplest choice for such a field grows linearly with this distance, a so-called **conic well potential**

$$U_{\text{att}}(q) = \zeta ||q - q_{\text{f}}||$$

in which ζ is a parameter used to scale the effects of the attractive potential. The gradient of such a field has unit magnitude everywhere except the goal configuration, where it is zero. This can lead to stability problems since there is a discontinuity in the attractive force at the goal position.

The **parabolic well potential** given by

$$U_{\text{att}}(q) = \frac{1}{2}\zeta ||q - q_{\text{f}}||^2$$

is continuously differentiable and increases monotonically with distance to the goal configuration. The attractive force is equal to the negative gradient of U_{att}, which is given by (Problem 7–9)

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(q) = -\zeta(q - q_{\text{f}}) \tag{7.1}$$

For the parabolic well the attractive force is a vector directed toward q_{f} with magnitude linearly related to the distance to q from q_{f}.

While this force converges linearly to zero as q approaches q_{f}, which is a desirable property, it grows without bound as q moves away from q_{f}. If q_{s} is very far from q_{f}, this may produce an initial attractive force that is very large. For this reason we may choose to combine the quadratic and

conic potentials so that the conic potential is active when q is distant from q_{f}, and the quadratic potential is active when q is near q_{f}. Of course, it is necessary that the gradient be defined at the boundary between the conic and quadratic fields. Such a field can be defined by

$$U_{\mathrm{att}}(q) = \begin{cases} \frac{1}{2}\zeta||q - q_{\mathrm{f}}||^2 & : \quad ||q - q_{\mathrm{f}}|| \leq d \\ \\ d\zeta||q - q_{\mathrm{f}}|| - \frac{1}{2}\zeta d^2 & : \quad ||q - q_{\mathrm{f}}|| > d \end{cases}$$

in which d is the distance that defines the transition from conic to parabolic well. In this case the force is given by

$$F_{\mathrm{att}}(q) \quad = \quad \begin{cases} -\zeta(q - q_{\mathrm{f}}) & : \quad ||q - q_{\mathrm{f}}|| \leq d \\ \\ -d\zeta \dfrac{(q - q_{\mathrm{f}})}{||q - q_{\mathrm{f}}||} & : \quad ||q - q_{\mathrm{f}}|| > d \end{cases}$$

The gradient is well defined at the boundary of the two fields since at the boundary $d = ||q - q_{\mathrm{f}}||$ and the gradient of the quadratic potential is equal to the gradient of the conic potential $F_{\mathrm{att}}(q) = -\zeta(q - q_{\mathrm{f}})$.

The Repulsive Field

In order to prevent collisions between the robot and obstacles we will define a **repulsive potential field** that grows as the configuration approaches the boundary of $\mathcal{QO}$. There are several criteria that such a repulsive field should satisfy. It should repel the robot from obstacles, never allowing the robot to collide with an obstacle, and, when the robot is far away from an obstacle, that obstacle should exert little or no influence on the motion of the robot. One way to achieve this is to define a potential function whose value approaches infinity as the configuration approaches an obstacle boundary, and whose value decreases to zero at a specified distance from the obstacle boundary.

We define $\rho(q)$ to be the distance from configuration q to the boundary of $\mathcal{QO}$,

$$\rho(q) = \min_{q' \in \partial\mathcal{QO}} \|q - q'\|$$

in which $\partial\mathcal{QO}$ denotes the boundary of the configuration space obstacle region. We define ρ_0 to be the distance of influence of an obstacle. This means that an obstacle will not repel the robot if the distance from q to the obstacle is greater than ρ_0.

One potential function that meets the criteria described above is given by

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & : \quad \rho(q) \leq \rho_0 \\ \\ 0 & : \quad \rho(q) > \rho_0 \end{cases}$$

in which η is a parameter used to scale the effects of the attractive potential. The repulsive force is equal to the negative gradient of U_{rep}. For $\rho(q) \leq \rho_0$, this force is given by (Problem 7–13)

$$F_{\text{rep}}(q) = \eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\nabla\rho(q) \tag{7.2}$$

If $\mathcal{QO}$ is convex and b is the point on the boundary of $\mathcal{QO}$ that is closest to q, then $\rho(q) = ||q - b||$, and its gradient is

$$\nabla\rho(q) = \frac{q - b}{||q - b||}$$

that is, the unit vector directed from b toward q.

If the obstacle is not convex, then the distance function ρ will not necessarily be differentiable everywhere, which implies discontinuity in the force vector. Figure 7.7 illustrates such a case. Here the obstacle region is defined by two rectangular obstacles. For all configurations to the left of the dashed line the force vector points to the right, while for all configurations to the right of the dashed line the force vector points to the left. Thus, when q crosses the dashed line, a discontinuity in force occurs. There are various ways to deal with this problem. The simplest of these is merely to ensure that the regions of influence of distinct obstacles do not overlap.

Gradient Descent Planning

Gradient descent is a well-known approach for solving optimization problems. The idea is simple. Starting at the initial configuration, take a small step in the direction of the negative gradient (which is the direction that decreases the potential as quickly as possible). This gives a new configuration, and the process is repeated until the final configuration is reached. More formally, a gradient descent algorithm constructs a sequence[3] of configurations, $q^0, q^1, \ldots, q^m$ such that $q^0 = q_{\text{s}}$ and $q^m = q_{\text{f}}$. The configuration

[3]Note that q^i is used to denote the value of q at the i^{th} iteration (not the i^{th} component of the vector q).

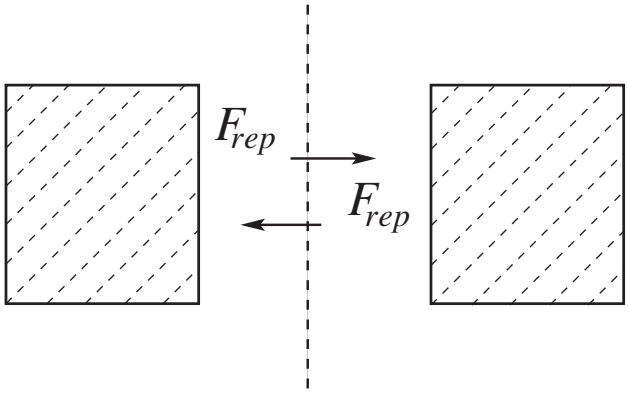


Figure 7.7: In this case the gradient of the repulsive potential given by Equation (7.2) is not continuous. In particular, the gradient changes discontinuously when q crosses the line midway between the two obstacles.

at step $i+1$ is given by

$$q^{i+1} = q^i - \alpha^i \nabla U(q^i) \tag{7.3}$$

The scalar parameter α^i scales the step size at the i^{th} iteration. Some variations of gradient descent replace the gradient $\nabla U(q^i)$ by a unit vector in the direction of the gradient; in this case α^i completely determines the step size at the i^{th} iteration. It is important that α^i be small enough that the robot is not allowed to "jump into" obstacles while being large enough that the algorithm does not require excessive computation time. In motion planning problems the choice for α^i is often made on an ad hoc or empirical basis, for example, based on the distance to the nearest obstacle or to the goal. A number of systematic methods for choosing α^i can be found in the optimization literature.

It is unlikely that we will ever exactly satisfy the condition $q^i = q_{\mathrm{f}}$ and for this reason gradient descent algorithms typically terminate when q^i is sufficiently near the goal configuration q_{f}, for example when $||q^i - q_{\mathrm{f}}|| < \epsilon$, where we choose ϵ to be a sufficiently small constant, based on the task requirements.

Escaping Local Minima

The problem that plagues all gradient descent algorithms is the possible existence of local minima in the potential field. For appropriate choice of α^i in (7.3), it can be shown that the gradient descent algorithm is guaranteed

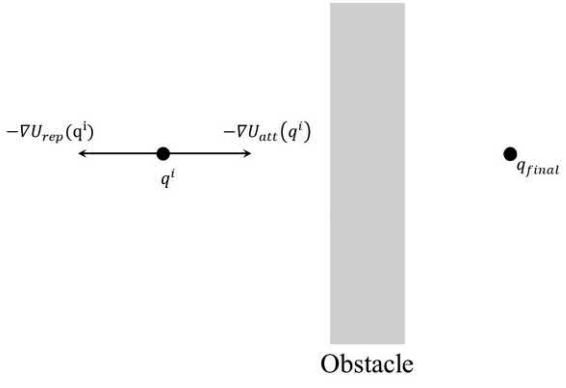


Figure 7.8: The configuration q^i is a local minimum in the potential field. At q^i the attractive force exactly cancels the repulsive force and the planner fails to make further progress.

to converge to a minimum in the field, but there is no guarantee that this minimum will be the global minimum. In our case this implies that there is no guarantee that this method will find a path to q_f. An simple example of this situation is shown in Figure 7.8.

This problem has long been known in the optimization community, where probabilistic methods such as simulated annealing have been developed to cope with it. Similarly, **randomized methods** have been developed to deal with this and other problems in robot motion planning. One method for escaping local minima combines gradient descent with randomization. This approach uses gradient descent until the planner finds itself stuck in a local minimum, and then uses a random walk to escape the local minimum. This requires solving two problems: determining when the planner is stuck in a local minimum and defining the random walk.

Typically, a heuristic is used to recognize a local minimum. For example, if several successive q^i lie within a small region of the configuration space, it is likely that there is a nearby local minimum. For example, if for some small positive ϵ_m we have $\|q^i - q^{i+1}\| < \epsilon_m$, $\|q^i - q^{i+2}\| < \epsilon_m$, and $\|q^i - q^{i+3}\| < \epsilon_m$ then assume q^i is near a local minimum, provided q^i is not sufficiently near the goal configuration.

Defining the random walk requires a bit more care. One approach is to simulate Brownian motion. The random walk consists of t random steps. A random step from $q = (q_1, \ldots, q_n)$ is obtained by randomly adding a small

fixed constant to each q_i,

$$q_{\text{random}-\text{step}} = (q_1 \pm v_1, \ldots, q_n \pm v_n)$$

with v_i a fixed small constant and the probability of adding $+v_i$ or $-v_i$ equal to 1/2 (that is, a uniform distribution). Without loss of generality, assume that $q = 0$. We can use probability theory to characterize the behavior of the random walk consisting of t random steps. In particular, if q^t is the configuration reached after t random steps, the probability density function for $q^t = (q_1, \ldots, q_n)$ is given by

$$p_i(q_i, t) \approx \frac{1}{v_i\sqrt{2\pi t}} \exp\left(-\frac{q_i^2}{2v_i^2 t}\right)$$

which is a zero mean Gaussian density function[4] with variance $v_i^2 t$. This is a result of the central limit theorem, which states that the probability density function for the sum of k independent, identically distributed random variables tends to a Gaussian density function as $k \to \infty$. The variance $v_i^2 t$ essentially determines the range of the random walk. If certain characteristics of local minima (for example, the size of the basin of attraction) are known in advance, these can be used to select the parameters v_i and t. Otherwise, they can be determined empirically or based on weak assumptions about the potential field.

7.3.2 Potential Fields for $\mathcal{Q} \neq \mathbb{R}^n$

For the case of $\mathcal{Q} \neq \mathbb{R}^n$, it is difficult to construct a potential field directly on the configuration space, and difficult to compute the gradient of such a field. The reasons for this include the difficulty of computing shortest distances to configuration space obstacles, the complex geometry of the configuration space itself, and the computational cost of computing an explicit representation of the boundary of the configuration space obstacle region. For this reason, when $\mathcal{Q} \neq \mathbb{R}^n$ we will define **workspace potential fields** directly on the workspace of the robot, and then map the gradients of these fields to a corresponding gradient for a configuration space potential function.

In particular, for an n-link arm, we will define a potential field for each of the origins of the n DH frames (excluding the fixed, frame 0). These

[4] A Gaussian density function is the classical bell shaped curve. The mean indicates the center of the curve (the peak of the bell) and the variance indicates the width of the bell. The probability density function (pdf) tells how likely it is that the variable q_i will lie in a certain interval. The higher the pdf values, the more likely that q_i will lie in the corresponding interval.

workspace potential fields will attract the origins of the DH frames to their goal locations while repelling them from obstacles. We will use these fields to define motions in the configuration space using the manipulator Jacobian matrix. A similar approach can be used for mobile robots. In this case, we define a set of **control points** on the robot that are sufficient to fully constrain its position, and define workspace potentials for each of these. For a mobile robot in the plane, two points are sufficient, while three (noncollinear) points are sufficient for free-flying robots.

The Attractive Field

To attract the robot to its goal configuration, we will define an attractive potential field $U_{\text{att},i}$ for o_i, the origin of the i^{th} DH frame. When all n origins reach their goal positions, the arm will have reached its goal configuration. As above, choices include the conic well and parabolic well potentials.

If we denote the position of the origin of the i^{th} DH frame by $o_i(q)$, then the conic well potential is given by

$$U_{\text{att},i}(q) = \zeta_i ||o_i(q) - o_i(q_\text{f})||$$

in which ζ_i is a parameter used to scale the effects of the attractive potential. The parabolic well potential is given by

$$U_{\text{att},i}(q) = \frac{1}{2}\zeta_i ||o_i(q) - o_i(q_\text{f})||^2$$

For the parabolic well, the workspace attractive force for o_i is equal to the negative gradient of $U_{\text{att},i}$, which is given by

$$F_{\text{att},i}(q) = -\nabla U_{\text{att},i}(q) = -\zeta(o_i(q) - o_i(q_\text{f}))$$

which is a vector directed toward $o_i(q_\text{f})$ with magnitude linearly related to the distance to $o_i(q)$ from $o_i(q_\text{f})$.

We can combine the conic and parabolic well potentials as we did in Section 7.3.1, which yields

$$U_{\text{att},i}(q) = \begin{cases} \frac{1}{2}\zeta_i ||o_i(q) - o_i(q_\text{f})||^2 & : \quad ||o_i(q) - o_i(q_\text{f})|| \leq d \\ \\ d\zeta_i ||o_i(q) - o_i(q_\text{f})|| - \frac{1}{2}\zeta_i d^2 & : \quad ||o_i(q) - o_i(q_\text{f})|| > d \end{cases}$$

in which d is the distance that defines the transition from conic to parabolic

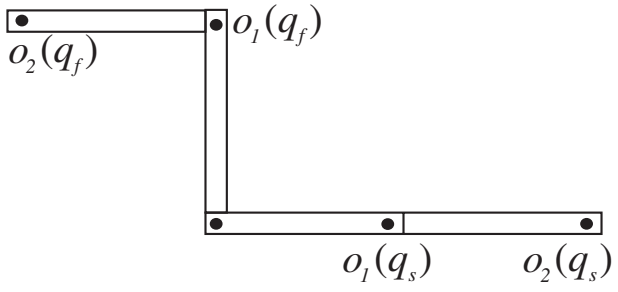


Figure 7.9: The initial configuration for the two-link arm is given by $\theta_1 = \theta_2 = 0$ and the final configuration is given by $\theta_1 = \theta_2 = \pi/2$. The origins for DH frames 1 and 2 are shown at both q_s and q_f.

well. In this case the workspace force for o_i is given by

$$F_{\mathrm{att},i}(q) = \begin{cases} -\zeta_i(o_i(q) - o_i(q_\mathrm{f})) & : \ ||o_i(q) - o_i(q_\mathrm{f})|| \leq d \\ -d\zeta_i \dfrac{(o_i(q) - o_i(q_\mathrm{f}))}{||o_i(q) - o_i(q_\mathrm{f})||} & : \ ||o_i(q) - o_i(q_\mathrm{f})|| > d \end{cases} \tag{7.4}$$

The gradient is well defined at the boundary of the two fields since at the boundary $d = ||o_i(q) - o_i(q_\mathrm{f})||$ and the gradient of the quadratic potential is equal to the gradient of the conic potential $F_{\mathrm{att},i}(q) = -\zeta_i(o_i(q) - o_i(q_\mathrm{f}))$.

Example 7.3 (Two-Link Planar Arm)**.** *Consider the two-link planar arm shown in Figure 7.9 with $a_1 = a_2 = 1$ and with initial and final configurations given by*

$$q_\mathrm{s} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad q_\mathrm{f} = \begin{bmatrix} \frac{\pi}{2} \\ \frac{\pi}{2} \end{bmatrix}$$

Using the forward kinematic equations for this arm (see Example 3.3.1) we obtain

$$o_1(q_\mathrm{s}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad o_1(q_\mathrm{f}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad o_2(q_\mathrm{s}) = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \qquad o_2(q_\mathrm{f}) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Using these coordinates for the origins of the two DH frames at their initial and goal configurations, assuming that d is sufficiently large, we obtain the

attractive forces

$$F_{\text{att},1}(q_s) = -\zeta_1(o_1(q_s) - o_1(q_f)) = \zeta_1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$
$$F_{\text{att},2}(q_s) = -\zeta_2(o_2(q_s) - o_2(q_f)) = \zeta_2 \begin{bmatrix} -3 \\ 1 \end{bmatrix}$$

The Repulsive Field

To prevent collisions, we will define a workspace repulsive potential field for the origin of each DH frame (excluding frame 0). Note that by defining repulsive potentials only for the origins of the DH frames we cannot ensure that collisions never occur (for example, the middle portion of a long link might collide with an obstacle), but it is fairly easy to modify the method to prevent such collisions as we will see below. For now, we will deal only with the origins of the DH frames.

We define $\rho(o_i(q))$ to be the distance in the workspace from the origin of DH frame i to the nearest obstacle

$$\rho(o_i(q)) = \min_{x \in \partial\mathcal{O}} \|o_i(q) - x\|$$

Likewise, we now define ρ_0 to be the workspace distance of influence of an obstacle. This means that an obstacle will not repel o_i if the distance from o_i to the obstacle is greater than ρ_0.

Our workspace repulsive potential is now given by

$$U_{\text{rep},i}(q) = \begin{cases} \frac{1}{2}\eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0} \right)^2 & : \quad \rho(o_i(q)) \leq \rho_0 \\ 0 & : \quad \rho(o_i(q)) > \rho_0 \end{cases}$$

The workspace repulsive force is equal to the negative gradient of $U_{\text{rep},i}$. For $\rho(o_i(q)) \leq \rho_0$, this force is given by

$$F_{\text{rep},i}(q) = \eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(o_i(q))} \nabla\rho(o_i(q)) \tag{7.5}$$

in which the notation $\nabla\rho(o_i(q))$ indicates the gradient $\nabla\rho(x)$ evaluated at $x = o_i(q)$. If the obstacle region is convex and b is the point on the obstacle boundary that is closest to o_i, then $\rho(o_i(q)) = ||o_i(q) - b||$, and its gradient is

$$\nabla\rho(x)\Big|_{x=o_i(q)} = \frac{o_i(q) - b}{||o_i(q) - b||}$$

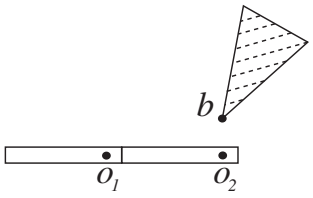


Figure 7.10: The obstacle shown repels o_2, but is outside the distance of influence for o_1. Therefore, it exerts no repulsive force on o_1.

that is, the unit vector directed from b toward $o_i(q)$.

Example 7.4 (Two-Link Planar Arm)**.** *Consider Example 7.3, with a single convex obstacle in the workspace as shown in Figure 7.10. Let* $\rho_0 = 1$*. This prevents the obstacle from repelling* o_1*, which is reasonable since link 1 can never contact the obstacle. The nearest obstacle point to* o_2 *is the vertex* b *of the polygonal obstacle. Suppose that* b *has the coordinates* $(2, 0.5)$*. Then the distance from* $o_2(q_s)$ *to* b *is* $\rho(o_2(q_s)) = 0.5$ *and* $\nabla\rho(o_2(q_s)) = [0, -1]^T$*. The repulsive force at the initial configuration for* o_2 *is then given by*

$$F_{\text{rep},2}(q_s) = \eta_2 \left(\frac{1}{0.5} - 1 \right) \frac{1}{0.25} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \eta_2 \begin{bmatrix} 0 \\ -4 \end{bmatrix}$$

This force has no effect on joint 1, but causes joint 2 to rotate slightly in the clockwise direction, moving link 2 away from the obstacle.

As mentioned above, defining repulsive fields only for the origins of the DH frames does not guarantee that the robot cannot collide with an obstacle. Figure 7.11 shows an example where this is the case. In this figure o_1 and o_2 are very far from the obstacle and therefore the repulsive influence may not be great enough to prevent link 2 from colliding with the obstacle. To cope with this problem we can use a set of **floating repulsive control points** $o_{float,i}$ typically one per link. The floating control points are defined as points on the boundary of a link that are closest to any workspace obstacle. Obviously the choice of the $o_{float,i}$ depends on the configuration q. For the case shown in Figure 7.11, $o_{float,2}$ would be located near the center of link 2, thus repelling the robot from the obstacle. The repulsive force acting on $o_{float,i}$ is defined in the same way as for the other control points using Equation (7.5).

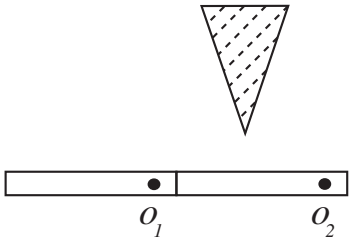


Figure 7.11: The repulsive forces exerted on the origins of the DH frames o_1 and o_2 may not be sufficient to prevent a collision between link 2 and the obstacle.

Mapping Workspace Forces to Joint Torques

We have shown how to construct potential fields in the robot's workspace that induce artificial forces on the origins o_i of the DH frames for the robot arm. In this section we describe how these artificial forces can be mapped to artificial joint torques.

As we derived in Chapter 4 using the principle of virtual work, if τ denotes the vector of joint torques induced by the workspace force F exerted at the end effector, then

$$J_v^T F = \tau$$

where J_v includes the top three rows of the manipulator Jacobian. We do not use the lower three rows, since we have considered only attractive and repulsive workspace forces, and not attractive and repulsive workspace torques. Note that for each o_i an appropriate Jacobian must be constructed, but this is straightforward given the techniques described in Chapter 4 and the A matrices for the arm. We denote the Jacobian for o_i by J_{o_i}.

Example 7.5 (Two-Link Planar Arm)**.** *Consider again the two-link arm of Example 7.3 with repulsive workspace forces as given in Example 7.4. The Jacobians that map joint velocities to linear velocities satisfy*

$$\dot{o}_i = J_{o_i}(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

For the two-link arm the Jacobian matrix for o_2 is merely the Jacobian that we derived in Chapter 4, namely

$$J_{o_2}(q_1, q_2) = \begin{bmatrix} -s_1 - s_{12} & -s_{12} \\ c_1 + c_{12} & c_{12} \end{bmatrix}$$

The Jacobian matrix for o_1 is similar, but takes into account that motion of joint 2 does not affect the velocity of o_1. Thus

$$J_{o_1}(q_1, q_2) = \begin{bmatrix} -s_1 & 0 \\ c_1 & 0 \end{bmatrix}$$

At $q_s = (0, 0)$ we have

$$J_{o_1}^T(q_s) = \begin{bmatrix} -s_1 & c_1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

and

$$J_{o_2}^T(q_s) = \begin{bmatrix} -s_1 - s_{12} & c_1 + c_{12} \\ -s_{12} & c_{12} \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix}$$

Using these Jacobians, we can easily map the workspace attractive and repulsive forces to joint torques. If we let $\zeta_1 = \zeta_2 = \eta_2 = 1$ we obtain

$$\tau_{\text{att},1}(q_s) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\tau_{\text{att},2}(q_s) = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\tau_{\text{rep},2}(q_s) = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -4 \end{bmatrix} = \begin{bmatrix} -8 \\ -4 \end{bmatrix}$$

The total artificial joint torque acting on the arm is the sum of the artificial joint torques that result from all attractive and repulsive potentials

$$\tau(q) = \sum_i J_{o_i}^T(q) F_{\text{att},i}(q) + \sum_i J_{o_i}^T(q) F_{\text{rep},i}(q) \tag{7.6}$$

It is essential that we add the joint torques and not the workspace forces. In other words, we must use the Jacobians to transform forces to joint torques before we combine the effects of the potential fields. For example, Figure 7.12 shows a case in which two workspace forces F_1 and F_2, act on opposite corners of a rectangle. It is easy to see that $F_1 + F_2 = 0$, but that the combination of these forces produces a pure torque about the center of the rectangle.

Example 7.6 (Two-Link Planar Arm)**.** *Consider again the two-link planar arm of Example 7.3, with joint torques as determined in Example 7.5. In this*

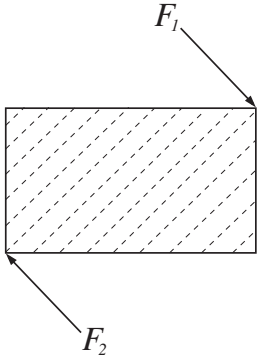


Figure 7.12: The two forces illustrated in the figure are vectors of equal magnitude in opposite directions. Vector addition of these two forces produces zero net force, but there is a net torque induced by these forces.

case the total joint torque induced by the attractive and repulsive workspace potential fields is given by

$$\begin{aligned}\tau(q_{\mathrm{s}}) &= \tau_{\mathrm{att},1}(q_{\mathrm{s}}) + \tau_{\mathrm{att},2}(q_{\mathrm{s}}) + \tau_{\mathrm{rep},2}(q_{\mathrm{s}}) \\ &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -8 \\ -4 \end{bmatrix} = \begin{bmatrix} -5 \\ -3 \end{bmatrix}\end{aligned}$$

These joint torques have the effect of causing each joint to rotate in a clockwise direction, away from the goal, due to the close proximity of o_2 to the obstacle. By choosing a smaller value for η_2, this effect can be overcome.

Application to Mobile Robots

The methods described in this section can easily be extended to the case of mobile robots. To do so, we define a set of control points on the robot $\{o_i\}_{i=1\ldots m}$ such that $q = q_{\mathrm{f}}$ when $o_i(q) = o_i(q_{\mathrm{f}})$ for $i = 1, \ldots m$. We then proceed as above, treating these o_i in the same way that we treated DH frames. The Jacobian matrix used in this case is given by

$$J_{o_i}(q) = \left[\frac{\partial o_i}{\partial q}\right]$$

and an appropriate gradient for the configuration space potential function is given by

$$\tau(q) = \sum_i J_{o_i}^T(q) F_{\mathrm{att},i}(q) + \sum_i J_{o_i}^T(q) F_{\mathrm{rep},i}(q)$$

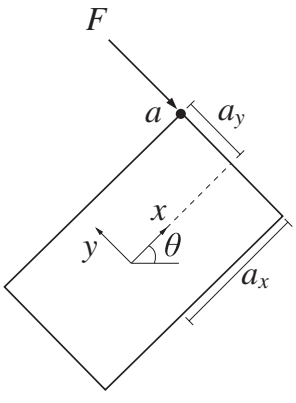


Figure 7.13: In this example, the robot is a polygon whose configuration can be represented as $q = (x, y, \theta)$, in which θ is the angle from the world x-axis to the x-axis of the robot's local frame. A force F is exerted on vertex a with local coordinates (a_x, a_y).

in which τ includes forces and moments applied with respect to the body-attached coordinate frame of the robot.

Example 7.7 (A Polygonal Robot in the Plane)**.** *Consider the polygonal robot shown in Figure 7.13. The vertex a has coordinates (a_x, a_y) in the robot's local coordinate frame. Therefore, if the robot's configuration is given by $q = (x, y, \theta)$, the forward kinematic map for vertex a (that is, the mapping from $q = (x, y, \theta)$ to the global coordinates of the vertex a) is given by*

$$a(x, y, \theta) = \begin{bmatrix} x + a_x \cos\theta - a_y \sin\theta \\ y + a_x \sin\theta + a_y \cos\theta \end{bmatrix}$$

The corresponding Jacobian matrix is given by

$$J_a(x, y, \theta) = \begin{bmatrix} 1 & 0 & -a_x \sin\theta - a_y \cos\theta \\ 0 & 1 & a_x \cos\theta - a_y \sin\theta \end{bmatrix}$$

Using the transpose of the Jacobian to map the workspace forces to generalized forces, we obtain

$$J_a^T(x, y, \theta) \begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ -F_x(a_x \sin\theta - a_y \cos\theta) + F_y(a_x \cos\theta - a_y \sin\theta) \end{bmatrix}$$

The bottom entry in this vector corresponds to the torque exerted about the origin of the robot frame.

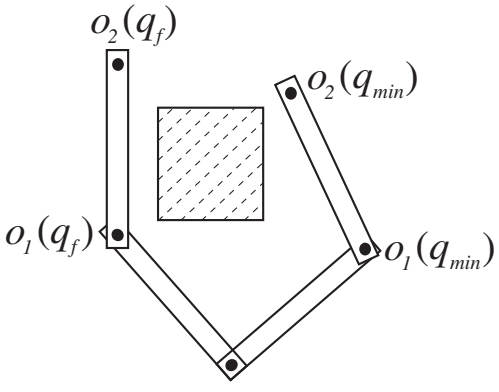


Figure 7.14: The configuration $q_{\min}$ is a local minimum in the potential field. At $q_{\min}$ the attractive force exactly cancels the repulsive force and the planner fails to make further progress.

Gradient Descent Planning

As above, the gradient descent algorithm constructs a sequence of configurations, $q^0, q^1, \ldots, q^m$ such that $q^0 = q_{\rm s}$ and $q^m = q_{\rm f}$, but in this case the k^{th} iteration is given by

$$q^{k+1} = q^k + \alpha^k \frac{\tau(q^k)}{||\tau(q^k)||}$$

in which τ is given by (7.6). The scalar α^k determines the step size at the k^{th} iteration, and the algorithm terminates when $||q^k - q_{\rm f}|| < \epsilon$, where we choose ϵ to be a sufficiently small constant, based on the task requirements.

The problem of local minima in the potential field also exists for this approach to defining workspace potentials and mapping workspace gradients to the configuration space. An example of this situation is shown in Figure 7.14.

There are a number of design choices that must be made when using this approach.

- The constant ζ_i in Equation (7.4) controls the relative influence of the attractive potential for control point o_i. It is not necessary that all of the ζ_i be set to the same value. Typically we assign a larger weight to one of the o_i than to the others, producing a "follow the leader" type of motion, in which the leader o_i is quickly attracted to its final position and the robot then reorients itself so that the other o_i reach their final positions.

- The constant η_i in Equation (7.5) controls the relative influence of the repulsive potential for o_i. As with the ζ_i it is not necessary that all of the η_i be set to the same value. In particular, we typically set the value of η_i to be much smaller for obstacles that are near the goal position of the robot (to avoid having these obstacles repel the robot from the goal).

- The constant ρ_0 in Equation (7.5) defines the distance of influence for obstacles. As with the η_i we can define a distinct value of ρ_0 for each obstacle. In particular, we do not want any obstacle's region of influence to include the goal position of any repulsive control point. We may also wish to assign distinct values of ρ_0 to the obstacles to avoid the possibility of overlapping regions of influence for distinct obstacles.

7.4 Sampling-Based Methods

The potential field approaches described above incrementally explore $\mathcal{Q}_{\text{free}}$ by generating a sequence of configurations $q^0, \ldots, q^m$ using a gradient descent approach. This exploration is inherently goal-driven (due to the attractive potential), and it is this bias that makes the approach susceptible to failure due to the presence of local minima in the potential field. As we have seen, applying a random walk can be an effective way to escape local minima, abandoning temporarily the goal-driven behavior in favor of a randomized strategy. Taken to an extreme, we could design a planner that completely abandons goal-driven search, instead relying completely on randomized strategies. This is the approach taken by **sampling-based planners**[5].

Sampling-based planners generate a sequence of configurations using a random sampling strategy. The simplest such strategy is merely to generate random samples from a uniform probability distribution on the configuration space. If two samples are sufficiently near to one another, planning a path between them can be accomplished using a simple, local planner. Iteratively applying this strategy produces a graph $G = (V, E)$ in which the vertex set V includes the randomly generated sample configurations, and edges correspond to local paths between sample configurations that lie in close proximity to one another. The graph G is referred to as a **configuration**

[5]While there are some sampling-based planners that use quasi-random, or even deterministic sampling strategies, the use of randomized approaches is far more prevalent, and here we consider only these approaches.

space roadmap.

In this section, we will describe two sampling-based planning algorithms. The first algorithm builds a **probabilistic roadmap** or **PRM**, which is a roadmap that attempts to uniformly cover the entire free configuration space. This approach is particularly useful when many planning problems are to be solved for a single workspace, so that the cost of building the roadmap can be amortized over many planning instances. The second algorithm builds a **rapidly-exploring random Tree** or **RRT**, which is a random tree whose root vertex corresponds to the initial configuration q_{s}. By using a clever sampling strategy to generate new vertices in the tree, this method is able to rapidly explore the free configuration space, and has proven to be effective for solving even difficult path planning problems.

7.4.1 Probabilistic Roadmaps (PRM)

In general, a configuration space roadmap is a one-dimensional network of curves that effectively represents $\mathcal{Q}_{\mathrm{free}}$. A roadmap is typically represented as a graph, in which edges correspond to curve segments, whose intersections correspond to vertices. When using roadmap methods, planning comprises three stages: (1) find a path from q_{s} to a configuration q_a in the roadmap, (2) find a path from q_{f} to a configuration q_b in the roadmap, (3) find a path in the roadmap from q_a to q_b. Steps (1) and (2) are typically much easier than finding a path from q_{s} to q_{f}. The visibility graph and generalized Voronoi diagram described in Section 7.2 are both examples of configuration space roadmaps (although for the visibility graph, both q_{s} and q_{f} are included in the roadmap by construction).

A **probabilistic roadmap**, or **PRM**, is a configuration space roadmap whose vertices correspond to randomly generated configurations, and whose edges correspond to collision-free paths between configurations. Constructing a PRM is a conceptually straightforward process. First, a set of random configurations is generated to serve as the vertices in the roadmap. Then, a simple, local path planner is used to generate paths that connect pairs of configurations. Finally, if the initial roadmap consists of multiple connected components[6], it is augmented during an enhancement phase, in which new vertices and edges are added in an attempt to connect disjoint components of the roadmap. To solve a path planning problem, the simple, local planner is used to add q_{s} and q_{f} to the roadmap, and the resulting roadmap is searched for a path from q_{s} to q_{f}. These four steps are illustrated in Figure

[6] A connected component is a maximal subgraph of the graph such that a path exists in the subgraph between any two vertices.

7.15. We now discuss these steps in more detail.

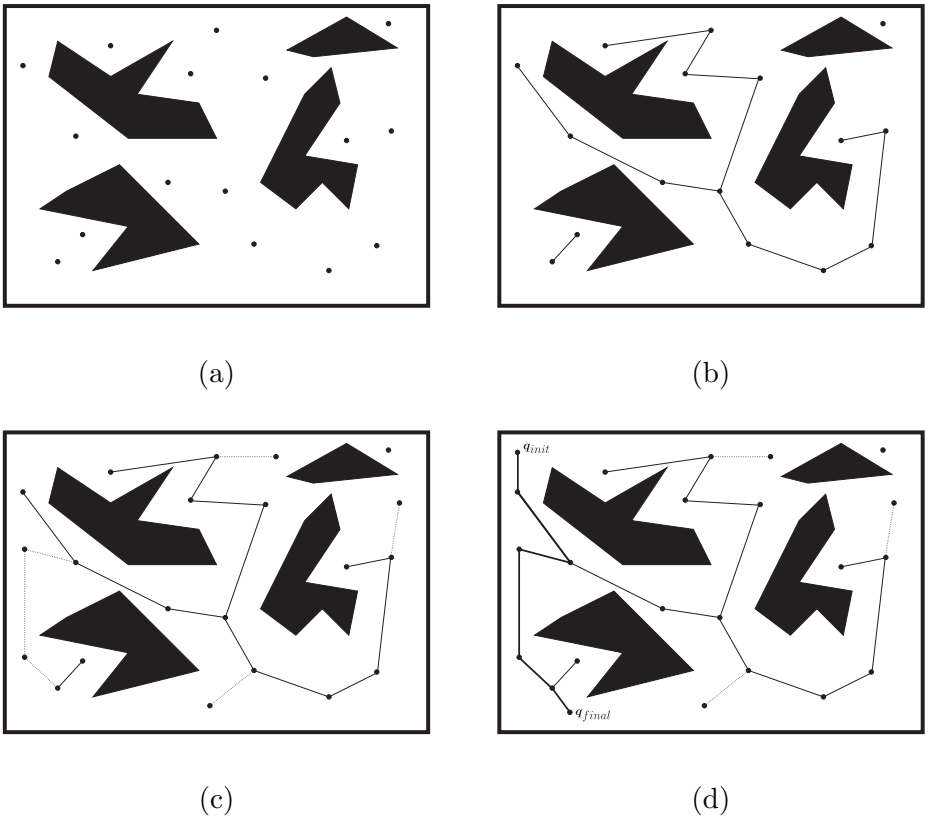


Figure 7.15: This figures illustrates the steps in the construction of a probabilistic roadmap for a two-dimensional configuration space containing polygonal obstacles. (a) First, a set of random samples is generated in the configuration space. Only collision-free samples are retained. (b) Each sample is connected to its nearest neighbors using a simple, straight-line path. If such a path causes a collision, the corresponding samples are not connected in the roadmap. (c) Since the initial roadmap contains multiple connected components, additional samples are generated and connected to the roadmap in an enhancement phase. (d) A path from q_{s} to q_{f} is found by connecting q_{s} and q_{f} to the roadmap and then searching this augmented roadmap for a path from q_{s} to q_{f}.

Sampling the Configuration Space

The simplest way to generate sample configurations is with uniform random sampling of the configuration space. Sample configurations that lie in $\mathcal{QO}$ are discarded. A simple collision checking algorithm can determine when this is the case. The disadvantage of this approach is that the number of samples it places in any particular region of $\mathcal{Q}_{\text{free}}$ is proportional to the volume of the region. Therefore, uniform sampling is unlikely to place samples in narrow passages of $\mathcal{Q}_{\text{free}}$. In the PRM literature, this is referred to as the **narrow passage problem**. It can be dealt with either by using more intelligent sampling schemes, or by using an enhancement phase during the construction of the PRM. In this section, we discuss the latter option.

Connecting Pairs of Configurations

Given a set of vertices that correspond to configurations, the next step in building the PRM is to determine which pairs of vertices should be connected by the local path planner. The typical approach is to attempt to connect each vertex to its k nearest neighbors, with k a parameter chosen by the user. Of course, to define the nearest neighbors, a distance function is required. Table 7.1 lists four distance functions that have been popular in the PRM literature. In this table, q and q' are the two configurations corresponding to different vertices in the roadmap, q_i refers to the value of the i^{th} coordinate of q, $\mathcal{A}$ is a set of reference points on the robot, and $\text{p}(q)$ refers to the workspace coordinates of reference point p at configuration q. Of these, the simplest, and perhaps most commonly used, is the 2-norm in configuration space.

Once pairs of neighboring vertices have been identified, a simple local planner is used to connect them. Often, a straight line in configuration space is used as the candidate plan, and thus, planning the path between two vertices is reduced to collision checking along a straight-line path in the configuration space. If a collision occurs on this path, it can be discarded, or a more sophisticated planner can be used to attempt to connect the vertices.

The simplest approach to collision detection along the straight-line path is to sample the path at a sufficiently fine discretization, and to check each sample for collision. This method works, provided the discretization is fine enough, but it is very inefficient. This is because many of the computations required to check for collision at one sample are repeated for the next sample (assuming that the robot has moved only a small amount between the two configurations). For this reason, incremental collision detection approaches have been developed. While these approaches are beyond the scope of this

2-norm in $\mathcal{Q}$:	$\|q' - q\| = \left[\sum_{i=1}^{n} (q'_i - q_i)^2\right]^{\frac{1}{2}}$
∞-norm in $\mathcal{Q}$:	$\max_n \|q'_i - q_i\|$
2-norm in workspace:	$\left[\sum_{\mathrm{p}\in\mathcal{A}} \|\mathrm{p}(q') - \mathrm{p}(q)\|^2\right]^{\frac{1}{2}}$
∞-norm in workspace:	$\max_{\mathrm{p}\in\mathcal{A}} \|\mathrm{p}(q') - \mathrm{p}(q)\|$

Table 7.1: Four commonly used distance functions.

text, a number of collision detection software packages are available in the public domain. Most developers of robot motion planners use one of these packages, rather than implementing their own collision detection routines.

Enhancement

After the initial PRM has been constructed, it is likely that it will consist of multiple connected components. Often these individual components lie in large regions of $\mathcal{Q}_{\text{free}}$ that are connected by narrow passages in $\mathcal{Q}_{\text{free}}$. The goal of the enhancement process is to connect as many of these disjoint components as possible.

One approach to enhancement is merely to attempt to connect pairs of vertices in two disjoint components, perhaps by using a more sophisticated planner such as described in Section 7.3. A common approach is to identify the largest connected component, and to attempt to connect the smaller components to it. The vertex in the smaller component that is closest to the larger component is typically chosen as the candidate for connection. A second approach is to choose a vertex randomly as a candidate for connection, and to bias the random choice based on the number of neighbors of the vertex; a vertex with fewer neighbors in the roadmap is more likely to be near a narrow passage, and should be a more likely candidate for connection.

Another approach to enhancement is to add more random vertices to the roadmap, in the hope of finding vertices that lie in or near the narrow passages. One approach is to identify vertices that have few neighbors, and

to generate sample configurations in regions around these vertices. The local planner is then used to attempt to connect these new configurations to the roadmap.

Path Smoothing

After the PRM has been generated, path planning amounts to connecting q_{s} and q_{f} to the roadmap using the local planner, and then performing path smoothing, since the resulting path will be composed of straight-line segments in the configuration space. The simplest path smoothing algorithm is to select two random points on the path and try to connect them with the local planner. This process is repeated until no significant progress is made.

7.4.2 Rapidly-Exploring Random Trees (RRTs)

In the classical PRM algorithm, the i^{th} sample configuration is obtained by sampling from a uniform probability distribution on $\mathcal{Q}$, independent of any previously generated samples[7]. As a result, it is possible, and even likely, that at any given iteration the newly generated sample configuration may be quite far from any existing vertices in the current roadmap. In such cases, the local planner is likely to fail to connect the new sample to any existing vertex, increasing the number of connected components in the roadmap. An alternative approach is to grow a single tree, starting from a vertex that corresponds to q_{s}, until some leaf vertex reaches the goal configuration. This is the approach embodied by **rapidly-exploring random trees** (**RRTs**).

The construction of an RRT is an iterative process in which a new vertex is added to an existing tree at each iteration. The process of adding a new vertex begins by generating a random sample, q_{sample}, from a uniform probability distribution on $\mathcal{Q}$; however, unlike the construction of PRMs, this vertex is not added to the existing tree. Instead, q_{sample} is used to determine how to "grow" the current tree. This is done by choosing the vertex q_{near} in the existing tree that is closest to q_{sample}, and taking a small step from q_{near} in the direction of q_{sample}. The configuration at which this step arrives, q_{new}, is added to the tree, and an edge is added from q_{near} to q_{new}. Figure 7.16 illustrates the process for a single iteration of the RRT construction algorithm.

While RRT-based algorithms have proven to be very effective in planning collision-free paths for robots, their true power lies in the method by which the new node q_{new} is generated and connected to the existing tree. For

[7] In the language of probability theory, the sample configurations correspond to independent random variables.

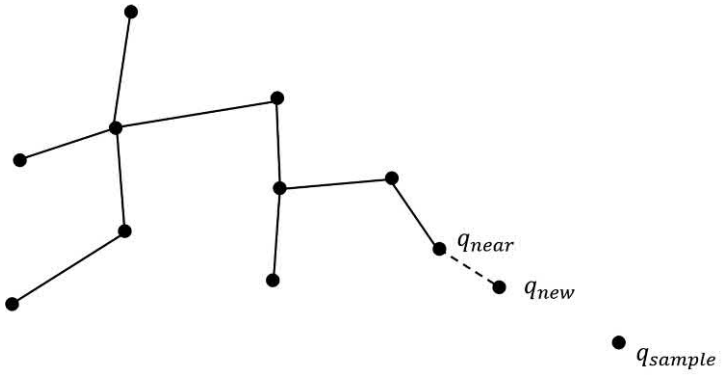


Figure 7.16: A new vertex is added to an existing tree by (i) generating a sample configuration generate q_{sample} from a uniform probability distribution on $\mathcal{Q}$, (ii) identifying q_{near}, the vertex in the current tree that is nearest to q_{sample}, and (iii) taking a small step from q_{near} toward q_{sample}.

PRMs, a new node is connected to the tree by solving a local path planning problem. For RRTs, the configuration q_{new} is chosen by "taking a step" toward q_{sample}. The simplest way to achieve this, of course, is to step along a straight-line path toward q_{sample}, but more general methods can be applied. Consider, for example, a robot whose system dynamics are described by

$$\dot{x} = f(x, u) \tag{7.7}$$

in which x denotes the state of the system and u denotes an input. If we define our RRT on the state space instead of the configuration space, we can generate the vertex x_{new} by integrating the dynamics (7.7) forward in time from the initial condition given by x_{near}

$$x_{\text{new}} = x_{\text{near}} + \int_0^{\Delta t} f(x, u)dt \tag{7.8}$$

Evaluating this integral requires knowing an appropriate control input u. Choosing u to ensure that x_{new} lies along the line connecting x_{near} to x_{sample} is a difficult problem, but luckily, ensuring that x_{new} lies exactly on this line is not essential to the efficacy of RRT algorithms. A popular way to determine x_{new} is to randomly select a set of candidate inputs, u^i, evaluate (7.8) for each of these, and retain the result that makes the best progress toward x_{sample}. Using this approach, RRTs have been applied to motion planning problems for car-like robots, unmanned air vehicles, autonomous underwater vehicles, spacecraft, satellites, and many others.

7.5 Trajectory Planning

In Section 7.1.3, we defined a path[8] from q_0 to q_f in configuration space as a continuous map, $\gamma : [0,1] \to \mathcal{Q}$, with $\gamma(0) = q_0$ and $\gamma(1) = q_\mathrm{f}$. A **trajectory** is a function of time $q(t)$ such that $q(t_0) = q_0$ and $q(t_f) = q_\mathrm{f}$. In this case, $t_f - t_0$ represents the amount of time taken to execute the trajectory. Since the trajectory is parameterized by time, we can compute velocities and accelerations along the trajectories by differentiation. If we think of the argument to γ as a time variable, then a path is a special case of a trajectory, one that will be executed in one unit of time. In other words, in this case γ gives a complete specification of the robot's trajectory, including the time derivatives (since one need only differentiate γ to obtain these).

As seen above, a path planning algorithm will not typically give the map γ; it will give only a sequence of points (called **via points**) along the path. This is also the case for other ways in which a path could be specified. In some cases, paths are specified by giving a sequence of end-effector poses, $T_6^0(k\Delta t)$. In this case, the inverse kinematic solution must be used to convert this to a sequence of joint configurations. A common way to specify paths for industrial robots is to physically lead the robot through the desired motion with a teach pendant, the so-called **teach and playback mode**. In some cases, this may be more efficient than deploying a path planning system, for example, in static environments when the same path will be executed many times. In this case, there is no need for calculation of the inverse kinematics; the desired motion is simply recorded as a set of joint angles (actually as a set of encoder values).

Below, we first consider **point-to-point** motion. In this case the task is to plan a trajectory from an initial configuration $q(t_0)$ to a final configuration $q(t_f)$. In some cases, there may be constraints on the trajectory (for example, if the robot must start and end with zero velocity). Nevertheless, it is easy to realize that there are infinitely many trajectories that will satisfy a finite number of constraints on the endpoints. It is common practice therefore to choose trajectories from a finitely parameterizable family, for example, polynomials of degree n, where n depends on the number of constraints to be satisfied. This is the approach that we will take in this text. Once we have seen how to construct trajectories between two configurations, it is straightforward to generalize the method to the case of trajectories specified by multiple via points.

[8] In Section 7.1.3 we used q_s to denote the initial configuration. In this section, we use q_0 to denote the first configuration in a sequence of two or more configurations that will be used to define a path.

7.5.1 Trajectories for Point-to-Point Motion

As described above, the problem is to find a trajectory that connects the initial and final configurations while satisfying other specified constraints at the endpoints, such as velocity and/or acceleration constraints. Without loss of generality, we will consider planning the trajectory for a single joint, since the trajectories for the remaining joints will be created independently and in exactly the same way. Thus, we will concern ourselves with the problem of determining $q(t)$, where $q(t)$ is a scalar joint variable.

We suppose that at time t_0 the joint variable satisfies

$$q(t_0) = q_0 \tag{7.9}$$
$$\dot{q}(t_0) = v_0 \tag{7.10}$$

and we wish to attain the values at t_f

$$q(t_f) = q_f \tag{7.11}$$
$$\dot{q}(t_f) = v_f \tag{7.12}$$

Figure 7.17 shows a suitable trajectory for this motion. In addition, we may wish to specify the constraints on initial and final accelerations. In this case we have two additional equations

$$\ddot{q}(t_0) = \alpha_0 \tag{7.13}$$
$$\ddot{q}(t_f) = \alpha_f \tag{7.14}$$

Below we will investigate several specific ways to compute trajectories using low-order polynomials. We begin with cubic polynomials, which allow specification of initial and final positions and velocities. We then describe quintic polynomial trajectories, which also allow the specification of the initial and final accelerations. After describing these two general polynomial trajectories, we describe trajectories that are pieced together from segments of constant acceleration.

Cubic Polynomial Trajectories

Consider first the case where we wish to generate a polynomial joint trajectory between two configurations, and that we wish to specify the start and end velocities for the trajectory. This gives four constraints that the trajectory must satisfy. Therefore, at a minimum we require a polynomial with four independent coefficients that can be chosen to satisfy these constraints. Thus, we consider a cubic trajectory of the form

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{7.15}$$

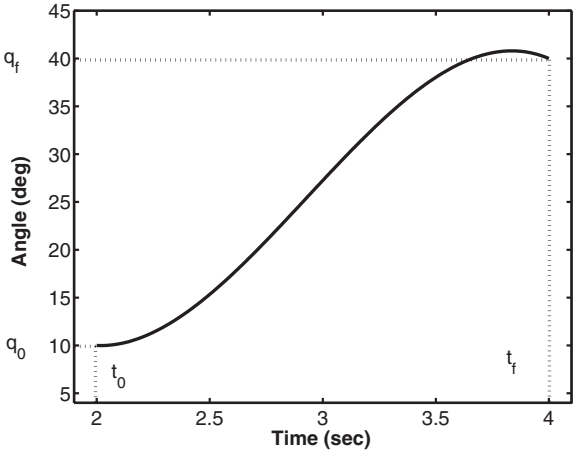


Figure 7.17: A typical joint space trajectory.

Then the desired velocity is given as

$$\dot{q}(t) = a_1 + 2a_2t + 3a_3t^2 \tag{7.16}$$

Combining Equations (7.15) and (7.16) with the four constraints yields four equations in four unknowns

$$\begin{aligned} q_0 &= a_0 + a_1t_0 + a_2t_0^2 + a_3t_0^3 \\ v_0 &= a_1 + 2a_2t_0 + 3a_3t_0^2 \\ q_f &= a_0 + a_1t_f + a_2t_f^2 + a_3t_f^3 \\ v_f &= a_1 + 2a_2t_f + 3a_3t_f^2 \end{aligned}$$

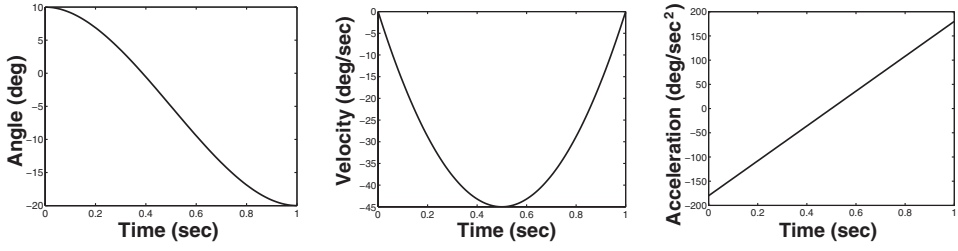
These four equations can be combined into a single matrix equation

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \tag{7.17}$$

It can be shown (Problem 7–19) that the determinant of the coefficient matrix in Equation (7.17) is equal to $(t_f - t_0)^4$ and, hence, Equation (7.17) always has a unique solution provided a nonzero time interval is allowed for the execution of the trajectory.

As an illustrative example, we may consider the special case that the initial and final velocities are zero. Suppose we take $t_0 = 0$ *and* $t_f = 1$ *sec,*

Example 7.8 (Cubic Polynomial Trajectory)**.**



(a) (b) (c)

Figure 7.18: (a) Cubic polynomial trajectory. (b) Velocity profile for cubic polynomial trajectory. (c) Acceleration profile for cubic polynomial trajectory.

with

$$v_0 = 0 \qquad v_f = 0$$

Thus, we want to move from the initial position q_0 to the final position q_f in 1 second, starting and ending with zero velocity. From Equation (7.17) we obtain

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ 0 \\ q_f \\ 0 \end{bmatrix}$$

This is then equivalent to the four equations

$$\begin{aligned} a_0 &= q_0 \\ a_1 &= 0 \\ a_2 + a_3 &= q_f - q_0 \\ 2a_2 + 3a_3 &= 0 \end{aligned}$$

These latter two equations can be solved to yield

$$\begin{aligned} a_2 &= 3(q_f - q_0) \\ a_3 &= -2(q_f - q_0) \end{aligned}$$

The required cubic polynomial function is therefore

$$q(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3$$

The corresponding velocity and acceleration curves are given as

$$\begin{aligned}\dot{q}(t) &= 6(q_f - q_0)t - 6(q_f - q_0)t^2 \\ \ddot{q}(t) &= 6(q_f - q_0) - 12(q_f - q_0)t\end{aligned}$$

Figure 7.18 shows these trajectories with $q_0 = 10°$, $q_f = -20°$.

Quintic Polynomial Trajectories

As can be seen in Figure 7.18, a cubic trajectory gives continuous positions and velocities at the start and finish points times but discontinuities in the acceleration. The derivative of acceleration is called the **jerk**. A discontinuity in acceleration leads to an impulsive jerk, which may excite vibrational modes in the manipulator and reduce tracking accuracy. For this reason, one may wish to specify constraints on the acceleration as well as on the position and velocity. In this case, we have six constraints (one each for initial and final configurations, initial and final velocities, and initial and final accelerations). Therefore we require a fifth order polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{7.18}$$

Using Equations (7.9)–(7.14) and taking the appropriate number of derivatives we obtain the following equations,

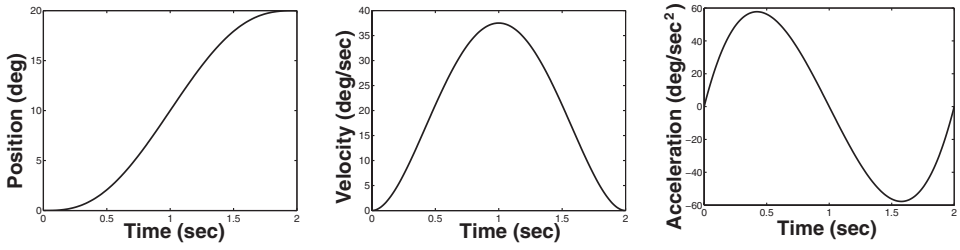
$$\begin{aligned}q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \\ v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \\ \alpha_0 &= 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \\ q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\ v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\ \alpha_f &= 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3\end{aligned}$$

which can be written as

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix} \tag{7.19}$$

Figure 7.19 shows a quintic polynomial trajectory with $q(0) = 0$, $q(2) = 20$ *with zero initial and final velocities and accelerations.*

Example 7.9 (Quintic Polynomial Trajectory)**.**



(a) (b) (c)

Figure 7.19: (a) Quintic polynomial trajectory, (b) its velocity profile, and (c) its acceleration profile.

Linear Segments with Parabolic Blends (LSPB)

Another way to generate suitable joint space trajectories is by using so-called **linear segments with parabolic blends** (**LSPB**). This type of trajectory has a **trapezoidal velocity profile** and is appropriate when a constant velocity is desired along a portion of the path. The LSPB trajectory is such that the velocity is initially "ramped up" to its desired value and then "ramped down" when it approaches the goal position. To achieve this we specify the desired trajectory in three parts. The first part from time t_0 to time t_b is a quadratic polynomial. This results in a linear "ramp" velocity. At time t_b, called the **blend time**, the trajectory switches to a linear function. This corresponds to a constant velocity. Finally, at $t_f - t_b$ the trajectory switches once again, this time to a quadratic polynomial so that the velocity is linear.

We choose the blend time t_b so that the position curve is symmetric as shown in Figure 7.20. For convenience suppose that $t_0 = 0$ and $\dot{q}(t_f) = 0 = \dot{q}(0)$. Then between times 0 and t_b we have

$$q(t) = a_0 + a_1 t + a_2 t^2$$

so that the velocity is

$$\dot{q}(t) \quad = \quad a_1 + 2a_2 t$$

The constraints $q_0 = 0$ and $\dot{q}(0) = 0$ imply that

$$\begin{aligned} a_0 &= q_0 \\ a_1 &= 0 \end{aligned}$$

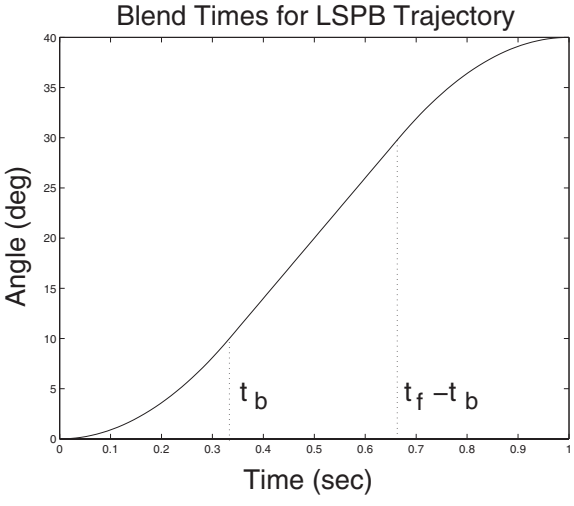


Figure 7.20: Blend times for LSPB trajectory.

At time t_b we want the velocity to equal a given constant, say V. Thus, we have

$$\dot{q}(t_b) = 2a_2 t_b = V$$

which implies that

$$a_2 = \frac{V}{2t_b}$$

Therefore the required trajectory between 0 and t_b is given as

$$\begin{aligned} q(t) &= q_0 + \frac{V}{2t_b}t^2 = q_0 + \frac{\alpha}{2}t^2 \\ \dot{q}(t) &= \frac{V}{t_b}t = \alpha t \\ \ddot{q} &= \frac{V}{t_b} = \alpha \end{aligned}$$

where α denotes the acceleration.

Now, between time t_b and $t_f - t_b$, the trajectory is a linear segment with velocity V

$$q(t) = q(t_b) + V(t - t_b)$$

Since, by symmetry,

$$q\left(\frac{t_f}{2}\right) = \frac{q_0 + q_f}{2}$$

we have

$$\frac{q_0 + q_f}{2} = q(t_b) + V(\frac{t_f}{2} - t_b)$$

which implies that

$$q(t_b) = \frac{q_0 + q_f}{2} - V(\frac{t_f}{2} - t_b)$$

Since the two segments must "blend" at time t_b, we require

$$q_0 + \frac{V}{2}t_b = \frac{q_0 + q_f - Vt_f}{2} + Vt_b$$

which, upon solving for the blend time t_b, gives

$$t_b = \frac{q_0 - q_f + Vt_f}{V} \tag{7.20}$$

Note that we have the constraint $0 < t_b \leq \frac{t_f}{2}$. This leads to the inequality

$$\frac{q_f - q_0}{V} < t_f \leq \frac{2(q_f - q_0)}{V}$$

To put it another way we have the inequality

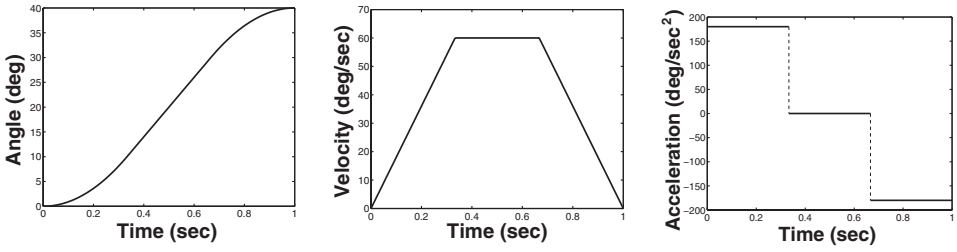
$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}$$

Thus, the specified velocity must be between these limits or the motion is not possible.

The portion of the trajectory between $t_f - t_b$ and t_f is now found by symmetry considerations (Problem 7–23). The complete LSPB trajectory is given by

$$q(t) = \begin{cases} q_0 + \frac{\alpha}{2}t^2 & 0 \leq t \leq t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \leq t_f - t_b \\ q_f - \frac{\alpha t_f^2}{2} + \alpha t_f t - \frac{\alpha}{2}t^2 & t_f - t_b < t \leq t_f \end{cases} \tag{7.21}$$

Figure 7.21(a) shows such an LSPB trajectory, where the maximum velocity $V = 60$. In this case $t_b = \frac{1}{3}$. The velocity and acceleration curves are given in Figures 7.21(b) and 7.21(c), respectively.



(a) (b) (c)

Figure 7.21: (a) LSPB trajectory. (b) Velocity profile for LSPB trajectory. (c) Acceleration profile for LSPB trajectory.

Minimum-Time Trajectories

An important variation of the LSPB trajectory is obtained by leaving the final time t_f unspecified and seeking the "fastest" trajectory between q_0 and q_f with a given constant acceleration α, that is, the trajectory with the final time t_f a minimum. This is sometimes called a **bang-bang** trajectory since the optimal solution is achieved with the acceleration at its maximum value $+\alpha$ until an appropriate **switching time** t_s at which time it abruptly switches to its minimum value $-\alpha$ (maximum deceleration) from t_s to t_f.

Returning to our simple example in which we assume that the trajectory begins and ends at rest, that is, with zero initial and final velocities, symmetry considerations would suggest that the switching time t_s is just $\frac{t_f}{2}$. This is indeed the case. For nonzero initial and/or final velocities, the situation is more complicated and we will not discuss it here. If we let V_s denote the velocity at time t_s then we have $V_s = \alpha t_s$ and using Equation (7.20) with $t_b = t_s$ we obtain

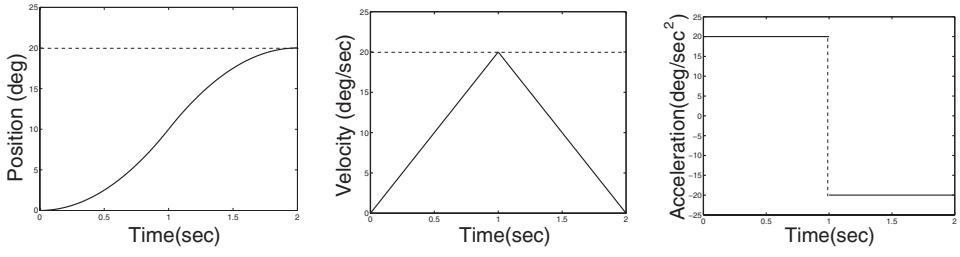
$$t_s = \frac{q_0 - q_f + V_s t_f}{V_s}$$

The symmetry condition $t_s = \frac{t_f}{2}$ implies that

$$V_s = \frac{q_f - q_0}{t_s}$$

and using the fact that $V_s = \alpha t_s$, we obtain

$$\frac{q_f - q_0}{t_s} = \alpha t_s$$



(a) (b) (c)

Figure 7.22: (a) Minimum-time trajectory. (b) Velocity profile for minimum-time trajectory. (c) Acceleration profile for minimum-time trajectory.

which implies that

$$t_s = \sqrt{\frac{q_f - q_0}{\alpha}}$$

Figure 7.22 shows the position, velocity, and acceleration for such a minimum-time trajectory.

7.5.2 Trajectories for Paths Specified by Via Points

Now that we have examined the problem of planning a trajectory between two configurations, we generalize our approach to the case of planning a trajectory that passes through a sequence of configurations, called **via points**. Consider the simple example of a path specified by three points, q_0, q_1, and q_2, such that the via points are reached at times t_0, t_1, and t_2, respectively. If in addition to these three constraints we impose constraints on the initial and final velocities and accelerations, we obtain the following set of constraints,

$$\begin{aligned} q(t_0) &= q_0 \ , \ \dot{q}(t_0) = v_0 \ , \ \ddot{q}(t_0) = \alpha_0 \\ q(t_1) &= q_1 \ , \ q(t_2) = q_2 \ , \ \dot{q}(t_2) = v_2 \ , \ \ddot{q}(t_2) = \alpha_2 \end{aligned}$$

which could be satisfied by generating a trajectory using the sixth-order polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 \qquad (7.22)$$

One advantage to this approach is that, since $q(t)$ is continuously differentiable, we need not worry about discontinuities in either velocity or

acceleration at the via point, q_1. However, to determine the coefficients for this polynomial, we must solve a linear system of dimension seven. The clear disadvantage to this approach is that as the number of via points increases, the dimension of the corresponding linear system also increases, making the method intractable when many via points are used.

An alternative to using a single high-order polynomial for the entire trajectory is to use low-order polynomials for trajectory segments between adjacent via points. These polynomials are sometimes referred to as interpolating polynomials or blending polynomials. With this approach, we must take care that velocity and acceleration constraints are satisfied at the via points, where we switch from one polynomial to another.

For the first segment of the trajectory, suppose that the initial and final times are t_0 and t_f, respectively, and the constraints on initial and final velocities are given by

$$\begin{array}{lll} q(t_0) = q_0 & ; & q(t_f) = q_1 \\ \dot{q}(t_0) = v_0 & ; & \dot{q}(t_f) = v_1 \end{array} \tag{7.23}$$

the required cubic polynomial for this segment of the trajectory can be computed from

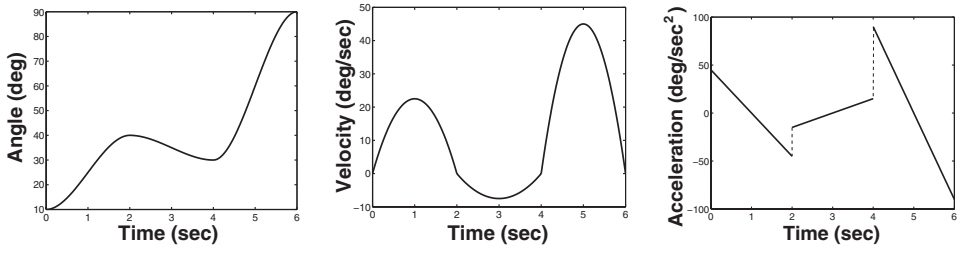
$$q(t_0) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \tag{7.24}$$

where

$$\begin{aligned} a_0 &= q_0 \\ a_1 &= v_0 \\ a_2 &= \frac{3(q_1 - q_0) - (2v_0 + v_1)(t_f - t_0)}{(t_f - t_0)^2} \\ a_3 &= \frac{2(q_0 - q_1) + (v_0 + v_1)(t_f - t_0)}{(t_f - t_0)^3} \end{aligned}$$

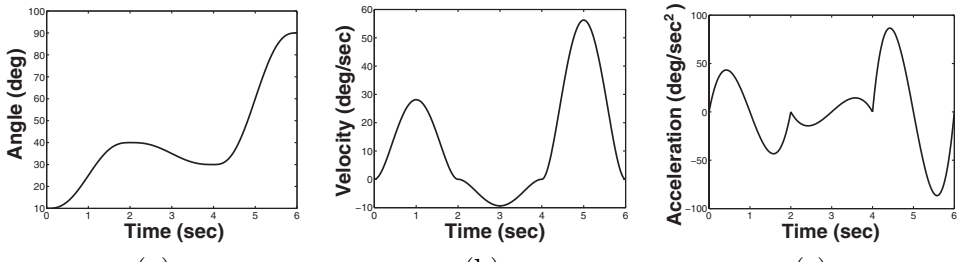
A sequence of moves can be planned using the above formula by using the end conditions q_f, v_f of the i^{th} move as initial conditions for the subsequent move.

Figure 7.23 shows a 6-second move, computed in three parts using Equation (7.24), where the trajectory begins at 10° and is required to reach 40° at 2 seconds, 30° at 4 seconds, and 90° at 6 seconds, with zero velocity at 0, 2, 4, and 6 seconds.



(a) (b) (c)

Figure 7.23: (a) Cubic spline trajectory made from three cubic polynomials. (b) Velocity profile for multiple cubic polynomial trajectory. (c) Acceleration profile for multiple cubic polynomial trajectory.



(a) (b) (c)

Figure 7.24: (a) Trajectory with multiple quintic segments. (b) Velocity profile for multiple quintic segments. (c) Acceleration profile for multiple quintic segments.

Figure 7.24 shows the same 6-second trajectory as above with the added constraints that the accelerations should be zero at the blend times.

7.6 Chapter Summary

In this chapter we studied methods for generating collision-free trajectories for robot manipulators. We divided the problem into two parts, first computing a collision-free path then by using interpolating polynomials to convert these paths into continuous trajectories.

We began by giving a more in-depth introduction to the concept of **configuration space**, including a catalog of configuration spaces for common robots and an explanation of how obstacles in the workspace map to configuration space obstacle regions. We described an algorithm to compute the configuration space obstacle region for the special case of a polygonal robot

that translates in the plane (i.e., $\mathcal{Q} = \mathbb{R}^2$) in an environment populated by polygonal obstacles. For this special case, we introduced three planning algorithms that exploit three unique graph-based representations of the free configuration space: the visibility graph, the generalized Voronoi diagram, and the trapezoidal decomposition. These approaches are often applicable for problems that involve ground-based mobile robots.

For more complex robots, for example, robots that have high dimensional configuration spaces, we introduced algorithms that incrementally search the configuration space for a free path. The first such algorithm operates by constructing an artificial potential field whose negative gradient acts as a kind of artificial force that pushes the robot away from obstacles, while guiding it toward the goal configuration. We first developed this method for the case of $\mathcal{Q} = \mathbb{R}^n$, and then extended it to the more general case of non-Euclidean configuration spaces by using the relationship $\tau = J^T F$. In both cases, gradient descent was used as the method to guide the incremental exploration of the configuration space, and thus, for both cases, there are potential problems arising from the existence of multiple local minima in the field. We briefly described how random walks could be used to escape these local minima, and used this notion of randomization to motivate the development of sampling-based planning algorithms.

Sampling-based planners construct a roadmap in the configuration space using a random sampling scheme. We described two approaches: the probabilistic roadmap (PRM), and the rapidly-exploring random tree (RRT). For the PRM, a set of random samples is generated, and each of these samples is connected to a set of its nearest neighbors using a simple local motion planner (often a simple straight-line planner in the configuration space). Once the roadmap has been constructed, planning amounts to connecting the initial and goal configurations to the roadmap (again, using the simple, local planner), then searching the roadmap for a connecting path. In the case of RRTs, a random tree is grown from the start configuration by iteratively generating a random sample in the configuration space, then taking a small step toward this sample from its nearest neighbor in the current tree. Both of these methods have proven effective for a large variety of path planning problems.

Finally, we showed that, given a sequence of set points, a trajectory can be constructed using a low-order polynomial defined in terms of initial and final conditions for joint variables and their derivatives. We described cubic and quintic trajectories, along with trajectories that are pieced together from segments of constant acceleration (including minimum time, or bang-bang trajectories).

Problems

7–1 Describe the configuration space for a mobile robot that can translate and rotate in the plane.

7–2 Describe the configuration space for the three-link manipulator shown in Figure 3.12.

7–3 Describe the configuration space for the two-link manipulator shown in Figure 3.13.

7–4 Describe the configuration space for the two-link manipulator shown in Figure 3.14.

7–5 Describe the configuration space for the three-link manipulator shown in Figure 3.15.

7–6 Describe the configuration space for a six-link anthropomorphic arm equipped with a spherical wrist.

7–7 Show that the visibility graph includes all shortest semi-free paths from q_s to q_f. Note, this is equivalent to showing that a necessary condition for a path to be a shortest semi-free path is that it be included in the visibility graph.

7–8 Suppose q_s lies in the Voronoi region for a particular feature, f. Show that a straight-line path from q_s that follows the gradient $\nabla d(q_s, f)$ will arrive to a Voronoi edge before reaching any other feature f'.

7–9 Verify Equation (7.1).

7–10 Derive the equations needed to compute the shortest distance from a point p to the line segment in the plane with vertices a_1 and a_2.

7–11 Derive the equations needed to compute the shortest distance from a point p to the polygon in the plane with vertices a_i, $i = 1, \ldots, n$.

7–12 Derive the equations needed to compute the shortest distance from a point p to the polygon in three dimensions with vertices a_i, $i = 1, \ldots, n$.

7–13 Verify Equation (7.5).

7–14 Consider a simple polygonal robot with four vertices, such that at $q = (0, 0, 0)$ the vertices are located at $a_1(0) = (0, 0)$, $a_2(0) = (1, 0)$,

$a_3(0) = (1, 1)$, and $a_4(0) = (0, 1)$. If two point obstacles are located at $o_1 = (3, 3)$ and $o_2 = (-3, -3)$, determine the artificial workspace and configuration space forces that act on the robot.

7–15 Write a computer program to implement the path planner described in Section 7.3.2 for a three-link planar arm moving among polygonal obstacles.

7–16 Write a simple computer program to perform collision checking for the case of a polygonal robot moving in the plane among polygonal obstacles. Your program should accept a configuration q as input, and should return a value that indicates whether q is a collision-free configuration.

7–17 Give a procedure for generating random samples of orientations in $SO(n)$ given that you have access to a random number generator that can generate samples from the uniform distribution on the unit interval. Your samples need not be uniformly distributed on $SO(n)$.

7–18 Write a computer program to implement the PRM planner described in Section 7.4.1 for a three-link planar arm moving among polygonal obstacles.

7–19 Show by direct calculation that the determinant of the coefficient matrix in Equation (7.17) is $(t_f - t_0)^4$.

7–20 Suppose we wish a manipulator to start from an initial configuration at time t_0 and track a conveyor. Discuss the steps needed in planning a suitable trajectory for this problem.

7–21 Suppose we desire a joint space trajectory $\dot{q}_i^d(t)$ for the i^{th} joint (assumed to be revolute) that begins at rest at position q_0 at time t_0 and reaches position q_1 in 2 seconds with a final velocity of 1 radian/sec. Compute a cubic polynomial satisfying these constraints. Sketch the trajectory as a function of time.

7–22 Compute a LSPB trajectory to satisfy the same requirements as in Problem 7–21. Sketch the resulting position, velocity, and acceleration profiles.

7–23 Fill in the details of the computation of the LSPB trajectory. In other words, compute the portion of the trajectory between times $t_f - t_b$ and t_f and verify Equations (7.21).

7–24 Write a Matlab m-file, lspb.m, to generate an LSPB trajectory, given appropriate initial data.

7–25 Rewrite the Matlab m-files, cubic.m, quintic.m, and lspb.m to turn them into Matlab functions. Document them appropriately.

Notes and References

The earliest work on robot planning was done in the late sixties and early seventies in a few university-based Artificial Intelligence (AI) labs [40], [45], and [129]. This research dealt with high level planning using symbolic reasoning that was much in vogue at the time in the AI community. Geometry was not often explicitly considered in early robot planners, in part because it was not clear how to represent geometric constraints in a computationally feasible manner. The configuration space and its application to path planning were introduced in [97]. This was the first rigorous, formal treatment of the geometric path planning problem, and it initiated a surge in path planning research.

The earliest work in geometric path planning developed methods to construct volumetric representations of the free configuration space. These included exact methods [147], and approximate methods [20], [73], and [97]. In the former case, the best known algorithms have exponential complexity and require exact descriptions of both the robot and its environment, while in the latter case, the size of the representation of configuration space grows exponentially in the dimension of the configuration space. The best known algorithm for the path planning problem, giving an upper bound on the amount of computation time required to solve the problem, appeared in [21]. That real robots rarely have an exact description of the environment, and a drive for faster planning systems led to the development of potential fields approaches [77], and [79].

By the early nineties, a great deal of research had been done on the geometric path planning problem, and this work is nicely summarized in the textbook [87]. This textbook helped to generate a renewed interest in the path planning problem, and it provided a common framework in which to analyze and express path planning algorithms.

In the early nineties, randomization was introduced in the robot planning community [10], originally to circumvent the problems with local minima in potential fields. Early randomized motion planners proved effective for a large range of problems, but sometimes required extensive computation time for some robots in certain environments [75]. This limitation, together

with the idea that a robot will operate in the same environment for a long period of time led to the development of the probabilistic roadmap planners [74, 132, 75]. Rapidly-exploring random trees were introduced in [90, 91].

Comprehensive reviews of motion planning research, including sensor-based approaches, can be found in [24, 88, 89].

Much work has been done in the area of collision detection in recent years [96], [115], [177], and [178]. This work is primarily focused on finding efficient, incremental methods for detecting collisions between objects when one or both are moving. A number of public domain collision detection software packages are currently available on the Internet.

Part III

CONTROL OF MANIPULATORS

Chapter 8

INDEPENDENT JOINT CONTROL

8.1 Introduction

The control problem for robot manipulators is to determine the time history of joint inputs required to cause the end effector to execute a commanded motion. The joint inputs may be joint forces and torques, or they may be inputs to the actuators, for example, voltage or current inputs to the motors, depending on the model used for controller design. The commanded motion is typically specified either as a sequence of end-effector positions and orientations, or as a continuous path.

There are many control techniques and methodologies that can be applied to the control of manipulators. The particular control method used can have a significant impact on the performance of the manipulator and consequently on the range of its possible applications. For example, continuous path tracking requires a different control architecture than does point-to-point motion.

In addition, the mechanical design of the manipulator itself will influence the type of control scheme needed. For example, the control problems encountered with a Cartesian manipulator are different from those encountered with an elbow manipulator. This creates a so-called **hardware/ software trade-off** between the mechanical structure of the system and the architecture/programming of the controller.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

Technological improvements are continually being made in the mechanical design of robots, which in turn improves their performance potential and broadens their range of applications. Realizing this increased performance, however, requires more sophisticated approaches to control. One can draw an analogy to the aerospace industry. Early aircraft were relatively easy to fly but possessed limited performance capabilities. As performance increased with technological advances, so did the problems of control to the extent that the latest vehicles, such as the space shuttle or forward swept wing fighter aircraft, cannot be flown without sophisticated computer control.

As an illustration of the effect of the mechanical design on the control problem, we may compare a robot actuated by permanent magnet DC motors with gear reduction to a direct-drive robot using high-torque motors with no gear reduction. In the first case, the motor dynamics are linear and well understood and the effect of the gear reduction is largely to decouple the system by reducing the inertia coupling among the joints. However, the presence of the gears introduces friction, drive train compliance, and backlash.

In the case of a direct-drive robot, the problems of backlash, friction, and compliance due to the gears are eliminated. However, the inertia coupling among the links is now significant, and the dynamics of the motors themselves may be much more complex. The result is that in order to achieve high performance from this type of manipulator, a different set of control problems must be addressed.

In this chapter we consider the simplest type of control strategy, namely, independent joint control. In this type of control each axis of the manipulator is controlled as a single-input/single-output (SISO) system. Any coupling effects due to the motion of the other links are treated as disturbances.

The basic structure of a single-input/single-output feedback control system is shown in Figure 8.1.

The design objective is to choose the compensator in such a way that the plant output follows or **tracks** a desired output, given by the reference signal. The control signal, however, is not the only input acting on the system. Disturbances, which are really inputs that we do not control, also influence the behavior of the output. Therefore, the controller must be designed, in addition, so that the effects of the disturbances on the plant output are reduced. If this is accomplished, the plant is said to **reject** the disturbances. The twin objectives of **tracking** and **disturbance rejection** are central to any control methodology.

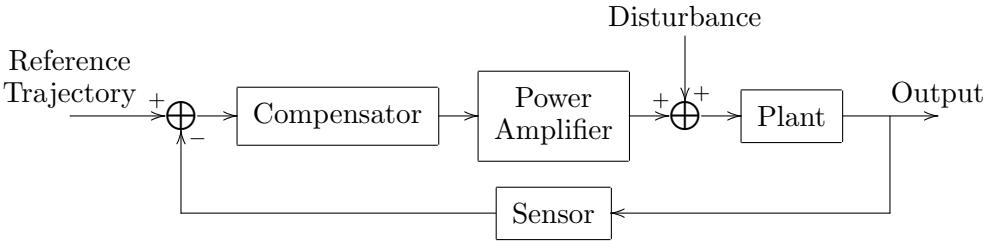


Figure 8.1: Basic structure of a feedback control system. The compensator measures the **error** between a **reference** and a measured **output** and produces signals to the plant that are designed to drive the error to zero despite the presence of disturbances.

8.2 Actuator Dynamics

Robot manipulators are equipped with actuators to move the joints through programmed motion trajectories in order to complete given tasks. These actuators may be electric, hydraulic, or pneumatic. In this section we restrict our attention to the dynamics of **permanent magnet DC motors**, as these are the simplest actuators to analyze and are commonly used in robot manipulators. Other types of electric motors, in particular **AC motors** and so-called **brushless DC motors**, are also used as actuators for robots but we will not discuss their dynamics here.

A DC motor works on the principle that a current-carrying conductor in a magnetic field experiences a force $F = i \times \phi$, where ϕ is the magnetic flux, i is the current in the conductor, and $\times$ is the vector cross product. The motor itself consists of a fixed **stator** and a movable **rotor** that rotates inside the stator as shown in Figure 8.2.

If the stator produces a radial magnetic flux ϕ and the current in the rotor (also called the **armature**) is i, then there will be a torque on the rotor causing it to rotate. The magnitude of this torque is

$$\tau_m = K_1 \phi i_a \tag{8.1}$$

where τ_m is the motor torque (Newton-meters), ϕ is the magnetic flux (webers), i_a is the armature current (amperes), and K_1 is a physical constant.

In addition, whenever a conductor moves in a magnetic field, a voltage V_b is generated across its terminals that is proportional to the velocity of the conductor in the field. This voltage, called the **back emf**, will tend to oppose the current flow in the conductor. Thus, in addition to the torque

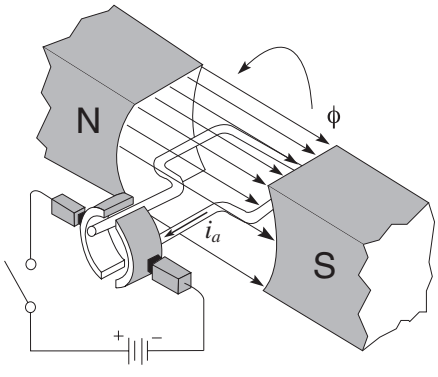


Figure 8.2: Principle of operation of a permanent magnet DC motor. The magnitude of the force (or torque) on the armature is proportional to the product of the current and magnetic flux. A **commutator** is required to periodically switch the direction of the current through the armature to keep it rotating in the same direction.

τ_m in Equation (8.1), we have the back emf relation

$$V_b = K_2 \phi \omega_m \tag{8.2}$$

where V_b denotes the back emf (volts), ω_m is the angular velocity of the rotor (radians per second), and K_2 is a proportionality constant.

DC motors can be classified according to the way in which the magnetic field is produced and the armature is designed. Here we discuss only the so-called **permanent magnet** motors whose stator consists of a permanent magnet. In this case we can take the flux ϕ to be a constant. The torque on the rotor is then controlled by controlling the armature current i_a.

Consider the schematic diagram of Figure 8.3, where

$$\begin{aligned}
V &= \text{armature voltage} \\
L &= \text{armature inductance} \\
R &= \text{armature resistance} \\
V_b &= \text{back emf} \\
i_a &= \text{armature current} \\
\theta_m &= \text{rotor position} \\
\tau_m &= \text{generated torque} \\
\tau_\ell &= \text{load torque} \\
\phi &= \text{magnetic flux due to the stator}
\end{aligned}$$

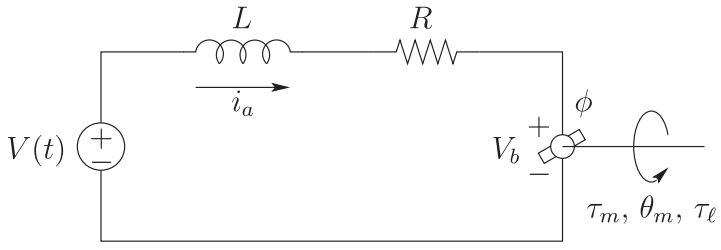


Figure 8.3: Circuit diagram for an armature controlled DC motor. The rotor windings have an effective inductance L and effective resistance R. The applied voltage V is the control input.

The differential equation for the armature current is then

$$L\frac{di_a}{dt} + Ri_a \quad = \quad V - V_b \tag{8.3}$$

Since the flux ϕ is constant, the torque developed by the motor is

$$\tau_m \quad = \quad K_1\phi i_a = K_m i_a \tag{8.4}$$

where K_m is the torque constant in N-m/amp. Also, from Equation (8.2) we have

$$V_b \quad = \quad K_2\phi\omega_m = K_b\omega_m = K_b\frac{d\theta_m}{dt} \tag{8.5}$$

where K_b is the back emf constant. It can be shown that the numerical values of K_m and K_b are the same provided MKS units[1] are used.

The torque constant can be determined from a set of torque-speed curves as shown in Figure 8.4 for various values of the applied voltage V.

When the motor is stalled, the blocked-rotor torque at the rated voltage V_r is denoted by τ_0. Using Equations (8.3) and (8.4) with $V_b = 0$ and $di_a/dt = 0$ we have

$$V_r = Ri_a = \frac{R\tau_0}{K_m} \tag{8.6}$$

Therefore the torque constant is

$$K_m \quad = \quad \frac{R\tau_0}{V_r} \tag{8.7}$$

[1]MKS units are based on the meter, kilogram, and second.

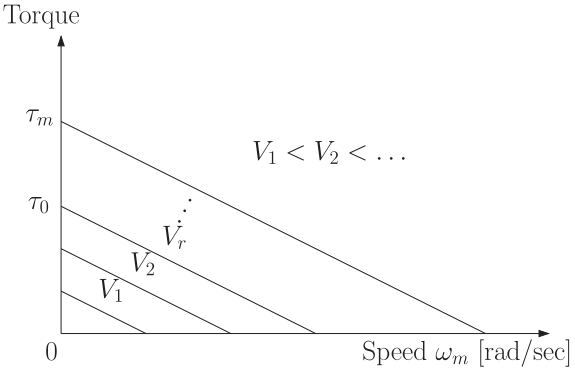


Figure 8.4: Typical torque-speed curves of a DC motor. Each line represents the torque versus speed for a given value of the applied voltage.

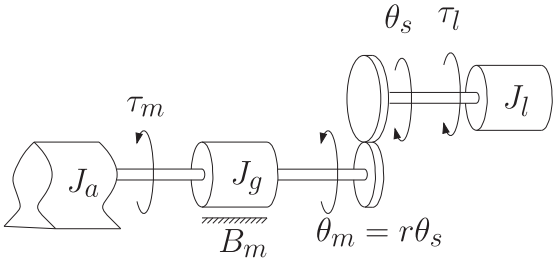


Figure 8.5: Lumped model of a single link with actuator and gear train. J_a, J_g, and J_ℓ are, respectively, the actuator, gear, and load inertias. B_m is the coefficient of motor friction and includes friction in the brushes and gears. The gear ratio is $r:1$ with $r \gg 1$.

8.3 Load Dynamics

In this section we consider the dynamics of the DC motor in series with a gear train and load as shown in Figure 8.5. The gear ratio is $r:1$, where r typically has values in the range 20 to 200 or more. The load is represented by the rotational inertia J_ℓ. Referring to Figure 8.5, we set $J_m = J_a + J_g$, the sum of the actuator and gear inertias.

In terms of the motor angle θ_m, the equation of motion of this system is then

$$\begin{aligned} J_m \frac{d^2\theta_m}{dt^2} + B_m \frac{d\theta_m}{dt} &= \tau_m - \tau_\ell / r \\ &= K_m i_a - \tau_\ell / r \end{aligned} \tag{8.8}$$

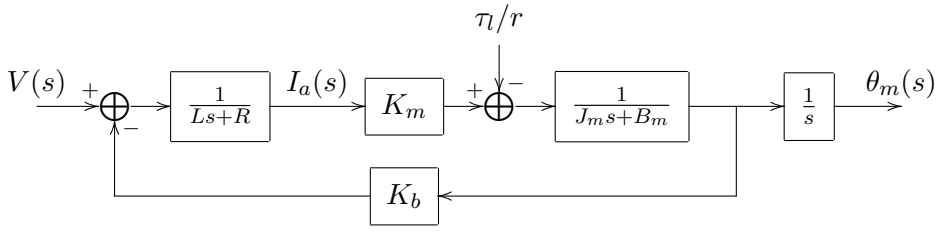


Figure 8.6: Block diagram for a DC motor system. The block diagram represents a third-order system from input voltage $V(s)$ to output position $\theta_m(s)$.

the latter equality coming from Equation (8.4). In the Laplace domain the three Equations (8.3), (8.5), and (8.8) may be combined and written as

$$(Ls + R)I_a(s) = V(s) - K_b s\Theta_m(s) \tag{8.9}$$

$$(J_m s^2 + B_m s)\Theta_m(s) = K_m I_a(s) - \tau_\ell(s)/r \tag{8.10}$$

The block diagram of the above system is shown in Figure 8.6.

The transfer function from $V(s)$ to $\Theta_m(s)$ is given, with $\tau_\ell = 0$, by (Problem 8–1)

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m}{s\left[(Ls + R)(J_m s + B_m) + K_b K_m\right]} \tag{8.11}$$

The transfer function from the load torque $\tau_\ell(s)$ to $\Theta_m(s)$ is given, with $V = 0$, by (Problem 8–1)

$$\frac{\Theta_m(s)}{\tau_\ell(s)} = \frac{-(Ls + R)/r}{s\left[(Ls + R)(J_m s + B_m) + K_b K_m\right]} \tag{8.12}$$

Notice that the magnitude of this latter transfer function, and hence the effect of the load torque on the motor angle, is reduced by the gear ratio r.

Frequently it is assumed that the "electrical time constant" L/R is much smaller than the "mechanical time constant" J_m/B_m. This is a reasonable assumption for many electromechanical systems and leads to a reduced order model of the actuator dynamics. If we divide numerator and denominator of Equations (8.11) and (8.12) by R and neglect the electrical time constant by setting L/R equal to zero, the transfer functions in Equations (8.11) and (8.12) become, respectively, (Problem 8–2)

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m/R}{s(J_m s + B_m + K_b K_m/R)} \tag{8.13}$$

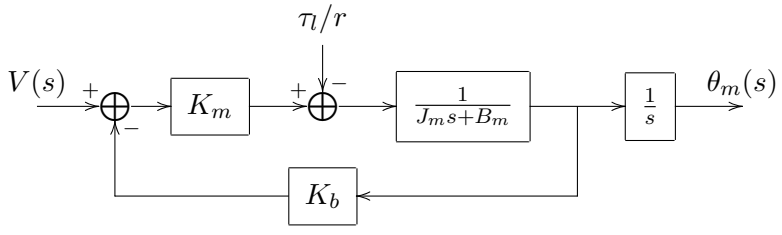


Figure 8.7: Block diagram for the reduced-order system. The block diagram now represents a second-order system.

and

$$\frac{\Theta_m(s)}{\tau_\ell(s)} = \frac{-1/r}{s(J_m(s) + B_m + K_b K_m/R)} \tag{8.14}$$

In the time domain Equations (8.13) and (8.14) represent, by superposition, the second-order differential equation

$$J_m\ddot{\theta}_m(t) + (B_m + K_b K_m/R)\dot{\theta}_m(t) = (K_m/R)V(t) - \tau_\ell(t)/r \tag{8.15}$$

The block diagram corresponding to the reduced-order system (8.15) is shown in Figure 8.7.

8.4 Independent Joint Model

In this section we refine the previous model by assuming that the load attached to the DC motor is a link of a multi-link manipulator rather than a simple rotational inertia in order to generate a more accurate description of the manipulator load dynamics. This section assumes knowledge of the Euler–Lagrange equations that we derived in Chapter 6 and may be skipped if the reader has not studied that chapter.

In Chapter 6 we obtained the following set of differential equations describing the motion of an n-degree-of-freedom manipulator (cf. Equation (6.66))

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{8.16}$$

We assume that the k-th component, τ_k, of the generalized force vector τ is a torque about the axis z_{k-1} if joint k is revolute, and is a force along z_{k-1} if joint k is prismatic. If the output side of the gear train is directly coupled to the joint axis, then the joint variables and motor variables are related by

$$q_k = \theta_{m_k}/r_k \ , \ k = 1, \ldots, n \tag{8.17}$$

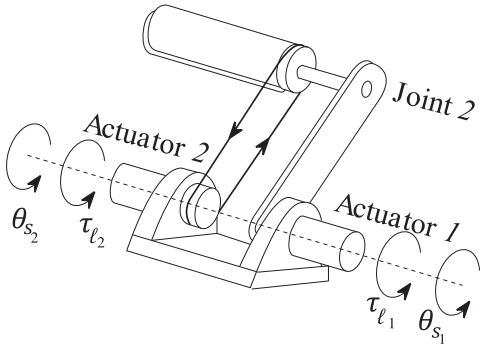


Figure 8.8: Two-link manipulator with remotely driven link.

where r_k is the k-th gear ratio. Similarly, the joint torques τ_k given by (8.16) and the actuator load torques τ_{ℓ_k} are related by

$$\tau_k = \tau_{\ell_k} \ , \ k = 1, \ldots, n \tag{8.18}$$

Remark 8.1. *In many manipulator designs, such as those incorporating belts, pulleys or chains between the actuators and joints the relationship between joint variables and actuator variables is more complicated. In general, one must incorporate a transformation of the form*

$$q_k = f_k(\theta_{s_1}, \ldots, \theta_{s_n}) \ , \ \tau_{\ell_k} = f_k(\tau_1, \ldots, \tau_n) \tag{8.19}$$

where $\theta_{s_k} = \theta_{m_k}/r_k$.

Example 8.1. *Consider the two-link manipulator shown in Figure 8.8 whose actuators are both located on the robot base. In this case we have*

$$q_1 = \theta_{s_1} \quad ; \quad q_2 = \theta_{s_1} + \theta_{s_2} \tag{8.20}$$

$$\tau_{\ell_1} = \tau_1 \quad ; \quad \tau_{\ell_2} = \tau_1 + \tau_2 \tag{8.21}$$

For the following discussion, assume for simplicity that the joint variables q_k, τ_{ℓ_k} and actuator variables θ_{s_k}, τ_k are related as follows

$$q_k = \theta_{s_k} = \theta_{m_k}/r_k \tag{8.22}$$

$$\tau_{\ell_k} = \tau_k \tag{8.23}$$

Then the equations of motion of the manipulator can be written componentwise, for $k = 1, \ldots, n$ as

$$\sum_{j=1}^{n} d_{jk}(q)\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k \tag{8.24}$$

$$J_{m_k}\ddot{\theta}_{m_k} + (B_{m_k} + K_{bk}K_{mk}/R_k)\dot{\theta}_{m_k} = (K_{mk}/R_k)V_k - \tau_k/r_k \tag{8.25}$$

Equation (8.25) represents the actuator dynamics and Equation (8.24) represents the nonlinear inertial, centripetal, coriolis, and gravitational coupling effects due to the motion of the manipulator. The simplest approach to the control of the above system is to consider the nonlinear term τ_k entering (8.25) and defined by (8.24) as an input disturbance to the motor and design an independent controller for each joint according to the model (8.25). The advantage of this approach is its simplicity since the motor dynamics represented by (8.25) are linear. Note that the term τ_k in (8.25) is divided by the gear ratio r_k. This is an important observation. The effect of the gear reduction is to reduce magnitude of the coupling nonlinearities given by (8.24), which adds to the validity of the independent joint control approach. However, for very high speed motion or for direct-drive manipulators without gear reduction at the joints, the coupling nonlinearities have a much larger effect on the performance of the system and so treating the nonlinear coupling effects as a disturbance will generally result in larger tracking errors. For this reason, we will introduce more advanced, nonlinear feedback control methods in Chapter 9.

Now, since $q_k = \theta_{m_k}/r_k$, the actual coefficient of $\ddot{\theta}_{m_k}$ in (8.25), includes the term $d_{kk}(q)/r_k^2$ from (8.24) and is thus given by

$$J_{kk} = r_k^2 J_m + d_{kk}(q) \tag{8.26}$$

which is, of course, configuration dependent and may vary over a large range. For example, Figure 8.9 shows the approximate range of inertias for the Stanford manipulator from [11].

Therefore, we may approximate the inertia coefficient J_{kk} by a constant **average** or **effective** inertia, called J_{eff_k}, for example, by taking the midpoint of values such as those in Table 8.9.

Setting

$$B_{eff_k} = B_{m_k} + k_{b_k} K_{m_k}/R_k \text{ and } u_k = K_{m_k} V_k / R_k \tag{8.27}$$

we write Equation (8.25) as

$$J_{eff_k} \ddot{\theta}_{m_k} + B_{eff_k} \dot{\theta}_{m_k} = u_k - d_k/r_k \tag{8.28}$$

where d_k is treated as a disturbance and defined by

$$d_k = \sum_{j \neq k} d_{jk}(q)\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) \tag{8.29}$$

Link k	J_{kk} min	J_{kk} max
1	1.417	6.176
2	3.590	6.950
3	7.257	7.257
4	0.108	0.123
5	0.114	0.114
5	0.02	0.02

Figure 8.9: Approximate range of effective inertias J_{kk} for the Stanford manipulator (kgm^2) from [11].

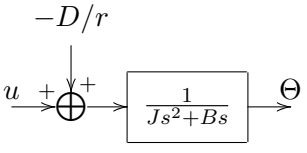


Figure 8.10: Block diagram of the simplified, open-loop system. The disturbance d/r represents all of the nonlinearities and coupling from the other links.

Note that both Equations (8.15) and (8.28) are of the form

$$J\ddot{\theta}(t) + B\dot{\theta}(t) \quad = \quad u(t) - d(t)/r \tag{8.30}$$

for suitable definitions of J and B and where $u(t)$ is a control input with units of torque and $d(t)$ represents a disturbance torque. Henceforth we will use this model (8.30) in the subsequent discussion of the compensator design. The block diagram in the Laplace domain corresponding to the reduced-order system (8.30) is shown in Figure 8.10.

8.5 PID Control

The PID (Proportional, Integral, Derivative) compensator is the most common type of controller used in most manipulators. The general form of a PID controller $u(t)$ is

$$u(t) = K_P e(t) + K_I \int_0^t e(\sigma)d\sigma + K_D \frac{de}{dt} \tag{8.31}$$

where $e(t) = \theta(t) - \theta^d(t)$ is the **tracking error**. The controller parameters are the **proportional gain** K_P, **integral gain** K_I, and **derivative gain**

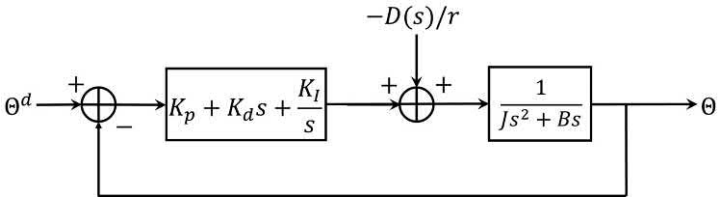


Figure 8.11: The system in Figure 8.10 with a PID compensator. K_P, K_I and K_D are the proportional, integral and derivative gains and Θ^d is the joint angle reference signal to be tracked.

K_D. In the Laplace domain we write Equation (8.31) as

$$U(s) = (K_P + K_D s + K_I/s)E(s) = C(s)E(s) \tag{8.32}$$

and we call

$$C(s) = K_P + K_D s + K_I/s = \frac{K_D s^2 + K_P s + K_I}{s} \tag{8.33}$$

the PID **compensator**.

Referring to Equation (8.33), the PID Compensator adds a pole (at $s = 0$) and two zeros (roots of $K_D s^2 + K_P s + K_I$) to the forward transfer function. The design problem, known as **tuning**, is then to choose the PID gains K_P, K_D, and K_I to achieve the desired performance. There are numerous results available for tuning PID compensators and we will not repeat them in any detail. The goal here is to provide insight into some of the main practical problems that arise in compensator design for a single link of a robot manipulator.

Two-Degree-of-Freedom Controller

In Figure 8.11, the compensator acts on the error signal. If the reference θ^d is a step reference, then the derivative term will introduce an impulse at $t = 0$. To avoid this we may use the alternative architecture shown in Figure 8.12, which is often called a **two-degree-of-freedom** controller. In this architecture, the input to the derivative term $K_D s$ is the output signal θ, rather than the error signal $\theta - \theta^d$, which avoids differentiating the step input θ^d at time $t = 0$, where the step function is discontinuous.

The Closed-Loop Systems

Whether we use the controller configuration in Figure 8.11 or Figure 8.12, we must analyze the appropriate transfer functions to assess the performance

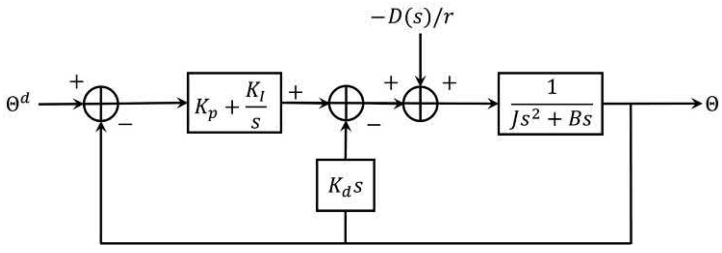


Figure 8.12: The system in Figure 8.10 with a two-degree-of-freedom PID compensator.

of the closed-loop systems. The transfer functions of relevance are, in each case, $G_1(s) = \dfrac{\Theta(s)}{\Theta^d(s)}$, the transfer function from the reference to the output, and $G_2(s) = \dfrac{\Theta(s)}{D(s)}$, the transfer function from the disturbance to the output.

The transfer function $G_1(s)$ is computed by setting the disturbance input $D(s)$ equal to zero. Likewise, the transfer function $G_2(s)$ is computed by setting the reference input $\Theta^d(s)$ equal to zero. By the **Principle of Superposition** then, the overall system response, in each case, is given by

$$\Theta(s) = G_1(s)\Theta^d(s) + G_2(s)D(s)$$

The first two rows in Table 8.1 show the four closed-loop transfer functions from the reference input $\Theta^d(s)$ to the output $\Theta(s)$ ($G_1(s)$) with the one- and two-degree-of-freedom architectures for both PID and PD compensators. The third row in Table 8.1 shows corresponding closed-loop transfer functions from the disturbance input $D(s)$ to the output $\Theta(s)$ ($G_2(s)$). In the latter case, the transfer functions from $D(s)$ to $\Theta(s)$ are the same for both the one- and two-degree-of-freedom architectures. The derivation of these transfer function is straigt!forward and left as an exercise (Problem 8–4).

Remark 8.2 (Derivative Filter)**.** *Note that the derivative term $K_D s$ in the PID compensator involves differentiation of the signal $\theta(t)$. In practice, a differentiator will amplify high-frequency signals and thus will perform poorly in the presence of noise. One may avoid using a pure differentiator by using a velocity sensor, such as a tachometer, to measure $\dot{\theta}$ directly or, more commonly, to use a filter of the form*

$$K_D f(s) = K_D \frac{Ns}{s+N} = K_D \frac{s}{\epsilon s + 1} \tag{8.34}$$

With PID Compensator	With PD Compensator
a) $\frac{K_D s^2 + K_P s + K_I}{Js^3 + (B + K_D)s^2 + K_P s + K_I}$	d) $\frac{K_D s + K_P}{Js^2 + (B + K_D)s + K_P}$
b) $\frac{K_P s + K_I}{Js^3 + (B + K_D)s^2 + K_P s + K_I}$	e) $\frac{K_P}{Js^2 + (B + K_D)s + K_P}$
c) $\frac{-s/r}{Js^3 + (B + K_D)s^2 + K_P s + K_I}$	f) $\frac{-1/r}{Js^2 + (B + K_D)s + K_P}$

Table 8.1: Closed-loop transfer functions using both the one- and two-degree-of-freedom architectures. a) and d) represent $G_1(s)$ in the one-degree-of-freedom architecture. b) and e) represent $G_1(s)$ in the two-degree-of-freedom architecture. c) and f) represent $G_2(s)$.

where $\epsilon = 1/N$. Note that $f(s) \to s$, an ideal differentiator, as $\epsilon \to 0$ and thus represents an approximate differentiator for small values of ϵ and low frequencies in s. The effect of such a derivative filter is both to introduce phase lag into the estimate of $\dot{\theta}(t)$ but also to mitigate the effect of noise amplification caused by ideal differentiation.

Set-Point Tracking

In this section, we consider the problem of **set-point tracking**, which is the problem of tracking a constant or step reference command θ^d and arises in point-to-point motion. Since the reference input is a step signal, we will use the two-degree-of-freedom compensator in Figure 8.12.

We first consider a PD compensator by setting the integral gain K_I equal to zero. In this case, we see from Table 8.1 that the closed-loop system is given by

$$\Theta(s) = \frac{K_P}{\Omega(s)}\Theta^d(s) - \frac{1/r}{\Omega(s)}D(s) \tag{8.35}$$

where $\Omega(s)$ is the closed-loop characteristic polynomial

$$\Omega(s) = Js^2 + (B + K_D)s + K_P \tag{8.36}$$

The closed-loop system will be stable for all positive values of K_P and K_D and bounded disturbances, and the tracking error $E(s)$ is given by

$$\begin{aligned} E(s) &= \Theta^d(s) - \Theta(s) \\ &= \frac{Js^2 + Bs}{\Omega(s)}\Theta^d(s) + \frac{1/r}{\Omega(s)}D(s) \end{aligned} \tag{8.37}$$

For a step reference input

$$\Theta^d(s) = \frac{\Theta^d}{s} \tag{8.38}$$

and a constant disturbance

$$D(s) = \frac{D}{s} \tag{8.39}$$

it follows directly from the final value theorem that the steady-state error e_{ss} satisfies

$$e_{ss} = \lim_{s\to 0} sE(s) = -\frac{D/r}{K_P} \tag{8.40}$$

Since the magnitude of the disturbance is proportional to the gear ratio $1/r$, we see that the steady state error is smaller for larger gear ratio and can be made arbitrarily small by making the position gain K_P large, which is to be expected since the system is type 1.

Using a PD compensator, the closed-loop system is second-order and hence the step response is determined by the closed-loop natural frequency ω and damping ratio ζ. Given a desired value for these quantities, the gains K_D and K_P can be found from the expression

$$s^2 + \frac{(B + K_D)}{J}s + \frac{K_P}{J} = s^2 + 2\zeta\omega s + \omega^2 \tag{8.41}$$

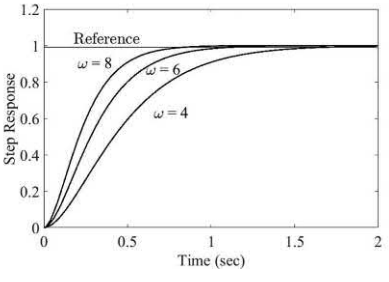
as

$$K_P = \omega^2 J, \qquad K_D = 2\zeta\omega J - B \tag{8.42}$$

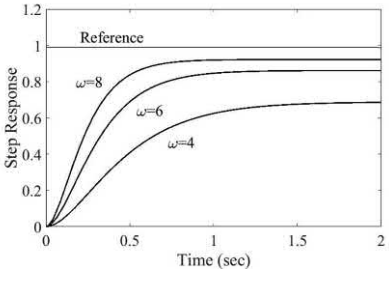
It is customary in robotics applications to take the damping ratio $\zeta = 1$ so that the response is critically damped. In this context ω determines the speed of response.

Example 8.2. *Taking $J = B = 1$, for illustrative purposes, the closed-loop characteristic polynomial is*

$$p(s) = s^2 + (1 + K_D)s + K_P \tag{8.43}$$



(a)



(b)

Figure 8.13: Second-order step responses with PD control. The speed of response, as measured by rise time, is faster for larger values of the natural frequency ω. Likewise, the steady-state error due to a constant disturbance is smaller for larger values of ω.

With $\theta^d = 1$ and no disturbance acting on the system, Figure 8.13(a) shows the resulting step responses for several values of the natural frequency ω. If there is a constant disturbance, we recall from Equation (8.40) that there will be a steady-state error $-\frac{D/r}{K_p}$. This steady-state error can be reduced by increasing the proportional gain K_p, but cannot be eliminated entirely for any finite K_P. Figure 8.13(b) shows the steady-state error for the same values of ω.

The Effect of Saturation

In theory, an arbitrarily fast response and arbitrarily small steady-state error to a constant disturbance could be achieved by simply increasing the gains in the PD compensator. In practice, however, there is a maximum speed of response achievable from the system due to limits on the maximum torque (or current) **saturation**, due to limits on the maximum torque output of the motors. Many manipulators, in fact, incorporate current limiters in the servo-system to prevent damage to the motors that migt! result from overdrawing current.

Figure 8.14 therefore includes a saturation function represents the maximum allowable output of the compensator. Figure 8.15 shows a comparison of the step responses with and without saturation. The disturbance input has been set to zero.

PID Control

In order to remove the steady-state error due to the disturbance entirely we can employ integral control. Adding an integral term K_I/s to the above

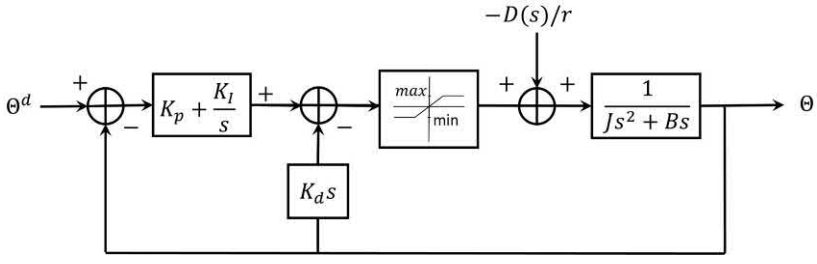


Figure 8.14: Second-order system with input saturation limiting the magnitude of the input signal. Increasing the magnitude of the compensator output signal beyond the saturation limit will not increase the input to the plant.

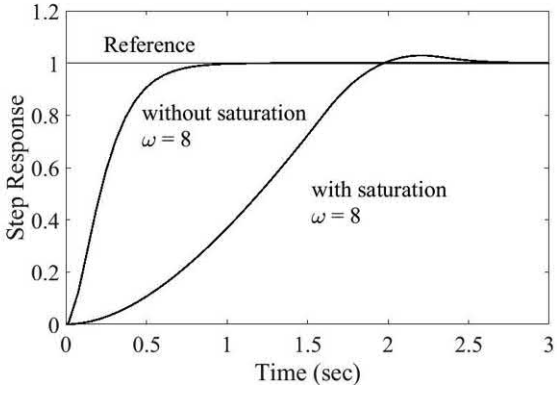


Figure 8.15: Second-order step response with PD control and saturation.

results in the closed-loop transfer functions b) and c) in Table 8.1. The PID control achieves exact steady tracking of step inputs while rejecting step disturbances, provided of course that the closed-loop system is stable.

With the PID compensator

$$U(s) = (K_P + \frac{K_I}{s})(\Theta^d(s) - \Theta(s)) - K_D s\Theta(s) \tag{8.44}$$

the closed-loop system is now the third order system

$$\Theta(s) = \frac{(K_P s + K_I)}{\Omega_2(s)}\Theta^d(s) - \frac{s}{\Omega_2(s)}D(s) \tag{8.45}$$

with characteristic polynomial

$$\Omega_2 = Js^3 + (B + K_D)s^2 + K_P s + K_I \tag{8.46}$$

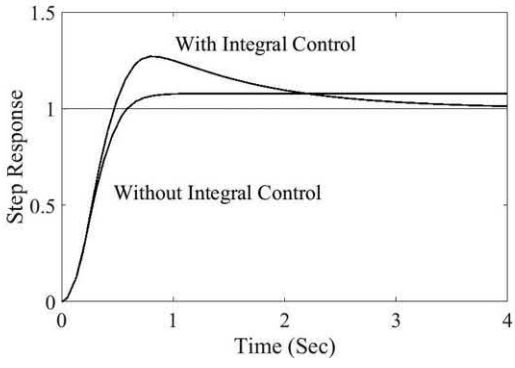


Figure 8.16: Response with integral control action showing that the steady-state error to a constant disturbance has been removed.

Applying the Routh–Hurwitz criterion to this polynomial, it follows that the closed-loop system is stable if the gains are positive, and in addition,

$$K_I \quad < \quad \frac{(B + K_D)K_P}{J} \tag{8.47}$$

A common design rule-of-thumb for PID control is to first set $K_I = 0$ and design the proportional and derivative gains, K_P and K_D, to achieve the desired transient behavior (rise time, settling time, and so forth) and then to choose K_I within the limits imposed by (8.47) to remove the steady-state error.

Example 8.3. *To the previous system we have added an integral control term in the compensator. The step responses are shown in Figure 8.16. We see that the steady-state error due to the disturbance is removed.*

8.6 Feedforward Control

The analysis in the previous section was carried out under the assumption that the reference signal and disturbance are constant and is not valid for tracking more general time-varying trajectories such as a cubic polynomial trajectory of the type generated in Chapter 7. In this section we introduce the notion of **feedforward control** as a method to track time-varying trajectories. We use the one-degree-off freedom architecture for simplicity.

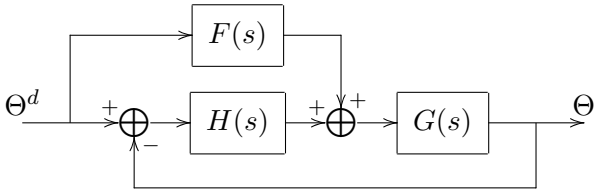


Figure 8.17: Feedforward control scheme. $F(s)$ is the feedforward transfer function which has the reference signal Θ^d as input. The output of the feedforward block is superimposed on the output of the compensator $H(s)$.

8.6.1 Trajectory Tracking

Suppose that $\theta^d(t)$ is a joint space reference trajectory and consider the block diagram of Figure 8.17, where $G(s)$ represents the forward transfer function of a given system and $H(s)$ is the compensator transfer function.

A feedforward control scheme consists of adding a feedforward path with transfer function $F(s)$ as shown. Let each of the three transfer functions be represented as ratios of polynomials

$$G(s) = \frac{q(s)}{p(s)}, H(s) = \frac{c(s)}{d(s)}, F(s) = \frac{a(s)}{b(s)} \tag{8.48}$$

We assume that $G(s)$ is strictly proper and $H(s)$ is proper. Simple block diagram manipulation shows that the closed-loop transfer function $T(s) = \frac{Y(s)}{R(s)}$ is given by (Problem 8–9)

$$T(s) \quad = \quad \frac{q(s)(c(s)b(s) + a(s)d(s))}{b(s)(p(s)d(s) + q(s)c(s))} \tag{8.49}$$

The closed-loop characteristic polynomial is $b(s)(p(s)d(s) + q(s)c(s))$. Therefore, for stability of the closed-loop system, we require that the compensator $H(s)$ and the feedforward transfer function $F(s)$ be chosen so that the polynomials $p(s)d(s) + q(s)c(s)$ and $b(s)$ are Hurwitz. This says that, in addition to stability of the closed-loop system, the feedforward transfer function $F(s)$ must itself be stable.

If we choose the feedforward transfer function $F(s)$ equal to $1/G(s)$, the inverse of the forward plant, that is, $a(s) = p(s)$ and $b(s) = q(s)$, then the closed-loop system becomes

$$q(s)(p(s)d(s) + q(s)c(s))Y(s) = q(s)(p(s)d(s) + q(s)c(s))R(s) \tag{8.50}$$

or, in terms of the tracking error $E(s) = R(s) - Y(s)$,

$$q(s)(p(s)d(s) + q(s)c(s))E(s) \quad = \quad 0 \tag{8.51}$$

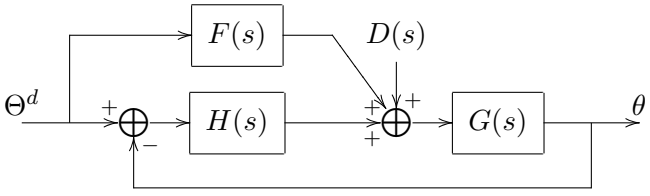


Figure 8.18: Feedforward control with disturbance $D(s)$.

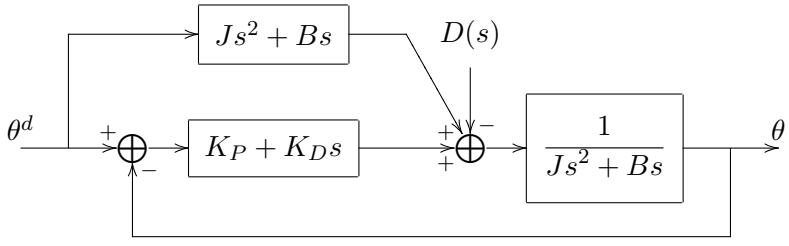


Figure 8.19: Feedforward compensator for the second-order system of Section 8.5.

Thus, assuming stability, the output $\theta(t)$ will track any reference trajectory $\theta^d(t)$. Note that we can only choose $F(s)$ in this manner provided that the numerator polynomial $q(s)$ of the forward plant is Hurwitz, that is, as long as all zeros of the forward plant are in the left half plane. Such systems are called **minimum phase**.

If there is a disturbance $D(s)$ entering the system as shown in Figure 8.18, then it is easily shown (Problem 8–10) that the tracking error $E(s)$ is given by

$$E(s) = \frac{q(s)d(s)}{p(s)d(s) + q(s)c(s)} D(s) \tag{8.52}$$

We have thus shown that, in the absence of disturbances, the closed-loop system will track **any** desired trajectory $\theta^d(t)$ provided that the closed-loop system is stable. The steady-state error is thus due only to the disturbance.

Let us apply this idea to the model of Section 8.5. Suppose that $\theta^d(t)$ is an arbitrary trajectory that we wish the system to track. In this case we have $G(s) = 1/(Js^2 + Bs)$ together with a PD compensator $H(s) = K_P + K_D s$.

We see that the plant transfer function $G(s)$ has no finite zeros and hence is minimum phase. Thus, we can choose the feedforward transfer $F(s)$ as $F(s) = Js^2 + Bs$. The resulting system is shown in Figure 8.19.

Note that $1/G(s)$ is not a proper rational function. However, since the derivatives of the reference trajectory $\theta^d(t)$ are known and precomputed,

the implementation of the above scheme does not require differentiation of an actual signal. It is easy to see from Equation (8.52) that the steady-state error to a step disturbance is now given by the same expression as in Equation (8.40) independent of the reference trajectory. As before, a PID compensator would result in zero steady-state error to a step disturbance. In the time domain the control law of Figure 8.19 can be written as

$$\begin{aligned} V(t) &= J\ddot{\theta}^d + B\dot{\theta}^d + K_D(\dot{\theta}^d - \dot{\theta}) + K_P(\theta^d - \theta) \\ &= f(t) + K_D\dot{e}(t) + K_P e(t) \end{aligned} \tag{8.53}$$

where $f(t)$ is the feedforward signal

$$f(t) = J\ddot{\theta}^d + B\dot{\theta}^d \tag{8.54}$$

and $e(t)$ is the tracking error $\theta^d(t) - \theta(t)$. Since the forward plant equation is

$$J\ddot{\theta} + B\dot{\theta} = V(t) - rd(t)$$

the closed-loop error $e(t) = \theta(t) - \theta^d(t)$ satisfies the second-order differential equation

$$J\ddot{e} + (B + K_D)\dot{e} + K_P e(t) = -d/r(t) \tag{8.55}$$

We note from Equation (8.55) that the characteristic polynomial of the closed-loop system is identical to Equation (8.36). However, the system (8.55) is now written in terms of the tracking error $e(t)$. Therefore, assuming that the closed-loop system is stable, the tracking error will approach zero asymptotically for any desired joint space trajectory in the absence of disturbances, that is, if $d = 0$.

8.6.2 The Method of Computed Torque

In this section we discuss the so-called method of **computed torque**, which as we shall see, can be viewed as a feedforward disturbance rejection scheme. This section assumes knowledge of the Euler–Lagrange equations from Chapter 6.

We see that the feedforward signal (8.54) results in asymptotic tracking of any trajectory in the absence of disturbances but does not otherwise improve the disturbance rejection properties of the system. However, although the term $d(t)$ represents a disturbance, it is not completely unknown since $d(t)$ satisfies (8.29). Thus we may consider adding to the above feedforward

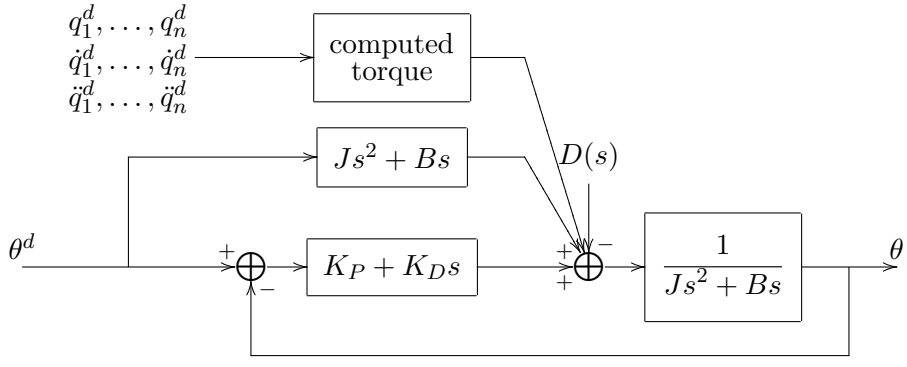


Figure 8.20: Computed torque feedforward disturbance cancellation. The term (8.56) is added to the output of the compensator to cancel the effect of the disturbance.

signal a term to anticipate and cancel the effects of the disturbance $d(t)$. This is known as **feedforward disturbance cancellation**.

Consider the block diagram in Figure 8.20. Given desired trajectories $q^d(t) = (q_1^d(t), \ldots, q_n^d(t))$ for each joint and desired joint velocities and accelerations $\dot{q}^d(t) = (\dot{q}_1^d(t), \ldots, \dot{q}_n^d(t))$ and $\ddot{q}^d(t) = (\ddot{q}_1^d(t), \ldots, \ddot{q}_n^d(t))$, respectively, we superimpose the term $d_k^d(t)$ at the k-th joint according to

$$d_k^d(t) = \sum_{j \neq k} d_{j,k}(q_j^d(t))\ddot{q}_j^d(t) + \sum_{i,j=1}^{n} c_{i,j,k}(q^d(t))\dot{q}_i^d(t)\dot{q}_j^d(t) + g_k(q^d(t)) \quad (8.56)$$

Since $d_k^d(t)$ has units of torque, the above feedforward disturbance cancellation control is called the **method of computed torque**. The expression (8.56) compensates in a feedforward manner the nonlinear coupling inertia, Coriolis, centripetal, and gravitational forces arising from the motion of the manipulator.

8.7 Drive-Train Dynamics

A second effect that limits the achievable performance of a manipulator is **flexibility** in the motor shaft and/or drive train. We refer to this as **joint flexibility** or **joint elasticity**. For many manipulators, particularly those using so-called **strain wave gears** or **harmonic gears**, for torque transmission, the joint flexibility is significant. In addition to torsional flexibility in the gears, joint flexibility is caused by effects such as shaft windup, bearing deformation, and compressibility of the hydraulic fluid in hydraulic robots.

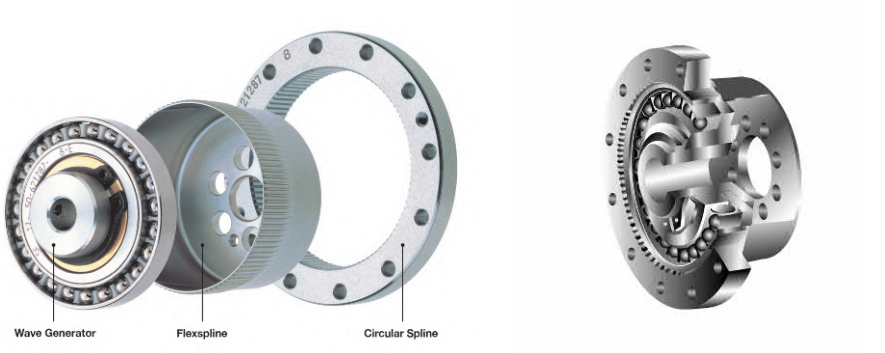


Figure 8.21: The Harmonic Drive® gear. The rotation of the elliptical wave generator meshes the teeth of the flexspline and circular spline resulting in low backlash and high torque transmission. (Courtesy of Harmonic Drive, LLC.)

Remark 8.3. *Let k_r be the effective stiffness at the joint. The joint resonant frequency is then $\omega_r = \sqrt{k_r/J}$. It is common engineering practice to limit ω in Equation (8.42) to no more than half of ω_r to avoid excitation of the joint resonance. However, in order to say more, we will model the flexibility and consider more advanced control design methods below.*

Harmonic Gears

Harmonic gears are a type of gear mechanism that are very popular for use in robots due to their low backlash, high torque transmission, and compact size.

A typical harmonic gear, the **Harmonic Drive® gear**, is shown in Figure 8.21 and consists of a rigid **circular spline**, a flexible **flexspline**, and an elliptical **wave generator**. The wave generator is attached to the actuator and hence is turned at high speed by the motor. The circular spline is attached to the load. As the wave generator rotates it deforms the flexspline causing a number of teeth of the flexspline to mesh with the teeth of the circular spline. The effective gear ratio is determined by the difference in the number of teeth of the flexspline and circular spline.

The low backlash and high torque throughput of the harmonic gears result from the relatively large number of teeth that are meshed at any given time. However, the principle of the harmonic gear relies on the flexibility of the flexspline. This flexibility is the limiting factor to the achievable performance in many cases.

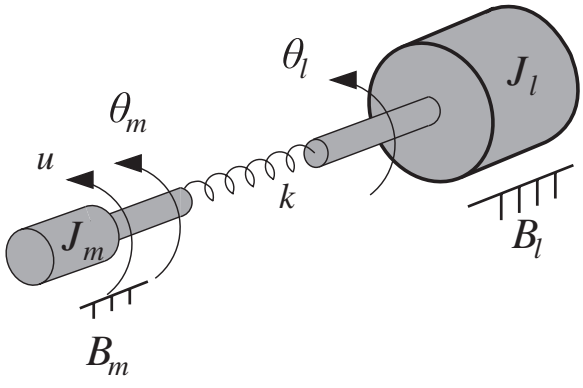


Figure 8.22: Idealized model to represent joint flexibility. The stiffness constant k represents the effective torsional stiffness of the harmonic gear.

Consider the idealized situation of Figure 8.22, consisting of an actuator connected to a load through a torsional spring representing the joint flexibility. For simplicity we take the motor torque u, rather than the armature voltage, as input. The equations of motion are

$$J_\ell \ddot{\theta}_\ell + B_\ell \dot{\theta}_\ell + k(\theta_\ell - \theta_m) = 0 \tag{8.57}$$

$$J_m \ddot{\theta}_m + B_m \dot{\theta}_m - k(\theta_\ell - \theta_m) = u \tag{8.58}$$

where J_ℓ, J_m are the load and motor inertias, B_ℓ and B_m are the load and motor damping constants, and u is the input torque applied to the motor shaft. The joint stiffness constant k represents the torsional stiffness of the harmonic gear. In the Laplace domain we can write the above system as

$$p_\ell(s)\Theta_\ell(s) = k\Theta_m(s) \tag{8.59}$$

$$p_m(s)\Theta_m(s) = k\Theta_\ell(s) + U(s) \tag{8.60}$$

where

$$p_\ell(s) = J_\ell s^2 + B_\ell s + k \tag{8.61}$$

$$p_m(s) = J_m s^2 + B_m s + k \tag{8.62}$$

This system is represented by the block diagram of Figure 8.23. The output to be controlled is, of course, the load angle θ_ℓ. The open-loop transfer function between U and Θ_ℓ is given by

$$\frac{\Theta_\ell(s)}{U(s)} = \frac{k}{p_\ell(s)p_m(s) - k^2} \tag{8.63}$$

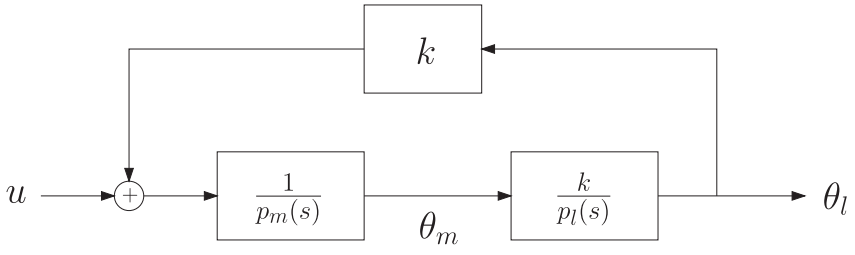


Figure 8.23: Block diagram for the system (8.59) and (8.60).

The open-loop characteristic polynomial $p_\ell p_m - k^2$ is

$$J_\ell J_m s^4 + (J_\ell B_m + J_m B_\ell)s^3 + (k(J_\ell + J_m) + B_\ell B_m)s^2 + k(B_\ell + B_m)s \quad (8.64)$$

We can obtain some insigt! into the behavior of the system by first neglecting the damping coefficients B_ℓ and B_m. In this case the open-loop characteristic polynomial would be

$$J_\ell J_m s^4 + k(J_\ell + J_m)s^2 \quad (8.65)$$

which has a double pole at the origin and a pair of complex conjugate poles on the $j\omega$-axis at $s = \pm j\omega$ where $\omega^2 = k\left(\frac{1}{J_\ell} + \frac{1}{J_m}\right)$. Note that the frequency of the imaginary poles increases with increasing joint stiffness k.

In practice the stiffness of the harmonic gear is large and the damping is small, which results in a difficult system to control. Assuming that the open-loop damping constants B_ℓ and B_m are small, the open-loop poles of the system (8.59) and (8.60) will be in the left half plane near the poles of the undamped system.

Suppose we implement a PD compensator $C(s) = K_P + K_D s$. At this point the analysis depends on whether the position/velocity sensors are placed on the motor shaft or on the load shaft, that is, whether the PD compensator is a function of the motor variables or the load variables. If the motor variables are measured then the closed-loop system is given by the block diagram of Figure 8.24.

If we measure the load angle θ_ℓ instead, the system with PD control is represented by the block diagram of Figure 8.25. The corresponding root locus is shown in Figure 8.27.

Figure 8.26 shows the response of the system with motor feedback (left) and with link feedback (right!) using the PD controller $K_D(s + a)$.

In order to perform a root locus we set $K_P + K_D s = K_D(s + a)$ with $a = K_P/K_D$. The root locus for the closed-loop systems in terms of K_D is

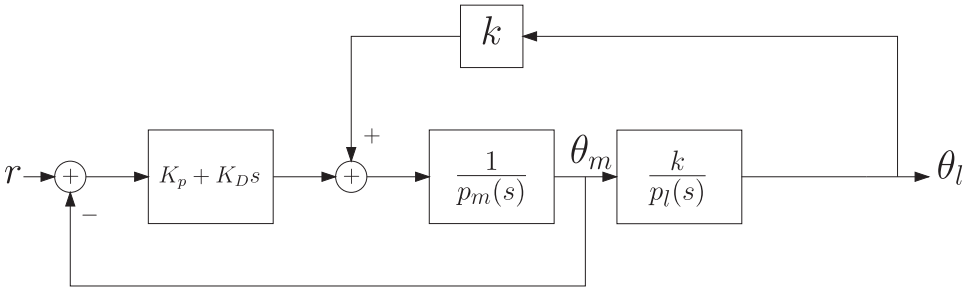


Figure 8.24: PD control with motor angle feedback.

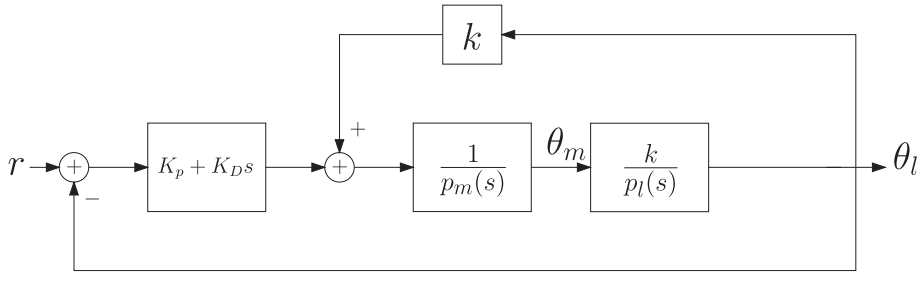


Figure 8.25: PD control with load angle feedback.

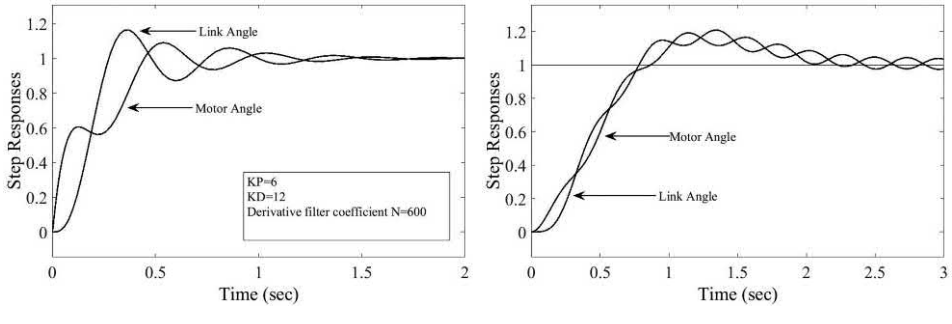


Figure 8.26: Step response — PD control with motor angle feedback (left) and with link angle feedback (right!).

shown in Figure 8.27 for the case of a) motor angle feedback and b) load angle feedback.

In the case that the motor angle θ_m is used in the PD control, we see that the closed-loop system is stable for all values of the gain K_D but that the presence of the open-loop zeros near the $j\omega$ axis may result in undesirable oscillations. Also the poor relative stability means that disturbances and

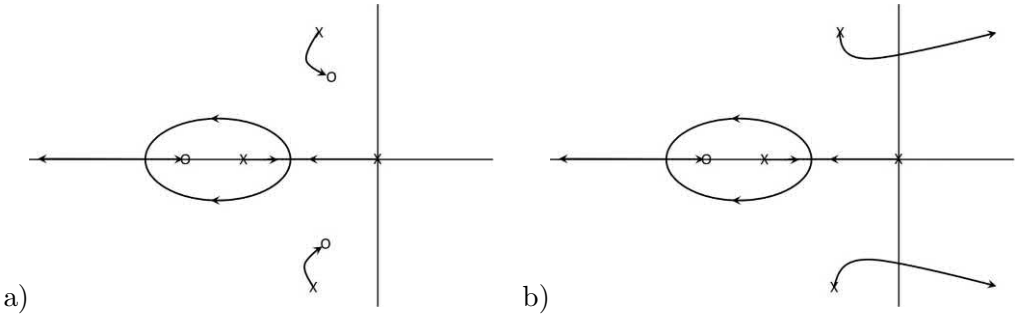


Figure 8.27: Root loci for the flexible joint systems. a) represents motor-angle feedback and b) represents link-angle feedback.

other unmodeled dynamics could render the system unstable.

In the case that the load angle θ_m is used in the PD control, we see that the closed-loop system is unstable for large K_D. The critical value of K_D, that is, the value of K_D for which the system becomes unstable, can be found from the Routh–Hurwitz criterion. The best that one can do in this case is to limit the gain K_D so that the closed-loop poles remain within the left half plane with a reasonable stability margin.

8.8 State Space Design

In this section we consider the application of state space methods for the control of the flexible joint system above.[2] The previous analysis has shown that PD control is inadequate for robot control unless the joint flexibility is negligible or unless one is content with relatively slow response of the manipulator. Not only does the joint flexibility limit the magnitude of the gain for stability reasons, it also introduces ligt!ly damped poles into the closed-loop system that may result in oscillation in the transient response. We can write the system given by Equations (8.57) and (8.58) in state space by choosing state variables

$$\begin{aligned} x_1 &= \theta_\ell \quad & x_2 &= \dot{\theta}_\ell \\ x_3 &= \theta_m \quad & x_4 &= \dot{\theta}_m \end{aligned} \tag{8.66}$$

[2]This section assumes more knowledge of control theory than previous sections.

In terms of these state variables the system given by Equations (8.57) and (8.58) becomes

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{k}{J_\ell}x_1 - \frac{B_\ell}{J_\ell}x_2 + \frac{k}{J_\ell}x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{k}{J_m}x_1 - \frac{B_\ell}{J_m}x_4 - \frac{k}{J_m}x_3 + \frac{1}{J_m}u \end{aligned} \tag{8.67}$$

which, in matrix form, can be written as

$$\dot{x} = Ax + bu \tag{8.68}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_\ell} & -\frac{B_\ell}{J_\ell} & \frac{k}{J_\ell} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_m} & 0 & -\frac{k}{J_m} & \frac{B_m}{J_m} \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_m} \end{bmatrix} \tag{8.69}$$

If we choose an output $y(t)$, say the measured load angle $\theta_\ell(t)$, then we have an output equation

$$y = x_1 = c^T x \tag{8.70}$$

where

$$c^T = [1, 0, 0, 0] \tag{8.71}$$

The relationship between the state space form given by Equations (8.68)–(8.71) and the transfer function (8.63) is found by taking Laplace transforms of Equations (8.68)–(8.70) with initial conditions set to zero. This yields

$$G(s) = \frac{\Theta_\ell(s)}{U(s)} = \frac{Y(s)}{U(s)} = c^T(sI - A)^{-1}b \tag{8.72}$$

where I is the $n \times n$ identity matrix. The poles of $G(s)$ are eigenvalues of the matrix A. For the system (8.68)–(8.71), the converse holds as well, that is, all of the eigenvalues of A are poles of $G(s)$. This is always true if the state space system is defined using a minimal number of state variables.

8.8.1 State Feedback Control

Given a linear system in state space form, such as Equation (8.68), a **linear state feedback control law** is an input u of the form

$$u(t) = -k^T x + u_r = -\sum_{i=1}^{4} k_i x_i + u_r \tag{8.73}$$

where k_i are constant gains to be determined and u_r is a reference input. In other words, the control is determined as a linear combination of the system states which, in this case, are the motor and load positions and velocities. Compare this to the previous PD/PID control, which was a function either of the motor position and velocity or of the load position and velocity, but not both. If we substitute the control law given by Equation (8.73) into Equation (8.68) we obtain

$$\dot{x} \;=\; (A - bk^T)x + bu_r \tag{8.74}$$

Thus, we see that the linear feedback control has the effect of changing the poles of the system from those determined by A to those determined by $A - bk^T$.

In the previous PD designs the closed-loop pole locations were restricted to lie on the root locus plots shown in Figure 8.27. Since there are more free parameters in Equation (8.73) than in the PD controller, it may be possible to achieve a much larger range of closed-loop poles. This turns out to be the case if the system given by Equation (8.68) satisfies a property known as **controllability**.

Definition 8.1. *A linear system is said to be* **controllable** *if, for each initial state $x(t_0)$ and each final state $x(t_f)$, there is a control input $t \to u(t)$ that transfers the system from $x(t_0)$ at time t_0 to $x(t_f)$ at time t_f.*

The above definition says, in essence, that if a system is controllable we can achieve any state whatsoever in finite time starting from an arbitrary initial state.

Let W_c be the $n \times n$ matrix[3]

$$W_c = [b, Ab, A^2b, \ldots, A^{n-1}b] \tag{8.75}$$

The matrix W_c is called the **controllability matrix** for the linear system defined by the pair (A, b). To check whether a system is controllable we have the following simple test.

[3]For a system with m control inputs W_c will have dimension $n \times nm$

Lemma 8.1. *A linear system of the form (8.68) is controllable if and only if* $\det W_c \neq 0$.

The fundamental importance of controllability of a linear system is shown by the following

Theorem 8.1. *Let* $\alpha(x) = s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0$ *be an arbitrary polynomial of degree n with real coefficients. Then there exists a state feedback control law of the form Equation (8.73) such that*

$$\det(sI - A + bk^T) \quad = \quad \alpha(s) \tag{8.76}$$

if and only if the system (8.68) is controllable.

This fundamental result says that, for a controllable linear system, we may achieve **arbitrary**[4] closed-loop poles using state feedback. Returning to the specific fourth-order system given by Equation (8.69), we see that the system is indeed controllable since

$$\det W_c \quad = \quad \frac{k^2}{J_m^4 J_\ell^2} \tag{8.77}$$

which is never zero since $k > 0$. Thus, we can achieve any desired set of closed-loop poles that we wish, which is much more than was possible using the previous PD compensator.

One way to design the feedback gains is through an optimization procedure. This takes us into the realm of optimal control theory. For example, we may choose as our goal the minimization of the performance criterion

$$J \quad = \quad \int_0^\infty \{x^T(t)Qx(t) + Ru^2(t)\}dt \tag{8.78}$$

where Q is a given symmetric, positive definite matrix and $R > 0$.

Choosing a control law to minimize Equation (8.78) frees us from having to decide beforehand what the closed-loop poles should be as they are automatically dictated by the weigt!ing matrices Q and R in Equation (8.78). It is shown in optimal control texts that the optimum linear control law that minimizes Equation (8.78) is given as

$$u \quad = \quad -k_*^T x \tag{8.79}$$

[4]Since the coefficients of the polynomial $a(s)$ are real, the only restriction on the pole locations is that they occur in complex conjugate pairs.

where

$$k_* = \frac{1}{R} b^T P \tag{8.80}$$

and P is the unique symmetric, positive definite $n \times n$ matrix satisfying the so-called **matrix algebraic Riccati equation**

$$A^T P + PA - \frac{1}{R} P b b^T P + Q = 0 \tag{8.81}$$

The control law (8.79) is referred to as a **linear quadratic (LQ) optimal control**, since the performance index is quadratic and the control system is linear.

8.8.2 Observers

The above result that any set of closed-loop poles may be achieved for a controllable linear system is remarkable. However, to achieve it we have had to pay a price, namely, that the control law must be a function of all of the states. In order to build a compensator that requires only the measured output, in this case θ_ℓ, we need to introduce the concept of an **observer**. An observer is a dynamical system (constructed in software) that attempts to estimate the full state $x(t)$ using only the system model, Equations (8.68)–8.71), and the measured output $y(t)$. A complete discussion of observers is beyond the scope of the present text. We give here only a brief introduction to the main idea of observers for linear systems.

Assuming that we know the parameters of the system (8.68) we could simulate the response of the system in software and recover the value of the state $x(t)$ at time t from the simulation and we could use this simulated or estimated state, call it $\hat{x}(t)$, in place of the true state in Equation (8.79). However, since the true initial condition $x(t_0)$ for Equation (8.68) will generally be unknown, this idea is not feasible. However the idea of using the model of the system given by Equation (8.68) is a good starting point to construct a state estimator in software. Let us, therefore, consider an estimate $\hat{x}(t)$ of $x(t)$ satisfying the system

$$\dot{\hat{x}} = A\hat{x} + bu + \ell(y - c^T \hat{x}) \tag{8.82}$$

Equation (8.82) is called an **observer** for Equation (8.68) and represents a model of the system (8.68) with an additional term $\ell(y - c^T \hat{x})$. This additional term is a measure of the error between the output $y(t) = c^T x(t)$

of the plant and the estimate of the output, $c^T\hat{x}(t)$. Since we know the coefficient matrices in Equation (8.82) and can measure y directly, we can solve the above system for $\hat{x}(t)$ starting from any initial condition, and use this $\hat{x}$ in place of the true state x in the feedback law (8.79). The additional term ℓ in Equation (8.82) is to be designed so that $\hat{x} \to x$ as $t \to \infty$, that is, so that the estimated state converges to the true (unknown) state, independent of the initial condition $x(t_0)$. Let us see how this is done.

Define $e(t) = x - \hat{x}$ as the **estimation error**. Combining Equations (8.68) and (8.82), since $y = c^T x$, we see that the estimation error satisfies the system

$$\dot{e} \quad = \quad (A - \ell c^T)e \tag{8.83}$$

From Equation (8.83) we see that the dynamics of the estimation error are determined by the eigenvalues of $A - \ell c^T$. Since ℓ is a design quantity, we can attempt to choose it so that $e(t) \to 0$ as $t \to \infty$, in which case the estimate $\hat{x}$ converges to the true state x. In order to do this we obviously want to choose ℓ so that the eigenvalues of $A - \ell c^T$ are in the left half plane. This is similar to the pole assignment problem considered previously. In fact it is dual, in a mathematical sense, to the pole assignment problem. It turns out that the eigenvalues of $A - \ell c^T$ can be assigned arbitrarily if and only if the pair (A, c) satisfies the property known as **observability**. Observability is defined by the following:

Definition 8.2. *A linear system is* **observable** *if every initial state $x(t_0)$ can be exactly determined from measurements of the output $y(t)$ and the input $u(t)$ in a finite time interval $t_0 \leq t \leq t_f$.*

Let W_o be the $n \times n$ matrix

$$W_o = \begin{bmatrix} c^T \\ c^T A \\ \vdots \\ c^T A^{n-1} \end{bmatrix} \tag{8.84}$$

The matrix W_o in Equation (8.84) is called the **observability matrix** for the pair (c^T, A). To check whether a system is observable we have the following

Lemma 8.2. *The pair (c^T, A) is observable if and only if* $\det W_0 \neq 0$.

In the system given by Equations (8.68)–(8.71) above we have that

$$\det W_o = \frac{k^2}{J_\ell^2} \tag{8.85}$$

and hence the system is observable.

Note that the problem of finding the observer gains ℓ so that the matrix $A - \ell c^T$ has a prescribed set of eigenvalues is very similar to that of finding the state feedback gains k so that $A - bk^T$ has prescribed eigenvalues. In fact the two problems are **dual** in a precise sense. Since the eigenvalues of A and A^T are the same, we can define the following equivalences:

$$A^T \leftrightarrow A,\ B^T \leftrightarrow c^T, k^T \leftrightarrow \ell, W_c \leftrightarrow W_o \tag{8.86}$$

Using the above equivalences we can take the observer gains ℓ as

$$\ell = \frac{1}{r} P c$$

where P satisfies

$$AP + PA^T - \frac{1}{r} P c c^T P + Q = 0 \tag{8.87}$$

for a given symmetric, positive definite matrix Q and $R > 0$. Now, if we use the estimated state $\hat{x}$ in place of the true state, we have the system (with $r = 0$)

$$\begin{aligned} \dot{x} &= Ax + bu \\ u &= -k^T \hat{x} \end{aligned}$$

It is easy to show from the above that the state x and estimation error e jointly satisfy the equation

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - bk^T & bk^T \\ 0 & A - \ell c^T \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} \tag{8.88}$$

and therefore the set of closed-loop poles of the system will consist of the union of the eigenvalues of $A - \ell c^T$ and the eigenvalues of $A - bk^T$.

This result is known as the **separation principle**. As the name suggests, the separation principle allows us to separate the design of the state feedback control law (8.79) from the design of the state estimator (8.82). A typical procedure is to place the observer poles to the left of the desired pole locations of $A - bk^T$. This results in rapid convergence of the estimated state to the true state, after which the response of the system is nearly the same as if the true state were being used in Equation (8.79). The reader can refer to Chapter 12 for a simulation of the observer-state feedback controller for the case of joint flexibility.

The result that the closed-loop poles of the system may be placed arbitrarily, under the assumption of controllability and observability, is a powerful theoretical result. There are always practical considerations to be taken into account, however. The most serious factor to be considered in observer design is noise in the measurement of the output. To place the poles of the observer very far to the left of the imaginary axis in the complex plane requires that the observer gains be large. Large gains can amplify noise in the output measurement and result in poor overall performance. Large gains in the state feedback control law (8.79) can result in saturation of the input, again resulting in poor performance. Also uncertainties in the system parameters, or nonlinearities such as a nonlinear spring characteristic and backlash, will reduce the achievable performance from the above design. Therefore, the above ideas are intended only to illustrate what may be possible by using more advanced concepts from control theory. In Chapter 9 we will develop more advanced, nonlinear control methods to control systems with uncertainties in the parameters.

8.9 Chapter Summary

This chapter is a basic introduction to robot control treating each joint of the manipulator as an independent single-input/single-output (SISO) system. In this approach one is primarily concerned with the actuator and drive-train dynamics.

Modeling

We first derived a reduced-order linear model for the dynamics of a permanent-magnet DC motor and showed that the transfer function from the motor voltage $V(s)$ to the motor shaft angle $\Theta_m(s)$ can be expressed as

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m/R}{s(J_m s + B_m + K_b K_m/R)}$$

while the transfer function from a load disturbance $D(s)$ to $\Theta_m(s)$ is

$$\frac{\Theta_m(s)}{D(s)} = \frac{-1/r}{s(J_m(s) + B_m + K_b K_m/R)}$$

PID Control

We then considered the set-point tracking problem using PD and PID compensators. A PD compensator is of the form

$$U(s) = K_P(\Theta^d(s) - \Theta(s)) - K_D s\Theta(s)$$

which results in a closed-loop system

$$\Theta(s) = \frac{K_P}{\Omega(s)}\Theta^d(s) - \frac{1}{\Omega(s)}D(s)$$

where

$$\Omega(s) = Js^2 + (B + K_D)s + K_P$$

is the closed-loop characteristic polynomial whose roots determine the closed-loop poles and, hence, the performance of the system.

A PID compensator is of the form

$$U(s) = (K_P + \frac{K_I}{s})(\Theta^d(s) - \Theta(s)) - K_D s\Theta(s)$$

The closed-loop system is now the third order system

$$\Theta(s) = \frac{(K_P s + K_I)}{\Omega_2(s)}\Theta^d(s) - \frac{rs}{\Omega_2(s)}D(s)$$

where

$$\Omega_2 = Js^3 + (B + K_D)s^2 + K_P s + K_I \tag{8.89}$$

We discussed methods to design the PD and PID gains for a desired transient and steady-state response. We then discussed the effects of saturation and flexibility on the performance of the system. Both of these effects limit the achievable performance of the closed-loop system.

Feedforward Control and Computed Torque

We next discussed the use of feedforward control as a method to track time varying reference trajectories such as the cubic polynomial trajectories that we derived in Chapter 7. A feedforward control scheme consists of adding a feedforward path from the reference signal to the control signal with transfer function $F(s)$. We showed that choosing $F(s)$ as the inverse of the forward plant allows tracking of arbitrary reference trajectories, provided that the forward plant is minimum phase.

We also introduced the notion of **computed torque control**, which is a feedforward disturbance cancellation method based on a computation of the nonlinear Euler-Lagrangian equations of motion. In the deal case the computed torque control, in effect, cancels the nonlinear dynamic coupling among the degrees of freedom of the manipulator and enhances the effectiveness of the linear feedback control methods in this chapter.

Joint Flexibility and State Space Methods

Next, we considered the effect of drive train dynamics in more detail. We derived a simple model for a single link system that included the joint elasticity and showed the limitations of PD control for this case. We then introduced state space control methods, which are much more powerful than the simple PD and PID control methods.

We introduced the fundamental notions of controllability and observability and showed that, if the state space model is both controllable and observable, we could design a linear control law to achieve any set of desired closed-loop poles. Specifically, given the linear system

$$\begin{aligned} \dot{x} &= Ax + bu \\ y &= c^T x \end{aligned}$$

then the state feedback control law $u = -k^T \hat{x}$ where $\hat{x}$ is the estimate of the state x computed from a linear observer

$$\dot{\hat{x}} = A\hat{x} + bu + \ell(y - c^T \hat{x})$$

results in the closed-loop system (in terms of the state x and estimation error $e = x - \hat{x}$)

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - bk^T & bk^T \\ 0 & A - \ell c^T \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}$$

The set of closed-loop poles of the system will therefore consist of the union of the eigenvalues of $A - \ell c^T$ and the eigenvalues of $A - bk^T$, a result known as the separation principle.

We also introduced the notion of linear quadratic optimal control and showed that the control

$$u = -k_*^T x$$

where

$$k_* = \frac{1}{R} b^T P$$

and P is the (unique) symmetric, positive definite $n \times n$ matrix satisfying the so-called matrix algebraic Riccati equation

$$A^T P + PA - \frac{1}{R} Pbb^T P + Q = 0$$

not only stabilizes the system but minimizes the quadratic performance measure

$$J = \int_0^\infty \{x^T(t) Q x(t) + R u^2(t)\} dt$$

Using the duality between control and observation, we showed that the observer gain ℓ could likewise be chosen as

$$\ell = \frac{1}{r} Pc$$

where P satisfies

$$AP + PA - \frac{1}{r} Pcc^T P + Q = 0$$

Problems

8–1 Using block diagram reduction techniques derive the transfer functions given by Equations (8.11) and (8.12).

8–2 Derive the transfer functions for the reduced-order model given by Equations (8.13) and (8.14).

8–3 Derive Equations (8.35) and (8.36).

8–4 Verify the computation of the closed-loop compensators in Table 8.1.

8–5 Verify the expression given by Equation (8.37) for the tracking error for the system in Figure 8.11. State the Final Value Theorem and use it to show that the steady-state error e_{ss} is indeed given by Equation (8.40).

8–6 Derive Equations (8.45) and (8.46).

8–7 Derive the inequality (8.47) using the Routh–Hurwitz criterion.

8–8 For the system of Figure 8.14 investigate the effect of saturation with various values of the PID gains and disturbance magnitude.

8–9 Verify Equation (8.49).

8–10 Verify Equation (8.52).

8–11 Derive Equations (8.63), (8.64), and (8.65).

8–12 Given the state space model defined by Equation (8.68) show that the transfer function

$$G(s) \quad = \quad c^T(sI - A)^{-1}b$$

is identical to Equation (8.63).

8–13 Search the control literature (for example, [71]) and find two or more algorithms for the pole assignment problem for linear systems.

8–14 Derive Equations (8.77) and (8.85).

8–15 Search the control literature to find out what is meant by **integrator windup**. Find out what is meant by **anti-windup** (or anti-reset windup). Simulate a PID control with anti-reset windup for the system of Figure 8.14. Compare the response with and without anti-reset windup.

8–16 Include the dynamics of a permanent magnet DC motor for the system given by Equations (8.57) and (8.58). What can you say now about controllability and observability of the system?

8–17 Choose appropriate state variables and write the system Equations (8.9) and (8.10) in state space form. What is the dimension of the state space?

8–18 Suppose in the flexible joint system represented by Equations (8.57) and (8.58) the following parameters are given

$$J_\ell = 10 \quad B_\ell = 1 \quad k = 100$$
$$J_m = 2 \quad B_m = 0.5$$

(a) Sketch the open-loop poles of the transfer functions given by Equation (8.63).

(b) Apply a PD compensator to the system (8.63). Sketch the root locus for the system. Choose a reasonable location for the compensator zero. Using the Routh criterion, find the value of the compensator gain when the root locus crosses the imaginary axis.

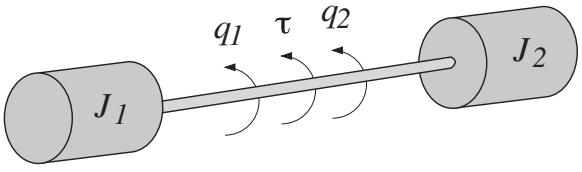


Figure 8.28: Coupled inertias in free space.

8–19 One of the problems encountered in space applications of robots is the fact that the base of the robot cannot be anchored, that is, cannot be fixed in an inertial coordinate frame. Consider the idealized situation shown in Figure 8.28, consisting of an inertia J_1 connected to the rotor of a motor whose stator is connected to an inertia J_2.

For example, J_1 could represent the space shuttle robot arm and J_2 the inertia of the shuttle itself. The simplified equations of motion are thus

$$\begin{aligned} J_1\ddot{q}_1 &= \tau \\ J_2\ddot{q}_2 &= \tau \end{aligned}$$

Write this system in state space form and show that it is uncontrollable. Discuss the implications of this and suggest possible solutions.

8–20 Given the linear second-order system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & -3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \end{bmatrix} u$$

find a linear state feedback control $u = k_1x_1 + k_2x_2$ so that the closed-loop system has poles at $s = -2, 2$.

8–21 Consider the block diagram of Figure 8.17. Suppose that $G(s) = 1/(2s^2 + s)$ and suppose that it is desired to track a reference signal $\theta^d(t) = \sin(t) + \cos(2t)$. If we further specify that the closed-loop system should have a natural frequency less than 10 radians with a damping ratio greater than 0.707, compute an appropriate compensator $C(s)$ and feedforward transfer function $F(s)$.

Notes and References

Although we treated only the dynamics of permanent magnet DC motors, the use of AC motors is increasing in robotics and other types of motion

control applications. AC motors do not require commutators and brushes and so are inherently more maintenance free and reliable. However, they are more difficult to control and require more sophisticated power electronics. With recent advances in power electronics together with their decreasing cost, AC motors may soon replace DC motors as the dominant actuation method for robot manipulators. A reference that treats different types of motors is [57].

A good background text on linear control systems is [84]. For a text that treats PID control in depth, consult [8].

The pole assignment theorem is due to Wonham [185]. The notion of controllability and observability, introduced by Kalman in [72], arises in several fundamental ways in addition to those discussed here. The interested reader should consult various references on the Kalman filter, which is a linear state estimator for systems whose output measurements are corrupted by (stochastic, white) noise. The linear observer that we discuss here was introduced by Luenberger [103] and is often referred to as the deterministic Kalman filter. Luenberger's main contribution in [103] was in the so-called reduced-order observer, which allows one to reduce the dimension of the observer.

Linear quadratic optimal control, in its present form, was introduced by Kalman in [72], where the importance of the Riccati equation was emphasized. The field of linear control theory is broad and there are many other techniques available to design state-feedback and output-feedback control laws in addition to the basic optimal control approach considered here. More recent control system design methods include the H_∞ approach [36] as well as approaches based on fuzzy logic [133] and neural networks [94] and [52].

The problem of drive-train dynamics in robotics was first pointed out by Good and Sweet, who studied the dynamics of the General Electric P-50 robot (see [172]). For this and other early robots the limiting factors to performance were current limiters that limited how much current could be drawn by the motors and elasticity in the joints due to gear flexibility. Both effects limit the maximum attainable safe speed at which the robot can operate. This work stimulated considerable research into the control of robots with input constraints (see [169]) and the control of robots with flexible joints (see [165]).

Finally, virtually all robot control systems today are implemented digitally. A treatment of digital control requires consideration of issues of sampling, quantization, resolution, as well as computer architecture, real-time programming and other issues that are not considered in this chapter. The interested reader should consult, for example, [48] for these latter subjects.

Chapter 9

NONLINEAR AND MULTIVARIABLE CONTROL

9.1 Introduction

In Chapter 8 we discussed techniques to derive control laws for each joint of a manipulator based on a single-input/single-output model. Coupling effects among the joints were regarded as disturbances to the individual systems. In reality, the dynamic equations of a robot manipulator form a complex, nonlinear, and multivariable system. In this chapter, therefore, we treat the robot control problem in the context of nonlinear, multivariable control. This approach allows us to provide more rigorous analysis of the performance of control systems, and also allows us to design robust and adaptive nonlinear control laws that guarantee stability and tracking of planned trajectories.

We first reformulate the manipulator dynamic equations in a form more suitable for the discussion to follow. We then treat the problem of inverse dynamics in both joint space and task space. The inverse dynamics control relies on exact cancellation of nonlinearities in the system, which is not possible in practical applications. For this reason, we discuss several methods for robust and adaptive control, which are aimed at providing good tracking performance despite uncertainty in knowledge of parameters or external disturbances. In distinguishing between robust control and adaptive control we follow the commonly accepted notion that a robust controller is a fixed

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

controller designed to satisfy performance specifications over a given range of uncertainties, whereas an adaptive controller incorporates some sort of online parameter estimation. This distinction is important. For example, in a repetitive motion task, the tracking errors produced by a fixed robust controller would tend to be repetitive as well, whereas tracking errors produced by an adaptive controller might be expected to decrease over time as the plant and/or control parameters are updated based on runtime information. At the same time, adaptive controllers that perform well in the face of parametric uncertainty may not perform well in the face of other types of uncertainty such as external disturbances or unmodeled dynamics. Therefore, an understanding of the trade-offs involved is important in deciding whether to employ robust or adaptive control design methods in a given situation. The final section outlines a torque optimization procedure to modify any of the controllers in this chapter to account for actuator saturation.

Dynamics Revisited

Recall the robot equations of motion given by Equation (6.65) and the actuator dynamics given by Equation (8.15)

$$\sum_{j=1}^{n} d_{kj}(q)\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k = \tau_k \tag{9.1}$$

$$J_{m_k}\ddot{\theta}_{m_k} + B_k\dot{\theta}_{m_k} = K_{m_k}/R_kV_k - \tau_k/r_k \tag{9.2}$$

for $k = 1, \ldots, n$, where $B_k = B_{m_k} + K_{b_k}K_{m_k}/R_k$. Multiplying Equation (9.2) by the gear ratio r_k and using the fact that the motor angles θ_{m_k} and the link angles q_k are related by

$$\theta_{m_k} = r_kq_k \tag{9.3}$$

we may write Equation (9.2) as

$$r_k^2J_m\ddot{q}_k + r_k^2B_k\dot{q}_k = r_kK_{m_k}/RV_k - \tau_k \tag{9.4}$$

Substituting Equation (9.4) into Equation (9.1) yields

$$r_k^2J_{m_k}\ddot{q}_k + \sum_{j=1}^{n} d_{kj}\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}\dot{q}_i\dot{q}_j + r_k^2B_k\dot{q}_k + g_k = r_k\frac{K_m}{R}V_k \tag{9.5}$$

for $k = 1, \ldots, n$. In matrix form Equation (9.5) can be written as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = u \tag{9.6}$$

where $M(q) = D(q) + J$ with J a diagonal matrix with diagonal elements $r_k^2 J_{m_k}$. The gravity vector $g(q)$ and the matrix $C(q, \dot{q})$ defining the Coriolis and centrifugal generalized forces are defined as before in Equations (6.67) and (6.68). The input vector u has components

$$u_k = r_k \frac{K_{m_k}}{R_k} V_k \ , \text{ for } k = 1, \ldots, n$$

and has units of torque. Note that the above formulation with J assumed to be a diagonal matrix amounts to ignoring the rotation of the motor about axes other than the axis of the joint to which it is attached. However, these off-diagonal effects are typically negligible.

Henceforth, we will ignore the friction and use Equation (9.6) with $B = 0$ as the plant model for our subsequent development. We leave it as an exercise for the reader (Problem 9–1) to show that the properties of passivity, skew symmetry, bounds on the inertia matrix, and linearity in the parameters continue to hold for the system (9.6).

9.2 PD Control Revisited

It is a rather remarkable fact that the simple PD control scheme that we discussed in Chapter 8 for the set-point control of a single-DOF model can be rigorously shown to work in the general case of Equation (9.6).[1] An independent joint PD control scheme can be written in vector form as

$$u = -K_P \tilde{q} - K_D \dot{q} \tag{9.7}$$

where $\tilde{q} = q - q^d$ is the error between the joint vector q and the desired joint vector q^d (the set point), and K_P, K_D are diagonal matrices of (positive) proportional and derivative gains, respectively. We first show that, in the absence of gravity, that is, if $g(q)$ is zero in Equation (9.6), the PD control law given in Equation (9.7) achieves global asymptotic tracking of the desired joint positions. This, in effect, reproduces the result derived previously but is more rigorous, in the sense that the nonlinear coupling terms are not approximated by a constant disturbance.

To show that the control law given in Equation (9.7) achieves asymptotic tracking consider the Lyapunov function candidate

$$V = 1/2 \dot{q}^T M(q) \dot{q} + 1/2 \tilde{q}^T K_P \tilde{q} \tag{9.8}$$

[1] The reader should review the discussion on Lyapunov stability in Appendix C.

The first term in Equation (9.8) is the kinetic energy of the robot and the second term accounts for the proportional feedback $K_P\tilde{q}$. Note that V represents the total energy that would result if the joint actuators were replaced by springs with stiffness constants represented by K_P and with equilibrium positions at q^d. Thus, V is positive except at the "goal" configuration $q = q^d$, $\dot{q} = 0$, at which point V is zero. The idea is to show that along any motion of the robot, the function V is decreasing to zero. This will imply that the robot is moving toward the desired goal configuration.

To show this we note that, since q^d is constant, the time derivative of V is given by

$$\dot{V} = \dot{q}^T M(q)\ddot{q} + 1/2\dot{q}^T\dot{M}(q)\dot{q} + \dot{q}^T K_P\tilde{q} \tag{9.9}$$

Solving for $M(q)\ddot{q}$ in Equation (9.6) with $g(q) = 0$ and substituting the resulting expression into Equation (9.9) yields

$$\begin{aligned} \dot{V} &= \dot{q}^T(u - C(q,\dot{q})\dot{q}) + 1/2\dot{q}^T\dot{M}(q)\dot{q} + \dot{q}^T K_P\tilde{q} \\ &= \dot{q}^T(u + K_P\tilde{q}) + 1/2\dot{q}^T(\dot{M}(q) - 2C(q,\dot{q}))\dot{q} \\ &= \dot{q}^T(u + K_P\tilde{q}) \end{aligned} \tag{9.10}$$

where in the last equality we have used the fact that $\dot{M} - 2C$ is skew symmetric. Substituting the PD control law (9.7) for u into the above yields

$$\dot{V} = -\dot{q}^T K_D\dot{q} \leq 0 \tag{9.11}$$

The above analysis shows that V is decreasing as long as $\dot{q}$ is not zero. This by itself is not enough to prove the desired result since it is conceivable that the manipulator could reach a position where $\dot{q} = 0$ but $q \neq q^d$. To show that this cannot happen we can use LaSalle's theorem (Appendix C). Suppose $\dot{V} \equiv 0$.[2] Then Equation (9.11) implies that $\dot{q} \equiv 0$ and hence $\ddot{q} \equiv 0$. From the equations of motion with PD control

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = -K_P\tilde{q} - K_D\dot{q}$$

we must then have

$$0 = -K_P\tilde{q}$$

which implies that $\tilde{q} = 0$. LaSalle's theorem then implies that the equilibrium is globally asymptotically stable.

[2]The notation $\dot{V} \equiv 0$ means that the expression is **identically** equal to zero, not simply zero at one instant of time.

In case there are gravitational terms present in Equation (9.6), then Equation (9.10) becomes

$$\dot{V} = \dot{q}^T(u - g(q) + K_P\tilde{q}) \tag{9.12}$$

The presence of the gravitational term in Equation (9.12) means that PD control alone cannot guarantee asymptotic tracking. In practice there will be a steady-state error or offset. Assuming that the closed-loop system is stable and that the joint velocity $\dot{q}$ goes to zero, the steady-state robot configuration q will satisfy

$$K_P(q - q^d) = g(q) \tag{9.13}$$

The physical interpretation of Equation (9.13) is that the configuration q must be such that the motor generates a steady-state "holding torque" $K_P(q - q^d)$ sufficient to balance the gravitational torque $g(q)$. Thus, we see that the steady-state error can be reduced by increasing the position gain K_P.

In order to remove this steady state error we can modify the PD control law as

$$u = -K_P\tilde{q} - K_D\dot{q} + g(q) \tag{9.14}$$

The modified control law given by Equation (9.14), in effect, cancels the gravitational terms and we achieve the same Equation (9.11) as before. The control law given by Equation (9.14) requires the computation of the gravitational terms $g(q)$ from the Lagrangian equations at each instant of time. If these terms are uncertain, then the control law (9.14) cannot be computed precisely. We will say more about this and related issues later in the context of robust and adaptive control.

The Effect of Joint Flexibility

In Chapter 8 we considered the effect of joint flexibility and showed for a lumped model of a single-link robot that a PD control could be designed for set-point tracking. In this section we will discuss the analogous result in the general case of an n-link manipulator.

We first derive a model similar to Equation (9.6) to represent the dynamics of an n-link robot with joint flexibility. For simplicity, assume that the joints are revolute and are actuated by permanent magnet DC motors. We model the flexibility of the i^{th} joint as a linear torsional spring with stiffness constant k_i, for $i = 1, \ldots, n$.

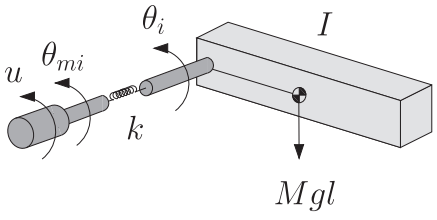


Figure 9.1: A single link of a flexible-joint manipulator. The joint elasticity is represented by a torsional spring between the link angle θ_i and the motor shaft angle θ_{mi}.

Referring to Figure 9.1, we let $q_1 = [\theta_1, \cdots, \theta_n]^T$ be the vector of DH-joint variables and $q_2 = [\frac{1}{r_{m_1}}\theta_{m_1}, \cdots, \frac{1}{r_{m_n}}\theta_{m_n}]^T$ be the vector of motor shaft angles (reflected to the link side of the gears). Then $q_1 - q_2$ is the vector of elastic joint deflections. Neglecting the effect of link motion on the kinetic energy of the actuators, as we did in Equation (9.6), the kinetic and potential energies of the manipulator are given by

$$\mathcal{K} = \frac{1}{2}\dot{q}_1^T D(q_1)\dot{q}_1 + \frac{1}{2}\dot{q}_2^T J\dot{q}_2 \tag{9.15}$$

$$\mathcal{P} = P(q_1) + \frac{1}{2}(q_1 - q_2)^T K(q_1 - q_2) \tag{9.16}$$

where J and K are diagonal matrices of inertia and stiffness constants, respectively.

$$J = \begin{bmatrix} J_1 & 0 & \cdots & 0 \\ 0 & J_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & J_n \end{bmatrix}, \quad K = \begin{bmatrix} k_1 & 0 & \cdots & 0 \\ 0 & k_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & k_n \end{bmatrix} \tag{9.17}$$

Note that we have simply augmented the kinetic energy of the rigid joint robot with the kinetic energy of the actuators and we have likewise augmented the potential energy due to gravity of the rigid joint model with the elastic potential energy of the linear springs at the joints. It is now straightforward to compute the Euler–Lagrange equations for this system as (Problem 9–2)

$$\begin{aligned} D(q_1)\ddot{q}_1 + C(q_1, \dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 + K(q_2 - q_1) &= u \end{aligned} \tag{9.18}$$

For the problem of set-point tracking with PD control, consider a control law of the form

$$u = -K_P\tilde{q}_2 - K_D\dot{q}_2 \tag{9.19}$$

where $\tilde{q}_2 = q_2 - q^d$ and q^d is a vector of constant set points, and suppose again that the gravity vector $g(q_1) = 0$. To show asymptotic tracking for the closed-loop system consider the Lyapunov function candidate

$$V = \frac{1}{2}\dot{q}_1^T D(q_1)\dot{q}_1 + \frac{1}{2}\dot{q}_2^T J\dot{q}_2 + \frac{1}{2}(q_1 - q_2)^T K(q_1 - q_2) + \frac{1}{2}\tilde{q}_2^T K_P\tilde{q}_2 \tag{9.20}$$

We leave it as an exercise (Problem 9–3) to show that an application of LaSalle's theorem proves global asymptotic stability of the system. Consequently, the motor and link angles are equal in the steady state. Thus, one may choose the set point q_2^d as a vector of desired DH variables.

If gravity is present, then it is not apparent how one can achieve gravity compensation in a manner similar to the rigid joint case. We will address this question later in the context of feedback linearization in Chapter 12.

9.3 Inverse Dynamics

We now consider the application of more complex nonlinear control techniques for trajectory tracking of rigid manipulators. The first algorithm that we consider is known as the method of **inverse dynamics**. The method of inverse dynamics is, as we shall see in Chapter 12, a special case of the method of **feedback linearization**.

After presenting the basic idea of inverse dynamics, both in joint space and in task space, we will discuss the practical situation of uncertainty in the parameters defining the manipulator dynamics. The problem of parametric uncertainty naturally leads to a discussion of robust and adaptive control, which we will discuss in the remaining sections of this chapter.

9.3.1 Joint Space Inverse Dynamics

Consider again the dynamic equations of an n-link rigid robot in matrix form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \;=\; u \tag{9.21}$$

The idea of inverse dynamics is to seek a nonlinear feedback control law

$$u \;=\; f(q,\dot{q},t) \tag{9.22}$$

which, when substituted into Equation (9.21), results in a linear closed-loop system. For general nonlinear systems, such a control law may be quite difficult or impossible to find. In the case of the manipulator dynamics given by Equations (9.21), however, the problem is actually easy. By inspecting Equation (9.21) we see that if we choose the control u according to the equation

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q) \tag{9.23}$$

then, since the inertia matrix M is invertible, the combined system given by Equations (9.21)–(9.23) reduces to

$$\ddot{q} = a_q \tag{9.24}$$

The term a_q represents a new input that is yet to be chosen. Equation (9.24) is known as the **double integrator system** as it represents n uncoupled double integrators. Equation (9.23) is called the **inverse dynamics control** and achieves a rather remarkable result, namely that the system given by Equation (9.24) is linear and decoupled. This means that each input a_{q_k} can be designed to control a SISO linear system. Moreover, assuming that a_{q_k} is a function only of q_k and $\dot{q}_k$, then the closed-loop system will be decoupled.

Since a_q can now be designed to control a linear second-order system, an obvious choice is to set

$$a_q = \ddot{q}^d(t) - K_0\tilde{q} - K_1\dot{\tilde{q}} \tag{9.25}$$

where $\tilde{q} = q - q^d$, $\dot{\tilde{q}} = \dot{q} - \dot{q}^d$, K_0, K_1 are diagonal matrices with diagonal elements consisting of position and velocity gains, respectively and the reference trajectory

$$t \rightarrow (q^d(t), \dot{q}^d(t), \ddot{q}^d(t)) \tag{9.26}$$

defines the desired time history of joint positions, velocities, and accelerations. Note that Equation (9.25) is nothing more than a PD control with feedforward acceleration as defined in Chapter 8.

Substituting Equation (9.25) into Equation (9.24), results in

$$\ddot{\tilde{q}}(t) + K_1\dot{\tilde{q}}(t) + K_0\tilde{q}(t) = 0 \tag{9.27}$$

A simple choice for the gain matrices K_0 and K_1 is

$$K_0 = \begin{bmatrix} \omega_1^2 & 0 & \cdots & 0 \\ 0 & \omega_2^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_n^2 \end{bmatrix}, \quad K_1 = \begin{bmatrix} 2\omega_1 & 0 & \cdots & 0 \\ 0 & 2\omega_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 2\omega_n \end{bmatrix} \tag{9.28}$$

which results in a decoupled closed-loop system with each joint response equal to the response of a critically damped linear second-order system with natural frequency ω_i. As before, the natural frequency ω_i determines the speed of response of the joint, or equivalently, the rate of decay of the tracking error.

The inverse dynamics approach is important as a basis for control and it is worthwhile to examine it from alternative viewpoints. We can give a second interpretation of the control law (9.23) as follows. Consider again the manipulator dynamics (9.21). Since $M(q)$ is invertible for all $q \in \mathbb{R}^n$, we may solve for the acceleration $\ddot{q}$ of the manipulator as

$$\ddot{q} \quad = \quad M^{-1}\{u - C(q,\dot{q})\dot{q} - g(q)\} \tag{9.29}$$

Suppose we were able to specify the acceleration as the input to the system. That is, suppose we had actuators capable of producing directly a commanded acceleration (rather than indirectly by producing a force or torque). Then the dynamics of the manipulator, which is after all a position control device, would be given as

$$\ddot{q} \quad = \quad a_q \tag{9.30}$$

where $a_q(t)$ is the input acceleration vector. This is again the familiar double integrator system. Note that Equation (9.30) is not an approximation in any sense; rather it represents the actual open-loop dynamics of the system provided that the acceleration is chosen as the input. The control problem for the system (9.30) is now easy and the acceleration input a_q can be chosen as before according to Equation (9.25).

In reality, however, such "acceleration actuators" are not available to us and we must be content with the ability to produce a generalized force (torque) u_i at each joint i. Comparing Equations (9.29) and (9.30) we see that the torque input u and the acceleration input a_q of the manipulator are related by

$$M^{-1}\{u - C(q,\dot{q})\dot{q} - g(q)\} \quad = \quad a_q \tag{9.31}$$

Solving Equation (9.31) for the input torque $u(t)$ yields

$$u \quad = \quad M(q)a_q + C(q,\dot{q})\dot{q} + g(q) \tag{9.32}$$

which is the same as the previously derived expression (9.23). Thus, the inverse dynamics can be viewed as an input transformation that transforms the problem from one of choosing torque input commands to one of choosing acceleration input commands.

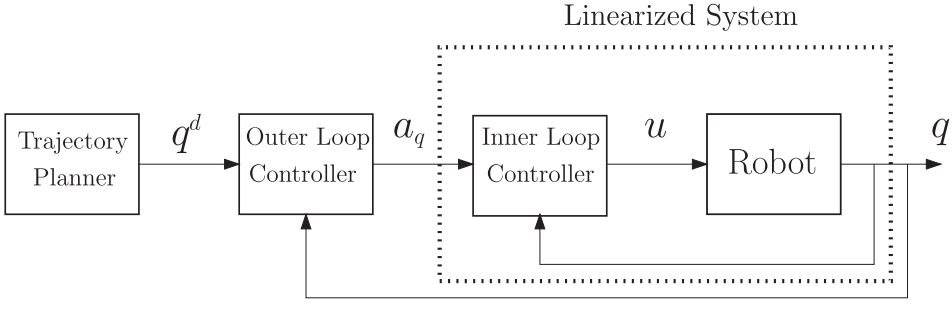


Figure 9.2: Inner-loop/outer-loop control architecture. The inner-loop control computes the vector u of input torques as a function of the measured joint positions and velocities and the given outer-loop control in order to compensate the nonlinearities in the plant model. The outer-loop control designed to track a given reference trajectory can then be based on a linear and decoupled plant model.

Note that the implementation of this control scheme requires the real-time computation of the inertia matrix and the vectors of Coriolis, centrifugal, and gravitational generalized forces. An important issue therefore in the control system implementation is the design of the computer architecture for the above computations.

Figure 9.2 illustrates the so-called **inner-loop/outer-loop** control architecture. By this we mean that the computation of the nonlinear control law (9.23) is performed in an inner loop with the vectors $q, \dot{q}$, and a_q as its inputs and u as output. The outer loop in the system is then the computation of the additional input term a_q. Note that the outer-loop control a_q is more in line with the notion of a feedback control in the usual sense of being error driven. The design of the outer-loop feedback control is in theory greatly simplified since it is designed for the plant represented by the dotted lines in Figure 9.2, which is now a linear system.

9.3.2 Task Space Inverse Dynamics

As an illustration of the importance of the inner-loop/outer-loop control architecture, we will show that tracking in task space can be achieved by modifying the outer-loop control a_q in Equation (9.24) while leaving the inner-loop control unchanged. Let $X \in \mathbb{R}^6$ represent the end-effector pose using any minimal representation of $SO(3)$. Since X is a function of the

joint variables $q \in \mathcal{Q}$, we have

$$\dot{X} = J(q)\dot{q} \tag{9.33}$$
$$\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{9.34}$$

where $J = J_a$ is the analytical Jacobian of Equation (4.88). Given the double integrator system (9.24) in joint space we see that if a_q is chosen as

$$a_q = J^{-1}\left\{a_X - \dot{J}\dot{q}\right\} \tag{9.35}$$

the result is a double integrator system in task space coordinates

$$\ddot{X} = a_X \tag{9.36}$$

Given a task space trajectory $X^d(t)$ satisfying the same smoothness and boundedness assumptions as the joint space trajectory $q^d(t)$, we may choose a_X as

$$a_X = \ddot{X}^d - K_0(X - X^d) - K_1(\dot{X} - \dot{X}^d) \tag{9.37}$$

so that the task space tracking error $\tilde{X} = X - X^d$ satisfies

$$\ddot{\tilde{X}} + K_1\dot{\tilde{X}} + K_0\tilde{X} = 0 \tag{9.38}$$

Therefore, a modification of the outer-loop control term a_q achieves a linear and decoupled system directly in the task space coordinates without the need to compute a joint space trajectory and without the need to modify the nonlinear inner-loop control.

Note that we have used a minimal representation for the orientation of the end effector in order to specify a trajectory $X \in \mathbb{R}^6$. In general, if the end-effector coordinates are given in $SE(3)$, then the Jacobian J in the above formulation will be the geometric Jacobian of Equation (4.41). In this case

$$\dot{X} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \omega \end{bmatrix} = J(q)\dot{q} \tag{9.39}$$

and the outer-loop control

$$a_q = J^{-1}(q)\left\{\begin{bmatrix} a_x \\ a_\omega \end{bmatrix} - \dot{J}(q)\dot{q}\right\} \tag{9.40}$$

applied to Equation (9.24) results in the system

$$\ddot{x} = a_x \in \mathbb{R}^3 \tag{9.41}$$
$$\dot{\omega} = a_\omega \in \mathbb{R}^3 \tag{9.42}$$
$$\dot{R} = S(\omega)R, \quad R \in SO(3), \; S \in so(3) \tag{9.43}$$

Although, in this latter case, the dynamics have not been linearized to a double integrator, the outer-loop terms a_v and a_ω may still be used to define control laws to track end-effector trajectories in $SE(3)$.

In both cases we see that nonsingularity of the Jacobian is necessary to implement the outer-loop control. Avoiding singularities, therefore, is an important aspect of motion planning and trajectory planning that we have considered in Chapter 7. Also, if the robot has more or fewer than six joints, then the Jacobians are not square. In this case, other schemes have been developed using, for example, the Jacobian pseudoinverse. See [32] for details.

9.3.3 Robust Inverse Dynamics

The inverse dynamics approach relies on exact cancellation of nonlinearities in the robot equations of motion. The practical implementation of inverse dynamics control requires consideration of various sources of uncertainties such as modeling errors, unknown loads, and computation errors. Let us return to the Euler–Lagrange equations of motion

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{9.44}$$

and write the inverse dynamics control input u as

$$u = \hat{M}(q)a_q + \hat{C}(q,\dot{q})\dot{q} + \hat{g}(q) \tag{9.45}$$

where the notation $\hat{(\cdot)}$ represents the computed or nominal value of $(\cdot)$ and indicates that the theoretically exact inverse dynamics control is not achievable in practice due to the uncertainties in the system. The error or mismatch $\tilde{(\cdot)} = \hat{(\cdot)} - (\cdot)$ is a measure of one's knowledge of the system parameters.

If we substitute Equation (9.45) into Equation (9.44) we obtain, after some algebra (Problem 9–8),

$$\ddot{q} = a_q + \eta(q,\dot{q},a_q) \tag{9.46}$$

where the nonlinear function $\eta(q,\dot{q},a_q)$ is given by

$$\eta(q,\dot{q},a_q) \quad = \quad M^{-1}(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{g}) \tag{9.47}$$

and is called the **uncertainty**. Henceforth, we will drop the arguments of $\eta(q,\dot{q},a_q)$ for simplicity. However, it is important to keep in mind that η is, in general, a function of both the state and the reference trajectory.

We define $E = E(q)$ as

$$E := M^{-1}\tilde{M} = M^{-1}\hat{M} - I \tag{9.48}$$

which allows us to express the uncertainty η as

$$\eta = Ea_q + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \tag{9.49}$$

The system (9.46) is still nonlinear and coupled due to the uncertainty $\eta(q, \dot{q}, a_q)$ and, therefore, we have no guarantee that the outer-loop control given by Equation (9.25) will satisfy desired tracking performance specifications.

In this section we show how to modify the outer-loop control (9.25) to guarantee global convergence of the tracking error for the system (9.46). The method we outline below is often called **Lyapunov redesign**. In this approach we set the outer-loop control a_q as

$$a_q = \ddot{q}^d(t) - K_0\tilde{q} - K_1\dot{\tilde{q}} + \delta a \tag{9.50}$$

where δa is an additional term to be designed. In terms of the tracking error

$$e = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix} = \begin{bmatrix} q - q^d \\ \dot{q} - \dot{q}^d \end{bmatrix} \tag{9.51}$$

we may write Equations (9.46) and (9.50) as

$$\dot{e} = Ae + B\{\delta a + \eta\} \tag{9.52}$$

where

$$A = \begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{9.53}$$

Thus, the double integrator is first stabilized by the linear feedback term $-K_0\tilde{q} - K_1\dot{\tilde{q}}$, and the additional control term δa should be designed to overcome the potentially destabilizing effect of the uncertainty η. The basic idea is to assume that we are able to compute a bound $\rho(e, t) \geq 0$ on the uncertainty η as

$$\|\eta\| \leq \rho(e, t) \tag{9.54}$$

and design the additional input term δa to guarantee ultimate boundedness of the error trajectory $e(t)$ in Equation (9.52). Note that the bound ρ is in general a function of the tracking error e and time reflecting the fact that η likewise depends on e as well as on t through a time-varying reference trajectory.

Returning to our expression for the uncertainty η and substituting for a_q from Equation (9.50) we have

$$\begin{aligned} \eta &= Ea_q + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \\ &= E\delta a + E(\ddot{q}^d - K_0\tilde{q} - K_1\dot{\tilde{q}}) + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \end{aligned} \tag{9.55}$$

Let us assume that we can find constants $\alpha < 1$, γ_1, and γ_2, together with possibly time-varying γ_3 such that

$$\|\eta\| \leq \alpha\|\delta a\| + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3(t) \tag{9.56}$$

We note that the condition $\alpha := \|E\| = \|M^{-1}\hat{M} - I\| < 1$ determines how close our estimate $\hat{M}$ must be to the true inertia matrix. Suppose that we have constant bounds on M^{-1} as

$$0 < \underline{\mathrm{M}} \leq \|M^{-1}(q)\| \leq \overline{M} < \infty \tag{9.57}$$

Remark 9.1. *Such constant bounds exist for all robots with revolute joints since the configuration space is compact. The inertia matrices of robots with both revolute and prismatic joints may or may not have constant bounds as in (9.57). See [53] for a complete characterization of manipulator designs satisfying (9.57).*

If we choose the estimated inertia matrix $\hat{M}$ as

$$\hat{M} = \frac{2}{\overline{M} + \underline{\mathrm{M}}} I \tag{9.58}$$

then it can be shown that

$$\|M^{-1}\hat{M} - I\| \leq \frac{\overline{M} - \underline{\mathrm{M}}}{\overline{M} + \underline{\mathrm{M}}} < 1 \tag{9.59}$$

The point is that there is always a choice for $\hat{M}$ that satisfies the condition $\|E\| < 1$ provided upper and lower bounds $\overline{M}$ and $\underline{\mathrm{M}}$ can be determined.

Next, assume, for the moment, that $\|\delta a\| \leq \rho(e,t)$ which must then be checked a posteriori. It follows that

$$\|\eta\| \leq \alpha\rho(e,t) + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3 =: \rho(e,t) \tag{9.60}$$

which, since $\alpha < 1$, defines ρ as

$$\rho(e,t) = \frac{1}{1-\alpha}(\gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3) \tag{9.61}$$

Since K_0 and K_1 are chosen so that the matrix A in Equation (9.52) is Hurwitz, we may choose $Q > 0$ and let $P > 0$ be the unique symmetric positive definite matrix satisfying the Lyapunov equation

$$A^T P + PA = -Q \tag{9.62}$$

Defining the control δa according to

$$\delta a = \begin{cases} -\rho(e,t)\dfrac{B^T P e}{\|B^T P e\|} & ; \quad \text{if} \quad \|B^T P e\| \neq 0 \\ \\ 0 & ; \quad \text{if} \quad \|B^T P e\| = 0 \end{cases} \tag{9.63}$$

it follows that the Lyapunov function

$$V = e^T P e \tag{9.64}$$

satisfies $\dot{V} < 0$ along solution trajectories of Equation (9.52). To show this result, we compute

$$\dot{V} \quad = \quad -e^T Q e + 2e^T P B\{\delta a + \eta\} \tag{9.65}$$

For simplicity set $w = B^T P e$ and consider the second term $w^T\{\delta a + \eta\}$ in the above expression. If $w = 0$ this term vanishes, and hence,

$$\dot{V} = -e^T Q e < 0 \tag{9.66}$$

independent of the choice of δa. For $w \neq 0$ we have

$$\delta a = -\rho \frac{w}{\|w\|} \tag{9.67}$$

and hence, using the Cauchy–Schwartz inequality we have

$$\begin{aligned} w^T(-\rho\frac{w}{\|w\|} + \eta) &\leq -\rho\|w\| + \|w\|\ \|\eta\| \\ &= \|w\|(-\rho + \|\eta\|) \leq 0 \end{aligned} \tag{9.68}$$

since $\|\eta\| \leq \rho$. Therefore

$$\dot{V} \leq -e^T Q e < 0 \tag{9.69}$$

and the result follows. Finally, note from Equation (9.67) that $\|\delta a\| \leq \rho$ as required.

Since the above control term δa is discontinuous on the subspace defined by $B^T P e = 0$, solution trajectories on this subspace are not well defined in

the usual sense. One may define solutions in a generalized sense, the so-called **Filippov solutions** [46]. In practice, the discontinuity in the control results in the phenomenon of **chattering**, where the control switches rapidly between the control values in (9.63).

To avoid control chattering, we may implement a continuous approximation to the above discontinuous control as

$$\delta a = \begin{cases} -\rho(e,t)\dfrac{B^T P e}{\|B^T P e\|} & ; \quad \text{if} \quad \|B^T P e\| > \epsilon \\ \\ -\dfrac{\rho(e,t)}{\epsilon} B^T P e & ; \quad \text{if} \quad \|B^T P e\| \leq \epsilon \end{cases} \tag{9.70}$$

In this case, since the control signal given by Equation (9.70) is continuous and differentiable except at $\|B^T P e\|$, a solution to the system (9.52) exists for any initial condition and we can prove the following result.

Theorem 9.1. *All trajectories of the system (9.52) are uniformly ultimately bounded (u.u.b.) using the continuous control law (9.70). (See Appendix C for the definition of uniform ultimate boundedness.)*

Proof: As before, choose $V(e) = e^T P e$ and compute

$$\begin{aligned} \dot{V} &= -e^T Q e + 2w^T (\delta a + \eta) && (9.71) \\ &\leq -e^T Q e + 2w^T (\delta a + \rho \frac{w}{\|w\|}) && (9.72) \end{aligned}$$

with $\|w\| = \|B^T P e\|$ as above. For $\|w\| \geq \epsilon$ the argument proceeds as above and $\dot{V} < 0$. For $\|w\| \leq \epsilon$ the second term in (9.72) becomes

$$2w^T \left(-\frac{\rho}{\epsilon} w + \rho \frac{w}{\|w\|} \right) = -2\frac{\rho}{\epsilon}\|w\|^2 + 2\rho\|w\|$$

This expression attains a maximum value of $\epsilon\frac{\rho}{2}$ when $\|w\| = \frac{\epsilon}{2}$. Thus, we have

$$\dot{V} \leq -e^T Q e + \epsilon\frac{\rho}{2} < 0 \tag{9.73}$$

provided

$$e^T Q e > \epsilon\frac{\rho}{2} \tag{9.74}$$

Using the relationship (see B.5)

$$\lambda_{min}(Q)\|e\|^2 \leq e^T Q e \leq \lambda_{max}(Q)\|e\|^2 \tag{9.75}$$

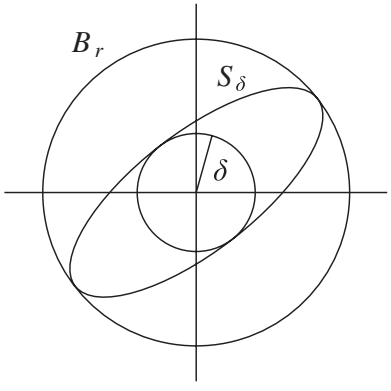


Figure 9.3: The uniform ultimate boundedness set. Since $\dot{V}$ is negative outside the ball B_δ, all trajectories will eventually enter the level set S_δ, the smallest level set of V containing B_δ. The system is thus u.u.b. with respect to B_r, the smallest ball containing S_δ.

from Equation (B.5), where $\lambda_{min}(Q)$, $\lambda_{max}(Q)$ denote the minimum and maximum eigenvalues, respectively, of the matrix Q, we have that $\dot{V} < 0$ if

$$\lambda_{min}(Q)\|e\|^2 > \epsilon\frac{\rho}{2} \tag{9.76}$$

or, equivalently

$$\|e\| > \left(\frac{\epsilon\rho}{2\lambda_{min}(Q)}\right)^{\frac{1}{2}} =: \delta \tag{9.77}$$

Let S_δ denote the smallest level set of V containing $B(\delta)$, the ball of radius δ and let B_r denote the smallest ball containing S_δ. Then all solutions of the closed-loop system are u.u.b. with respect to B_r. The situation is shown in Figure 9.3. All trajectories will eventually reach the boundary of S_δ since $\dot{V}$ is negative definite outside of S_δ and hence remain inside the ball B_r.

Note that the radius of the ultimate boundedness set, and hence, the magnitude of the steady-state tracking error, is proportional to the product of the uncertainty bound ρ and the constant ϵ.

9.3.4 Adaptive Inverse Dynamics

This section is concerned with the **adaptive** control problem in which an estimation scheme is used to generate estimated values of parameters that are then used in lieu of the true parameters.

Once the linear parametrization property for manipulators became widely known in the mid-1980's, the first globally convergent adaptive control re-

sults began to appear. These first results were based on the inverse dynamics approach discussed above. Consider the plant given by Equation (9.44) and the control given by Equation (9.45) as above, but now suppose that the parameters appearing in Equation (9.45) are not fixed as in the robust control approach, but are time-varying estimates of the true parameters. Substituting Equation (9.45) into Equation (9.44) and setting

$$a_q = \ddot{q}^d - K_1(\dot{q} - \dot{q}^d) - K_0(q - q^d) \tag{9.78}$$

it can be shown (Problem 9–11), using the linear parametrization property, that

$$\ddot{\tilde{q}} + K_1\dot{\tilde{q}} + K_0\tilde{q} = \hat{M}^{-1}Y(q, \dot{q}, \ddot{q})\tilde{\theta} \tag{9.79}$$

where $Y(q, \dot{q}, \ddot{q})$ is the regressor function defined in Equation (6.122) and $\tilde{\theta} = \hat{\theta} - \theta$, where $\hat{\theta}$ is the estimate of the parameter vector θ. In state space we write the system (9.79) as

$$\dot{e} = Ae + B\Phi\tilde{\theta} \tag{9.80}$$

where

$$A = \begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad \Phi = \hat{M}^{-1}Y(q, \dot{q}, \ddot{q}) \tag{9.81}$$

with K_0 and K_1 chosen as before as diagonal matrices of positive gains so that A is a Hurwitz matrix. Let P be the unique symmetric, positive definite matrix P satisfying the matrix Lyapunov equation

$$A^TP + PA = -Q \tag{9.82}$$

and choose the parameter update law as

$$\dot{\hat{\theta}} = -\Gamma^{-1}\Phi^TB^TPe \tag{9.83}$$

where Γ is a constant, symmetric, positive definite matrix. Then, global convergence to zero of the tracking error with all internal signals remaining bounded can be shown using the Lyapunov function

$$V = e^TPe + \tilde{\theta}^T\Gamma\tilde{\theta} \tag{9.84}$$

To see this we calculate $\dot{V}$ as (Problem 9–12)

$$\dot{V} = -e^TQe + 2\tilde{\theta}^T\{\Phi^TB^TPe + \Gamma\dot{\hat{\theta}}\} \tag{9.85}$$

the latter term following since θ is constant, that is, $\dot{\tilde{\theta}} = \dot{\hat{\theta}}$. Using the parameter update law (9.83) we have

$$\dot{V} = -e^TQe \leq 0 \tag{9.86}$$

Remark 9.2. *It is important to understand why the Lyapunov function V in Equation (9.86) is only negative semi-definite whereas V in Equation (9.69) is negative definite. In the robust approach the states of the system are $\tilde{q}$ and $\dot{\tilde{q}}$. In the adaptive control approach, the fact that $\tilde{\theta}$ satisfies the differential equation (9.83) means that the complete state vector now includes $\tilde{\theta}$ and the state equations are given by the coupled system, Equations (9.80) and (9.83). For this reason we included the positive definite term $\frac{1}{2}\tilde{\theta}^T\Gamma\tilde{\theta}$ in the Lyapunov function (9.84).*

From Equation (9.86) it follows that the position tracking errors converge to zero asymptotically and the parameter estimation errors $\tilde{\theta}$ remain bounded. We will not go through the details of the proof in this section. The argument is similar to that of the passivity-based, adaptive control approach of the next section so we will defer the details until later. In order to implement this adaptive inverse dynamics scheme, however, one notes that the acceleration $\ddot{q}$ is needed in the parameter update law and that $\hat{M}$ must be invertible. The need for the joint acceleration in the parameter update law presents a serious challenge to its implementation. Acceleration sensors are noisy and introduce additional cost whereas calculating the acceleration by numerical differentiation of position or velocity signals is not feasible in most cases. The invertibility of $\hat{M}$ can be enforced in the algorithm by resetting the parameter estimate whenever $\hat{\theta}$ would otherwise result in $\hat{M}$ becoming singular. The passivity-based approaches that we treat next remove both of these impediments.

9.4 Passivity-Based Control

The inverse dynamics based approaches in the previous section rely, in the ideal case, on exact cancellation of all nonlinearities in the system dynamics. In this section we discuss control techniques based on the passivity or skew symmetry property of the Euler–Lagrange equations. These methods do not rely on complete cancellation of nonlinearities and hence, do not lead to a linear closed-loop system even in the known-parameter case. However, as we shall see, the passivity-based methods have other advantages with respect to robust and adaptive control.

To motivate the discussion to follow, consider again the Euler–Lagrange equations

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{9.87}$$

and choose the control input according to

$$u = M(q)a + C(q,\dot{q})v + g(q) - Kr \tag{9.88}$$

where the quantities v, a, and r are given as

$$\begin{aligned} v &= \dot{q}^d - \Lambda\tilde{q} \\ a &= \dot{v} = \ddot{q}^d - \Lambda\dot{\tilde{q}} \\ r &= \dot{q} - v = \dot{\tilde{q}} + \Lambda\tilde{q} \end{aligned} \tag{9.89}$$

where K and Λ are diagonal matrices of constant, positive gains. Substituting the control law (9.88) into the plant model (9.87) leads to

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = 0 \tag{9.90}$$

Note that, in contrast to the inverse dynamics control approach, the closed-loop system (9.90) is still a coupled nonlinear system. Stability and asymptotic convergence of the tracking error to zero therefore are not obvious and require additional analysis.

Consider the Lyapunov function candidate

$$V = \frac{1}{2}r^T M(q)r + \tilde{q}^T\Lambda K\tilde{q} \tag{9.91}$$

Calculating $\dot{V}$ yields

$$\begin{aligned} \dot{V} &= r^T M\dot{r} + \tfrac{1}{2}r^T\dot{M}r + 2\tilde{q}^T\Lambda K\dot{\tilde{q}} \\ &= -r^T Kr + 2\tilde{q}^T\Lambda K\dot{\tilde{q}} + \tfrac{1}{2}r^T(\dot{M} - 2C)r \end{aligned} \tag{9.92}$$

Using the skew symmetry property and the definition of r, Equation (9.92) reduces to

$$\begin{aligned} \dot{V} &= -\tilde{q}^T\Lambda^T K\Lambda\tilde{q} - \dot{\tilde{q}}^T K\dot{\tilde{q}} \\ &= -e^T Qe \end{aligned} \tag{9.93}$$

where

$$Q = \begin{bmatrix} \Lambda^T K\Lambda & 0 \\ 0 & K \end{bmatrix} \tag{9.94}$$

Therefore the equilibrium $e = 0$ in error space is globally asymptotically stable.

In fact, assuming constant norm bounds on the inertia matrix $M(q)$, it is fairly straightforward to prove global exponential stability of the tracking error. In the case of the inverse dynamics control considered previously, we concluded exactly the same result in a much simpler way. Therefore the advantages, if any, of the passivity-based control over the inverse dynamics control are unclear at this point. We will see in the next two sections that the real advantage of the passivity-based approach occurs for the robust and

adaptive control problems. In the robust control approach considered next we will see that the assumption $\|E\| = \|M^{-1}\hat{M} - I\| < 1$ can be eliminated and that the computation of the uncertainty bounds is greatly simplified. In the adaptive control approach we will see that the requirements on acceleration measurement and boundedness of the estimated inertia matrix $\hat{M}$ can be eliminated. Thus, the passivity-based approach has some very important advantages over the inverse dynamics approach for the robust and adaptive control problems.

9.4.1 Passivity-Based Robust Control

In this section we use the passivity-based approach above to derive an alternative robust control algorithm that exploits both the skew symmetry property and linearity in the parameters and leads to a much easier design in terms of computation of uncertainty bounds. We modify the control input (9.88) as

$$u = \hat{M}(q)a + \hat{C}(q,\dot{q})v + \hat{g}(q) - Kr \tag{9.95}$$

where K, Λ, v, a, and r are given by (9.89). In terms of the linear parametrization of the robot dynamics, the control (9.95) becomes

$$u = Y(q,\dot{q},a,v)\hat{\theta} - Kr \tag{9.96}$$

and the combination of Equation (9.95) with Equation (9.87) yields

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y(\hat{\theta} - \theta) \tag{9.97}$$

We now choose the term $\hat{\theta}$ in Equation (9.96) as

$$\hat{\theta} = \theta_0 + \delta\theta \tag{9.98}$$

where θ_0 is a fixed nominal parameter vector and $\delta\theta$ is an additional control term. The system (9.97) then becomes

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y(q,\dot{q},a,v)(\tilde{\theta} + \delta\theta) \tag{9.99}$$

where $\tilde{\theta} = \theta_0 - \theta$ is a constant vector and represents the parametric uncertainty in the system. If the uncertainty can be bounded by finding a nonnegative constant $\rho \geq 0$ such that

$$\|\tilde{\theta}\| = \|\theta - \theta_0\| \leq \rho \tag{9.100}$$

then the additional term $\delta\theta$ can be designed according to

$$\delta\theta = \begin{cases} -\rho\dfrac{Y^T r}{||Y^T r||} & ; \quad \text{if} \quad ||Y^T r|| > \epsilon \\ \\ -\dfrac{\rho}{\epsilon} Y^T r & ; \quad \text{if} \quad ||Y^T r|| \leq \epsilon \end{cases} \tag{9.101}$$

Using the same Lyapunov function candidate from Equation (9.91) above, we can show uniform ultimate boundedness of the tracking error. Carrying out the details of the calculation of $\dot{V}$ yields

$$\dot{V} \quad = \quad -e^T Q e + r^T Y(\tilde{\theta} + \delta\theta) \tag{9.102}$$

Uniform ultimate boundedness of the tracking error follows with the control $\delta\theta$ from Equation (9.101) exactly as in the proof of Theorem 9.1. The details are left as an exercise (Problem 9–14).

Comparing this approach with the approach in Section 9.3.3 we see that finding a constant bound ρ for the constant vector $\tilde{\theta}$ is much simpler than finding a time-varying and state-dependent bound for η in Equation (9.47). The bound ρ in this case depends only on the inertia parameters of the manipulator, while $\rho(x,t)$ in Equation (9.54) depends on the manipulator state vector, the reference trajectory and, in addition, requires some assumptions on the estimated inertia matrix $\hat{M}(q)$.

9.4.2 Passivity-Based Adaptive Control

In the adaptive approach the vector $\hat{\theta}$ in Equation (9.96) is taken to be a time-varying estimate of the true parameter vector θ. Combining the control law, Equation (9.95), with Equation (9.87) yields

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y\tilde{\theta} \tag{9.103}$$

The parameter estimate $\hat{\theta}$ may be computed using standard methods of adaptive control such as gradient or least squares. For example, using the gradient update law

$$\dot{\hat{\theta}} = -\Gamma^{-1} Y^T(q,\dot{q},a,v) r \tag{9.104}$$

together with the Lyapunov function

$$V = \frac{1}{2} r^T M(q) r + \tilde{q}^T \Lambda K \tilde{q} + \frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta} \tag{9.105}$$

results in global convergence of the tracking errors to zero and boundedness of the parameter estimates. Computing $\dot{V}$ along trajectories of the system (9.103), we obtain

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} + \tilde{\theta}^T \{\Gamma \dot{\hat{\theta}} + Y^T r\} \tag{9.106}$$

Substituting the expression for $\dot{\hat{\theta}}$ from the gradient update law (9.104) into Equation (9.106) yields

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} = -e^T Q e \leq 0 \tag{9.107}$$

where e and Q are defined as before, showing that the closed-loop system is stable in the sense of Lyapunov. It follows that the tracking error $e(t)$ converges asymptotically to zero and the estimation error $\tilde{\theta}$ is bounded. In order to show this we note that since $\dot{V}$ is nonincreasing from Equation (9.107), the value of $V(t)$ can be no greater than its value at $t = 0$. Since V consists of a sum of nonnegative terms, this means that each of the terms r, $\tilde{q}$, and $\tilde{\theta}$ are bounded as functions of time.

With regard to the tracking error, $\tilde{q}$, $\dot{\tilde{q}}$, we also note that $\dot{V}$ is quadratic in the error vector $e(t)$. Integrating both sides of Equation (9.107) gives

$$V(t) - V(0) = -\int_0^t e^T(\sigma) Q e(\sigma) d\sigma < \infty \tag{9.108}$$

As a consequence, the tracking error vector $e(t)$ is a **square integrable function**. Such functions, under some mild additional restrictions, must tend to zero as $t \to \infty$. Specifically, we may appeal Barbalat's Lemma from Appendix C. Since both $r = \dot{\tilde{q}} + \Lambda \tilde{q}$ and $\tilde{q}$ have already been shown to be bounded, it follows that $\dot{\tilde{q}}$ is also bounded. Therefore, we have that $\tilde{q}$ is square integrable and its derivative is bounded. Hence, the tracking error $\tilde{q} \to 0$ as $t \to \infty$.

To show that the velocity tracking error also converges to zero, one must appeal to the equations of motion (9.103), from which one may argue that the acceleration $\ddot{q}$ is bounded. It follows that the velocity error $\dot{\tilde{q}}$ asymptotically converges to zero provided that the reference acceleration $\ddot{q}^d(t)$ is bounded.

9.5 Torque Optimization

In previous sections we considered the effect of parameter uncertainty in the design of control laws for n-link manipulators and derived both robust

and adaptive control schemes to deal with uncertain or unknown inertia parameters. In this section we consider the problem of actuator saturation in the design of nonlinear control laws. Actuator saturation, together with joint elasticity and friction, have long been recognized as the primary factors that limit the performance of manipulators. Actuator saturation may be considered as part of the motion planning problem, for example, by including bounds on the velocities and accelerations when planning desired trajectories and then correlating these bounds with bounds on the actuator torques.

In this section we discuss an alternative method that views the problem of designing a nonlinear control law subject to actuator saturation as a constrained torque optimization problem. Given a nominal control law designed as a first step without considering input constraints, the goal is to find a control law "closest" to the nominal control with respect to a given norm. The optimal control is thus determined by solving a **quadratic programming problem** at each time step.

Consider again the Euler–Lagrange equations for an n-link manipulator

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{9.109}$$

and suppose that the torque input $u = (u_1, \cdots, u_n)$ is bounded as

$$u_{i,min} \leq u_i \leq u_{i,max},\ i = 1, \ldots, n \tag{9.110}$$

For simplicity we will assume that these bounds are constant.

Example 9.1. *Consider the two-link, planar RR manipulator example from Section (6.4) and suppose that the nominal control law u_0 is chosen as an inverse dynamics control (9.23)*

$$u = M(q)a + C(q,\dot{q})\dot{q} + g(q) \tag{9.111}$$
$$a = \ddot{q}^d + K_d(\dot{q}^d - q) + K_p(q^d - q) \tag{9.112}$$

where $q^d(t)$ is the reference trajectory. For simplicity, we will take q^d as a constant step change in the joint angles in this example so that

$$a = K_p(q^d - q) - K_d\dot{q} \tag{9.113}$$

Figure (9.4) shows the joint responses with and without torque saturation, where we have taken

$$q_1^d = 10 \qquad q_2^d = 20 \tag{9.114}$$
$$-90 \leq u_1 \leq 90 \qquad -30 \leq u_2 \leq 30 \tag{9.115}$$

We see that the actual step responses deviate considerably from the ideal (unconstrained) responses with both overshoot and undershoot.

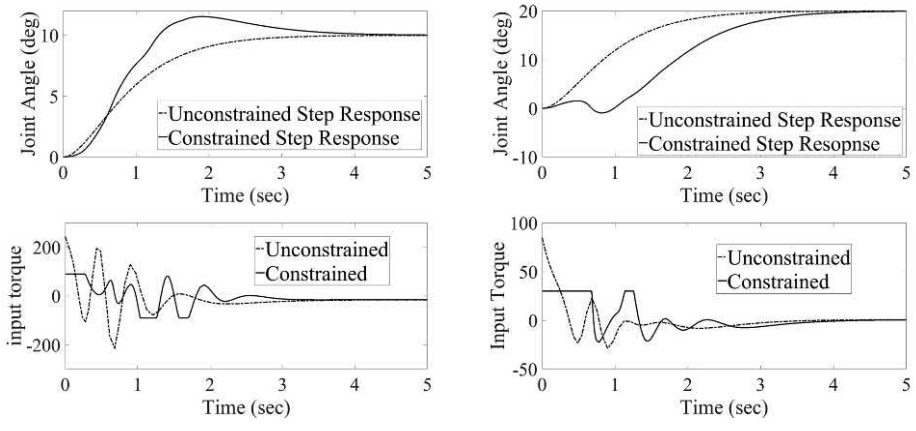


Figure 9.4: Joint responses and input torques with saturated and unsaturated inverse dynamics control.

Next, note that the inequality constraints (9.110) can be written as

$$Nu \leq c$$

where $N \in \mathbb{R}^{2n \times n}$ and $c \in \mathbb{R}^{2n \times 1}$ are given by

$$N = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & 1 \\ \vdots & \vdots & & -1 \end{bmatrix}, \; c = \begin{bmatrix} u_{1,max} \\ -u_{1,min} \\ \vdots \\ u_{n,max} \\ -u_{n,min} \end{bmatrix} \tag{9.116}$$

Let u_0 be a control law designed by any method discussed so far, such as an inverse dynamics control, robust control, adaptive control, and so on. The torque optimization problem that we consider is then to choose the actual control input u to satisfy

$$\min_u \frac{1}{2}(u - u_0)^T \Pi (u - u_0) \tag{9.117}$$

$$\text{subject to} \quad Nu \leq c \tag{9.118}$$

where Π is a symmetric, positive definite $n \times n$ matrix to be chosen by the designer. Note that Π need not necessarily be a constant matrix. Thus the minimization (9.117)–(9.118) is a quadratic programming that is solved pointwise, i.e. at each instant of time t.

Primal-Dual Method

The optimization problem (9.117)–(9.118) is called the **primal problem**. The primal problem can be shown to be equivalent to the optimization problem

$$\max_{\lambda \geq 0} \min_{u} \{ \frac{1}{2}(u - u_0)^T \Pi (u - u_0) + \lambda^T (Nu - c) \} \tag{9.119}$$

where $\lambda \in \mathbb{R}^n$ is a vector of **Lagrange multipliers**.

From (9.119) we see that the minimization over u is unconstrained. Therefore, after a straightforward calculation the minimizer u^* is given by

$$u^* = u_0 - \Pi^{-1} N^T \lambda \tag{9.120}$$

Substituting u^* from (9.120) back into (9.119) we get

$$\max_{\lambda \geq 0} \{ -\frac{1}{2} \lambda^T P \lambda - \lambda^T d \} \tag{9.121}$$

or equivalently

$$\min_{\lambda \geq 0} \{ \frac{1}{2} \lambda^T P \lambda + \lambda^T d \} \tag{9.122}$$

where

$$P = N \Pi^{-1} N^T \ ; \ d = c - N u_0 \tag{9.123}$$

The minimization problem (9.122) is called the **dual problem**. Note that the dual problem is also a quadratic program.

Solving the dual problem for $\lambda = \lambda^*$ gives the optimal control input u^{**} as

$$u^{**} = u_0 - \Pi^{-1} N^T \lambda^* \tag{9.124}$$

The advantage of the primal-dual method is that the dual problem has only the constraint $\lambda \geq 0$ and is therefore often easier to solve than the primal problem.

Example 9.2. *Going back to the previous example, with u_0 given as an inverse dynamics control*

$$u \ = \ M(q)a + C(q, \dot{q})\dot{q} + g(q) \tag{9.125}$$

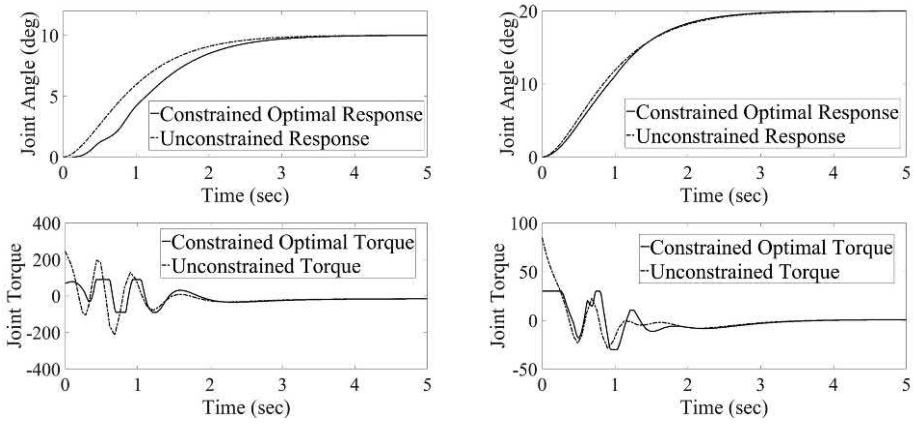


Figure 9.5: Optimal joint trajectories and input torques using (9.128) compared with the unsaturated case.

if we choose Π *in (9.117) as*

$$\Pi = M^{-1}QM^{-1} \qquad (\text{so that } \Pi^{-1} = MQM) \tag{9.126}$$

where Q *is a constant symmetric positive definite* $n \times n$ *matrix, then the optimal control law* u^{**} *can be written as*

$$\begin{aligned} u^{**} &= u_0 - \Pi^{-1}N^T\lambda^* && (9.127) \\ &= M(q)(a - QMN^T\lambda^*) + C(q,\dot{q})\dot{q} + g(q) && (9.128) \end{aligned}$$

Therefore we see that the optimization strategy amounts to modifying the outer-loop control term to account for the actuator saturation. This significance of this result is that the nominal inner-loop control remains unchanged and therefore we may incorporate the above torque optimization method into any control design methodology.

Figure (9.5) shows the joint responses and joint torques using the control law (9.128) computed by solving the dual problem. The tracking performance of the unconstrained control is nearly recovered.

9.6 Chapter Summary

In this chapter we discussed the nonlinear control problem for robot manipulators. We developed models of both rigid and flexible-joint robots. We then developed several control algorithms and discussed pros and cons of

each as well as implementation aspects. Among the algorithms we discussed were PD control, inverse dynamics, and passivity-based control. Moreover we showed how to formulate robust and adaptive versions of the latter two approaches.

Plant Models

The rigid and flexible-joint robot models are, respectively:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

$$\begin{aligned} D(q_1)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 + K(q_2 - q_1) &= u \end{aligned}$$

PD Control

A PD control law in joint space is of the form

$$u = -K_P\tilde{q} - K_D\dot{q}$$

Global asymptotic tracking for the rigid model can be shown using the Lyapunov function candidate below together with LaSalle's theorem in case the gravity vector $g(q) = 0$.

$$V = 1/2\dot{q}^T M(q)\dot{q} + 1/2\tilde{q}^T K_P\tilde{q}$$

PD Control with Gravity Compensation

With gravity present, the PD plus gravity compensation algorithm below also results in global asymptotic tracking for the rigid model.

$$u = -K_P\tilde{q} - K_D\dot{q} + g(q)$$

Joint Space Inverse Dynamics

The inverse dynamics control law consists of the following two expressions, the first being the **inner-loop** control and the second being the **outer-loop** control.

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q)$$

$$a_q = \ddot{q}^d(t) - K_0\tilde{q} - K_1\dot{\tilde{q}}$$

The inverse dynamics algorithm results in a closed-loop system that is linear and decoupled.

Task Space Inverse Dynamics

We showed that the modified outer-loop term below results in a linear, decoupled system in task space coordinates $X \in \mathbb{R}^6$, where X is a minimal representation of $SE(3)$ and J is the analytical Jacobian.

$$\begin{aligned} a_q &= J^{-1}\left\{a_X - \dot{J}\dot{q}\right\} \\ a_X &= \ddot{X}^d - K_0(X - X^d) - K_1(\dot{X} - \dot{X}^d) \end{aligned}$$

Robust Inverse Dynamics

We presented a Lyapunov-based approach for robust inverse dynamics control.

$$u = \hat{M}(q)a_q + \hat{C}(q,\dot{q})\dot{q} + \hat{g}(q)$$

where $\hat{M}$, $\hat{C}$, and $\hat{g}(q)$ are the nominal values of M, C, and g. From this we derived the state space model

$$\dot{e} = Ae + B\{\delta a + \eta\}$$

where η represents the uncertainty resulting from inexact cancellation of nonlinearities and

$$A = \begin{bmatrix} 0 & I \\ -K_0 & -K_1 \end{bmatrix} \; ; \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

The additional control input δa was chosen as

$$\delta a = \begin{cases} -\rho(e,t)\dfrac{B^T P e}{\|B^T P e\|} & ; \quad \text{if} \quad \|B^T P e\| > \epsilon \\ \\ -\dfrac{\rho(e,t)}{\epsilon} B^T P e & ; \quad \text{if} \quad \|B^T P e\| \leq \epsilon \end{cases}$$

and shown to achieve uniform ultimate boundedness of the tracking errors. This is a practical notion of asymptotic stability in the sense that the tracking errors can be made small.

Adaptive Inverse Dynamics

The adaptive version of the inverse dynamics control results in a system of the form

$$\begin{aligned} \dot{e} &= Ae + B\Phi\tilde{\theta} \\ \dot{\hat{\theta}} &= -\Gamma^{-1}\Phi^T B^T P e \end{aligned}$$

where θ represents the unknown parameters (masses, moments of inertia, etc.) The second equation above is used to estimate the parameters online. The Lyapunov function candidate below can be used to show asymptotic convergence of the tracking errors to zero and boundedness of the parameter estimation error.

$$V = e^T P e + \frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta}$$

Passivity-Based Robust Control

We introduced the notion of passivity-based control as an alternative to inverse dynamics control. This approach exploits the passivity property of the robot dynamics rather than attempting to cancel the nonlinearities as in the inverse dynamics approach. We presented a control algorithm of the form

$$u = \hat{M}(q)a + \hat{C}(q,\dot{q})v + \hat{g}(q) - Kr$$

where the quantities v, a, and r are given as

$$\begin{aligned} v &= \dot{q}^d - \Lambda\tilde{q} \\ a &= \dot{v} = \ddot{q}^d - \Lambda\dot{\tilde{q}} \\ r &= \dot{q} - v = \dot{\tilde{q}} + \Lambda\tilde{q} \end{aligned}$$

and K is a diagonal matrix of positive gains. This results in a closed-loop system

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y(\hat{\theta} - \theta)$$

In the robust passivity-based approach the term $\hat{\theta}$ is chosen as

$$\hat{\theta} = \theta_0 + \delta\theta$$

where θ_0 is a fixed nominal parameter vector and $\delta\theta$ is an additional control term. The additional term $\delta\theta$ can be designed according to

$$\delta\theta = \begin{cases} -\rho\dfrac{Y^T r}{\|Y^T r\|} & ; \quad \text{if} \quad \|Y^T r\| > \epsilon \\ \\ -\dfrac{\rho}{\epsilon}Y^T r & ; \quad \text{if} \quad \|Y^T r\| \leq \epsilon \end{cases}$$

where ρ is a bound on the parameter uncertainty. Uniform ultimate boundedness of the tracking errors follows using the Lyapunov function candidate

$$V = \frac{1}{2}r^T M(q) r + \tilde{q}^T \Lambda K \tilde{q}$$

Passivity-Based Adaptive Control

In the adaptive version of this approach we derived the system

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y\tilde{\theta}$$

$$\dot{\hat{\theta}} = -\Gamma^{-1}Y^T(q,\dot{q},a,v)r$$

and used the Lyapunov function candidate

$$V = \frac{1}{2}r^T M(q)r + \tilde{q}^T \Lambda K \tilde{q} + \frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta}$$

to show global convergence of the tracking errors to zero and boundedness of the parameter estimates.

Torque Optimization

Finally, we considered the problem of actuator saturation and formulated an optimization method to reduce the tracking errors caused by the input saturation. Given a nominal control u_0, computed using any of the methods in this chapter, the actual control u is computed as a solution of a pointwise quadratic programming problem

$$\begin{aligned} &\min_u \frac{1}{2}(u-u_0)^T \Pi (u-u_0) \\ &\text{subject to} \quad Nu \leq c \end{aligned}$$

where the constraints $Nu \leq c$ account for the constraints on the input torque. We showed that the implementation of the above torque optimization requires only the modification of the outer-loop control term.

Problems

9–1 Verify the properties of skew symmetry, passivity and linearity in the parameters for the system given by Equation (9.6). Compute bounds on the inertia matrix, $M(q)$, in terms of bounds on $D(q)$. Show that $M(q)$ is positive definite.

9–2 Form the Lagrangian for an n-link manipulator with joint flexibility using Equations (9.15) and (9.16). From this derive the equations of motion (9.18).

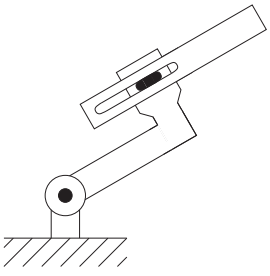


Figure 9.6: Two-link RP manipulator.

9–3 Complete the proof of stability of PD control for the flexible-joint robot without gravity terms using the Lyapunov function candidate (9.20) and LaSalle's theorem. Show that $q_1 = q_2$ in the steady state.

9–4 Given the flexible-joint model defined by Equation (9.18) with the PD control law (9.19), what are the steady-state values of q_1 and q_2 if the gravity term is present? How could one define the reference angle q^d in the case that gravity is present?

9–5 Suppose that the PD control law given by Equation (9.19) is implemented using the link variables

$$u = K_p\tilde{q} - K_D\dot{q}_1$$

where $\tilde{q} = q_1 - q^d$. Show that the equilibrium $\tilde{q} = 0 = \dot{q}_1$ is unstable. Hint: Use Lyapunov's First Method, that is, show that the equilibrium is unstable for the linearized system.

9–6 Simulate an inverse dynamics control law for a two-link elbow manipulator whose equations of motion were derived in Chapter 6. Investigate what happens if there are constraints on the input torque.

9–7 For the system of Problem 9–6, what happens to the response of the system if the coriolis and centrifugal terms are dropped from the inverse dynamics control law in order to facilitate computation? What happens if incorrect values are used for the link masses? Investigate via computer simulation.

9–8 Carry out the details to derive the uncertain system (9.46) and (9.47).

9–9 Consider the two-link RP manipulator shown in Figure 9.6. Show that

the rotational inertial about the first link is not bounded as a function of the prismatic joint variable d_2. Discuss the implications of this for the problem of robust control.

9–10 Add an outer-loop correction term δa to the control law of Problem 9–7 to overcome the effects of uncertainty. Base your design on the second method of Lyapunov as in Section 9.3.3.

9–11 Derive the error equation (9.79) using the linearity-in-the-parameters property of the robot dynamics.

9–12 Verify the expression for $\dot{V}$ in Equation (9.85).

9–13 Consider the coupled nonlinear system

$$\begin{aligned} \ddot{y}_1 + 3y_1y_2 + y_2^2 &= u_1 + y_2u_2 \\ \ddot{y}_2 + \cos y_1\dot{y}_2 + 3(y_1 - y_2) &= u_2 - 3(\cos y_1)^2y_2u_1 \end{aligned}$$

where u_1, u_2 are the inputs and y_1, y_2 are the outputs.

(a) What is the dimension of the state space?

(b) Choose state variables and write the system as a system of first order differential equations in state space.

(c) Find an inverse dynamics control so that the closed-loop system is linear and decoupled, with each subsystem having natural frequency 10 radians and damping ratio 1/2.

9–14 Complete the proof of uniform ultimate boundedness for the passivity-based robust control law given by Equation (9.95) applied to the rigid robot model.

9–15 Prove the inequality (9.59).

Notes and References

Many of the fundamental theoretical problems in motion control of robot manipulators were solved during an intense period of research from about the mid-1980's until the early-1990s, during which time researchers first began to exploit the structural properties of manipulator dynamics such as feedback linearizability, skew symmetry and passivity, multiple time-scale behavior, and other properties. For a more advanced treatment of some of these topics, the reader is referred to [167] and [32].

In the area of PD and PID control of manipulators, the earliest results are contained in [174]. These results were based on the Hamiltonian formulation of robot dynamics and effectively exploited the passivity property. The use of energy as a Lyapunov function is described in [78].

The problem of joint flexibility was first brought to the forefront of robotics research in [128], [172], and [173]. The model presented here to describe the dynamics of flexible-joint robots is due to [165].

The earliest results on computed torque appeared in [110] and [136]. A related approach known as the method of **resolved motion acceleration control** is due to [104]. In [83] these various control schemes are all compared to the method of inverse dynamics and shown to be essentially equivalent.

The robust inverse dynamics control approach here follows closely the general methodology in [28]. The earliest application of this method to the manipulator control problem was in [31] and [168]. This technique is closely related to the so-called **method of sliding modes**, which has been applied to manipulator control in [154]. A very complete survey of robust control of robots up to about 1990 is found in [1]. Other results in robust control from an operator theoretic viewpoint are [170] and [58]. The passivity-based robust control result here is due to [166].

The question of when the inertia matrix of a robot manipulator is bounded is treated in detail in [53].

The adaptive inverse dynamics control result presented here is due to [29]. Other notable results in this area appeared in [114]. The first results in passivity-based adaptive control of manipulators was in [67] and [153]. The Lyapunov stability proof presented here is due to [161]. A unifying treatment of adaptive manipulator control from a passivity perspective was presented in [131]. Other works based on passivity are [19] and [12].

One of the problems with the adaptive control approaches considered here is the so-called **parameter drift** problem. The Lyapunov stability proofs presented show that the parameter estimates are bounded but there is no guarantee that the estimated parameters converge to their true values. It can be shown that the estimated parameters converge to the true parameters provided the reference trajectory satisfies the condition of **persistency of excitation**

$$\alpha I \leq \int_{t_0}^{t_0+T} Y^T(q^d, \dot{q}^d, \ddot{q}^d) Y(q^d, \dot{q}^d, \ddot{q}^d) dt \leq \beta I \tag{9.129}$$

for all t_0, where α, β, and T are positive constants.

Chapter 10

FORCE CONTROL

In previous chapters we considered the problem of tracking motion trajectories using a variety of elementary and advanced control methods. These position control schemes are adequate for tasks such as materials transfer, spray painting, or spot welding where the manipulator is not interacting significantly with objects in the workplace (hereafter referred to as the **environment**). However, tasks such as assembly, grinding, and deburring, which involve extensive contact with the environment, are often better handled by controlling the **forces**[1] of interaction between the manipulator and the environment rather than simply controlling the position of the end effector. For example, consider an application where the manipulator is required to wash a window, or to write with a felt tip marker. In both cases a pure position control scheme is unlikely to work. Slight deviations of the end effector from a planned trajectory would cause the manipulator either to lose contact with the surface or to press too strongly on the surface. For a highly rigid structure such as a robot, a slight position error could lead to extremely large forces of interaction with disastrous consequences (broken window, smashed pen, damaged end effector, etc.). The above applications are typical in that they involve both force control and trajectory control. In the window washing application, for example, one clearly needs to control the forces normal to the plane of the window and position in the plane of the window.

We first introduce the notion of **natural and artificial constraints**, which allows one to partition a given task into position-controlled directions and force-controlled directions similar to the window washing example above. We then introduce **network models** that are useful to represent

[1] Hereafter we use *force* to mean force and/or torque and *position* to mean position and/or orientation.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

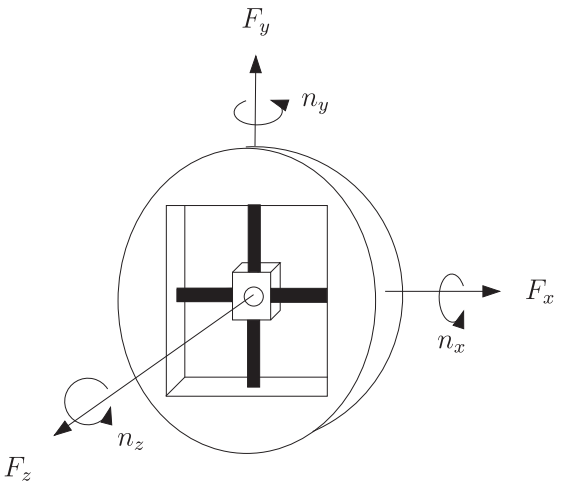


Figure 10.1: A wrist force sensor. The array of strain gauges provides data of both force and torque relative to the local coordinate frame of the sensor.

the robot/environment interaction. These network models are similar to those used in circuit theory and, in fact, we use two commonly used circuit models, the **Thévenin** and **Norton** equivalent networks to model both the robot and environment after which the robot/environment interaction can be viewed as essentially a problem of **impedance matching**. We then include the interaction force in the task space dynamics of the robot using the dynamic models and inverse dynamics control results of the previous chapters. We then give details of two common design methods for robot force control, namely, **impedance control** and **hybrid impedance control**. Impedance control allows one to modify the apparent inertia, stiffness, and damping of the robot as seen by the environment. Hybrid impedance control takes this a step further and allows one to control the robot position or contact forces in addition to specifying the robot/environment impedance.

Force and Torque Sensing

A force control strategy is one that modifies position trajectories based on the sensed forces. There are three main types of sensors for force feedback, **wrist force** sensors, **joint torque** sensors, and **tactile** or hand sensors. A wrist force sensor such as that shown in Figure 10.1 usually consists of an array of strain gauges and can delineate the three components of the vector force along the three axes of the sensor coordinate frame, and the three

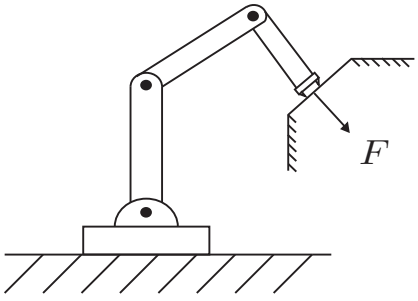


Figure 10.2: Robot end effector in contact with a rigid surface. The surface prevents motion of the end effector in the direction normal to the surface, which results in interaction forces between the robot and environment.

components of the torque about these axes. A joint torque sensor consists of strain gauges located on the actuator shaft. Tactile sensors are usually located on the fingers of the gripper and are useful for sensing gripping force and for shape detection. For the purposes of controlling the end-effector/environment interactions, the six-axis wrist sensor usually gives the best results and we shall henceforth assume that the manipulator is equipped with such a device.

10.1 Coordinate Frames and Constraints

Force control tasks can be thought of in terms of constraints imposed by the robot/environment interaction. A manipulator moving through free space within its workspace is unconstrained in motion and can exert no forces since there is no source of reaction force from the environment. A wrist force sensor in such a case would record only the inertial forces due to any acceleration of the end effector. As soon as the manipulator comes in contact with the environment, say a rigid surface as shown in Figure 10.2, one or more degrees of freedom in motion may be lost since the manipulator cannot move through the environment surface. At the same time, the manipulator can exert forces against the environment.

10.1.1 Reciprocal Bases

In order to describe the robot/environment interaction, let $\xi = [v, \omega]$ represent the instantaneous linear and angular velocity of the end effector and let $F = [f, n]$ represent the instantaneous force and moment acting on the end

effector. The vectors ξ and F are each elements of six dimensional vector spaces, which we denote by $\mathcal{M}$ and $\mathcal{F}$, the motion and force spaces, respectively. The vectors ξ and F are called **twists** and **wrenches**, respectively, in more advanced texts [118] although we will continue to refer to them simply as velocity and force for simplicity.

Definition 10.1.

1. *If $\{e_1, \ldots, e_6\}$ is a basis for the vector space $\mathcal{M}$, and $\{f_1, \ldots, f_6\}$ is a basis for the vector space $\mathcal{F}$, we say that these basis vectors are* **reciprocal** *provided that*

$$\begin{array}{rcll} e_i^T f_j & = & 0 & \text{if } i \neq j \\ e_i^T f_j & = & 1 & \text{if } i = j \end{array} \tag{10.1}$$

2. *A twist $\xi \in \mathcal{M}$ and a wrench $F \in \mathcal{F}$ are called* **reciprocal** *if*

$$\xi^T F = v^T f + \omega^T n = 0 \tag{10.2}$$

Twists are elements of the tangent space $T_q\mathcal{Q}$ at each configuration q and wrenches are elements of the dual tangent space $T_q^*\mathcal{Q}$. Equation (10.2) means simply that the twists and wrenches are orthogonal at each $q \in \mathcal{Q}$. The advantage of using reciprocal basis vectors is that the product $\xi^T F$ is then **invariant** with respect to a linear change of basis from one reciprocal coordinate system to another. Thus, the **reciprocity condition** given by Equation (10.2) is invariant with respect to choice of reciprocal bases of $\mathcal{M}$ and $\mathcal{F}$. We shall show below how the reciprocity relation given by Equation (10.2) can be used to design reference inputs to execute motion and force control tasks.

Metrics on $SO(3)$ and $SE(3)$

Since $\mathcal{M}$ and $\mathcal{F}$ are each six dimensional vector spaces, it is tempting to identify each with $\mathbb{R}^6$. However, it turns out that inner product expressions such as $\xi_1^T \xi_2$ or $F_1^T F_2$ for vectors ξ_i, F_i belonging to $\mathcal{M}$ and $\mathcal{F}$, respectively, are not necessarily well defined. For example, the expression

$$\xi_1^T \xi_2 = v_1^T v_2 + \omega_1^T \omega_2 \tag{10.3}$$

is not invariant with respect to either choice of units or basis vectors in $\mathcal{M}$.

Example 10.1. *Suppose that*

$$\begin{array}{rcl} \xi_1 & = & [v_1, \omega_1] = [1, 1, 1, 2, 2, 2] \\ \xi_2 & = & [v_2, \omega_2] = [2, 2, 2, -1, -1, -1] \end{array}$$

where the linear velocity is in meters/sec and angular velocity is in radians/sec. Then clearly, $\xi_1^T \xi_2 = 0$ and so one could infer that ξ_1 and ξ_2 are orthogonal vectors in $\mathcal{M}$. However, suppose now that the linear velocity is represented in units of centimeters/sec. Then

$$\begin{aligned} \xi_1 &= [1 \times 10^2, 1 \times 10^2, 1 \times 10^2, 2, 2, 2]^T \\ \xi_2 &= [2 \times 10^2, 2 \times 10^2, 2 \times 10^2, -1, -1, -1]^T \end{aligned}$$

and clearly $\xi_1^T \xi_2 \neq 0$. Thus, the usual notion of orthogonality is not meaningful in $\mathcal{M}$.

Remark 10.1. *It is possible to define inner product like operations, that is, symmetric, bilinear forms on $\mathcal{M}$ and $\mathcal{F}$ that have the necessary invariance properties. These are the so-called* **Klein form**, *$KL(\xi_1, \xi_2)$, and* **Killing form**, *$KI(\xi_1, \xi_2)$, defined according to*

$$KL(\xi_1, \xi_2) = v_1^T \omega_2 + \omega_1^T v_2 \tag{10.4}$$

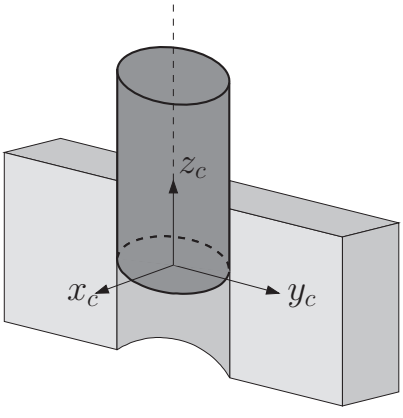
$$KI(\xi_1, \xi_2) = \omega_1^T \omega_2 \tag{10.5}$$

One can show that the equality $KL(\xi_1, \xi_2) = 0$ (respectively, $KI(\xi_1, \xi_2) = 0$) is independent of the units or the basis chosen to represent ξ_1 and ξ_2. For example, the condition $KI(\xi_1, \xi_2) = 0$ means that the axes of rotation defining ω_1 and ω_2 are orthogonal. However, a detailed discussion of these concepts is beyond the scope of this text. As the reader may suspect, the need for a careful treatment of these concepts is related to the geometry of $SO(3)$ as we have seen before in other contexts.

10.1.2 Natural and Artificial Constraints

In this section we discuss so-called **natural constraints**, which are defined using the reciprocity condition given by Equation (10.2). We then discuss the notion of **artificial constraints**, which are used to define reference inputs for motion and force control tasks.

We begin by defining a so-called **compliance frame** $o_c x_c y_c z_c$ (also called a **constraint frame**) in which the task to be performed is easily described. For example, in the window washing application, we can define a frame at the tool with the z_c-axis along the surface normal direction. The task specification is then expressed in terms of maintaining a constant force in the z_c direction while following a prescribed trajectory in the x_c–y_c plane. Such a position constraint in the z_c direction, arising from the presence of a rigid surface, is a natural constraint. The force that the robot exerts against the rigid surface in the z_c direction, on the other hand, is not constrained by the



Natural Constraints	Artificial Constraints
$v_x = 0$	$f_x = 0$
$v_y = 0$	$f_y = 0$
$f_z = 0$	$v_z = v_d$
$\omega_x = 0$	$n_x = 0$
$\omega_y = 0$	$n_y = 0$
$n_z = 0$	$\omega_z = 0$

Figure 10.3: Inserting a peg into a hole, showing natural constraints imposed by the environment and artificial constraints chosen by the designer.

environment. A desired force in the z_c direction would then be considered as an artificial constraint that must be maintained by the control system. Likewise, while the window prevents motion in the z_c direction, it does not impose any limitations on motion in the x_c and y_c directions. The application of washing the window could be achieved by specifying a desired trajectory in the x_c–y_c plane, which would then be considered constitute artificial constraints.

Figure 10.3 shows a typical task, that of inserting a peg into a hole. With respect to a compliance frame $o_c x_c y_c z_c$, as shown at the end of the peg, we may take the standard orthonormal basis in $\mathbb{R}^6$ for both $\mathcal{M}$ and $\mathcal{F}$, in which case

$$\xi^T F = v_x f_x + v_y f_y + v_z f_z + \omega_x n_x + \omega_y n_y + \omega_z n_z \tag{10.6}$$

If we assume that the walls of the hole and the peg are perfectly rigid and there is no friction, it is easy to see that

$$\begin{array}{lll} v_x = 0, & v_y = 0, & f_z = 0 \\ \omega_x = 0, & \omega_y = 0, & n_z = 0 \end{array} \tag{10.7}$$

These relationships in (10.7) are termed **natural constraints**, since they are imposed by the environment. For example, the walls of the hole prevent motion in the x_c and y_c directions giving rise to the natural constraints $v_x = 0$ and $v_y = 0$. On the other hand, there is nothing to prevent the peg from spinning about the z_c-axis, hence there is no natural constraint on ω_z. Each of these natural constraints can be inferred from the geometry of the task.

With the above natural constraints, it is easy to see that the reciprocity condition $\xi^T F = 0$ is satisfied for any F. Examining Equation (10.6) we see that the variables

$$f_x, \quad f_y, \quad v_z, \quad n_x, \quad n_y, \quad \omega_z \tag{10.8}$$

are therefore unconstrained by the environment. In other words, given the natural constraints from Equation (10.7), the reciprocity condition $\xi^T F = 0$ holds for all values of the above variables in Equation (10.8). We may therefore arbitrarily assign reference values, called **artificial constraints**, for these variables with the idea that the control system should be designed to enforce these artificial constraints in order to carry out the task at hand. For example, in the peg-in-hole task, we specify $f_x = f_y = 0$ to prevent the robot from exerting forces against the sides of the hole where motion is not possible. Likewise, since there is no natural constraint on motion in the z_c direction, we may specify $v_c = v_d$ as the desired rate that the peg should be inserted into the hole. Equation (10.9) gives a set of such artificial constraints for the peg-in-hole insertion task

$$\begin{array}{lll} f_x = 0, & f_y = 0, & v_z = v^d \\ n_x = 0, & n_y = 0, & \omega_z = 0 \end{array} \tag{10.9}$$

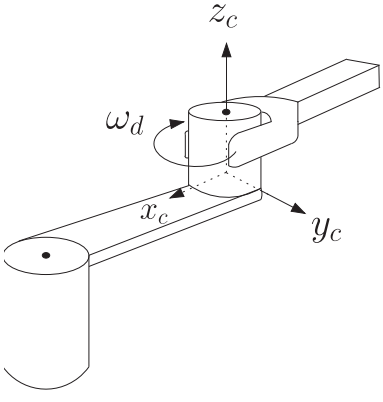
The natural and artificial constraints are summarized in Figure 10.3.

As a second example, Figure 10.4 shows natural and artificial constraints for the task of turning a crank. The reader should verify the natural constraints and the justification for the choice of the artificial constraints.

10.2 Network Models and Impedance

The reciprocity condition $\xi^T F = 0$ means that the forces of constraint do no work in directions compatible with motion constraints and holds under the ideal conditions of no friction and perfect rigidity of both the robot and the environment. In practice, compliance and friction in the robot/environment interface will alter the strict separation between motion constraints and force constraints.

For example, consider the situation in Figure 10.5. Since the environment deforms in response to a force, there is clearly both motion and force normal to the surface. Thus, the product $\xi(t)F(t)$ along this direction will not be zero. Let k represent the stiffness of the surface so that $F = kx$.



Natural Constraints	Artificial Constraints
$v_x = 0$	$f_x = 0$
$f_y = 0$	$v_y = 0$
$v_z = 0$	$f_z = 0$
$\omega_x = 0$	$n_x = 0$
$\omega_y = 0$	$n_y = 0$
$n_z = 0$	$\omega_z = \omega_d$

Figure 10.4: The task of turning a crank with the resulting natural and artificial constraints.

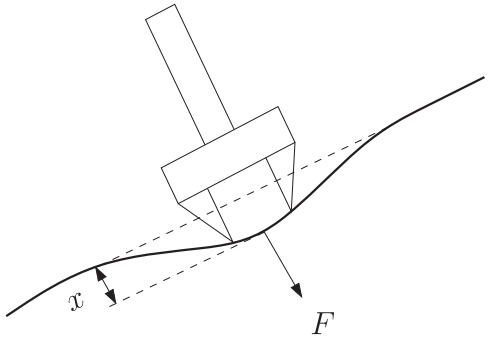


Figure 10.5: A robot in contact with a compliant environment. Both motion and force are permitted normal to the surface because of the deformation of the surface.

Then

$$\begin{aligned}\int_0^t \xi(u)F(u)du &= \int_0^t \dot{x}(u)kx(u)du = k\int_0^t \frac{d}{du}\frac{1}{2}kx^2(u)du \\ &= \frac{1}{2}k(x^2(t) - x^2(0))\end{aligned}$$

is the change of the potential energy due to the material deformation. The environment stiffness k determines the amount of force needed to produce a given motion. The higher the value of k, the more the environment "impedes" the motion of the end effector.

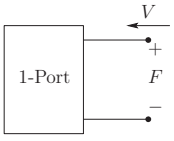


Figure 10.6: A one-port network can be thought of a black box representation of a system that describes the relationship between the port variables.

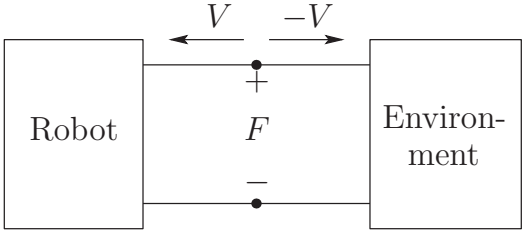


Figure 10.7: Robot/environment interaction as an interconnection of one-port networks.

In this section we introduce the notion of **mechanical impedance**, which captures the relation between force and motion. We introduce so-called **network models**, which are particularly useful for modeling the interaction between the robot and the environment.

We model the robot and the environment as **one-port networks** as shown in Figure 10.6. The dynamics of the robot and environment determine the relations among the **port variables**, V_r, F_r, and V_e, F_e, respectively. The forces F_r, F_e are known as **effort** or **across** variables while the velocities V_r, V_e are known as **flow** or **through** variables. With this description, the product of the port variables, $V^T F$, represents instantaneous **power** and the integral of this product

$$\int_0^t V^T(\sigma) F(\sigma) d\sigma$$

is the **energy** dissipated by the network over the time interval $[0, t]$.

The robot and the environment are then coupled through their interaction ports, as shown in Figure 10.7, which describes the energy exchange between the robot and the environment.

10.2.1 Impedance Operators

The relationship between the effort and flow variables may be described in terms of an **impedance operator**. For linear, time-invariant systems, we

may utilize the s-domain or Laplace domain to define the impedance.

Definition 10.2. *Given the one-port network in Figure 10.6 the impedance, $Z(s)$ is defined as the ratio of the Laplace transform of the effort to the Laplace transform of the flow,*

$$Z(s) = \frac{F(s)}{V(s)} \tag{10.10}$$

Example 10.2. *Suppose a mass-spring-damper system is described by the differential equation*

$$M\ddot{x} + B\dot{x} + Kx = F \tag{10.11}$$

Taking the Laplace transforms of both sides (assuming zero initial conditions) it follows that

$$Z(s) = F(s)/V(s) = Ms + B + K/s \tag{10.12}$$

10.2.2 Classification of Impedance Operators

It seems intuitive that different types of environments would dictate different control strategies. For example, as we have seen, pure position control would be difficult in contact with a very stiff environment as in the window washing example. Similarly, interaction forces would be difficult to control if the environment is very soft. In this section we introduce terminology to classify robot and environment impedance operators that will prove useful in the analysis to follow.

Definition 10.3. *An impedance $Z(s)$ in the Laplace variable s is said to be*

1. **Inertial** *if and only if $|Z(0)| = 0$*
2. **Resistive** *if and only if $|Z(0)| = B$ for some constant $0 < B < \infty$*
3. **Capacitive** *if and only if $|Z(0)| = \infty$*

Example 10.3. *Figure 10.8 shows examples of environment types. Figure 10.8(a) shows a mass on a frictionless surface. The impedance is $Z(s) = Ms$, which is inertial. Figure 10.8(b) shows a mass moving in a viscous medium with resistance B. Then $Z(s) = Ms + B$, which is resistive. Figure 10.8(c) shows a linear spring with stiffness K. Then $Z(s) = K/s$, which is capacitive.*

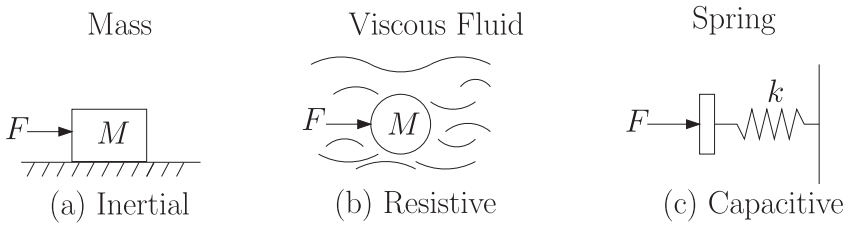


Figure 10.8: Examples of (a) inertial, (b) resistive, and (c) capacitive environments.

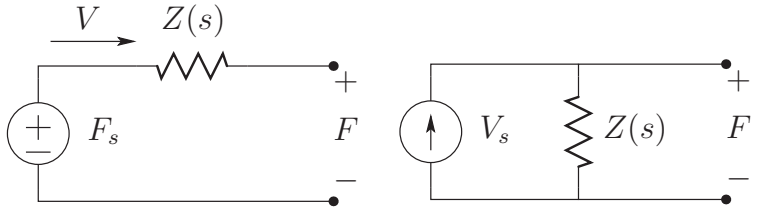


Figure 10.9: Thévenin (left) and Norton (right) equivalent networks.

10.2.3 Thévenin and Norton Equivalents

In linear circuit theory it is common to use so-called **Thévenin** and **Norton** equivalent circuits for analysis and design. It is easy to show that any one-port network consisting of passive elements (resistors, capacitors, inductors) and current or voltage sources can be represented either as an impedance $Z(s)$ in series with an effort source (Thévenin equivalent) or as an impedance $Z(s)$ in parallel with a flow source (Norton equivalent). The independent sources F_s and V_s may be used to represent reference signal generators for force and velocity, respectively, or they may represent external disturbances.

10.3 Task Space Dynamics and Control

Since a manipulator task, such as grasping an object, or inserting a peg into a hole, is typically specified relative to the end-effector frame, it is natural to derive the control algorithm directly in the task space rather than in joint space.

When the manipulator is in contact with the environment, the dynamic equations of Chapter 6 must be modified to include the reaction torque $J^T F_e$ corresponding to the end-effector force F_e, where J is the manipulator Jacobian. The modified equations of motion of the manipulator in joint

space anre given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + J^T(q)F_e = u \tag{10.13}$$

Let us consider a modified inverse dynamics control law of the form

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q) + J^T(q)a_f \tag{10.14}$$

where a_q and a_f are outer-loop controls with units of acceleration and force, respectively. Using the relationship between joint space and task space variables derived in Chapter 9

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{10.15}$$

$$a_x = J(q)a_q + \dot{J}(q)\dot{q} \tag{10.16}$$

we substitute Equations (10.14)–(10.16) into Equation (10.13) to obtain

$$\ddot{x} = a_x + W(q)(F_e - a_f) \tag{10.17}$$

where $W(q) = J(q)M^{-1}(q)J^T(q)$ is called the **mobility tensor**.

There is often a conceptual advantage to separating the position and force control terms by assuming that a_x is a function only of position and velocity and a_f is a function only of force. However, for simplicity, we shall take $a_f = F_e$ to cancel the environment force and thus recover the task space double integrator system

$$\ddot{x} = a_x \tag{10.18}$$

and we will assume that any additional force feedback terms are included in the outer-loop term a_x. This entails no loss of generality as long as the Jacobian (hence $W(q)$) is invertible. This will become clear later in this chapter.

10.3.1 Impedance Control

In this section we discuss the notion of **impedance control**. We begin with an example that illustrates in a simple way the effect of force feedback.

Example 10.4. *Consider the one-dimensional system in Figure 10.10 consisting of a mass M on a frictionless surface subject to an environmental force F and control input u. The equation of motion of the system is*

$$M\ddot{x} = u - F \tag{10.19}$$

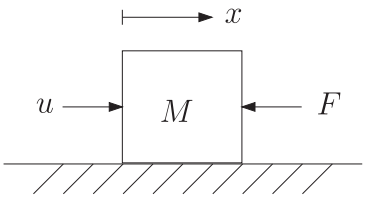


Figure 10.10: A mass M on a frictionless surface, subject to a force F.

With $u = 0$, the object "appears to the environment" as a pure inertia with mass M. Suppose the control input u is chosen as a force feedback term $u = -mF$. Then the closed-loop system is

$$M\ddot{x} = -(1+m)F \implies \frac{M}{1+m}\ddot{x} = -F \tag{10.20}$$

Hence, the object now appears to the environment as an inertia with mass $\frac{M}{1+m}$. Thus, the force feedback has the effect of changing the **apparent inertia** *of the system.*

The idea behind impedance control is to regulate the apparent inertia, and also the apparent damping, and stiffness, through force feedback as in the above example. For instance, in a grinding operation, it may be useful to reduce the apparent stiffness of the end effector normal to the part so that excessively large normal forces are avoided.

Next we show that impedance control may be realized within our standard inner-loop/outer-loop control architecture by a suitable choice of the outer-loop term a_x in Equation (10.18). Let $x^d(t)$ be a reference trajectory defined in task space coordinates and let M_d, B_d, K_d, be 6×6 matrices specifying desired inertia, damping, and stiffness, respectively. Let $\tilde{x}(t) = x(t) - x^d(t)$ be the tracking error in task space and set

$$a_x = \ddot{x}^d - M_d^{-1}(B_d\dot{\tilde{x}} + K_d\tilde{x} + F) \tag{10.21}$$

where F is the measured environmental force. Substituting Equation (10.21) into Equation (10.18) yields the closed-loop system

$$M_d\ddot{\tilde{x}} + B_d\dot{\tilde{x}} + K_d\tilde{x} = -F \tag{10.22}$$

which results in desired impedance properties of the end effector. Note that for $F = 0$ tracking of the reference trajectory, $x^d(t)$, is achieved, whereas for nonzero environmental force, tracking is not achieved, in general. We will address this in the next section.

10.3.2 Hybrid Impedance Control

In this section we introduce the notion of **hybrid impedance control**. We again take as our starting point the linear, decoupled system given by Equation (10.18). The impedance control formulation in the previous section is independent of the environment dynamics. It is reasonable to expect that stronger results may be obtained by incorporating a model of the environment dynamics into the design. For example, we will illustrate below how one may control the manipulator impedance while simultaneously regulating either position or force, which is not possible with the pure impedance control law given by Equation (10.21).

We consider a one-dimensional system representing one component of the outer-loop system (10.18)

$$\ddot{x}_i = a_{x_i} \tag{10.23}$$

and we henceforth drop the subscript i for simplicity. We assume that the impedance Z_e of the environment in this direction is fixed and known, a priori. The impedance of the robot Z_r is of course determined by the control input. The hybrid impedance control design proceeds as follows based on the classification of the environment impedance into inertial, resistive, or capacitive impedances:

1. If the environment impedance $Z_e(s)$ is capacitive, use a Norton network representation. Otherwise, use a Thévenin network representation.[2]

2. Choose a desired robot impedance $Z_r(s)$ and represent it as the **dual** to the environment impedance. Thévenin and Norton networks are considered dual to one another. This implies, in addition, that the robot impedance $Z_r(s)$ should be non-capacitive if $Z_e(s)$ is capacitive (likewise non-inertial if $Z_e(s)$ is inertial).

3. Couple the robot and environment one-ports and design the outer-loop control a_x to achieve the desired impedance of the robot while tracking a reference position or force.

We illustrate this procedure on two examples, a capacitive environment and an inertial environment, respectively.

Example 10.5 (Capacitive Environment)**.** *In the case that the environment impedance is capacitive we have the robot/environment interconnection as shown in Figure 10.11 where the environment one-port is the Norton network*

[2] In fact, for a resistive environment, either representation may be used.

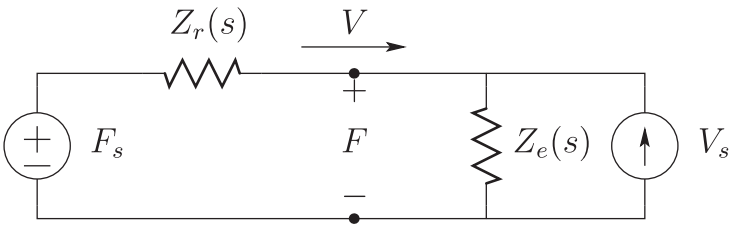


Figure 10.11: Capacitive environment case. The robot impedance is non-capacitive.

and the robot one-port is the Thévenin network. Suppose that $V_s = 0$*, that is, suppose there are no environmental disturbances, and that* F_s *represents a reference force. From the circuit diagram it is straightforward to show that*

$$\frac{F}{F_s} = \frac{Z_e(s)}{Z_e(s) + Z_r(s)} \tag{10.24}$$

Then the steady-state force error e_{ss} *to a step reference force* $F_s = \frac{F^d}{s}$ *is given by the final value theorem as*

$$e_{ss} = \frac{-Z_r(0)}{Z_r(0) + Z_e(0)} = 0 \tag{10.25}$$

since $Z_e(0) = \infty$ *(capacitive environment) and* $Z_r \neq 0$ *(non-capacitive robot).*

The implications of the above calculation are that we can track a constant force reference value, while simultaneously specifying a given impedance Z_r *for the robot.*

In order to realize this result we need to design outer-loop control term a_x *in Equation (10.23) using only position, velocity, and force feedback. This imposes a practical limitation on the achievable robot impedance functions,* Z_r*.*

Suppose that the desired robot impedance Z_r^{-1} *can be written as*

$$Z_r(s) = M_c s + Z_{rem}(s) \tag{10.26}$$

where the remainder term $Z_{rem}(s)$ *is a proper rational function. We now choose the outer-loop term* a_x *as*

$$a_x = -\frac{1}{M_c} Z_{rem} \dot{x} + \frac{1}{M_c}(F_s - F) \tag{10.27}$$

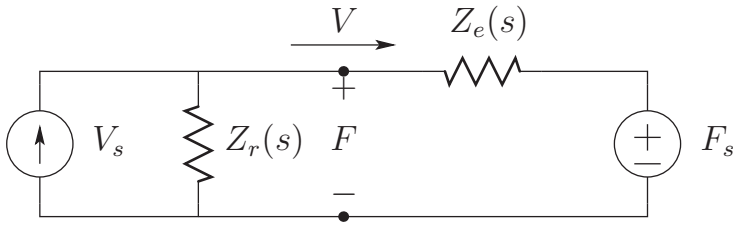


Figure 10.12: Inertial environment case. The robot impedance is non-inertial.

Substituting this into the double integrator system $\ddot{x} = a_x$ *yields*

$$Z_r(s)\dot{x} = F_s - F \tag{10.28}$$

Thus, we have shown that, for a capacitive environment, force feedback can be used to regulate contact force and specify a desired robot impedance simultaneously.

Example 10.6 (Inertial Environment)**.** *In the case that the environment impedance is inertial we have the robot/environment interconnection as shown in Figure 10.12, where the environment one-port is a Thévenin network and the robot one-port is a Norton network. Suppose that* $F_s = 0$ *and that* V_s *represents a reference velocity. From the circuit diagram it is straightforward to see that*

$$\frac{V}{V_s} = \frac{Z_r(s)}{Z_e(s) + Z_r(s)} \tag{10.29}$$

Then the steady-state velocity error e_{ss} *to a step reference velocity command* $V_s = \frac{V^d}{s}$ *is given by the final value theorem as*

$$e_{ss} = \frac{-Z_e(0)}{Z_r(0) + Z_e(0)} = 0 \tag{10.30}$$

since $Z_e(0) = 0$ *(inertial environment) and* $Z_r \neq 0$ *(non-inertial robot).*

To achieve this non-inertia robot impedance we take, as before,

$$Z_r(s) = M_c s + Z_{rem}(s) \tag{10.31}$$

and set

$$a_x = \ddot{x}^d + \frac{1}{M_c} Z_{rem}(\dot{x}^d - \dot{x}) + \frac{1}{M_c} F \tag{10.32}$$

Then, substituting this into the double integrator equation $\ddot{x} = a_x$ *yields*

$$Z_r(s)(\dot{x} - \dot{x}^d) = F \tag{10.33}$$

Thus, we have shown that, for an inertial environment, position control can be used to regulate a reference velocity and specify a desired robot impedance simultaneously.

10.4 Chapter Summary

This chapter covers some of the basic ideas in robot force control. A force control strategy is one that modifies position trajectories based on the sensed force.

Natural and Artificial Constraints

We first described so-called natural and artificial constraints using the notion of reciprocity. Given six dimensional velocity (or twist) and force (or wrench) vectors V and F, respectively, an ideal robot/environment contact task satisfies

$$V^T F = v_x f_x + v_y f_y + v_z f_z + \omega_x n_x + \omega_y n_y + \omega_z n_z = 0$$

In general, the chosen task imposes environmental constraints on six of the above variables. These are the natural constraints. The remaining variables can be arbitrarily assigned artificial constraints that are then maintained by the control system in order to complete the task.

Network Models and Impedance

We next introduced the notion of mechanical impedance to model the realistic case that the robot and environment are not perfectly rigid. The impedance is a measure of the ratio of force and velocity and is analogous to electrical impedance as a ratio of voltage and current. For this reason we introduced one-port network models of mechanical systems and modeled the robot/environment interaction as a connection of one-port networks.

Task Space Dynamics and Control

When the manipulator is in contact with the environment, the dynamic equations must be modified to include the reaction torque $J^T F_e$ corresponding to the end-effector force F_e. Thus, the equations of motion of the manipulator in joint space are given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + J^T(q)F_e \quad = \quad u$$

We therefore introduced a modified inverse dynamics control law of the form

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q) + J^T(q)a_f$$

where a_q and a_f are outer-loop controls with units of acceleration and force, respectively. The resulting system can be written as

$$\ddot{x} = a_x + W(q)(F_e - a_f)$$

where

$$a_x = J(q)a_q + \dot{J}(q)\dot{q}$$

is the outer-loop control in task space and $W(q) = J(q)M^{-1}(q)J^T(q)$ is the mobility tensor.

Impedance Control

Using this model we introduced the notions of impedance control and hybrid impedance control. The impedance control methodology is to design the outer-loop control terms a_x and a_f according to

$$\begin{aligned} a_x &= \ddot{x}^d - M_d^{-1}(B_d\dot{e} + K_d e + F_e) \\ a_f &= F_e \end{aligned}$$

to achieve the closed-loop system

$$M_d\ddot{e} + B_d\dot{e} + K_d e = -F_e \tag{10.34}$$

which results in desired impedance properties of the end effector.

Hybrid Impedance Control

Using our network models we introduced a classification of robot/environment impedance operators $Z(s)$ as

1. Inertial if and only if $|Z(0)| = 0$
2. Resistive if and only if $|Z(0)| = B$ for some constant $0 < B < \infty$
3. Capacitive if and only if $|Z(0)| = \infty$

Using this impedance classification scheme we were able to derive so-called hybrid impedance control laws that allowed us both to regulate impedance and to regulate position and force.

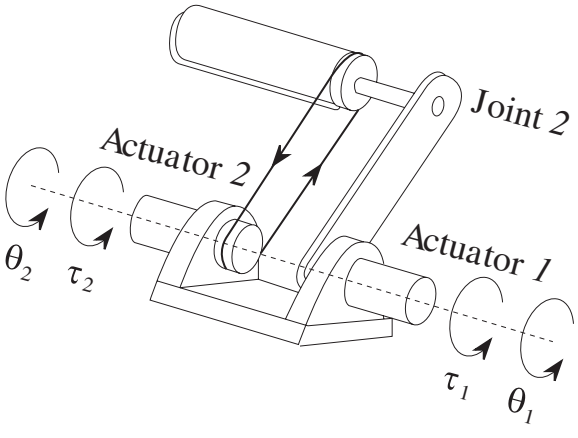


Figure 10.13: Two-link manipulator with remotely driven link.

Problems

10–1 Given the two-link planar manipulator of Figure 4.11, find the joint torques τ_1 and τ_2 corresponding to the end-effector force $[-1, -1]^T$.

10–2 Consider the two-link planar manipulator with remotely driven links shown in Figure 10.13. Find an expression for the motor torques needed to balance a force F at the end effector. Assume that the motor gear ratios are r_1, r_2, respectively.

10–3 What are the natural and artificial constraints for the task of inserting a square peg into a square hole? Sketch the compliance frame for this task.

10–4 Describe the natural and artificial constraints associated with the task of opening a box with a hinged lid. Sketch the compliance frame.

10–5 Discuss the task of opening a long two-handled drawer. How would you go about performing this task with two manipulators? Discuss the problem of coordinating the motion of the two arms. Define compliance frames for the two arms and describe the natural and artificial constraints.

10–6 Given the following tasks, classify the environments as either inertial, capacitive, or resistive according to Definition 10.3.

1. Turning a crank
2. Inserting a peg in a hole

3. Polishing the hood of a car
4. Cutting cloth
5. Shearing a sheep
6. Placing stamps on envelopes
7. Cutting meat

Notes and References

Among the earliest results in robot force control was the work of Mason [113], who introduced the notion of natural and artificial constraints and Raibert and Craig [140], who introduced the notion of hybrid position/force control based on the decomposition of the force control problem into position-controlled and force-controlled directions relative to a compliance frame. Our use of the reciprocity condition is an outgrowth of this early work. The use of twists and wrenches to define global geometric notions in this context was introduced to the robotics community in [38].

The notion of impedance control is due to Hogan [63]. The hybrid impedance control concept and the classification of environments as inertial, resistive, or capacitive is taken from Anderson and Spong [4]. Alternate formulations of force control can be found in [32].

Chapter 11

VISION-BASED CONTROL

In Chapter 10 we described methods to control the forces and torques applied by a manipulator interacting with its environment. Force feedback is most useful when the end effector is in physical contact with the environment. During free motion, such as moving a gripper toward a grasping configuration, force feedback provides no information that can be used to guide the motion of the gripper. In such situations, noncontact sensing, such as computer vision, can be used to control the motion of the end effector relative to environment.

In this chapter we consider the problem of vision-based control. Unlike force control, with vision-based control the quantities of concern are typically not measured directly by the sensor. For example, if the task is to grasp an object, the quantities of concern are pose variables that describe the position of the object and the configuration of the gripper. A vision sensor provides a two-dimensional image of the workspace, but does not explicitly contain any information regarding the pose of the objects in the scene. There is, of course, a relationship between this image and the geometry of the robot's workspace, but the task of inferring the 3D structure of the scene from an image is a difficult one, and a full solution to this problem is not required for most robotic manipulation tasks. The problem faced in vision-based control is that of extracting a relevant set of parameters from an image and using these parameters to control the motion of the manipulator in real time to perform the desired task.

Over the years several approaches have been developed for vision-based control. These vary based on how the image data are used, the relative configuration of camera and manipulator, choices of coordinate systems, etc. We begin this chapter with a discussion of these issues. Following this discussion, we provide a brief introduction to computer vision. We focus

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

on those aspects of computer vision that are directly relevant to vision-based control, namely imaging geometry and features that can be extracted directly from image data. We then develop the differential kinematics that relate camera motion to changes in these features, deriving the so-called **interaction matrix**. We use the interaction matrix to develop image-based control laws, motivated by Lyapunov theory. When introducing these concepts, we focus primarily on **image-based visual servo control** for **eye-in-hand camera systems**. In such systems, the problem is to control the motion of a hand-held camera based on information that can be directly extracted from an image, without resorting to computing a representation of the 3D scene geometry. After developing the main theory for image-based eye-in-hand systems, the remainder of the chapter considers a number of other issues related to vision-based control, including generalization of camera-manipulator configuration, alternative control schemes, and criteria for optimality in control design.

11.1 Design Considerations

A number of questions confront the designer of a vision-based control system. What kind of camera should be used? Should a zoom lens or a lens with fixed focal length be used? How many cameras should be used? Where should the cameras be placed? What image information should be used to control the motion? Should the vision system be used to infer a three-dimensional description of the scene, or should two-dimensional image data be used directly? For the questions of camera and lens selection, in this chapter we will consider only systems that use a single camera with a fixed focal length lens. We briefly discuss the remaining questions below.

11.1.1 Camera Configuration

Perhaps the first decision to be made when constructing a vision-based control system is where to place the camera. There are essentially two options: the camera can be mounted in a fixed location in the workspace or it can be attached to the robot. These are often referred to as **fixed-camera** and **eye-in-hand** configurations, respectively.

With a fixed-camera configuration, the camera is positioned so that it can observe the manipulator and any objects to be manipulated. There are several advantages to this approach. Since the camera position is fixed, the field of view does not change as the manipulator moves. The geometric relationship between the camera and the workspace is fixed, and can be

calibrated offline. One disadvantage to this approach is that as the manipulator moves through the workspace, it can occlude the camera's field of view. This can be particularly important for tasks that require high precision. For example, if an insertion task is to be performed, it may be difficult to find a position from which the camera can view the entire insertion task without occlusion from the end effector. This problem can be ameliorated to some extent by using multiple cameras, but for robots performing tasks in dynamic or cluttered environments, it may be difficult, or even impossible to determine a set of fixed camera positions that can view all relevant aspects of the scene throughout the complete task execution.

With an eye-in-hand system[1], the camera is often attached to the manipulator above the wrist so that the motion of the wrist does not affect the camera motion. In this way, the camera can observe the motion of the end effector at a fixed resolution and without occlusion as the manipulator moves through the workspace. One difficulty that confronts the eye-in-hand configuration is that the geometric relationship between the camera and the workspace changes as the manipulator moves. The field of view can change drastically for even small motion of the manipulator, particularly if the link to which the camera is attached experiences a change in orientation.

For either the fixed-camera or eye-in-hand configuration, motion of the manipulator will produce changes in the images obtained by the camera. The analysis of the relationships between manipulator motion and changes in the image for the two cases is similar mathematically, and in this text we will consider only the case of eye-in-hand systems.

11.1.2 Image-Based vs. Position-Based Approaches

There are two basic ways to approach the problem of vision-based control, and these are distinguished by the way in which the data provided by the vision system are used. These two approaches can also be combined in various ways to yield what are known as partitioned control schemes.

The first approach to vision-based control is known as **position-based visual servo control**. With this approach, the vision data are used to build a partial 3D representation of the world. For example, if the task is to grasp an object, a model of the imaging geometry (such as the pinhole camera model discussed below) can be used to determine the 3D coordinates of the grasp points relative to the camera coordinate frame. For eye-in-hand systems, if these 3D coordinates can be obtained in real time they can be

[1]The terminology **eye-in-hand** is standard within the visual servo community, even though the camera is typically attached to a link of the manipulator, and not directly to the end effector.

used to determine a motion that will reduce the error between the current and desired end-effector pose. The main difficulties with position-based methods are related to the difficulty of building the 3D representation in real time. For example, these methods tend not to be robust with respect to errors in camera calibration. An additional problem arises from the fact that the error signal used by the control algorithm is defined in terms of the end-effector pose. It is possible that a motion could reduce this pose error while simultaneously inducing camera motion that could, for example, cause the object of interest to leave the camera field of view.

A second method known as **image-based visual servo control** uses the image data directly to control the robot motion. An error function is defined in terms of quantities that can be directly measured in an image (for example, image coordinates of points or the orientations of lines in an image) and a control law is constructed that maps this error directly to robot motion. Typically, relatively simple control laws are used to map the image error to robot motion. We will describe image-based control in some detail in this chapter.

It is possible to combine multiple approaches, using different control algorithms to control different aspects of the camera motion. For example, we might use a position-based approach to control the orientation of the camera while using an image-based approach to control its position. Such methods essentially partition the degrees of freedom of the camera motion into disjoint sets, and are thus known as **partitioned methods**. We briefly describe one particular partitioned method in Section 11.7.

11.2 Computer Vision for Vision-Based Control

Many types of images have been used for controlling robot motion, including grayscale images, color images, ultrasound images, and range images. In this chapter, we restrict our attention to the case of two-dimensional grayscale images. Such an image is formed by using a lens to focus light onto a two-dimensional sensor, the most common of which are CMOS[2] and charge-coupled device (CCD) sensors. The sensor consists of a two-dimensional array of individual sensing elements, each of which corresponds to an image **pixel** (derived from the term **picture element**), whose value corresponds to the intensity of the light incident on the particular sensing element. A digital camera is able to transmit these values directly to a computer, for example via a USB interface, where the image can be represented

[2]CMOS stands for complementary metal-oxide-semiconductor, a technology that has been widely adopted for use in integrated circuits.

as a two-dimensional array of intensity values. These intensity values could be represented as integers, 8-bit unsigned integers (implying that pixel values are integers ranging from 0 to 255), floating point numbers, or even double precision floating point numbers.

A typical image can contain several megabytes of data; however, the amount of data contained in an image often far exceeds the information content of the image. For this reason, most vision-based control algorithms begin by identifying useful **features** in the image, typically defined in terms of pixel values in a small region of the image. To be useful for vision-based control, features should be relatively easy to detect, have desirable robustness properties (e.g., invariance with respect to scale, orientation of the camera, lighting, etc.), and be distinctive relative to possible image content (i.e., they should be uniquely recognizable in an image). If features are well chosen, they can be used for object identification, 3D scene reconstruction, or for tracking specific objects as the camera (or the object) moves. For the case of vision-based control, feature identification and tracking are essential steps in relating the changes in an image to the corresponding camera motion.

In the remainder of this section we first describe the geometry of image formation and then discuss approaches to image feature detection and tracking. Later, by investigating how specific features are affected by camera motion, we will derive specific vision-based control algorithms.

11.2.1 The Geometry of Image Formation

For vision-based control, it is typically not necessary to explicitly consider the photometric aspects of image formation, such as issues related to focus, depth of field, or lens distortions. Therefore, in this chapter we describe only the geometry of the image formation process.

Figure 11.1 illustrates the basic geometry of the image formation process under the pinhole lens model, a commonly used approximation for the imaging process. With this model, the lens is considered to be an ideal pinhole that is located at the **focal center** of the lens, also referred to as the **center of projection**. Light rays pass through this pinhole, intersecting the image plane[3].

We define a camera-centric coordinate system as follows. The image

[3]Note that in our mathematical model, illustrated in Figure 11.1, we have placed the pinhole behind the image plane in order to simplify the model. Introductory texts on imaging often place the image plane behind the pinhole, resulting in an "upside-down" image. Our model provides an equivalent geometry, but eliminates the possible confusion of dealing with the "upside-down" image.

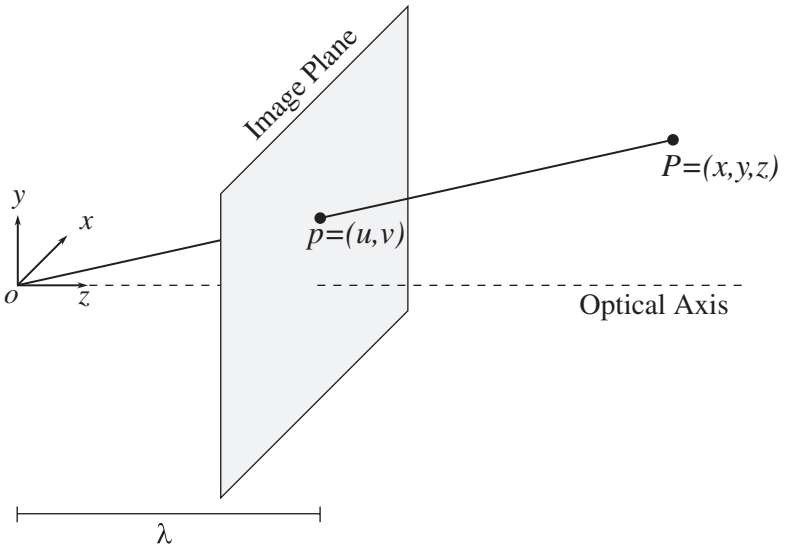


Figure 11.1: The camera coordinate frame is placed at distance λ behind the image plane, with z-axis perpendicular to the image plane and aligned with the optical axis of the lens.

plane is the plane that contains the sensing array. The x-axis and y-axis form a basis for the image plane and are typically taken to be parallel to the horizontal and vertical axes (respectively) of the sensing array. The z-axis is perpendicular to the image plane and aligned with the optical axis of the lens, that is, it passes through the focal center of the lens. The origin of the camera frame is located at a distance λ behind the image plane, referred to as the **focal length** of the camera lens. The point at which the optical axis intersects the image plane is known as the **principal point**. With this assignment of the camera frame, any point in the image plane will have coordinates (u, v, λ). Thus, we can use (u, v) to parameterize the image plane, and we will refer to (u, v) as **image-plane coordinates**.

Let P be a point in the world with coordinates (x, y, z) relative to the camera frame, and let p denote the projection of P onto the image plane with coordinates (u, v, λ). Under the pinhole model, the points P, p, and the origin of the camera frame will be collinear, as shown in Figure 11.1. Thus, for some unknown positive constant k we have

$$k \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix}$$

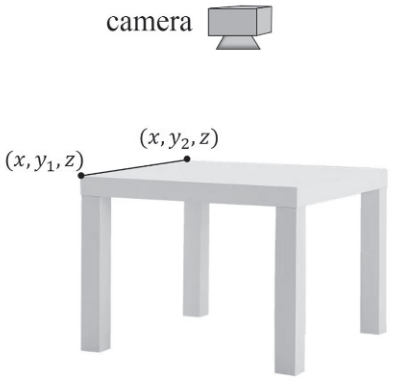


Figure 11.2: Camera viewing a table from overhead, as in Example 11.1.

which can be rewritten as the system of equations

$$kx = u \tag{11.1}$$
$$ky = v \tag{11.2}$$
$$kz = \lambda \tag{11.3}$$

Equation (11.3) implies that $k = \lambda/z$, which can be substituted into Equations (11.1) and (11.2) to obtain

$$u = \lambda\frac{x}{z} \quad ; \quad v = \lambda\frac{y}{z} \tag{11.4}$$

These are the well-known equations for **perspective projection**

Note that the perspective projection equations are expressed in the camera coordinate frame. If coordinates are to be specified with respect to some other frame, for example the end-effector frame or the inertial frame, it is necessary to know the coordinate transformation that relates the camera frame to this other frame. This is achieved by **camera calibration**, which is discussed in Appendix E. It should also be noted that pixels in a digital image have discrete integer coordinates, referred to as **pixel coordinates**. The relationship between pixel coordinates and image-plane coordinates is also discussed in Appendix E.

Example 11.1 (Distance and Scale)**.** *Consider a camera viewing a square table from above, such that the image plane is parallel to the tabletop, and the camera frame axes are parallel to the sides of the table, as shown in Figure 11.2. Let the corners of the left-most edge of the table have coordinates* (x, y_1, z) *and* (x, y_2, z) *w.r.t. the camera frame. Note that the two corners*

Figure 11.3: The boundaries of the various sidewalks in this scene are all parallel, and therefore have a common vanishing point in the image.

share a common x-coordinate because the y-axis of the camera is parallel to the table edge, and share a common z-coordinate because the table is parallel to the image plane. In the image, the length of the edge is easily computed as

$$L = \mid v_2 - v_1 \mid \; = \; \frac{\lambda}{z} \mid y_2 - y_1 \mid$$

and the area of the tabletop in the image is given by

$$A = \; (v_2 - v_1)^2 \; = \; \frac{\lambda^2}{z^2} \, (y_2 - y_1)^2$$

These two relationships illustrate two effects of perspective projection, namely that distance between image points decreases linearly with inverse-depth, and area occupied in an image decreases with the square of the inverse-depth. In vision-based control, these effects often manifest as gain terms in the feedback control design.

Example 11.2 (Vanishing Points)**.** *Consider an environment that contains a collection of parallel lines such as shown in Figure 11.3. Suppose the*

equations for these lines (in world coordinates) are given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \gamma \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

in which $(x_i, y_i, z_i)^T$ *is some point on the* i^{th} *line, and* $n = (n_x, n_y, n_z)^T$ *is the direction of the lines, both specified relative to the camera frame, and* γ *is a parameter that determines distance along the line. Although the lines are parallel, due to the effects of perspective projection, the images of these lines will intersect (except when the lines are parallel to the image plane). This intersection point is known as the* **vanishing point**. *To find the vanishing point, one merely need note that in projective geometry two parallel lines intersect at a point that lies on the line at infinity*[4]. *Thus, we can find the vanishing point by examining the case where* $\gamma \to \infty$.

If we let (u_∞, v_∞) *denote the image-plane coordinates for the vanishing point, we have*

$$\begin{aligned} u_\infty &= \lim_{\gamma \to \infty} \lambda \frac{x}{z} \\ &= \lim_{\gamma \to \infty} \lambda \frac{x_i + \gamma n_x}{z_i + \gamma n_z} \\ &= \lambda \frac{n_x}{n_z} \end{aligned}$$

and by similar reasoning,

$$v_\infty = \lambda \frac{n_y}{n_z}$$

Note that when $n_z = 0$ *the lines are parallel to the image plane, and the images of the lines do not have a finite point of intersection.*

11.2.2 Image Features

Image features are defined by patterns that can be detected in an image, and that have an associated set of measurable attributes. As an example, object corners are often easy to detect in images using specially designed algorithms known as **corner detectors**; their attributes might include the image coordinates of the corner and the angle between the two edges incident at the corner.

[4]The line at infinity extends the real plane to include points at infinity in a manner analogous to adding $+\infty$ and $-\infty$ to the set of real numbers to obtain the extended real number system.

One of the basic problems in visual servo control is to track a set of features through a sequence of images, and to infer things about camera motion from the corresponding changes in the attributes of those features. Thus, to be useful for visual servo control, an image feature should have two properties. First, the feature should convey geometric information about the position and orientation of the camera relative to its environment, and camera motion should correspond in a well-defined way to changes in the attributes of the feature. Second, the feature should be easy to detect and track in an image sequence. For example, if an object corner is used as a feature, it should be easy to detect in an image, and to track in successive frames of an image sequence.

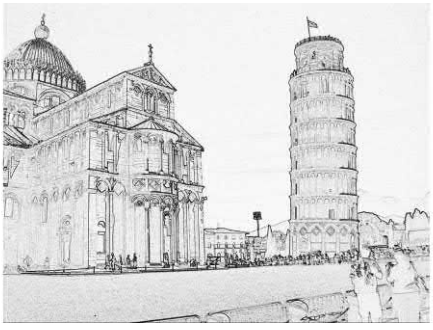
The computer vision literature is replete with examples of features, including features that are based directly on intensity values, features that are based on image gradient information, features that are based on aggregate properties of image regions, and many more. Rather than provide a catalog of possible features, below we focus on the class of gradient-based features. Even in this restricted case, we do not develop the full details, but rather we present a conceptual introduction, leaving details of implementation to the reader. This should not be a significant burden, as numerous feature detection algorithms have been implemented in open-source computer vision software packages that are readily accessible.

Gradient-Based Features

Even though an image consists of a discrete set of pixels whose intensity values are represented as unsigned integers, when defining image features it is often convenient to treat the image as a continuous plane, and to consider the image intensity to be a real-valued function, $I : \mathbb{R}^2 \to \mathbb{R}$, that maps points on this image plane to their corresponding intensity values. Using this approach, we can define image features in terms of the local structure of the function I. The gradient of a function provides a good approximation to its local structure, and therefore a number of image features have been defined in terms of the image gradient, ∇I.

Perhaps the simplest image feature that can be defined in this manner is an image edge. If an image contains an edge, there will be a sharp change in the image intensity in the direction perpendicular to the edge. This gives rise to a gradient with large magnitude. Many edge detection algorithms compute discrete approximations to the image gradient, and classify a pixel as belonging to an edge when the magnitude of the gradient exceeds a given threshold at that pixel. The Sobel edge detector is a well-known method that applies weighted local averaging to a first-order difference approximation of

(a)



(b)

Figure 11.4: The Sobel edge detector applies local image smoothing and a discrete approximation of the gradient operator at each pixel. (a) An intensity image. (b) The result of applying the Sobel operator to the image.

the image gradient. Figure 11.4(a) shows an intensity image, and Figure 11.4(b) shows the result of applying the Sobel operator to the image.

Edges are not particularly good features for visual servo control, because they give good localization information only in the direction perpendicular to the edge. For example, if we slide a small window along an edge in an image, the sub-image contained in that window changes very little in appearance. Thus, edges do not provide good localization information in directions parallel to the edge. Corners, however, have good localization properties, since at a corner the intensity changes significantly in directions that are parallel to, as well as perpendicular to, the feature. Stated another way, if an image contains a corner located at (u_0, v_0), a window centered at (u_0, v_0) will be significantly different from windows centered at $(u_0+\delta_u, v_0+\delta_v)$ for small values of δ_u and δ_v. This observation is the basis for a number of corner detection algorithms.

A common way to evaluate the similarity of two windows that are offset from one another is to evaluate the integral of squared differences between the two windows. Consider a reference window of size $2\ell \times 2\ell$ centered at (u_0, v_0) and a $2\ell \times 2\ell$ target window centered at $(u_0 + \delta_u, v_0 + \delta_v)$, as shown in Figure 11.5. We can evaluate the similarity of these two windows by integrating the squared difference between the intensity value of each point in the reference image and its corresponding point in the shifted image

$$S(\delta_u, \delta_v) = \int_{v_0-\ell}^{v_0+\ell} \int_{u_0-\ell}^{u_0+\ell} [I(u, v) - I(u + \delta_u, v + \delta_v)]^2 du dv \qquad (11.5)$$

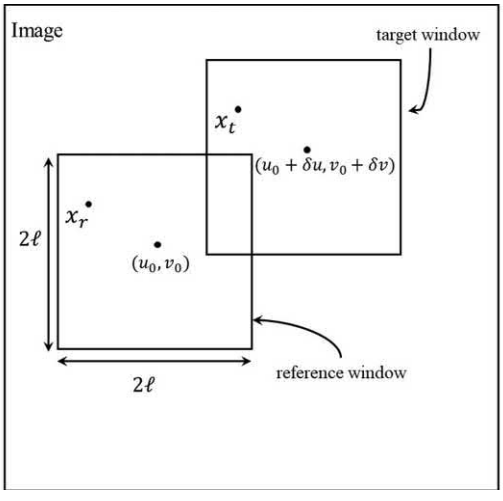


Figure 11.5: The reference window is centered at (u_0, v_0), and the target window is centered at $(u_0 + \delta_u, v_0 + \delta_v)$. For each point x_r in the reference image, the squared-difference integral computes a difference for its corresponding point x_t in the target image.

We can approximate the integrand in Equation (11.5) in terms of the image gradient $\nabla I = (I_u, I_v)^T$. Applying the Taylor series expansion for I about the point (u, v) we obtain

$$I(u + \delta_u, v + \delta_v) \approx I(u, v) + I_u(u, v)\delta_u + I_v(u, v)\delta_v \tag{11.6}$$

and substituting this into Equation (11.5) we obtain

$$S(\delta_u, \delta_v) \approx \int_{v_0-\ell}^{v_0+\ell} \int_{u_0-\ell}^{u_0+\ell} \left(I_u(u, v)\delta_u + I_v(u, v)\delta_v\right)^2 dudv \tag{11.7}$$

$$= \begin{bmatrix} \delta_u & \delta_v \end{bmatrix} M \begin{bmatrix} \delta_u \\ \delta_v \end{bmatrix} \tag{11.8}$$

in which M, called the **second moment matrix**, or **structure tensor**, is given by

$$M = \begin{bmatrix} \int I_u^2(u, v) & \int I_u(u, v)I_v(u, v) \\ \int I_u(u, v)I_v(u, v) & \int I_v^2(u, v) \end{bmatrix} \tag{11.9}$$

Note that M is defined in terms of the image gradient at the point (u, v), and does not depend on δ_u or δ_v. Thus, M is a gradient-based feature descriptor

that characterizes the local variation of the image intensity about the image point (u, v).

Since M is a symmetric positive semi-definite matrix (this follows from the fact that the integrand in Equation (11.7) is positive, except for the degenerate case in which I is constant over the window), Equation (11.8) is a quadratic form that defines an ellipse, and which can be characterized by its eigenvalues $\lambda_1 \geq \lambda_2$ (see, for example, Section 4.12). If λ_1 and λ_2 are both large, then there is significant variation in all directions, indicating the likelihood of a corner at the point (u_0, v_0). If λ_1 is large and $\lambda_2 \approx 0$, then there is little variation along one direction, corresponding to an edge at the point (u_0, v_0). If both $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$, then there is very little variation in the image around the point (u_0, v_0), indicating an absence of useful features.

For images comprised of discrete pixels, the matrix M is computed using summations over discrete pixel windows, for an appropriate discrete approximation to the image gradient.

Remark 11.1. *Features that can be described using variations of the structure tensor described above are at the heart of several well-known feature detectors, including the Harris and Shi-Tomasi feature detectors. The* **histogram of oriented gradients (HOG)** *is a gradient-based feature descriptor that does not rely on the structure tensor.*

Feature Detection and Tracking

The two problems of feature detection and feature tracking are different but closely related. The feature **detection** problem is to find a particular feature in an image, given no prior knowledge about the location of the feature. In contrast, the feature **tracking** problem is to find a feature in the current image, assuming that it has been detected in the previous image. In the latter case, the feature location in the previous image, along with any known information about camera motion, can be used to guide the feature detection in the current image. If a model of camera motion is given (even an uncertain model), the tracking problem can be approached using standard techniques from estimation theory, such as the Kalman filter or extended Kalman filter (EKF). With this approach, the position and orientation of the feature (or the camera) can be considered as the state, and the process and observation models are derived using the camera motion model and image formation model. While numerous methods for feature tracking have been developed, these lie beyond the scope of this text.

11.3 Camera Motion and the Interaction Matrix

Once we have chosen a set of features and designed a feature tracker, it is possible detect a feature as it moves in an image sequence, and to compute the values of its various attributes in each image frame. For vision-based control, it is useful to relate camera motion to the changes in these feature attributes. In this section, we develop the mathematical formulation for this relationship.

Recall the inverse velocity problem discussed in Chapter 4. Even though the inverse kinematics problem is difficult to solve and often ill-posed, the inverse velocity problem is typically fairly easy to solve: one merely inverts the manipulator Jacobian matrix, assuming the Jacobian is nonsingular. This can be understood mathematically by noting that while the inverse kinematic equations represent a nonlinear mapping between possibly complicated geometric spaces, for a given configuration q, the mapping of velocities is a linear map between linear subspaces. For example, for the two-link planar arm, the inverse kinematic equations map an end-effector position, $(x, y) \in \mathbb{R}^2$ to the joint variables (θ_1, θ_2) that lie on the surface of a torus; however the inverse velocity relationship maps a velocity vector in $\mathbb{R}^2$ to a velocity that lies in the tangent plane to the torus at (θ_1, θ_2). Likewise, the relationship between the camera velocity and the corresponding differential changes in the attributes of image features is a linear mapping between linear subspaces. We will now give a more rigorous explanation of this basic idea.

Let $s(t)$ denote the vector of feature attributes that can be measured in an image. Its derivative $\dot{s}(t)$ is referred to as an **image feature velocity**. For example, if a single image point is used as a feature, we could consider the image-plane coordinates of the point as a feature attribute, giving

$$s(t) = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}$$

In this case $\dot{s}(t)$ would be the image plane velocity of the image point.

The image feature velocity is linearly related to the camera velocity. Let the camera velocity ξ consist of linear velocity[5] v and angular velocity ω

$$\xi = \begin{bmatrix} \mathrm{v} \\ \omega \end{bmatrix} \tag{11.10}$$

so that the origin of the camera frame is moving with linear velocity v and the camera frame is rotating about the axis ω, which passes through the

[5]To avoid confusion with image coordinates u, v, in this chapter we use the symbol v to denote linear velocity.

origin of the camera frame. There is no difference between ξ as used here and as used in Chapter 4; in each case, ξ encodes the linear and angular velocity of a moving frame. In Chapter 4 the frame was attached to the end effector while here it is attached to the moving camera.

The relationship between $\dot{s}$ and ξ is given by

$$\dot{s} = L(s, q)\xi \tag{11.11}$$

Here, the matrix $L(s, q)$ is known as the **interaction matrix**. The interaction matrix is a function of both the configuration of the robot, as was also true for the manipulator Jacobian described in Chapter 4, and of the image feature values s.

The interaction matrix L is also called the **image Jacobian matrix**. This is due, at least in part, to the analogy that can be drawn between the manipulator Jacobian discussed in Chapter 4 and the interaction matrix. In each case, a velocity ξ is related to the differential change in a set of parameters, either joint velocities or image feature velocities, by a linear transformation. Strictly speaking, the interaction matrix is not a Jacobian matrix, since ξ is not actually the derivative of some set of pose parameters. However, using techniques analogous to those used to develop the analytic Jacobian in Section 4.8, it is straightforward to construct an actual Jacobian matrix that represents a linear transformation from the derivatives of a set of pose parameters to the image feature velocities, which are derivatives of the image feature attributes.

The specific form of the interaction matrix depends on the features that are used to define s. The simplest features are coordinates of points in the image, and we will focus our attention on this case.

11.4 The Interaction Matrix for Point Features

In this section we derive the interaction matrix for the case of a moving camera observing a point that is fixed in space. This scenario is useful for postioning a camera relative to some object that is to be manipulated. For example, a camera can be attached to a manipulator that will be used to grasp a stationary object. Vision-based control can then be used to bring the manipulator to a grasping configuration that may be defined in terms of image features. In Section 11.4.4 we extend the development to the case of multiple feature points.

At time t, the orientation of the camera frame is given by a rotation matrix $R_c^0 = R(t)$, which specifies the orientation of the camera frame relative to the fixed frame. We denote by $o(t)$ the position of the origin of the

camera frame relative to the fixed frame. We denote by P the fixed point in the workspace, and by $s = [u, v]^T$ the feature vector corresponding to the projection of P in the image (see Figure 11.1).

Our goal is to derive the interaction matrix L that relates the velocity of the camera ξ to the derivatives of the coordinates of the projection of the point in the image $\dot{s}$. We begin by finding an expression for the velocity of the point P relative to the moving camera. We then use the perspective projection equations to relate this velocity to the image velocity $\dot{s}$. Finally, after a bit of algebraic manipulation we arrive to the interaction matrix that satisfies $\dot{s} = L\xi$.

11.4.1 Velocity Relative to a Moving Frame

We now derive an expression for the velocity of a fixed point in the world frame relative to a moving camera frame. We denote by p^0 the coordinates of P relative to the world frame. Note that p^0 does not vary with time, since P is fixed with respect to the world frame. If we denote by $p^c(t)$ the coordinates of P relative to the moving camera frame at time t, using Equation (2.58) we have

$$p^0 = R(t)p^c(t) + o(t)$$

Thus, at time t we can solve for the coordinates of P relative to the camera frame by

$$p^c(t) = R^T(t)\left[p^0 - o(t)\right] \tag{11.12}$$

Now, to find the velocity of the point P relative to the moving camera frame we merely differentiate this equation, as was done in Chapter 4. We will drop the explicit reference to time in these equations to simplify notation. Using the product rule for differentiation we obtain

$$\frac{d}{dt}p^c = \dot{R}^T\left(p^0 - o\right) - R^T\dot{o} \tag{11.13}$$

From Equation (4.18) we have $\dot{R} = S(\omega)R$, and thus $\dot{R}^T = R^T S(\omega)^T = -R^T S(\omega)$. This allows us to write Equation (11.13) as

$$\begin{aligned}\dot{R}^T\left(p^0 - o\right) - R^T\dot{o} &= -R^T S(\omega)\left(p^0 - o\right) - R^T\dot{o} \\ &= -R^T S(\omega)RR^T\left(p^0 - o\right) - R^T\dot{o} \\ &= -R^T\omega \times R^T\left(p^0 - o\right) - R^T\dot{o}\end{aligned}$$

In this equation the rotation matrix R^T is applied to three vectors, yielding three new vectors whose coordinates are expressed with respect to the

camera frame. From Equation (11.12) we see that $R^T(p^0 - o) = p^c$. The vector ω gives the angular velocity vector for the moving frame in coordinates expressed with respect to the fixed frame, that is, $\omega = \omega^0$. Therefore, $R^T\omega = R_0^c\omega^0 = \omega^c$ gives the angular velocity vector for the moving frame in coordinates expressed with respect to the moving frame. Finally, note that $R^T\dot{o} = \dot{o}^c$. Using these conventions we can immediately write the equation for the velocity of P relative to the moving camera frame

$$\dot{p}^c = -\omega^c \times p^c - \dot{o}^c \tag{11.14}$$

Relating this to the velocity ξ we see that ω^c is the angular velocity of the camera frame expressed relative to the moving camera frame, and $\dot{o}^c$ is the linear velocity v of the camera frame, also expressed relative to the moving camera frame. It is interesting to note that the velocity of a fixed point relative to a moving frame is merely -1 times the velocity of a moving point relative to a fixed frame.

Example 11.3 (Camera Motion in the Plane)**.** *Consider a camera whose optical axis is parallel to the world z-axis. If the camera motion is constrained to rotation about the optical axis and translation parallel to the x-y plane, we have*

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad o(t) = \begin{bmatrix} x \\ y \\ z_0 \end{bmatrix}$$

in which z_0 is the fixed height of the camera frame relative to the world frame. This gives

$$\omega^c = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}, \quad \dot{o}^c = \begin{bmatrix} \mathrm{v}_x \\ \mathrm{v}_y \\ 0 \end{bmatrix}$$

If the point P has coordinates (x, y, z) relative to the camera frame, we have

$$\dot{p}^c = -\omega^c \times p^c - \dot{o}^c = \begin{bmatrix} y\dot{\theta} - \mathrm{v}_x \\ -x\dot{\theta} - \mathrm{v}_y \\ 0 \end{bmatrix}$$

11.4.2 Constructing the Interaction Matrix

Using Equation (11.14) and the equations of perspective projection, it is not difficult to derive the interaction matrix for point features. To simplify notation, we define the coordinates for P relative to the camera frame as $p^c =$

$[x, y, z]^T$. By this convention, the velocity of P relative to the moving camera frame is merely the vector $\dot{p}^c = [\dot{x}, \dot{y}, \dot{z}]^T$. We will denote the coordinates for the angular velocity vector by $\omega^c = [\omega_x, \omega_y, \omega_z]^T = R^T\omega$, where ω is the camera angular velocity expressed relative to the world coordinate frame. To further simplify notation, we assign coordinates $R^T\dot{o} = [\mathrm{v}_x, \mathrm{v}_y, \mathrm{v}_z]^T = \dot{o}^c$. Using these conventions, we can write Equation (11.14) as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = -\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} \mathrm{v}_x \\ \mathrm{v}_y \\ \mathrm{v}_z \end{bmatrix}$$

which can be written as the system of three equations

$$\dot{x} = y\omega_z - z\omega_y - \mathrm{v}_x \tag{11.15}$$

$$\dot{y} = z\omega_x - x\omega_z - \mathrm{v}_y \tag{11.16}$$

$$\dot{z} = x\omega_y - y\omega_x - \mathrm{v}_z \tag{11.17}$$

Since u and v are the image coordinates of the projection of P in the image, using Equation (11.4), we can express x and y as

$$x = \frac{uz}{\lambda}, \qquad y = \frac{vz}{\lambda}$$

Substituting these into Equations (11.15)–(11.17) we obtain

$$\dot{x} = \frac{vz}{\lambda}\omega_z - z\omega_y - \mathrm{v}_x \tag{11.18}$$

$$\dot{y} = z\omega_x - \frac{uz}{\lambda}\omega_z - \mathrm{v}_y \tag{11.19}$$

$$\dot{z} = \frac{uz}{\lambda}\omega_y - \frac{vz}{\lambda}\omega_x - \mathrm{v}_z \tag{11.20}$$

These equations express the velocity $\dot{p}^c$ in terms of the image coordinates u, v, the depth z of the point P, and the angular and linear velocity of the camera. We will now find expressions for $\dot{u}$ and $\dot{v}$ and then combine these with Equations (11.18)–(11.20).

Using the quotient rule for differentiation with the equations of perspective projection we obtain

$$\dot{u} = \frac{d}{dt}\frac{\lambda x}{z} = \lambda\frac{z\dot{x} - x\dot{z}}{z^2}$$

Substituting Equations (11.18) and (11.20) into this expression gives

$$\begin{aligned} \dot{u} &= \frac{\lambda}{z^2}\left(z\left[\frac{vz}{\lambda}\omega_z - z\omega_y - \mathrm{v}_x\right] - \frac{uz}{\lambda}\left[\frac{uz}{\lambda}\omega_y - \frac{vz}{\lambda}\omega_x - \mathrm{v}_z\right]\right) \\ &= -\frac{\lambda}{z}\mathrm{v}_x + \frac{u}{z}\mathrm{v}_z + \frac{uv}{\lambda}\omega_x - \frac{\lambda^2 + u^2}{\lambda}\omega_y + v\omega_z \end{aligned} \tag{11.21}$$

We can apply the same technique for $\dot{v}$ to obtain

$$\dot{v} = \frac{d}{dt}\frac{\lambda y}{z} = \lambda\frac{z\dot{y} - y\dot{z}}{z^2}$$

and substituting Equations (11.19) and (11.20) into this expression gives

$$\begin{aligned}\dot{v} &= \frac{\lambda}{z^2}\left(z\left[-\frac{uz}{\lambda}\omega_z + z\omega_x - \mathrm{v}_y\right] - \frac{vz}{\lambda}\left[\frac{uz}{\lambda}\omega_y - \frac{vz}{\lambda}\omega_x - \mathrm{v}_z\right]\right)\\ &= -\frac{\lambda}{z}\mathrm{v}_y + \frac{v}{z}\mathrm{v}_z + \frac{\lambda^2+v^2}{\lambda}\omega_x - \frac{uv}{\lambda}\omega_y - u\omega_z \qquad (11.22)\end{aligned}$$

Equations (11.21) and (11.22) can be combined and written in matrix form as

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & v\\ 0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2+v^2}{\lambda} & -\frac{uv}{\lambda} & -u\end{bmatrix}\begin{bmatrix}\mathrm{v}_x\\ \mathrm{v}_y\\ \mathrm{v}_z\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \qquad (11.23)$$

The matrix in Equation (11.23) is the interaction matrix for a point. To make explicit its dependence on u, v, and z, Equation (11.23) is often written as

$$\dot{s} = L_p(u, v, z)\xi \qquad (11.24)$$

Example 11.4 (Camera Motion in the Plane)**.** *Returning to the situation described in Example 11.3. Suppose that the point P has coordinates $p^0 = [x_p, y_p, 0]^T$ relative to the world frame. Relative to the camera frame, P has coordinates given by*

$$\begin{aligned}p = R^T(p^0 - o) &= \begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\left(\begin{bmatrix}x_p\\ y_p\\ 0\end{bmatrix} - \begin{bmatrix}x_c\\ y_c\\ z_0\end{bmatrix}\right)\\ &= \\ &= \begin{bmatrix}\cos\theta(x_p - x_c) + \sin\theta(y_p - y_c)\\ -\sin\theta(x_p - x_c) + \cos\theta(y_p - y_c)\\ -z_0\end{bmatrix}\end{aligned}$$

The image coordinates for P are thus given by

$$\begin{aligned}u &= -\lambda\frac{\cos\theta(x_p - x_c) + \sin\theta(y_p - y_c)}{z_0}\\ v &= -\lambda\frac{-\sin\theta(x_p - x_c) + \cos\theta(y_p - y_c)}{z_0}\end{aligned}$$

These can be substituted into Equation (11.24) to yield

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & v \\ 0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2+v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix} \begin{bmatrix} \mathrm{v}_x \\ \mathrm{v}_y \\ 0 \\ 0 \\ 0 \\ \dot{\theta} \end{bmatrix}$$

$$= \frac{\lambda}{z_0} \begin{bmatrix} \mathrm{v}_x + (\sin\theta(x_p - x_c) - \cos\theta(y_p - y_c))\dot{\theta} \\ \mathrm{v}_y + (\cos\theta(x_p - x_c) + \sin\theta(y_p - y_c))\dot{\theta} \end{bmatrix}$$

11.4.3 Properties of the Interaction Matrix for Points

Equation (11.23) can be decomposed as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} \\ 0 & -\frac{\lambda}{z} & \frac{v}{z} \end{bmatrix} \begin{bmatrix} \mathrm{v}_x \\ \mathrm{v}_y \\ \mathrm{v}_z \end{bmatrix} + \begin{bmatrix} \frac{uv}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & v \\ \frac{\lambda^2+v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

which can be written as

$$L_{\mathrm{v}}(u, v, z)\mathrm{v} + L_{\omega}(u, v)\omega$$

in which $L_{\mathrm{v}}(u, v, z)$ contains the first three columns of the interaction matrix, and is a function of both the image coordinates of the point and its depth, while $L_{\omega}(u, v)$ contains the last three columns of the interaction matrix, and is a function of only the image coordinates of the point, that is, it does not depend on depth. This can be particularly beneficial in real-world situations when the exact value of z may not be known. In this case, errors in the value of z merely cause a scaling of the matrix $L_{\mathrm{v}}(u, v, z)$, and this kind of scaling effect can be compensated for by using fairly simple control methods. This kind of decomposition is at the heart of the partitioned method that we discuss in Section 11.7.

For a camera with six degrees of freedom (e.g., a camera attached to the end effector of a six-link arm), we have $\xi \in \mathbb{R}^6$, while only the two values u and v are observed in the image. Thus, one would expect that not all camera motions cause observable changes in the image. More precisely, $L \in \mathbb{R}^{2\times 6}$ and therefore has a null space of dimension 4. Therefore, the system

$$0 = L(s, q)\xi$$

has solution vectors ξ that lie in a four-dimensional subspace of $\mathbb{R}^6$. For the case of a single point, it can be shown (Problem 11–12) that the null space of the interaction matrix given in Equation (11.23) is spanned by the four vectors

$$\begin{bmatrix} u \\ v \\ \lambda \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ u \\ v \\ \lambda \end{bmatrix}, \begin{bmatrix} uvz \\ -(u^2+\lambda^2)z \\ \lambda vz \\ -\lambda^2 \\ 0 \\ u\lambda \end{bmatrix}, \begin{bmatrix} \lambda(u^2+v^2+\lambda^2)z \\ 0 \\ -u(u^2+v^2+\lambda^2)z \\ uv\lambda \\ -(u^2+\lambda^2)z \\ u\lambda^2 \end{bmatrix} \tag{11.25}$$

The first two of these vectors have particularly intuitive interpretations. The first corresponds to motion of the camera frame along a line that contains both the lens focal center and the point P. Such lines are called **projection rays**. The second corresponds to rotation of the camera frame about a projection ray that contains P.

11.4.4 The Interaction Matrix for Multiple Points

It is straightforward to generalize the development above to the case in which several points are used to define the image feature vector. Consider the case for which the feature vector consists of the coordinates of n image points. Here, the i^{th} feature point has an associated depth z_i and we define the feature vector s and the vector of depth values z by

$$s = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}, \qquad z = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

For this case, the composite interaction matrix L_c that relates camera velocity to image feature velocity is a function of the image coordinates of the n points and also of the n depth values, is obtained by stacking the n

interaction matrices for the individual feature points,

$$
\begin{aligned}
\dot{s} &= L_c(s,z)\xi \\
&= \begin{bmatrix} L_1(u_1,v_1,z_1) \\ \vdots \\ L_n(u_n,v_n,z_n) \end{bmatrix} \xi \\
&= \begin{bmatrix}
-\frac{\lambda}{z_1} & 0 & \frac{u_1}{z_1} & \frac{u_1 v_1}{\lambda} & -\frac{\lambda^2+u_1^2}{\lambda} & v_1 \\
0 & -\frac{\lambda}{z_1} & \frac{v_1}{z_1} & \frac{\lambda^2+v_1^2}{\lambda} & -\frac{u_1 v_1}{\lambda} & -u_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\frac{\lambda}{z_n} & 0 & \frac{u_n}{z_n} & \frac{u_n v_n}{\lambda} & -\frac{\lambda^2+u_n^2}{\lambda} & v_n \\
0 & -\frac{\lambda}{z_n} & \frac{v_n}{z_n} & \frac{\lambda^2+v_n^2}{\lambda} & -\frac{u_n v_n}{\lambda} & -u_n
\end{bmatrix} \xi
\end{aligned}
$$

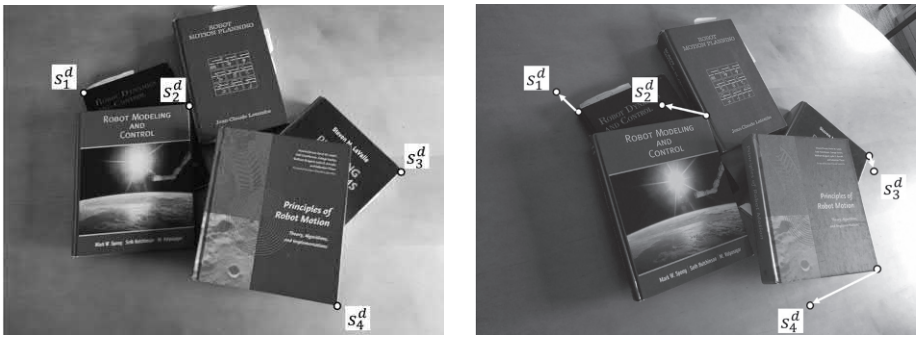
Thus, we have $L_c \in \mathbb{R}^{2n\times 6}$. Since each image point gives rise to two feature values (the u, v coordinates of the point), three image points provide six feature values, which is sufficient to solve for ξ given the image measurements $\dot{s} \in \mathbb{R}^6$, provided that L_c is full rank.

11.5 Image-Based Control Laws

With image-based control, the goal configuration is defined by a desired configuration of image features, denoted by $s^d \in \mathbb{R}^k$, where the dimension k depends on the task. The image error function is then given by

$$e(t) = s(t) - s^d$$

The image-based control problem is to find a mapping from this error function to a commanded camera motion. As we have seen in previous chapters, there are a number of control approaches that can be used to determine the joint-level inputs to achieve a desired trajectory. Therefore, in this chapter we will treat the manipulator as a kinematic positioning device, that is, we will ignore manipulator dynamics and develop controllers that compute desired end-effector trajectories. The underlying assumption is that these trajectories can then be tracked by a lower level manipulator controller.



(a) (b)

Figure 11.6: The goal image is shown on the left. When the camera reaches the desired configuration, the features $s_1^d, s_2^d, s_3^d, s_4^d$ will be located as shown in this image. The initial image is shown on the right, along with four error vectors that correspond to the four image features.

Example 11.5 (A Simple Positioning Task)**.** *Visual servo tasks are often specified by defining the desired location of a set of features in an image. Consider the task of positioning a camera relative to a pile of books, as shown in Figure 11.6. For this example, we define the task by specifying the location of four features, $s_1^d, s_2^d, s_3^d, s_4^d$, in the goal image, as shown in Figure 11.6(a). Figure 11.6(b) shows an initial image, and the corresponding error vectors for each of the four features. An image-based control law will map these error vectors to a corresponding camera motion that will reduce the error.*

The most common approach to image-based control is to compute a desired camera velocity ξ and use this as the control input. Relating image feature velocities to the camera velocity is typically done by solving $\dot{s} = L(s, q)\xi$ for ξ, which gives the camera velocity that will produce a desired value for $\dot{s}$. In some cases, this can be done simply by inverting the interaction matrix, but in other cases the pseudoinverse must be used. Below we describe various pseudoinverses of the interaction matrix and then explain how these can be used to construct an image-based control law.

11.5.1 Computing Camera Motion

For the case of k feature values and m components of the camera body velocity ξ, we have $L \in \mathbb{R}^{k \times m}$. In general we will have $m = 6$, but in some cases we may have $m < 6$, for example if the camera is attached to a SCARA arm used to manipulate objects on a moving conveyor. When L is full rank

(i.e., when $\text{rank}(L) = \min(k, m)$), it is possible to compute ξ for a given $\dot{s}$. There are three cases that must be considered: $k = m$, $k > m$, and $k < m$.

- When $k = m$ and L is full rank, we have $\xi = L^{-1}\dot{s}$. This is the case when the number of feature values is equal to the number of degrees of freedom for the camera.

- When $k < m$, L^{-1} does not exist, and the system is underconstrained. In the visual servo application, this implies that we are not observing enough feature velocities to uniquely determine the camera motion ξ, that is, there are certain components of the camera motion that cannot be observed (namely, those that lie in the null space of L). In this case we can compute a solution given by

$$\xi = L^{+}\dot{s} + (I_m - L^{+}L)b$$

  where L^{+} is the right pseudoinverse for L, given by

$$L^{+} = L^{T}(LL^{T})^{-1}$$

  I_m is the $m \times m$ identity matrix, and $b \in \mathbb{R}^m$ is an arbitrary vector. Note the similarity between this equation and Equation (4.112) which gives the solution for the inverse velocity problem (that is, solving for joint velocities to achieve a desired end-effector velocity) for redundant manipulators.

  In general, for $k < m$, $(I - LL^{+}) \neq 0$, but all vectors of the form $(I - LL^{+})b$ lie in the null space of L, which implies that those components of the camera velocity that are unobservable lie in the null space of L. If we let $b = 0$, we obtain the value for ξ that minimizes the norm

$$\|\dot{s} - L\xi\|$$

- When $k > m$ and L is full rank, we will typically have an inconsistent system, especially when the feature values s are obtained from measured image data. In the visual servo application, this implies that we are observing more feature velocities than are required to uniquely determine the camera motion ξ. In this case the rank of the null space of L is zero, since the dimension of the column space of L equals $\text{rank}(L)$. In this situation, we can use the least squares solution

$$\xi = L^{+}\dot{s} \tag{11.26}$$

in which we use the left pseudoinverse, which is given by

$$L^+ = (L^T L)^{-1} L^T \tag{11.27}$$

11.5.2 Proportional Control Schemes

Lyapunov theory (see Appendix C) can be used to analyze the stability of dynamical systems, but it can also be used to aid in the design of stable control systems. Consider again the visual servo problem given by

$$\begin{aligned} \dot{s} &= L(s,q)\xi \\ e(t) &= s(t) - s^d \end{aligned}$$

and define a candidate Lyapunov function as

$$V(t) = \frac{1}{2}\|e(t)\|^2 = \frac{1}{2}e^T e$$

Note that this choice of V is a valid candidate Lyapunov function only if $e \neq 0$ for every non-goal configuration (this condition is not satisfied, for example, when there is a nontrivial null space for L in a neighborhood of $e = 0$). The derivative of this function is

$$\dot{V} = \frac{d}{dt}\frac{1}{2}e^T e = e^T \dot{e}$$

Thus, if we could design a controller such that

$$\dot{e} = -\lambda e \tag{11.28}$$

with $\lambda > 0$ we would have

$$\dot{V} = -\lambda e^T e = -2\lambda V < 0$$

and this would ensure asymptotic stability of the closed-loop system. In fact, if we could design such a controller, we would have exponential stability, which ensures that the closed-loop system is asymptotically stable even under small perturbations, for example, small errors in camera calibration.

For the case of visual servo control, it is often possible to design such a controller. In the case of s^d a constant[6], the derivative of the error function is given by

$$\dot{e}(t) = \frac{d}{dt}(s(t) - s^d) = \dot{s}(t) = L\xi$$

[6]When $s^d(t)$ is time-varying, it is necessary to design a trajectory tracking control law.

and substituting this into Equation (11.28) we obtain

$$-\lambda e(t) = L\xi$$

If $k = m$ and L has full rank, then L^{-1} exists, and we have

$$\xi = -\lambda L^{-1} e(t)$$

and the system is exponentially stable.

When $k > m$ we obtain the control

$$\xi = -\lambda L^{+} e(t)$$

with $L^{+} = (L^T L)^{-1} L^T$. Unfortunately, in this case we do not obtain exponential stability. To see this, consider again the Lyapunov function given above. We have

$$\dot{V} = e^T \dot{e} = e^T L\xi = -\lambda e^T L L^{+} e$$

But in this case, the matrix LL^{+} is only positive semidefinite, not positive definite, and therefore we cannot demonstrate asymptotic stability by Lyapunov theory. This follows because $L^{+} \in \mathbb{R}^{m \times k}$, and since $k > m$, it has a nonzero nullspace. This follows since $\text{rank}(L) = \min(k, m) < k$ implies that L has at most $m < k$ linearly independent columns, and therefore there must exist some linear combination of columns $\sum_i \alpha_i L_i = 0$ such that not all $\alpha_i = 0$. The corresponding vector $[\alpha_1 \dots \alpha_k]^T$ lies in the null space of L. Therefore, $eLL^{+}e = 0$ for certain values of e, and we can demonstrate only stability, not asymptotic stability.

Remark 11.2. *In practice, we will not know the exact value of L or L^{+} since these depend on knowledge of depth information that must be estimated by the computer vision system. In this case, we will have an estimate for the interaction matrix $\widehat{L}$ and we can use the control $\xi = -\widehat{L}^{+} e(t)$. It is easy to show, by a proof analogous to the one above, that the resulting visual servo system will be stable when $L\widehat{L}^{+}$ is positive definite. This helps explain the robustness of image-based control methods to calibration errors in the computer vision system.*

11.5.3 Performance of Image-Based Control Systems

While the image-based control law described above performs well with respect to the image error, it can sometimes induce large camera motions that cause task failure, for example, if the required camera motion exceeds the

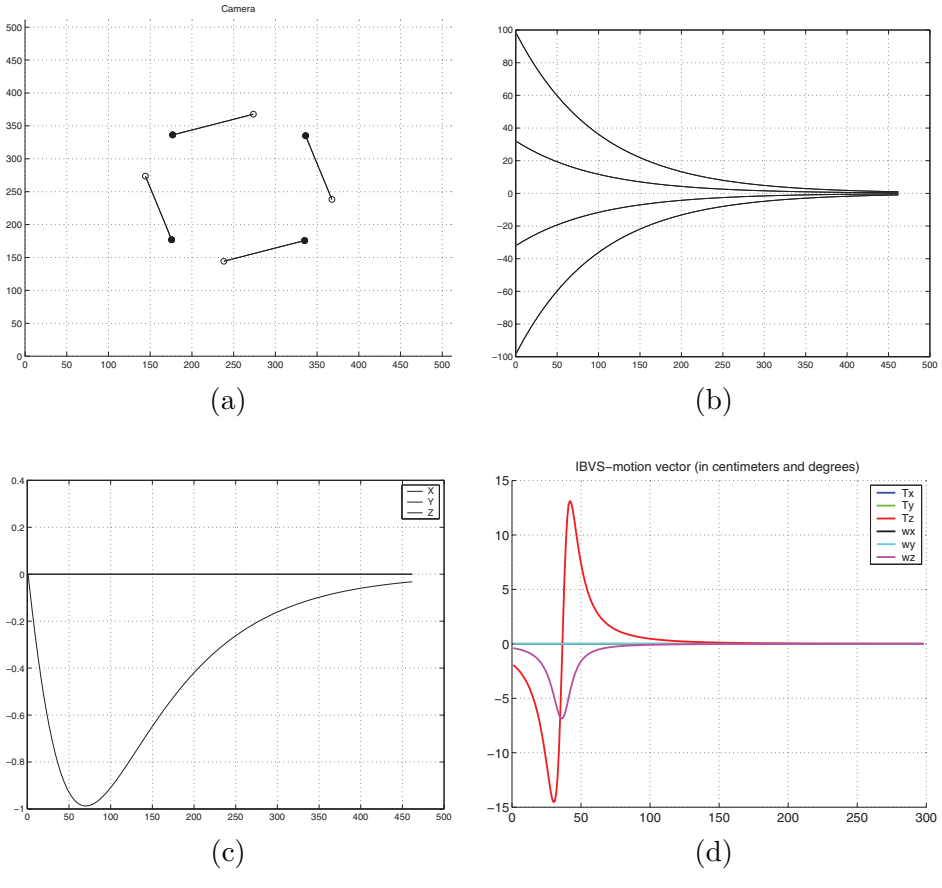
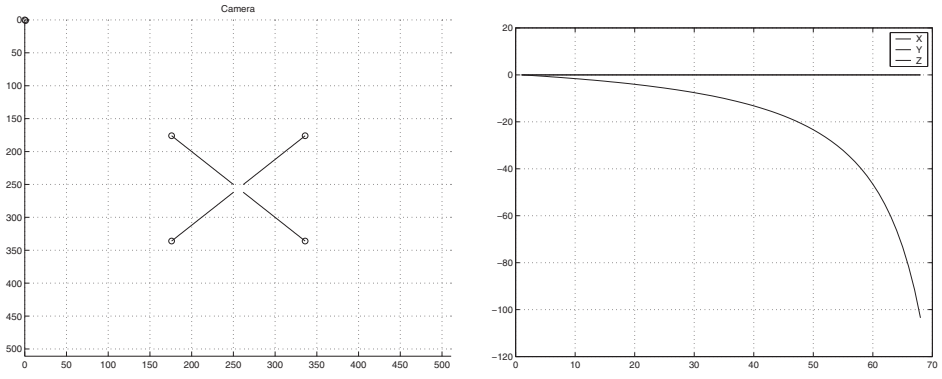


Figure 11.7: In (a) the desired feature point locations are shown in dark circles, and the initial feature locations are shown as unfilled circles. As can be seen in (a), when image-based control is used, the feature points move on straight-line trajectories in the image to the desired positions, while the image error decreases exponentially to zero, as shown in part (b). Unfortunately, the required camera motion includes a significant retreat along the camera z-axis, as illustrated in (c) and (d).

physical range of the manipulator. Such a case is illustrated in Figure 11.7. For this example, the camera image plane is parallel to the plane that contains the four feature points. When the feature points reach their desired positions in the image, the pose of the camera will differ from its initial pose by a pure rotation about the camera's z-axis. Figure 11.7(a) shows the image feature trajectories for the four feature points when image-based



(a) (b)

Figure 11.8: The required camera motion is a rotation by π about the camera z-axis. In (a) the feature points move along straight-line trajectories in the image, but in (b) this requires the camera to retreat to $z = -\infty$.

control is applied. As can be seen in the figure, the feature points move on straight lines to their goal positions in the image. Figure 11.7(b) shows the image feature errors for the four points; the errors converge exponentially to zero. Unfortunately, as can be seen in Figure 11.7(c), to achieve this performance the camera retreats by a full one meter along its z-axis. Such a large motion is not possible for most manipulators. Figure 11.7(d) shows the corresponding camera velocities. The velocities along and about the camera x- and y-axes are very small, but the linear velocity along camera z-axis varies significantly.

The most extreme version of this problem occurs when the effective camera motion is a rotation by π about the camera's optical axis. This case is shown in Figure 11.8. In Figure 11.8(a) the feature points again move on straight-line trajectories in the image. However, in this case, these trajectories pass through the image center. This occurs only when the camera has retreated infinitely far along its z-axis. The corresponding camera position is shown in Figure 11.8(b).

These two examples are special cases that illustrate one of the key problems that confront image-based visual servo systems. Such systems explicitly control the error in the image, but exert no explicit control over the trajectory of the camera. Thus, it is possible that the required camera motions will exceed the capabilities of the robot manipulator. Partitioned methods provide one way to cope with these problems, and we describe one such method in Section 11.7.

11.6 End Effector and Camera Motions

The output of a visual servo controller is a camera velocity ξ_c, typically expressed in coordinates relative to the camera frame. If the camera frame were coincident with the end-effector frame, we could use the manipulator Jacobian to determine the joint velocities that would achieve the desired camera motion as described in Section 4.11. In most applications, the camera frame is not coincident with the end-effector frame, but is rigidly attached to it. Suppose the two frames are related by the constant homogeneous transformation

$$T_c^6 = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{11.29}$$

In this case, we can use Equation (4.81) to determine the required velocity of the end effector to achieve the desired camera velocity. This gives

$$\xi_6^6 = \begin{bmatrix} R & S(d)R \\ 0_{3\times 3} & R \end{bmatrix} \xi_c^c$$

If we wish to express the end-effector velocity with respect to the base frame, we merely apply a rotational transformation to the two free vectors v_6 and ω_6, and this can be written as the matrix equation

$$\xi_6^0 = \begin{bmatrix} R_6^0 & 0_{3\times 3} \\ 0_{3\times 3} & R_6^0 \end{bmatrix} \xi_6^6$$

Example 11.6 (Eye-in-hand System with SCARA Arm)**.** *Consider the camera system described in Example 11.4. Recall that in this example the camera motion was restricted to three degrees of freedom, namely,* $\xi_c = [\mathrm{v}_x, \mathrm{v}_y, 0, 0, 0, \dot{\theta}]^T$*. Suppose that this camera is attached to the end effector of a SCARA manipulator, such that the optical axis of the camera is aligned with the z-axis of the end-effector frame. In this case, we can express the orientation of the camera frame relative to the end-effector frame by*

$$R_c^6 = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

in which α *gives the angle from* x_6 *to* x_c*. Let the origin of the camera frame relative to the end-effector frame be given by* $d_c^6 = [10, 5, 0]^T$*. The relationship between end effector and camera velocities is then given by*

$$\xi_6^6 = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 & 0 & 5 \\ \sin\alpha & \cos\alpha & 0 & 0 & 0 & -10 \\ 0 & 0 & 1 & -5 & 10 & 0 \\ 0 & 0 & 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & 0 & 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{v}_x \\ \mathrm{v}_y \\ 0 \\ 0 \\ 0 \\ \dot{\theta} \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{v}_x \cos\alpha - v_y \sin\alpha + 5\dot{\theta} \\ \mathrm{v}_x \sin\alpha + v_y \cos\alpha - 10\dot{\theta} \\ 0 \\ 0 \\ 0 \\ \dot{\theta} \end{bmatrix}$$

This can be used with the Jacobian matrix of the SCARA arm (derived in Chapter 4) to solve for the joint velocities required to achieve the desired camera motion.

11.7 Partitioned Approaches

Although image-based methods are versatile and robust to calibration and sensing errors, as discussed in Section 11.5.3 they sometimes fail when the required camera motion is large. Consider again the case illustrated in Figure 11.8. A pure rotation of the camera about the optical axis would cause each feature point to trace a trajectory in the image that lies on a circle, while image-based methods, in contrast, cause each feature point to move in a straight line from its current image position to its desired position. In the latter case, the induced camera motion is a retreat along the optical axis, in this case to $z = -\infty$, at which point $\det L = 0$ and the controller fails. This problem is a consequence of the fact that image-based control does not explicitly take camera motion into account. Instead, image-based control determines a desired trajectory in the image feature space, and maps this trajectory, using the interaction matrix, to a camera velocity. One way to deal with such situations is to use a **partitioned method**.

Partitioned methods use the interaction matrix to control only a subset of the camera degrees of freedom, and use other methods to control the remaining degrees of freedom. Consider, for example Equation (11.23). We

can write this equation as

$$\begin{aligned}
\dot{s} &= \begin{bmatrix} L_{\mathrm{v}_x} & L_{\mathrm{v}_y} & L_{\mathrm{v}_z} & L_{\omega_x} & L_{\omega_y} & L_{\omega_z} \end{bmatrix} \xi \\
&= \begin{bmatrix} L_{\mathrm{v}_x} & L_{\mathrm{v}_y} & L_{\omega_x} & L_{\omega_y} \end{bmatrix} \begin{bmatrix} \mathrm{v}_x \\ \mathrm{v}_y \\ \omega_x \\ \omega_y \end{bmatrix} + \begin{bmatrix} L_{\mathrm{v}_z} & L_{\omega_z} \end{bmatrix} \begin{bmatrix} \mathrm{v}_z \\ \omega_z \end{bmatrix} \\
&= L_{xy}\xi_{xy} + L_z\xi_z \qquad (11.30)
\end{aligned}$$

Here, $\dot{s}_z = L_z\xi_z$ gives the component of $\dot{s}$ due to the camera motion along and rotation about the optical axis, while $\dot{s}_{xy} = L_{xy}\xi_{xy}$ gives the component of $\dot{s}$ due to velocity along and rotation about the camera x- and y-axes.

Equation (11.30) allows us to partition the control into two components, ξ_{xy} and ξ_z. Suppose that we have established a control scheme to determine the value ξ_z. Using an image-based method to find ξ_{xy} we would solve Equation (11.30) as

$$\xi_{xy} = L_{xy}^{+}(\dot{s} - L_z\xi_z) \qquad (11.31)$$

This equation has an intuitive explanation. The term $-L_{xy}^{+}L_z\xi_z$ is the required value of ξ_{xy} to cancel the feature motion $\dot{s}_z$. The control $\xi_{xy} = L_{xy}^{+}\dot{s}$ gives the velocity along and rotation about the camera x- and y-axes that produce the desired $\dot{s}$ once image feature motion due to ξ_z has been taken into account.

If we use the Lyapunov design method described above, we set $\dot{e} = -\lambda e$, and obtain

$$-\lambda e = \dot{e} = \dot{s} = L_{xy}\xi_{xy} + L_z\xi_z$$

which leads to

$$\xi_{xy} = -L_{xy}^{+}(\lambda e + L_z\xi_z)$$

We can consider $(\lambda e + L_z\xi_z)$ as a modified error that incorporates the original image feature error while taking into account the feature error that will be induced by the translation along and rotation about the optical axis due to ξ_z.

The only remaining task is to construct a control law to determine the value of ξ_z. To determine ω_z, we can use the angle θ_{ij} from the horizontal axis of the image plane to the directed line segment joining two feature points. For numerical conditioning it is advantageous to select the longest line segment that can be constructed from the feature points, allowing this choice to change during the motion as the feature point configuration changes. The value for such an ω_z is given by

$$\omega_z = \gamma_{\omega_z}(\theta_{ij}^{d} - \theta_{ij})$$

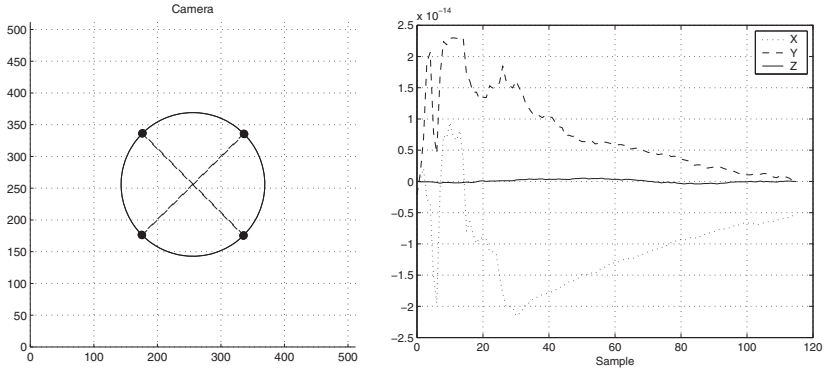


Figure 11.9: For the case of pure image-based control, each feature point would move on a straight line from initial to goal location, as illustrated by the straight line segments in the figure; all feature points would pass simultaneously through the center of the circle in this case. Using the partitioned controller, the feature points move along the circle shown in the figure, until they reach their goal positions. The figure on the right shows the translational motion of the camera; note that motion along the z-axis is essentially zero.

in which θ_{ij}^d is the desired value, and γ_{ω_z} is a scalar gain coefficient.

We can use the apparent size of an object in the image to determine v_z. Let σ^2 denote the area of some polygon in the image. We define v_z as

$$v_z = \gamma_{v_z} \ln \left(\frac{\sigma^d}{\sigma} \right)$$

The advantages to using the apparent size as a feature are that it is a scalar that is invariant with respect to rotation about the optical axis (thus decoupling camera rotation from z-axis translation), and that it can be easily computed.

Figures 11.9 and 11.10 illustrate the performance of this partitioned controller for the case of desired rotation by π about the optical axis. Note in Figure 11.10 that the camera does not retreat (σ is constant), the angle θ monotonically decreases, and the feature points move in a circle. The feature coordinate error is initially increasing, unlike the classical image-based methods, in which feature error is monotonically decreasing.

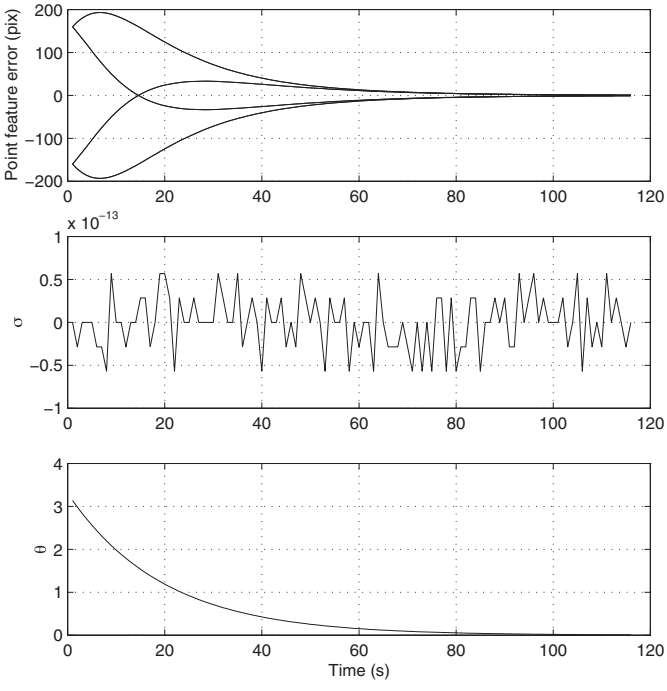


Figure 11.10: The feature error trajectories for the motion illustrated in Figure 11.9, from top to bottom: error for the four feature points, error in σ, error in θ.

11.8 Motion Perceptibility

Recall that the notion of manipulability described in Section 4.12 gave a quantitative measure of the scaling from joint velocities to end-effector velocities. **Motion perceptibility** is an analogous concept that relates camera velocity to the velocity of features in the image. Intuitively, motion perceptibility quantifies the magnitude of changes to image features that result from motion of the camera.

Consider the set of all camera velocities ξ such that

$$\|\xi\|^2 = \xi_1^2 + \xi_2^2 + \ldots \xi_m^2 \leq 1.$$

Suppose that there are redundant image features, that is, $k > m$. We may use Equation (11.26) to obtain

$$\begin{aligned} \|\xi\|^2 &= \xi^T \xi \\ &= (L^+ \dot{s})^T (L^+ \dot{s}) \\ &= \dot{s}^T (L^{+T} L^+) \dot{s} \leq 1 \end{aligned} \tag{11.32}$$

Now, consider the singular value decomposition of L (see Appendix B) given by

$$L = U \Sigma V^T. \tag{11.33}$$

in which U and V are orthogonal matrices and $\Sigma \in \mathbb{R}^{k \times m}$ with

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \sigma_m \\ & & 0 & & \end{bmatrix}$$

and the σ_i are the singular values of L and $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_m$.

For this case, the pseudoinverse of the interaction matrix L^+ is given by Equation (11.27). Using this with Equations (11.32) and (11.33) we obtain

$$\dot{s}^T U \begin{bmatrix} \sigma_1^{-2} & & & & \\ & \sigma_2^{-2} & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \sigma_m^{-2} \\ & & 0 & & \end{bmatrix} U^T \dot{s} \leq 1 \tag{11.34}$$

Consider the orthogonal transformation of $\dot{s}$ given by

$$\tilde{\dot{s}} = U^T \dot{s}$$

Substituting this into Equation (11.34) we obtain

$$\sum_{i=1}^{m} \frac{1}{\sigma_i^2} \tilde{\dot{s}}_i \leq 1 \tag{11.35}$$

Equation (11.35) defines an ellipsoid in an m-dimensional space. We shall refer to this ellipsoid as the **motion perceptibility ellipsoid**. We may use the volume of the m-dimensional ellipsoid given in Equation (11.35) as

a quantitative measure of the perceptibility of motion. The volume of the motion perceptibility ellipsoid is given by

$$K\sqrt{\det(L^T L)}$$

in which K is a scaling constant that depends on m, the dimension of the ellipsoid. Because the constant K depends only on m, it is not relevant for the purpose of evaluating motion perceptibility, since m will be fixed for any particular problem. Therefore, we define the motion perceptibility, which we shall denote by ρ, as

$$\rho = \sqrt{\det(L^T L)} = \sigma_1 \sigma_2 \cdots \sigma_m$$

The motion perceptibility measure ρ has the following properties, which are direct analogs of properties derived for manipulability:

- In general, $\rho = 0$ holds if and only if $\text{rank}(L) < \min(k, m)$, that is, when L is not full rank.

- Suppose that there is some error in the measured visual feature velocity $\Delta\dot{s}$. We can bound the corresponding error in the computed camera velocity $\Delta\xi$ by
$$(\sigma_1)^{-1} \leq \frac{||\Delta\xi||}{||\Delta\dot{s}||} \leq (\sigma_m)^{-1}.$$

There are other quantitative methods that could be used to evaluate the perceptibility of motion. For example, in the context of feature selection the condition number for the interaction matrix, given by $\|L\|\|L^{-1}\|$, could be used to select image features.

11.9 Summary

In this chapter, we introduced vision-based control, including an overview of those aspects of computer vision that are necessary to fully understand and implement vision-based control algorithms.

We began by discussing basic choices that confront the designer of a vision-based control scheme, distinguishing between fixed-camera and eye-in-hand systems (the latter being the focus of the current chapter), and between position-based and image-based control architectures. Image-based methods map image data directly to control signals, and are the primary focus of the chapter.

Our discussion of computer vision was limited to the geometric aspects of image formation. We presented perspective projection as a model for image formation. In this case, the projection onto the image plane of a point with coordinates (x, y, z) is given by the perspective projection equations

$$u = \lambda \frac{x}{z}, \qquad v = \lambda \frac{y}{z}$$

We then described how gradient information can be used to design a rudimentary corner detector.

Image-based visual servo control is a method for using an error measured in the image to directly control the motion of a robot. The key relationship exploited by all image-based methods is given by

$$\dot{s} = L(s, q)\xi$$

in which $L(s, q)$ is the interaction matrix and s is a vector of measured image feature values. When a single image point is used as the feature, this relationship is given by

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{\lambda} & -\frac{\lambda^2 + u^2}{\lambda} & v \\ 0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2 + v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

In image-based control the image error is defined by

$$e(t) = s(t) - s^d$$

and by using the square of the error norm as a candidate Lyapunov function, we derive the control law

$$\xi = -\lambda L^{-1} e(t)$$

when the interation matrix is square and nonsingular or

$$\xi = -\lambda L^{+} e(t)$$

with $L^{+} = (L^T L)^{-1} L^T$ when $L \in \mathbb{R}^{k \times m}$ and $k > m$.

In general, the camera coordinate frame and the end effector frame of the robot are not coincident. In this case, it is necessary to relate the camera velocity to the end effector velocity. This relationship is given by

$$\xi_6^6 = \begin{bmatrix} R_c^6 & S(d_c^6) R_c^6 \\ 0_{3\times 3} & R_c^6 \end{bmatrix} \xi_c^c$$

in which R_c^6 and d_c^6 specify the fixed relative orientation and position of the camera frame with respect to the end effector frame, and ξ_6 and ξ_c denote the end effector and camera velocities, respectively.

In some cases, it is advantageous to use different control laws for the different degrees of freedom. In this chapter, we described one way to partition the control system using the relationship

$$\dot{s} = L_{xy}\xi_{xy} + L_z\xi_z$$

After defining two new image features, we controlled the z-axis translation and rotation using

$$\begin{aligned} \omega_z &= \gamma_{\omega_z}(\theta_{ij}^d - \theta_{ij}) \\ v_z &= \gamma_{v_z} \ln\left(\frac{\sigma^d}{\sigma}\right) \end{aligned}$$

in which θ_{ij} is the angle between the horizontal axis of the image plane and the directed line segment joining two feature points, σ^2 is the area of a polygon in the image, and γ_{ω_z} and γ_{v_z} are scalar gain coefficients.

Finally, we defined motion perceptibility as a property of visual servo systems that is analogous to the manipulability measure for manipulators. For $k > m$ motion percpetibility is defined by

$$\rho = \sqrt{\det(L^T L)} = \sigma_1\sigma_2\cdots\sigma_m$$

in which σ_i are the singular values of the interaction matrix.

Problems

11–1 For a camera with focal length $\lambda = 10$, find the image plane coordinates for the 3D points whose coordinates with respect to the camera frame are given below. Indicate if any of these points will not be visible to a physical camera.

1. $(25, 25, 50)$
2. $(-25, -25, 50)$
3. $(20, 5, -50)$
4. $(15, 10, 25)$
5. $(0, 0, 50)$
6. $(0, 0, 100)$

11–2 Repeat Problem 11–1 for the case when the coordinates of the points are given with respect to the world frame. Suppose the optical axis of the camera is aligned with the world x-axis, the camera x-axis is parallel to the world y-axis, and the center of projection has coordinates $(0, 0, 100)$.

11–3 A stereo camera system consists of two cameras that share a common field of view. By using two cameras, stereo vision methods can be used to compute 3D properties of the scene. Consider stereo cameras with coordinate frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ such that

$$H_2^1 = \begin{bmatrix} 1 & 0 & 0 & B \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, B is called the **baseline distance** between the two cameras. Suppose that a 3D point P projects onto these two images with image plane coordinates (u_1, v_1) in the first camera and (u_2, v_2) in the second camera. Determine the depth of the point P.

11–4 Show that the projection of a 3D line is a line in the image.

11–5 Show that the vanishing points for all 3D horizontal lines must lie on the line $v = 0$ of the image plane.

11–6 Suppose the vanishing point for two parallel lines has the image coordinates (u_∞, v_∞). Show that the direction vector for the 3D line is given by

$$u = \frac{1}{\sqrt{u_\infty^2 + v_\infty^2 + \lambda^2}} \begin{bmatrix} u_\infty \\ v_\infty \\ \lambda \end{bmatrix}$$

in which λ is the focal length of the imaging system.

11–7 Two parallel lines define a plane. Consider a set of pairs of parallel lines such that the corresponding planes are all parallel. Show that the vanishing points for the images of these lines are collinear. Hint: let n be the normal vector for the parallel planes and exploit the fact that $u_i \cdot n = 0$ for the direction vector u_i associated to the i^{th} line.

11–8 A cube has twelve edges, each of which defines a line in three space. We can group these lines into three groups, such that in each of the groups there are four parallel lines. Let (a_1, a_2, a_3), (b_1, b_2, b_3), and

(c_1, c_2, c_3) be the direction vectors for these three sets of parallel lines. Each set of parallel lines gives rise to a vanishing point in the image. Let the three vanishing points be $V_a = (u_a, v_a)$, $V_b = (u_b, v_b)$, and $V_c = (u_c, c_c)$, respectively.

(a) If C is the optical center of the camera, show that the three angles $\angle V_a C V_b$, $\angle V_a C V_c$ and $\angle V_b C V_c$ are each equal to $\frac{\pi}{2}$. Hint: In the world coordinate frame, the image plane is the plane $z = \lambda$.

(b) Let h_a be the altitude from V_a to the line defined by V_b and V_c. Show that the plane containing both h_a and the line through points C and V_a is orthogonal to the line defined by V_b and V_c.

(c) Let h_a be the altitude from V_a to the line defined by V_b and V_c, h_b the altitude from V_b to the line defined by V_a and V_c, and h_c the altitude from V_c to the line defined by V_a and V_b. We define the following three planes:

- P_a is the plane containing both h_a and the line through points C and V_a.
- P_b is the plane containing both h_b and the line through points C and V_b.
- P_c is the plane containing both h_c and the line through points C and V_c.

Show that each of these planes is orthogonal to the image plane (it is sufficient to show that P_i is orthogonal to the image plane for a specific value of i).

1. The three vanishing points V_a, V_b, V_c define a triangle, and the three altitudes h_a, h_b, h_c intersect at the orthocenter of this triangle. For this special case, where the three direction vectors are mutually orthogonal, what is the significance of this point?

11–9 Use your results on vanishing points to draw a nice cartoon scene with a road or two, some houses and maybe a roadrunner and coyote.

11–10 Suppose that a circle lies in a plane parallel to the image plane. Show that the perspective projection of the circle is a circle in the image plane and determine its radius.

11–11 Give an expression for the interaction matrix for two points p_1 and

p_2 that satisfies

$$\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} = L\xi$$

in which (u_1, v_1) and (u_2, v_2) are the image coordinates of the projection of p_1 and p_2, respectively, and ξ is the velocity of the moving camera.

11–12 Show that the four vectors given in Equation (11.25) form a basis for the null space for the interaction matrix given in Equation (11.23).

11–13 What is the dimension of the null space for the interaction matrix for two points? Give a basis for this null space.

11–14 Consider a stereo camera system attached to a robot manipulator. Derive the interaction matrix L that satisfies

$$\begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = L\xi$$

in which (u_l, v_l) and (u_r, v_r) are the image coordinates of the projection of p in the left and right images, respectively, and ξ is the velocity of the moving stereo camera system.

11–15 Consider a stereo camera system mounted to a fixed tripod observing the manipulator end effector. If the end effector velocity is given by ξ, derive the interaction matrix L that satisfies

$$\begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = L\xi$$

in which (u_l, v_l) and (u_r, v_r) are the image coordinates of the projection of p in the left and right images, respectively.

11–16 Consider a fixed camera that observes the motion of a robot arm. Derive the interaction matrix that relates the velocity of the end effector frame to the image coordinates of the projection of the origin of the end effector frame.

11–17 Consider a camera mounted above a conveyor belt such that the optical axis is parallel to the world z-axis. The camera can translate and rotate about its optical axis, so in this case we have $\xi = [v_x, v_y, v_z, \dot{\theta}]^T$. Suppose the camera observes a planar object whose moments are given by

$$m_{ij} = \sum_{r,c} r^i c^j \mathcal{I}(r, c).$$

Derive the interaction matrix that satisfies

$$\begin{bmatrix} \dot{m}_{00} \\ \dot{m}_{10} \\ \dot{m}_{01} \end{bmatrix} = L\xi$$

11–18 Write a simulator for an image-based visual servo controller that uses the coordinates of four image points as features. Find initial and goal images for which the classical image-based control scheme will stop in a local minimum, that is, where the error lies in null space of the interaction matrix. Find initial and goal images for which the system diverges.

11–19 Use the simulator you implemented for Problem 11–18 to demonstrate the robustness of image-based methods to errors in depth estimation.

11–20 Use the simulator you implemented for Problem 11–18 to demonstrate the robustness of image-based methods to errors in the estimation of camera orientation. For example, use Euler angles to construct a rotation matrix that can be used to perturb the world coordinates of the four points used as features.

11–21 It is often acceptable to use a fixed value of depth $z = z^d$ when constructing the interaction matrix. Here, z^d denotes the depth when the camera is in its goal position. Use the simulator you implemented for Problem 11–18 to compare the performance of image-based control using the true value of depth vs. using the fixed value z^d.

11–22 Write a simulator for a partitioned controller that uses the coordinates of four image points as features. Find initial and goal images for which the controller fails. Compare the performance of this controller to the one you implemented for Problem 11–18.

Notes and References

Computer vision research dates back to the early sixties. In the early eighties several computer vision texts appeared. These books approached computer vision from the perspective of cognitive modeling of human vision [111], image processing [144], and applied robotic vision [66]. A comprehensive review of computer vision techniques through the early nineties can be found in [61], and an introductory treatment of methods in 3D vision can be found in [175]. Detailed treatments of the geometric aspects of computer vision can be found in [42] and [106]. A comprehensive review of the state of the art in computer vision at the turn of the century can be found in [47]. More recently, [26] has provided an updated treatment of robot vision, including feature detection and tracking, and vision-based control.

There are numerous design considerations when designing feature detectors and feature tracking algorithms. Should tracking be done directly in the image (using the position and orientation of the feature in the image as the state), or in the world frame (using the position and orientation of the camera as the state)? What should be the underlying tracking approach – a linearized approach such as the EKF, or a general nonlinear estimator? Which features are easiest to track? Which are most robust to variations in lighting? Which are most distinctive? These design considerations have led to a rich body of literature on the feature tracking problem in computer vision, and many general feature tracking algorithms have by now been made available in various open-source software libraries such as OpenCV [17] and Matlab's Robotics and Computer Vision Toolboxes [26]. While the specifics of these algorithms may vary considerably, most can be seen as variations of the classical prediction-correction approach that lies at the heart of most state estimators. These issues are treated in standard computer vision texts (e.g., [47, 106, 26]).

Vision-based control of robotic systems dates back to the 1960's and the robot known as Shakey that was built at SRI. However, the vision system for Shakey was much too slow for real-time control applications. Some of the earliest results of real-time, vision-based control were reported in [5] and [6], which described a robot that played ping pong.

The interaction matrix was first introduced in [146], where it was referred to as the **feature sensitivity matrix**. In [44] it was referred to as a Jacobian matrix, subsequently referred to in the literature as the image Jacobian, and in [39] it was given the name interaction matrix, the term we use in this text.

The performance problems associated with image-based methods were

first rigorously investigated in [23]. This paper charted a course for the next several years of research in visual servo control.

The first partitioned method for visual servo was introduced in [107], which describes a system in which the three rotational degrees of freedom are controlled using position-based methods and the three translational degrees of freedom are controlled using image-based methods. Other partitioned methods have been reported in [33] and [116]. The method described in this chapter was reported in [27].

Motion perceptibility was introduced in [150] and [149]. The notion of **resolvability**, introduced in [124] and [125], is similar. In [43] the condition number of the interaction matrix is used for the purpose of feature selection.

Chapter 12

FEEDBACK LINEARIZATION

In this chapter we present some basic, but fundamental, ideas from **geometric nonlinear control theory**. We first give some background from differential geometry to set the notation and define basic quantities, such as **manifold**, **vector field**, **Lie bracket**, and so forth that we will need later. The main tool that we will use in this chapter is the **Frobenius theorem**, which we introduce in Section 12.1.2.

We then discuss the notion of **feedback linearization of nonlinear systems**. This approach generalizes the concept of inverse dynamics of rigid manipulators discussed in Chapter 9. The idea of feedback linearization is to construct a nonlinear control law as an **inner-loop control**, which, in the ideal case, exactly linearizes the nonlinear system after a suitable state space change of coordinates. The designer can then design the **outer-loop control** in the new coordinates to satisfy the traditional control design specifications such as tracking and disturbance rejection.

In the case of rigid manipulators the inverse dynamics control of Chapter 9 and the feedback linearizing control are the same. The main difference between inverse dynamics control and feedback linearization, as we shall see, is to find the "right" set of coordinates with respect to which the dynamics can be rendered linear by feedback. In the case of inverse dynamics, no change of coordinates is necessary.

As we shall see, the full power of the feedback linearization technique for manipulator control becomes apparent if one includes in the dynamic description of the manipulator the transmission dynamics, such as elasticity resulting from shaft windup and gear elasticity.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

12.1 Background

In recent years an impressive volume of literature has emerged in the area of differential geometric methods for nonlinear systems, treating not only feedback linearization but also other problems such as disturbance decoupling, estimation, observers, and adaptive control. It is our intent here to give only that portion of the theory that finds an immediate application to robot control, and even then to give only the simplest versions of the results.

12.1.1 Manifolds, Vector Fields, and Distributions

The fundamental notion in differential geometry is that of a **differentiable manifold**, (manifold for short) which is a topological space that is locally diffeomorphic[1] to Euclidean space $\mathbb{R}^m$. For our purposes here a manifold may be thought of as a subset of $\mathbb{R}^n$ defined by the zero set of a smooth vector valued function[2] $h : \mathbb{R}^n \to \mathbb{R}^p$, for $p < n$,

$$\begin{aligned} h_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ h_p(x_1, \ldots, x_n) &= 0 \end{aligned}$$

We assume that the differentials $dh_1, \ldots, dh_p$ are linearly independent at each point, in which case the dimension of the manifold is $m = n - p$. Given an m-dimensional manifold M we may attach at each point $x \in M$ a **tangent space** T_xM, which is an m-dimensional vector space specifying the set of possible velocities (directional derivatives) at x.

Example 12.1. *Consider the unit sphere S^2 in $\mathbb{R}^3$ defined by*

$$h(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$$

S^2 is a two-dimensional submanifold of $\mathbb{R}^3$. At points in the upper hemisphere, $z = \sqrt{1 - x^2 - y^2}$, and the tangent space is spanned by the vectors

$$\begin{aligned} v_1 &= [1, 0, -x/\sqrt{1 - x^2 - y^2}]^T \\ v_2 &= [0, 1, -y/\sqrt{1 - x^2 - y^2}]^T \end{aligned}$$

[1] A **diffeomorphism** is simply a differentiable function whose inverse exists and is also differentiable. We shall assume both the function and its inverse to be infinitely differentiable. Such functions are customarily referred to as C^∞ **diffeomorphisms**.

[2] Our definition amounts to the special case of an **embedded submanifold** of dimension $m = n - p$ in $\mathbb{R}^n$.

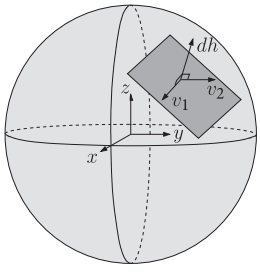


Figure 12.1: The sphere as a two-dimensional manifold in $\mathbb{R}^3$.

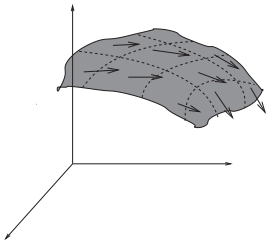


Figure 12.2: Pictorial representation of a vector field on a manifold.

The differential of h is

$$dh = (2x, 2y, 2z) = (2x, 2y, 2\sqrt{1 - x^2 - y^2})$$

which is perpendicular to the tangent plane at x, y, z.

Definition 12.1. *A* **smooth vector field** *on a manifold M is an infinitely differentiable function $f : M \to TM$ represented as a column vector*

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} \in T_xM \text{ for all } x \in M$$

Another useful notion is that of **cotangent space** and **covector field**. The cotangent space T_x^*M is the dual space of the tangent space. It is an m-dimensional vector space specifying the set of possible differentials of functions at x. Mathematically, T_x^*M is the space of all linear functionals on T_xM, that is, the space of functions from T_xM to $\mathbb{R}$.

Definition 12.2. *A* **smooth covector field** *is a function $w : M \to T^*M$, which is infinitely differentiable, represented as a row vector,*

$$w(x) = \begin{bmatrix} w_1(x), \ldots, w_m(x) \end{bmatrix} \in T_x^*M \text{ for all } x \in M$$

Henceforth, whenever we use the term function, vector field, or covector field, it is assumed to be smooth. Since T_xM and T_x^*M are m-dimensional vector spaces, they are isomorphic and the only distinction we will make between vectors and covectors below is whether or not they are represented as column vectors or row vectors.

We may also have multiple vector fields defined simultaneously on a given manifold. Such a set of vector fields will span a subspace of the tangent space at each point. Likewise, we will consider multiple covector fields spanning a subspace of the cotangent space at each point. These notions give rise to so-called **distributions** and **codistributions**.

Definition 12.3. *Let $X_1(x), \ldots, X_k(x)$ be vector fields on M. A* **distribution** Δ *is the linear span of the vectors (at each $x \in M$)*

$$\Delta = \text{span } \{X_1(x), \ldots, X_k(x)\} \tag{12.1}$$

Likewise, let $w_1(x), \ldots, w_k(x)$ be covector fields on M. A **codistribution** Ω *is the linear span of the covectors (at each $x \in M$)*

$$\Omega = \text{span}\{w_1(x), \ldots, w_k(x)\} \tag{12.2}$$

Remark 12.1. *We will assume that the distribution Δ is linearly independent at each $x \in M$, in which case Δ is called a* **regular distribution**. *Henceforth, whenever we say distribution, it will be understood to mean regular distribution. The same convention will apply to codistribution and* **regular codistribution**.

A distribution therefore assigns a vector space $\Delta(x)$ to each point $x \in M$. $\Delta(x)$ is a k-dimensional subspace of the m-dimensional tangent space T_xM. A codistribution likewise defines a k-dimensional subspace $\Omega(x)$ at each x of the m-dimensional cotangent space T_x^*M.

Vector fields are used to define differential equations and their associated flows. We restrict our attention here to nonlinear systems of the form

$$\begin{aligned} \dot{x} &= f(x) + g_1(x)u_1 + \cdots + g_m(x)u_m \\ &= f(x) + G(x)u \end{aligned} \tag{12.3}$$

where $f(x)$, $g_1(x), \ldots, g_m(x)$ are smooth vector fields on M, and where we define $G(x) = [g_1(x), \ldots, g_m(x)]$ and $u = [u_1, \ldots, u_m]^T$. In the remainder of this chapter we will assume that $M = \mathbb{R}^n$.

Definition 12.4. *Let f and g be two vector fields on $\mathbb{R}^n$. The* **Lie bracket** *of f and g, denoted by $[f, g]$, is a vector field defined by*

$$[f, g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g \tag{12.4}$$

where $\frac{\partial g}{\partial x}$ *(respectively,* $\frac{\partial f}{\partial x}$*) denotes the* $n \times n$ *Jacobian matrix whose* ij^{th} *entry is* $\frac{\partial g_i}{\partial x_j}$ *(respectively,* $\frac{\partial f_i}{\partial x_j}$*).*

Example 12.2. *Suppose that vector fields* $f(x)$ *and* $g(x)$ *on* $\mathbb{R}^3$ *are given as*

$$f(x) = \begin{bmatrix} x_2 \\ \sin x_1 \\ x_1 + x_3^2 \end{bmatrix} \qquad g(x) = \begin{bmatrix} 0 \\ x_2^2 \\ 1 \end{bmatrix}$$

Then the vector field $[f, g]$ *is computed according to Equation (12.4) as*

$$\begin{aligned} [f,g] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2x_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ \sin x_1 \\ x_1 + x_3^2 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ \cos x_1 & 0 & 0 \\ 1 & 0 & 2x_3 \end{bmatrix} \begin{bmatrix} 0 \\ x_2^2 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -x_2^2 \\ 2x_2 \sin x_1 \\ -2x_3 \end{bmatrix} \end{aligned}$$

We also denote $[f, g]$ as $ad_f(g)$ and define $ad_f^k(g)$ inductively by

$$ad_f^k(g) = [f, ad_f^{k-1}(g)] \tag{12.5}$$

with $ad_f^0(g) = g$.

Definition 12.5. *Let* $f : \mathbb{R}^n \to \mathbb{R}^n$ *be a vector field on* $\mathbb{R}^n$ *and let* $h : \mathbb{R}^n \to \mathbb{R}$ *be a scalar function. The* **Lie derivative** *of* h *with respect to* f*, denoted* $L_f h$*, is defined as*

$$L_f h = \frac{\partial h}{\partial x} f(x) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} f_i(x) \tag{12.6}$$

The Lie derivative is simply the directional derivative of h in the direction of f. We denote by $L_f^2 h$ the Lie derivative of $L_f h$ with respect to f, that is,

$$L_f^2 h = L_f(L_f h) \tag{12.7}$$

In general we define

$$L_f^k h = L_f(L_f^{k-1} h) \quad \text{for } k = 1, \ldots, n \tag{12.8}$$

with $L_f^0 h = h$.

The following technical lemma gives an important relationship between the Lie bracket and Lie derivative and is crucial to the subsequent development.

Lemma 12.1. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a scalar function and f and g be vector fields on $\mathbb{R}^n$. Then we have the following identity*

$$L_{[f,g]}h = L_f L_g h - L_g L_f h \tag{12.9}$$

Proof: Expand Equation (12.9) in terms of the coordinates $x_1, \ldots, x_n$ and equate both sides. The i^{th} component $[f,g]_i$ of the vector field $[f,g]$ is given as

$$[f,g]_i = \sum_{j=1}^{n} \frac{\partial g_i}{\partial x_j} f_j - \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} g_j$$

Therefore, the left-hand side of Equation (12.9) is

$$\begin{aligned} L_{[f,g]}h &= \sum_{i=1}^{n} \frac{\partial h}{\partial x_i}[f,g]_i \\ &= \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} \left(\sum_{j=1}^{n} \frac{\partial g_i}{\partial x_j} f_j - \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} g_j \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial h}{\partial x_i} \left(\frac{\partial g_i}{\partial x_j} f_j - \frac{\partial f_i}{\partial x_j} g_j \right) \end{aligned}$$

If the right-hand side of Equation (12.9) is expanded similarly it can be shown, with a little algebraic manipulation, that the two sides are equal. The details are left as an exercise (Problem 12–1).

12.1.2 The Frobenius Theorem

In this section we present a basic result in differential geometry known as the **Frobenius theorem**. The Frobenius theorem can be thought of as an existence theorem for solutions to certain systems of first-order partial differential equations. Although a rigorous proof of this theorem is beyond the scope of this text, we can gain an intuitive understanding of it by considering the following system of partial differential equations

$$\frac{\partial z}{\partial x} = f(x,y,z) \tag{12.10}$$

$$\frac{\partial z}{\partial y} = g(x,y,z) \tag{12.11}$$

In this example there are two partial differential equations in a single dependent variable z. A solution to Equations (12.10) and (12.11) is a function

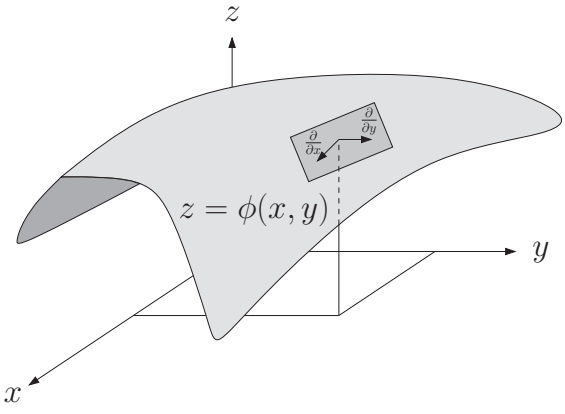


Figure 12.3: Integral manifold in $\mathbb{R}^3$.

$z = \phi(x, y)$ satisfying

$$\frac{\partial \phi}{\partial x} = f(x, y, \phi(x, y)) \tag{12.12}$$

$$\frac{\partial \phi}{\partial y} = g(x, y, \phi(x, y)) \tag{12.13}$$

We can think of the function $z = \phi(x, y)$ as defining a surface in $\mathbb{R}^3$ as in Figure 12.3. The function $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ defined by

$$\Phi(x, y) = (x, y, \phi(x, y)) \tag{12.14}$$

then characterizes both the surface and the solution of Equations (12.10) and (12.11). At each point (x, y) the tangent plane to the surface is spanned by two vectors found by taking partial derivatives of Φ in the x and y directions, respectively, that is, by

$$\begin{aligned} X_1 &= [1, 0, f(x, y, \phi(x, y))]^T \\ X_2 &= [0, 1, g(x, y, \phi(x, y))]^T \end{aligned} \tag{12.15}$$

The vector fields X_1 and X_2 are linearly independent and span a two-dimensional subspace at each point. Notice that X_1 and X_2 are completely specified by Equations (12.10) and (12.11). Geometrically, one can now think of the problem of solving this system of first-order partial differential equations as the problem of finding a surface in $\mathbb{R}^3$ whose tangent space at each point is spanned by the vector fields X_1 and X_2. Such a surface, if it can be found, is called an **integral manifold** for Equations (12.10) and

(12.11). If such an integral manifold exists then the set of vector fields, equivalently, the system of partial differential equations, is called **completely integrable**.

Let us reformulate this problem in yet another way. Suppose that $z = \phi(x, y)$ is a solution of Equations (12.10) and (12.11). Then it is a simple computation (Problem 12–2) to check that the function

$$h(x, y, z) \quad = \quad z - \phi(x, y) \tag{12.16}$$

satisfies the system of partial differential equations

$$\begin{array}{rcl} L_{X_1} h & = & 0 \\ L_{X_2} h & = & 0 \end{array} \tag{12.17}$$

Conversely, suppose a scalar function h can be found satisfying (12.17), and suppose that we can solve the equation

$$h(x, y, z) \quad = \quad 0 \tag{12.18}$$

for z, as $z = \phi(x, y)$.[3] Then it can be shown (Problem 12–3) that ϕ satisfies Equations (12.10) and (12.11). Hence, complete integrability of the set of vector fields $\{X_1, X_2\}$ is equivalent to the existence of h satisfying Equations (12.17). With the preceding discussion as background we state the following.

Definition 12.6. *A distribution $\Delta = span\{X_1, \ldots, X_m\}$ on $\mathbb{R}^n$ is said to be* **completely integrable** *if and only if there are $n - m$ linearly independent functions $h_1, \ldots, h_{n-m}$ satisfying the system of partial differential equations*

$$L_{X_i} h_j = 0 \quad \textit{for } 1 \leq i \leq m \ , \ 1 \leq j \leq n - m \tag{12.19}$$

Another important concept is the notion of **involutivity**, as defined next.

Definition 12.7. *A distribution $\Delta = span\{X_1, \ldots, X_m\}$ is said to be* **involutive** *if and only if there are scalar functions $\alpha_{ijk} : \mathbb{R}^n \to \mathbb{R}$ such that*

$$[X_i, X_j] \quad = \quad \sum_{k=1}^{m} \alpha_{ijk} X_k \quad \text{for all } i, j, k \tag{12.20}$$

[3]The **implicit function theorem** states that Equation (12.18) can be solved for z as long as $\frac{\partial h}{\partial z} \neq 0$.

Involutivity simply means that if one forms the Lie bracket of any pair of vector fields in Δ then the resulting vector field can be expressed as a linear combination of the original vector fields $X_1, \ldots, X_m$. An involutive distribution is thus closed under the operation of taking Lie brackets. Note that the coefficients in this linear combination are allowed to be smooth functions on $\mathbb{R}^n$.

In the simple case of Equations (12.10) and (12.11) one can show that involutivity of the set $\{X_1, X_2\}$ defined by Equation (12.19) is equivalent to interchangeability of the order of partial derivatives of h, that is, $\frac{\partial^2 h}{\partial x \partial y} = \frac{\partial^2 h}{\partial y \partial x}$. The Frobenius theorem, stated next, gives the conditions for the existence of a solution to the system of partial differential equations (12.19).

Theorem 12.1 (Frobenius)**.** *A distribution Δ is completely integrable if and only if it is involutive.*

The importance of the Frobenius theorem is that it allows one to determine whether or not a given distribution is integrable without having to actually solve the partial differential equations. The involutivity condition can, in principle, be computed from the given vector fields alone.

12.2 Feedback Linearization

To introduce the idea of feedback linearization consider the following simple system,

$$\dot{x}_1 = a\sin(x_2) \tag{12.21}$$
$$\dot{x}_2 = -x_1^2 + u \tag{12.22}$$

Note that we cannot simply choose u in the above system to cancel the nonlinear term $a\sin(x_2)$. However, if we first change variables by setting

$$y_1 = x_1 \tag{12.23}$$
$$y_2 = a\sin(x_2) = \dot{x}_1 \tag{12.24}$$

then, by the chain rule, y_1 and y_2 satisfy

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= a\cos(x_2)(-x_1^2 + u) \end{aligned} \tag{12.25}$$

We see that the nonlinearities can now be cancelled by the control input

$$u = \frac{1}{a\cos(x_2)} v + x_1^2 \tag{12.26}$$

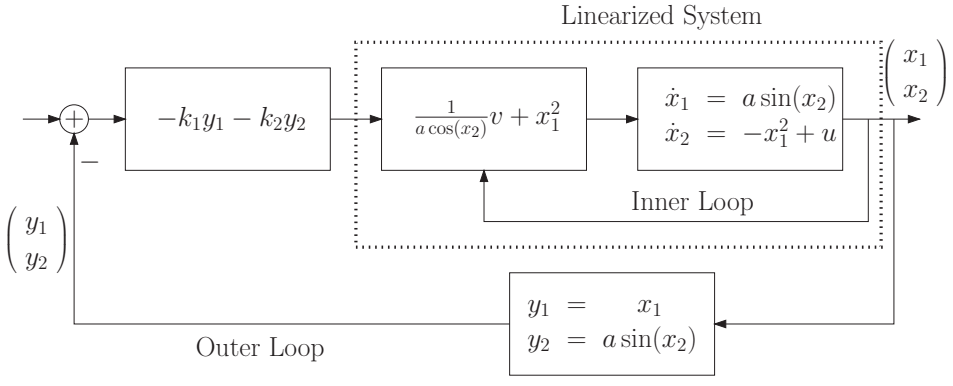


Figure 12.4: Inner-loop/outer-loop control architecture for feedback linearization.

which results in the linear system in the (y_1, y_2) coordinates

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= v \end{aligned} \tag{12.27}$$

The term v has the interpretation of an outer-loop control and can be designed to place the poles of the second-order linear system given by Equation (12.27) in the coordinates (y_1, y_2). For example, the outer-loop control

$$v = -k_1y_1 - k_2y_2 \tag{12.28}$$

applied to Equation (12.27) results in the closed-loop system

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -k_1y_1 - k_2y_2 \end{aligned} \tag{12.29}$$

which has characteristic polynomial

$$p(s) = s^2 + k_2s + k_1 \tag{12.30}$$

and hence, the closed-loop poles of the system with respect to the coordinates (y_1, y_2) are completely specified by the choice of k_1 and k_2. Figure 12.4 illustrates the inner-loop/outer-loop implementation of the above control strategy. The response in the y variables is easy to determine. The corresponding response of the system in the original coordinates (x_1, x_2) can be found by inverting the transformation given by Equations (12.23) and (12.24). The result is

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= \sin^{-1}(y_2/a) \qquad -a < y_2 < +a \end{aligned} \tag{12.31}$$

This example illustrates several important features of feedback linearization. The first thing to note is the local nature of the result. We see from Equations (12.23) and (12.24) that the transformation and the control make sense only in the region $-\infty < x_1 < \infty$, $-\frac{\pi}{2} < x_2 < \frac{\pi}{2}$. Second, in order to control the linear system given by Equation (12.27), the coordinates (y_1, y_2) must be available for feedback. This can be accomplished by measuring them directly if they are physically meaningful variables, or by computing them from the measured (x_1, x_2) coordinates using the transformation given by Equations (12.23) and (12.24). In the latter case the parameter a must be known precisely.

In Section 12.3 we give necessary and sufficient conditions under which a general single-input nonlinear system can be transformed into a linear system using a nonlinear change of variables and nonlinear feedback as in the above example.

12.3 Single-Input Systems

The idea of feedback linearization is easiest to understand in the context of single-input systems. In this section we give necessary and sufficient conditions for single-input nonlinear system to be locally feedback linearizable. As an illustration, we apply this result to the control of a single-link manipulator with joint elasticity.

Definition 12.8. *A single-input nonlinear system*

$$\dot{x} = f(x) + g(x)u \tag{12.32}$$

where $f(x)$ and $g(x)$ are vector fields on $\mathbb{R}^n$, $f(0) = 0$, and $u \in \mathbb{R}$, is said to be **feedback linearizable** *if there exists a diffeomorphism $T : U \to \mathbb{R}^n$, defined on an open region U in $\mathbb{R}^n$ containing the origin, and nonlinear feedback*

$$u = \alpha(x) + \beta(x)v \tag{12.33}$$

with $\beta(x) \neq 0$ on U such that the transformed state

$$y = T(x) \tag{12.34}$$

satisfies the linear system of equations

$$\dot{y} = Ay + bv \tag{12.35}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & & & 0 \\ 0 & 0 & 1 & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & 1 \\ 0 & 0 & \cdot & \cdot & 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \tag{12.36}$$

Remark 12.2. *The linear system (12.35) with A and b given by (12.36) is known as the* **Brunovsky canonical form**. *Any controllable single-input linear system can be transformed into the Brunovsky form with a linear change of coordinates and linear state feedback. Thus, it is without loss of generalization that we specify the Brunovsky form as the target linear system in the definition of nonlinear feedback linearization.*

The nonlinear transformation given by Equation (12.34) and the nonlinear control law, Equation (12.33), when applied to the nonlinear system (12.32), result in a linear controllable system (12.35). The diffeomorphism $T(x)$ can be thought of as a nonlinear change of coordinates in the state space. The idea of feedback linearization is that, if one first changes to the coordinate system $y = T(x)$, then there exists a nonlinear control law to cancel the nonlinearities in the system. The feedback linearization is said to be **global** if the region U is all of $\mathbb{R}^n$.

We next derive necessary and sufficient conditions on the vector fields f and g in Equation (12.32) for the existence of such a nonlinear coordinate transformation. Let us set

$$y = T(x) \tag{12.37}$$

and see what conditions the transformation $T(x)$ must satisfy. Differentiating both sides of Equation (12.37) with respect to time yields

$$\dot{y} = \frac{\partial T}{\partial x}\dot{x} \tag{12.38}$$

where $\frac{\partial T}{\partial x}$ is the Jacobian matrix of the transformation $T(x)$. Using Equations (12.32) and (12.35), Equation (12.38) can be written as

$$\frac{\partial T}{\partial x}(f(x) + g(x)u) = AT(x) + bv \tag{12.39}$$

In component form with

$$T = \begin{bmatrix} T_1 \\ \cdot \\ \cdot \\ \cdot \\ T_n \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 & & & 0 \\ 0 & 0 & 1 & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & 1 \\ 0 & 0 & \cdot & \cdot & 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \tag{12.40}$$

we see that the first equation in Equation (12.39) is

$$L_f T_1 + L_g T_1 u = T_2 \tag{12.41}$$

Similarly, the other components of T satisfy

$$\begin{aligned} L_f T_2 + L_g T_2 u &= T_3 \\ &\vdots \\ L_f T_n + L_g T_n u &= v \end{aligned} \tag{12.42}$$

Since we assume that $T_1, \ldots, T_n$ are independent of u while v is not independent of u we conclude from (12.42) that

$$L_g T_1 = L_g T_2 = \cdots = L_g T_{n-1} = 0 \tag{12.43}$$

$$L_g T_n \neq 0 \tag{12.44}$$

This leads to the system of partial differential equations

$$L_f T_i = T_{i+1} \; ; \quad i = 1, \ldots, n-1 \tag{12.45}$$

together with

$$L_f T_n + L_g T_n u = v \tag{12.46}$$

Using Lemma 12.1 together with Equations (12.43) and (12.44) we can derive a system of partial differential equations in terms of T_1 alone as follows. Using $h = T_1$ in Lemma 12.1 we have

$$L_{[f,g]} T_1 = L_f L_g T_1 - L_g L_f T_1 = 0 - L_g T_2 = 0 \tag{12.47}$$

Thus, we have shown

$$L_{[f,g]} T_1 = 0 \tag{12.48}$$

By proceeding inductively it can be shown (Problem 12–4) that

$$L_{ad_f^k}(g)T_1 = 0 \quad k = 0, 1, \ldots n-2 \tag{12.49}$$

$$L_{ad_f^{n-1}}(g)T_1 \neq 0 \tag{12.50}$$

If we can find T_1 satisfying the system of partial differential equations (12.49), then $T_2, \ldots, T_n$ are found inductively from Equation (12.45) and the control input u is found from (12.46) as

$$u = \frac{1}{L_g T_n}(v - L_f T_n) \tag{12.51}$$

We have thus reduced the problem to solving the system given by Equation (12.49) for T_1. When does such a solution exist?

First note that the vector fields g, $ad_f(g), \ldots, ad_f^{n-1}(g)$ must be linearly independent. If they are not linearly independent, then for some index i, we would have

$$ad_f^i(g) = \sum_{k=0}^{i-1} \alpha_k ad_f^k(g) \tag{12.52}$$

and then $ad_f^{n-1}(g)$ would be a linear combination of g, $ad_f(g), \ldots, ad_f^{n-2}(g)$ and Equation (12.50) could not hold. Now, by the Frobenius theorem, Equation (12.49) has a solution if and only if the distribution $\Delta = \text{span}\{g, ad_f(g), \ldots, ad_f^{n-2}(g)\}$ is involutive. Putting this together we have shown the following

Theorem 12.2. *The nonlinear system*

$$\dot{x} = f(x) + g(x)u \tag{12.53}$$

is feedback linearizable if and only if there exists an open region U containing the origin in $\mathbb{R}^n$ in which the following conditions hold:

1. *The vector fields $\{g, ad_f(g), \ldots, ad_f^{n-1}(g)\}$ are linearly independent in U.*
2. *The distribution $\Delta = \text{span}\{g, ad_f(g), \ldots, ad_f^{n-2}(g)\}$ is involutive in U.*

Remark 12.3. *A few remarks about the above theorem are the following:*

1. *For a linear system,*

$$\dot{x} = Fx + gu$$

it is easily shown that $ad_f^k(g) = -1^k F^k g$ (Problem 12–5) and therefore the first condition above is equivalent to controllability of the pair F, g.

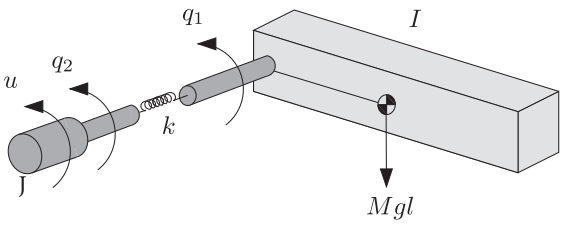


Figure 12.5: Single-link, flexible joint robot.

2. *Since each term $ad_f^k(g)$ is constant, the involutivity condition is trivially satisfied, since the Lie bracket of two constant vector fields is zero. Therefore, controllability in the linear case is necessary and sufficient to transform the system into the Brunovsky canonical form with a linear change of variables and linear state feedback.*

3. *A necessary condition for the nonlinear system*

$$\dot{x} = f(x) + g(x)u$$

to be feedback linearizable is that the linear approximation about the origin

$$\dot{x} = Fx + gu \quad \text{where } F = \frac{\partial f}{\partial x}(0);\ g = g(0)$$

is controllable.

4. *For second-order systems, controllability of the linear approximation about the origin is both necessary and sufficient for feedback linearizability. This follows immediately from the fact that the distribution Δ in the case $n = 2$ is $\Delta = \text{span}\{g(x)\}$ since a one-dimensional distribution is always involutive.*

Example 12.3 (Flexible-Joint Robot)**.** *Consider the single-link, flexible-joint manipulator shown in Figure 12.5. Ignoring damping for simplicity, the equations of motion are*

$$\begin{aligned} I\ddot{q}_1 + Mgl\sin(q_1) + k(q_1 - q_2) &= 0 \\ J\ddot{q}_2 + k(q_2 - q_1) &= u \end{aligned} \tag{12.54}$$

Note that since the nonlinearity enters into the first equation the control u cannot simply be chosen to cancel it as in the case of the rigid manipulator

equations. In state space we set

$$\begin{array}{ll} x_1 = q_1 & x_2 = \dot{q}_1 \\ x_3 = q_2 & x_4 = \dot{q}_2 \end{array} \tag{12.55}$$

and write the system (12.54) as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{Mgl}{I}\sin(x_1) - \frac{k}{I}(x_1 - x_3) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{k}{J}(x_1 - x_3) + \frac{1}{J}u \end{aligned} \tag{12.56}$$

The system is thus of the form (12.32) with

$$f(x) = \begin{bmatrix} x_2 \\ -\dfrac{Mgl}{I}\sin(x_1) - \dfrac{k}{I}(x_1 - x_3) \\ x_4 \\ \dfrac{k}{J}(x_1 - x_3) \end{bmatrix}, \qquad g(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{J} \end{bmatrix} \tag{12.57}$$

Therefore, with $n = 4$, the necessary and sufficient conditions for feedback linearization of this system are that

$$\textit{rank}\left[g, ad_f(g), ad_f^2(g), ad_f^3(g)\right] = 4 \tag{12.58}$$

and that the distribution

$$\Delta = \text{span}\{g, ad_f(g), ad_f^2(g)\} \tag{12.59}$$

is involutive. Performing the indicated calculations it is easy to check that (Problem 12–10)

$$[g, ad_f(g), ad_f^2(g), ad_f^3(g)] = \begin{bmatrix} 0 & 0 & 0 & -\frac{k}{IJ} \\ 0 & 0 & \frac{k}{IJ} & 0 \\ 0 & -\frac{1}{J} & 0 & \frac{k}{J^2} \\ \frac{1}{J} & 0 & -\frac{k}{J^2} & 0 \end{bmatrix} \tag{12.60}$$

which has rank 4 for k, I, $J \neq 0$. Also, since the vector fields $\{g, ad_f(g), ad_f^2(g)\}$ are constant, the distribution Δ is involutive. To see this it suffices to note that the Lie bracket of two constant vector fields is zero. Hence, the

Lie bracket of any two members of the set of vector fields in Equation (12.59) is zero, which is trivially a linear combination of the vector fields themselves. It follows that the system given by Equation (12.54) is feedback linearizable. The new coordinates

$$y_i = T_i \qquad i = 1, \ldots, 4 \tag{12.61}$$

are found from the conditions given by Equation (12.49), with $n = 4$, that is

$$\begin{aligned} L_g T_1 &= 0 \\ L_{[f,g]} T_1 &= 0 \\ L_{ad_f^2}(g) T_1 &= 0 \\ L_{ad_f^3}(g) T_1 &\neq 0 \end{aligned} \tag{12.62}$$

Carrying out the above calculations leads to the system of equations (Problem 12–11)

$$\frac{\partial T_1}{\partial x_2} = 0 \; ; \; \frac{\partial T_1}{\partial x_3} = 0 \; ; \; \frac{\partial T_1}{\partial x_4} = 0 \tag{12.63}$$

and

$$\frac{\partial T_1}{\partial x_1} \neq 0 \tag{12.64}$$

From this we see that the function T_1 should be a function of x_1 alone. Therefore, we take the simplest solution

$$y_1 = T_1 = x_1 \tag{12.65}$$

and compute from Equation (12.45) (Problem 12–12)

$$\begin{aligned} y_2 &= T_2 = L_f T_1 = x_2 \\ y_3 &= T_3 = L_f T_2 = -\frac{Mgl}{I} \sin(x_1) - \frac{k}{I}(x_1 - x_3) \\ y_4 &= T_4 = L_f T_3 = -\frac{Mgl}{I} \cos(x_1) x_2 - \frac{k}{I}(x_2 - x_4) \end{aligned} \tag{12.66}$$

The feedback linearizing control input u is found from the condition

$$u = \frac{1}{L_g T_4}(v - L_f T_4) \tag{12.67}$$

as (Problem 12–13)

$$u = \frac{IJ}{k}(v - a(x)) = \beta(x) v + \alpha(x) \tag{12.68}$$

where

$$\begin{aligned} a(x) &= \frac{Mgl}{I}\sin(x_1)\left(x_2^2 + \frac{Mgl}{I}\cos(x_1) + \frac{k}{I}\right) \\ &\quad + \frac{k}{I}(x_1 - x_3)\left(\frac{k}{I} + \frac{k}{J} + \frac{Mgl}{I}\cos(x_1)\right) \end{aligned} \tag{12.69}$$

Therefore in the coordinates $y_1, \ldots, y_4$ with the control law given by Equation (12.68) the system becomes

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= y_3 \\ \dot{y}_3 &= y_4 \\ \dot{y}_4 &= v \end{aligned} \tag{12.70}$$

or, in matrix form,

$$\dot{y} = Ay + bv \tag{12.71}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{12.72}$$

It is interesting to note that the above feedback linearization is actually global. In order to see this we need only compute the inverse of the change of variables given by Equations (12.65) and (12.66). By inspection we see that

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 \\ x_3 &= y_1 + \frac{I}{k}\left(y_3 + \frac{Mgl}{I}\sin(y_1)\right) \\ x_4 &= y_2 + \frac{I}{k}\left(y_4 + \frac{Mgl}{I}\cos(y_1)y_2\right) \end{aligned} \tag{12.73}$$

The inverse transformation is well defined and differentiable everywhere and, hence, the feedback linearization for the system given by Equation (12.54) holds globally. The transformed variables $y_1, \ldots, y_4$ are themselves physically meaningful. We see that

$$\begin{aligned} y_1 = x_1 &= \textit{link position} \\ y_2 = x_2 &= \textit{link velocity} \\ y_3 = \dot{y}_2 &= \textit{link acceleration} \\ y_4 = \dot{y}_3 &= \textit{link jerk} \end{aligned} \tag{12.74}$$

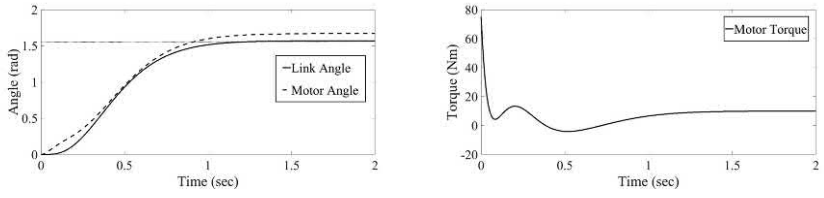


Figure 12.6: Step response and motor torque of the flexible joint robot. The difference between the final values of the motor angle and the link angle are determined by the joint stiffness and the steady-state value of the gravitational torque.

Since the motion trajectory of the link is typically specified in terms of these quantities, they are natural variables to use for feedback.

Example 12.4. *Once the system is linearized to Brunovsky form, we can choose the outer-loop control v as*

$$v = -k_1(y_1 - y_1^{ref}) - k_2 y_2 - k_3 y_3 - k_4 y_4$$

to execute a step change y_1^{ref} in the link angle.

Figure 12.6 shows the response of the system with parameter values $I = J = 1$, $mgL = 10$, $k = 100$, reference value $y_1^{ref} = \pi/2$, and outer loop gains $k_1 = 4788$, $k_2 = 2319$, $k_3 = 420$, $k_4 = 34$, chosen so that the closed poles of the linearized system are at $-7, -8, -9, -9.5$. Note that the final value of the input torque is $u = 10$, which is necessary to balance the gravitational torque $mgL = 10$.

Example 12.5. *The next example shows the response for the above flexible joint robot, where the reference input is a cubic polynomial trajectory. With a cubic polynomial the desired jerk is a constant. Therefore, let us specify a trajectory*

$$y_1^d(t) \;=\; y_1^d = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{12.75}$$

so that

$$\begin{aligned} y_2^d &= \dot{y}_1^d = a_1 + 2a_2 t + 3a_3 t^2 \\ y_3^d &= \dot{y}_2^d = 2a_2 + 6a_3 t \\ y_4^d &= \dot{y}_3^d = 6a_3 \end{aligned}$$

Then a linear control law that tracks this trajectory, which is essentially equivalent to the feedforward/feedback scheme of Chapter 9, is given by

$$v \;=\; -k_0(y_1 - y_1^d) - k_1(y_2 - y_2^d) - k_2(y_3 - y_3^d) - k_3(y_4 - y_4^d) \tag{12.76}$$

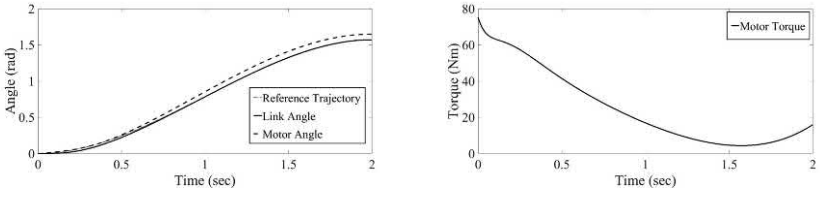


Figure 12.7: Tracking response and motor torque of the flexible joint robot with a cubic polynomial reference trajectory for the link angle y_1.

Applying this control law to the fourth-order linear system given by Equation (12.68) we see that the tracking error $e(t) = y_1 - y_1^d$ *satisfies the fourth-order linear equation*

$$\frac{d^4e}{dt^4} + k_3\frac{d^3e}{dt^3} + k_2\frac{d^2e}{dt^2} + k_1\frac{de}{dt} + k_0e \quad = \quad 0 \tag{12.77}$$

and, hence, the error dynamics are completely determined by the choice of gains $k_0, \ldots, k_3$.

Figure 12.7 shows the closed loop response of the system.

Remark 12.4. *Notice that the feedback control law given by Equation (12.76) is stated in terms of the variables* $y_1, \ldots, y_4$. *Thus, it is important to consider how these variables are to be determined so that they may be used for feedback in case they cannot be measured directly. Although the first two variables, representing the link position and velocity, are easy to measure, the remaining variables, representing link acceleration and jerk, are difficult to measure with any degree of accuracy using present technology. One could measure the original variables* $x_1, \ldots, x_4$, *which represent the motor and link positions and velocities, and compute* $y_1, \ldots, y_4$ *using the transformation Equations (12.65) and (12.66). In this case the parameters appearing in the transformation equations would have to be known precisely. One may also implement a nonlinear observer to estimate the full state, given only the measured link position* x_1.

Nonlinear Observer with Output Injection

It is interesting to note that the special structure of the dynamic equations (12.54), in the single-link case, allows us to design a nonlinear observer to estimate the full state vector, $x = [x_1, x_2, x_3, x_4]^T$ supposing that only the link angle x_1 is measured. To see this, set $y = x_1 = c^T x$ as an output equation with $c^T = [1, 0, 0, 0]$. We can then write the state space equations

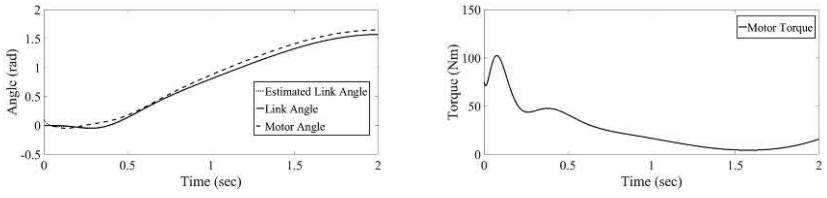


Figure 12.8: Tracking response and motor torque of the flexible joint robot with an observer. The link angle and estimated link angle are nearly indistinguishable.

(12.56) as

$$\dot{x} = Ax + bu + \phi(y)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{I} & 0 & \frac{k}{I} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J} & 0 & -\frac{k}{J} & 0 \end{bmatrix}, \; b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J} \end{bmatrix}, \; \phi = \begin{bmatrix} \frac{Mgl}{I}\sin(x_1) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.78}$$

Since $\phi(y)$ is a nonlinear function depending only on the output y, a so-called **nonlinear observer with output injection** takes the form

$$\dot{\hat{x}} = A\hat{x} + bu + \phi(y) + \ell(y - c^T\hat{x})$$

The estimation error $e = \hat{x} - x$ is then given by the linear system

$$\dot{e} = (A - \ell c^T)e \tag{12.79}$$

It is easily checked that (c^T, A) is an observable pair and so the observer gains ℓ can now be chosen so that $A - \ell c^T$ is a Hurwitz matrix. Figure 12.8 shows the closed-loop response of the system using the estimated state $\hat{y}_1$ in the control input. The observer gains were chosen as $\ell_1 = 46$, $\ell_2 = 591$, $\ell_3 = 14.3$, $\ell_4 = -419$.

12.4 Multi-Input Systems

In the general case of an n-link manipulator the dynamic equations represent a multi-input nonlinear system. The conditions for feedback linearization of

multi-input systems are more difficult to state, but the conceptual idea is the same as the single-input case. That is, one seeks a coordinate system in which the nonlinearities can be exactly cancelled by one or more of the inputs. In the multi-input system we can also decouple the system, that is, linearize the system in such a way that the resulting linear system is composed of subsystems, each of which is affected by only a single one of the outer-loop control inputs.

Since we are concerned only with the application of these ideas to manipulator control, we will not need the most general results in multi-input feedback linearization. Instead, we will use the physical insight gained by our detailed derivation of this result in the single-link case to derive a feedback linearizing control both for n-link rigid manipulators and for n-link manipulators with elastic joints. For fully-actuated rigid manipulators, no coordinate change from the usual generalized coordinates $q, \dot{q}$ is needed. For n-link flexible-joint robots, the required coordinate transformation can be defined in terms of the link position, velocity, acceleration, and jerk just as in the single-input case.

Example 12.6. *We will first verify what we have stated previously, namely that for an n-link rigid manipulator the feedback linearizing control is identical to the inverse dynamics control of Chapter 9. To see this, consider the rigid robot equations of motion given by Equation (9.2.6), which we write in state space as*

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -M(x_1)^{-1}(C(x_1, x_2)x_2 + g(x_1)) + M(x_1)^{-1}u \end{aligned} \tag{12.80}$$

with $x_1 = q$, $x_2 = \dot{q}$. In this case a feedback linearizing control is found by simply inspecting Equation (12.80) as

$$u = M(x_1)v + C(x_1, x_2)x_2 + g(x_1) \tag{12.81}$$

Substituting Equation (12.81) into Equation (12.80) yields

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= v \end{aligned} \tag{12.82}$$

Equation (12.82) represents a set of n second-order systems of the form

$$\begin{aligned} \dot{x}_{1i} &= x_{2i} \\ \dot{x}_{2i} &= v_i \ , \quad i = 1, \ldots, n \end{aligned} \tag{12.83}$$

Comparing Equation (12.81) with Equation (9.3.4) we see indeed that the feedback linearizing control for a rigid manipulator is precisely the inverse dynamics control of Chapter 9.

Example 12.7. *Including the joint flexibility in the dynamic description of an n-link robot results in a Lagrangian system with 2n degrees of freedom. Recall the Euler–Lagrange equations of motion for the flexible-joint robot from Chapter 9*

$$\begin{aligned} D(q_1)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 - K(q_1 - q_2) &= u \end{aligned} \tag{12.84}$$

In state space, which is now $\mathbb{R}^{4n}$, we define state variables in block form

$$\begin{aligned} x_1 &= q_1 \qquad x_2 = \dot{q}_1 \\ x_3 &= q_2 \qquad x_4 = \dot{q}_2 \end{aligned} \tag{12.85}$$

Then from Equation (12.84) we have:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -D(x_1)^{-1}\{h(x_1,x_2) + K(x_1 - x_3)\} \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= J^{-1}K(x_1 - x_3) + J^{-1}u \end{aligned} \tag{12.86}$$

where we define $h(x_1,x_2) = C(x_1,x_2)x_2 + g(x_1)$ for simplicity. This system is of the form

$$\dot{x} = f(x) + G(x)u \tag{12.87}$$

In the single-link case we saw that the system could be linearized by nonlinear feedback if we took as state variables the link position, velocity, acceleration, and jerk. Following the single-input example, we can attempt to do the same thing in the multi-link case and derive a feedback linearizing transformation blockwise as follows. Set

$$\begin{aligned} &y_1 = T_1(x) = x_1 \\ &y_2 = T_2(x) = \dot{y}_1 = x_2 \\ &y_3 = T_3(x) = \dot{y}_2 = \dot{x}_2 = -D^{-1}\{h(x_1,x_2) + K(x_1 - x_3)\} \\ &y_4 = T_4(x) = \dot{y}_3 = -\frac{d}{dt}[D^{-1}]\{h(x_1,x_2) + K(x_1 - x_3)\} \\ &-D^{-1}\left\{\frac{\partial h}{\partial x_1}x_2 + \frac{\partial h}{\partial x_2}[-D^{-1}(h(x_1,x_2) + K(x_1 - x_3))] + K(x_2 - x_4)\right\} \\ &= a_4(x_1,x_2,x_3) + D(x_1)^{-1}Kx_4 \end{aligned} \tag{12.88}$$

where for simplicity we define the function a_4 to be everything in the definition of y_4 except the last term, which is $D^{-1}Kx_4$. Note that x_4 appears only in this last term so that a_4 depends only on x_1, x_2, x_3.

As in the single-link case, the above mapping is a global diffeomorphism. Its inverse can be found by inspection to be

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 \\ x_3 &= y_1 + K^{-1}(D(y_1)y_3 + h(y_1, y_2)) \\ x_4 &= K^{-1}D(y_1)(y_4 - a_4(y_1, y_2, y_3)) \end{aligned} \tag{12.89}$$

The linearizing control law can now be found from the condition

$$\dot{y}_4 = v \tag{12.90}$$

Computing $\dot{y}_4$ from Equation (12.88) yields

$$\begin{aligned} v &= \frac{\partial a_4}{\partial x_1}x_2 - \frac{\partial a_4}{\partial x_2}D^{-1}(h + K(x_1 - x_3)) + \frac{\partial a_4}{\partial x_3}x_4 \\ &+ \frac{d}{dt}[D^{-1}]Kx_4 + D^{-1}K(J^{-1}K(x_1 - x_3) + J^{-1}u) \\ &= a(x) + b(x)u \end{aligned} \tag{12.91}$$

where $a(x)$ denotes all the terms in Equation (12.91) but the last term, which involves the input u, and $b(x) = D^{-1}(x)KJ^{-1}$.

Solving the above expression for u yields

$$u = b(x)^{-1}(v - a(x)) = \alpha(x) + \beta(x)v \tag{12.92}$$

where $\beta(x) = JK^{-1}D(x)$ and $\alpha(x) = -b(x)^{-1}a(x)$.

With the nonlinear change of coordinates given by Equation (12.88) and nonlinear feedback given by Equation (12.92) the transformed system now has the linear block form

$$\begin{aligned} \dot{y} &= \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{bmatrix} y + \begin{bmatrix} 0 \\ 0 \\ 0 \\ I \end{bmatrix} v \\ &= Ay + Bv \end{aligned} \tag{12.93}$$

where $I = n \times n$ identity matrix, $0 = n \times n$ zero matrix, $y^T = (y_1^T, y_2^T, y_3^T, y_4^T) \in \mathbb{R}^{4n}$, and $v \in \mathbb{R}^n$. The system (12.93) represents a set of n decoupled quadruple integrators. The outer-loop design can now proceed as before, because not only is the system linearized, but it consists of n subsystems each identical to the fourth-order system (12.70).

12.5 Chapter Summary

This chapter introduced some basic concepts from differential geometric nonlinear control theory and serves as a foundation to explore more advanced literature.

Manifolds, Vector Fields, Distributions

We introduced basic definitions from differential geometry, such as a differentiable manifold, vector field, and distribution. We introduced some geometric operations such as the Lie derivative and Lie bracket and showed how they are related. We stated the Frobenius Theorem, which is an important tool for nonlinear analysis.

Feedback Linearization

We derived the necessary and sufficient conditions for feedback linearization of single-input nonlinear systems. This important result serves as a basis for controller design for a wide range of physical systems. In particular, we showed how feedback linearization can be used to design globally stable tracking controllers for flexible joint robots.

Problems

12–1 Complete the proof of Lemma 12.1 by direct calculation.

12–2 Show that the function $h = z - \phi(x, y)$ satisfies the system given by Equation (12.17) if ϕ is a solution of Equations (12.10) and (12.11) and X_1, X_2 are defined by Equation (12.15).

12–3 Show that if $h(x, y, z)$ satisfies Equation (12.17) and $\frac{\partial h}{\partial z} \neq 0$, then Equation (12.18) can be solved for z as $z = \phi(x, y)$ where ϕ satisfies Equations (12.10) and (12.11). Also show that $\frac{\partial h}{\partial z} = 0$ can occur only in the case of the trivial solution $h = 0$ of Equation (12.17).

12–4 Verify Equations (12.49) and (12.50).

12–5 Show that if F and g are $n \times n$ and $n \times 1$, respectively, that

$$ad_F^k(g) = -1^k F g$$

12–6 Is the second-order system below feedback linearizable about the origin? Prove your answer.

$$\begin{aligned} \dot{x}_1 &= x_2 - x_1^2 \\ \dot{x}_2 &= x_1^2 + u \end{aligned}$$

12–7 Show that the system below is locally feedback linearizable.

$$\begin{aligned} \dot{x}_1 &= x_1^3 + x_2 \\ \dot{x}_2 &= x_2^3 + u \end{aligned}$$

Find explicitly the change of coordinates and nonlinear feedback to linearize the system.

12–8 Derive Equation (12.54), which gives the equations of motion for the single-link manipulator with joint elasticity of Figure 12.5 using Lagrange's equations.

12–9 Repeat Problem 12–8 if there is viscous friction both on the link side and on the motor side of the spring in Figure 12.5.

12–10 Perform the calculations necessary to verify Equation (12.60).

12–11 Derive the system of partial differential equations (12.63) from the condition (12.62). Also verify Equation (12.64).

12–12 Compute the change of coordinates (12.66).

12–13 Verify Equations (12.68) and (12.69).

12–14 Verify Equations (12.73).

12–15 Design and simulate a linear outer-loop control law v for the system given by Equation (12.54) so that the link angle $y_1(t)$ follows a desired trajectory $y_1^d(t) = \theta_\ell^d(t) = \sin 8t$. Use various techniques such as pole placement, linear quadratic optimal control, etc.

12–16 Consider again a single-link manipulator (either rigid or elastic joint). Add to your equations of motion the dynamics of a permanent magnet DC motor. What can you say now about feedback linearizability of the system?

12–17 What happens to the inverse coordinate transformation given by Equation (12.73) as the joint stiffness $k \to \infty$? Give a physical interpretation. Use this to show that the system given by Equation (12.54) reduces to the equation governing the rigid joint manipulator in the limit as $k \to \infty$.

12–18 Consider the single-link manipulator with elastic joint of Figure 12.5 but suppose that the spring characteristic is nonlinear, that is, suppose that the spring force F is given by $F = \phi(q_1 - q_2)$, where ϕ is a diffeomorphism. Derive conditions under which the system is feedback linearizable and carry out the details of the feedback linearizing transformation. Specialize the result to the case of a cubic spring characteristic $\phi = k_1(q_1 - q_2) + (q_1 - q_2)^3$. The cubic spring characteristic is a more accurate description for many manipulators than is the linear spring, especially for elasticity arising from gear flexibility.

Notes and References

A rigorous treatment of differential geometry can be found, for example, in a number of texts, for example, [15] or [156]. A comprehensive treatment of differential geometric methods in control is [69]. For specific applications in robotics of these advanced methods, the reader is referred to [118] and [32].

Our treatment of feedback linearization for single-input, affine, nonlinear systems follows closely the pioneering result of Su [171]. The first application of the method of feedback linearization for the single-link flexible-joint robot appeared in Marino and Spong [109]. The corresponding result for the case of n-link flexible-joint robots is due to Spong [165]. Dynamic feedback linearization for flexible-joint robots was treated in DeLuca [34]. An complete reference for flexible-joint robots is in Readman [142]. The problem of designing nonlinear observers is treated in [81] and [82].

Part IV

CONTROL OF UNDERACTUATED SYSTEMS

Chapter 13

UNDERACTUATED ROBOTS

13.1 Introduction

In this chapter we consider the control of an important class of robots known as **underactuated robots**. By an underactuated robot, or more generally an **underactuated mechanical system**, we mean one in which the number of independent control inputs is fewer than the number of generalized coordinates.

Underactuation arises in several ways, for example, from intentional design as in the so-called **Acrobot** and **Pendubot** described below, in which one or more of the joints are unactuated, or it may arise because of the mathematical model used for control design, such as when joint flexibility is included in the model. In the previous chapter we discussed the control of n-link flexible-joint robots, which have $2n$ degrees of freedom and n control inputs and hence fall into the class of underactuated robots.

Systems with **unilateral constraints** or **nonholonomic constraints** are also often underactuated. For example, in legged locomotion, the contact between the foot and the ground represents a unilateral constraint and is unactuated. Walking robots are therefore inherently underactuated. Wheeled robots, swimming robots, space robots, and flying robots are all examples of underactuated robots.

The class of underactuated robots is thus large and complex and the control problems are more difficult than for fully-actuated robots. As we

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

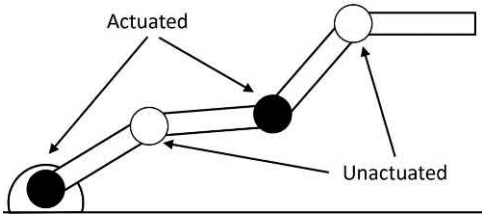


Figure 13.1: An underactuated serial-link robot.

saw in previous chapters, fully actuated robots possess a number of strong properties that facilitate control design. In particular, fully-actuated manipulator arms are globally feedback linearizable. This is generally not true for most underactuated systems, flexible-joint robots being an exception. As a result, the control problems for underactuated systems often require the development of new tools for controller design.

We will present modeling and control results in this chapter for a specific class of robots that includes underactuated serial-link robots. The tools we will use include **partial feedback linearization**, **switching control**, and **energy/passivity methods**. In Chapter 14 we will present additional control results for nonholonomic mobile robots.

13.2 Modeling

Figure 13.1 shows an underactuated serial-link robot. The shaded joints represent actuated degrees of freedom and the unshaded joints represent unactuated degrees of freedom. We assume that there are n joints with $m \leq n$ of the joints actuated and the remaining $n - m$ joints unactuated. The actuated degrees of freedom are called **active joints** and the unactuated degrees of freedom are called **passive joints**. The difference $\ell = n - m$ is called the **degree of underactuation**.

Upper-Actuated and Lower-Actuated Models

It is convenient for later analysis and control design to model the robot, by renumbering the joint variables if necessary, as either **upper-actuated** or **lower-actuated** as we define next.

Definition 13.1. *An* **upper-actuated system** *is one in which the first m, or proximal, joints are active and the remaining $\ell = n - m$, or distal, joints are passive.*

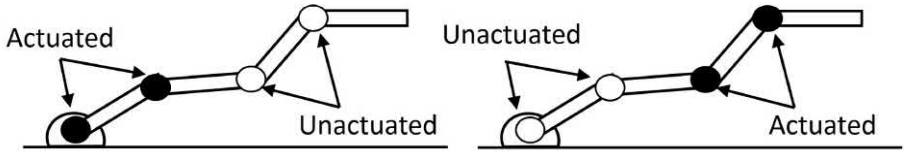


Figure 13.2: Upper-actuated (left) and lower-actuated (right) robots.

A **lower-actuated system** *is one in which the last m joints are active and the first $\ell = n - m$ joints are passive.*

Note that the terms upper and lower refer to the analogy to upper and lower arms rather than the joint numbers. This will become more clear in the examples that follow.

The dynamic equations of motion of a general n-DOF underactuated system can be derived using the tools from Chapter 6 and expressed as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \phi(q) = Bu \tag{13.1}$$

where $M(q)$ is the $n \times n$ inertia matrix, $C(q,\dot{q})$ is the Coriolis and centrifugal matrix and the vector $\phi(q)$ contains the generalized forces derived from the potential energy, such as the gravitational forces and elastic forces, if present. For simplicity, we will ignore the actuator dynamics and take the inertia matrix $M(q)$ to be identical to the matrix $D(q)$ as defined in Chapter 6.

The matrix B is an $n \times m$ matrix of rank m reflecting the fact that there are m independent actuators. For simplicity we take the matrix $B = B_u$ for an upper-actuated robot and $B = B_l$ for a lower-actuated robot, as

$$B_u = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad B_l = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

where I is the $m \times m$ identity matrix and 0 is an $\ell \times m$ matrix of zeros.

With the vector $q \in \mathbb{R}^n$ of generalized coordinates partitioned as $q = (q_1, q_2)$ with $q_1 \in \mathbb{R}^\ell$ and $q_2 \in \mathbb{R}^m$, we write the dynamic equations of a lower-actuated system as

$$M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 = 0 \tag{13.2}$$

$$M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 = u \tag{13.3}$$

where

$$M(q) = \begin{array}{c} \\ \ell \\ m \end{array} \begin{array}{c} \begin{array}{cc} \ell & m \end{array} \\ \left[\begin{array}{c|c} M_{11} & M_{12} \\ \hline M_{21} & M_{22} \end{array}\right] \end{array} \tag{13.4}$$

is a partition of the symmetric, positive definite inertia matrix into blocks

$$M_{11} \in \mathbb{R}^{\ell \times \ell}, \quad M_{12} \in \mathbb{R}^{\ell \times m}, \quad M_{21} \in \mathbb{R}^{m \times \ell}, \quad M_{22} \in \mathbb{R}^{m \times m}$$

the vectors $c_1(q, \dot{q}) \in \mathbb{R}^\ell$ and $c_2(q, \dot{q}) \in \mathbb{R}^m$ contain Coriolis and centrifugal terms, $\phi_1(q) \in \mathbb{R}^\ell$ and $\phi_2(q) \in \mathbb{R}^m$ are derived from the potential energy, and $u \in \mathbb{R}^m$ represents the input generalized forces at the active joints.

Likewise, with a similar partitioning of the generalized coordinates $q = (q_1, q_2)$ with $q_1 \in \mathbb{R}^m$ and $q_2 \in \mathbb{R}^\ell$ and similar partitioning of $M(q)$, $C(q, \dot{q})\dot{q}$, and $\phi(q)$ the dynamics of an upper-actuated system can be written as

$$\begin{aligned} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 &= u & (13.5) \\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 &= 0 & (13.6) \end{aligned}$$

In this case, the sub-blocks of the inertia matrix have dimensions shown below.

$$M_{11} \in \mathbb{R}^{m \times m}, \quad M_{12} \in \mathbb{R}^{m \times \ell}, \quad M_{21} \in \mathbb{R}^{\ell \times m}, \quad M_{22} \in \mathbb{R}^{\ell \times \ell}$$

and also $c_1(q, \dot{q}) \in \mathbb{R}^m$, $c_2(q, \dot{q}) \in \mathbb{R}^\ell$, $\phi_1(q) \in \mathbb{R}^m$, and $\phi_2(q) \in \mathbb{R}^\ell$.

Second-Order Constraints

Since the right-hand side of Equation (13.2) (or (13.6)) is equal to zero, this equation in effect defines constraints on the generalized coordinates. In particular, any reference trajectory $q^d(t), \dot{q}^d(t), \ddot{q}^d(t)$ must satisfy (13.2) (or(13.6)). Hence underactuated robots cannot track arbitrary trajectories, which is another important difference between fully-actuated and underactuated robots.

Example 13.1. *Recall the flexible-joint robot model from Chapter 12*

$$\begin{aligned} D(q_1)\ddot{q}_1 + C(q_1, \dot{q}_1)\dot{q} + g(q_1) + K(q_1 - q_2) &= 0 & (13.7) \\ J\ddot{q}_2 + K(q_2 - q_1) &= u & (13.8) \end{aligned}$$

which is in the lower-actuated form (13.2)–(13.3) with $M_{11} = D(q_1)$, $M_{12} = M_{21} = 0$, $M_{22} = J$, $c_1 = C(q_1, \dot{q}_1)\dot{q}_1$, $c_2 = 0$, $\phi_1 = g(q_1) + K(q_1 - q_2)$, *and* $\phi_2 = K(q_2 - q_1)$.

In this form, it is straightforward to show that the control input u can be designed so that the motor angle $q_2(t)$ *follows a desired motor angle trajectory* $q_2^d(t)$. *The resulting motion of the link angle* $q_1(t)$ *will then be determined by Equation (13.7).*

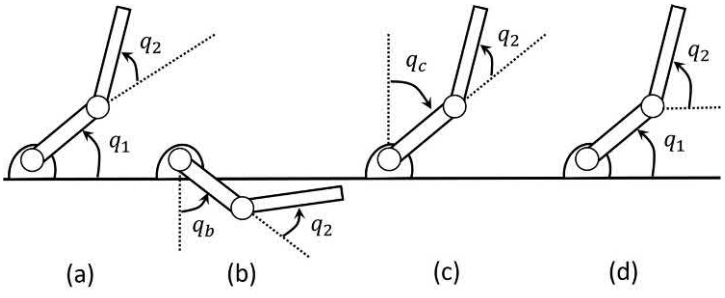


Figure 13.3: Illustrating common reference angle conventions.

Alternatively, as we showed in Chapter 12, this system is globally feedback linearizable in the state coordinates $y_1, \ldots, y_4$ with $y_1 = q_1$, the vector of link angles, and y_2, y_3, y_4 successive derivatives of y_1. In these coordinates, any desired trajectory $y_1^d(t) = q_1^d(t)$ can be tracked but no independent trajectory for the motor angles q_2 can be tracked. Rather, the resulting trajectory for q_2 is determined implicitly by the trajectory for q_1 and Equation (13.8). In either case, a desired trajectory may be specified for one, but not both, of the generalized coordinates.

13.3 Examples of Underactuated Robots

A Note About Angle Convention

In this section we give several examples of underactuated systems that we will use to illustrate various theoretical concepts and control design methods. Referring to Figure 13.3 we first note that the dynamic equations of motion that we derived in Chapter 6 used the DH convention for the joint angles q_1 and q_2 shown in (a). With a few exceptions, we will follow this convention throughout the present chapter. Depending on the context, other conventions such as (b), (c), or (d) are also used in books and research articles. The reader should note the particular convention used in each example.

13.3.1 The Cart-Pole System

The **cart-pole system** or **inverted pendulum on a cart**, shown in Figure 13.4, is a classic example used to illustrate nonlinear dynamics and test various control strategies. The inverted pendulum is representative of several practical systems and applications. The overhead crane transporting a load can be modeled as a cart-pole system. In this case, the control chal-

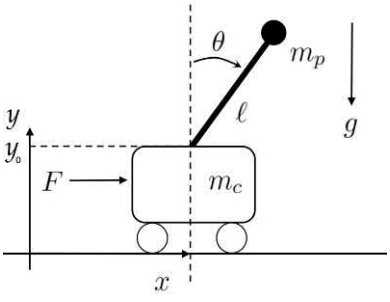


Figure 13.4: The inverted pendulum on a cart.

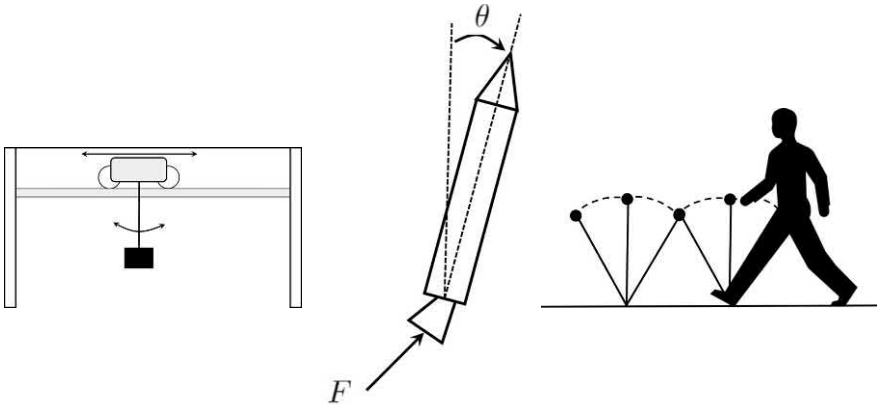


Figure 13.5: An overhead crane, a gimballed rocket and bipedal walking as examples of the inverted pendulum.

lenge is to transport the load while minimizing the pendular swing motion of the load, referred to as sway. The pitch dynamics of a rocket ascending vertically with gimballed thrust is similar to an inverted pendulum. Fuel slosh in the tanks also exhibits pendulum-like dynamic behavior. Likewise, the inverted pendulum is often used as a simple model to study problems of balance and walking in bipedal locomotion. Figure 13.5 illustrates these examples of pendulum-like dynamics.

Referring to Figure 13.4, the cart moves linearly in the x direction subject to an input force F. The pendulum is unactuated, that is, there is no input torque acting at the pivot connecting the pendulum to the cart. Note that the cart-pole system is kinematically identical to a two degree-of-freedom PR robot.

To derive the equations of motion, we let x denote the cart position and θ denote the pendulum angle relative to the vertical position. With the

angle convention shown, the (x, y) coordinates of the cart and pendulum mass can be written, respectively, as

$$\begin{bmatrix} x \\ y \end{bmatrix} \text{ and } \begin{bmatrix} x + \ell \sin(\theta) \\ y_0 + \ell \cos(\theta) \end{bmatrix} \tag{13.9}$$

Thus, the cart kinetic energy, K_c, and the pole kinetic energy, K_p, are

$$K_c = \frac{1}{2} m_c \dot{x}^2 \text{ and } K_p = \frac{1}{2} m_p \frac{d}{dt}((x + \ell \sin(\theta))^2 + (y_0 + \ell \cos(\theta))^2) \tag{13.10}$$

The potential energy of the cart-pole system is

$$P = mg\ell \cos(\theta) \tag{13.11}$$

The Euler-Lagrange equations are therefore given by (Problem 13–1)

$$\begin{aligned} (m_p + m_c)\ddot{x} + m_p \ell \cos(\theta)\ddot{\theta} - m_c \dot{\theta}^2 \sin(\theta) &= F & (13.12) \\ m_p \ell \cos(\theta)\ddot{x} + m_p \ddot{\theta} - m_p \ell g \sin(\theta) &= 0 & (13.13) \end{aligned}$$

System (13.12)–(13.13) is in the form of an upper-actuated system if we take $q_1 = x$, $q_2 = \theta$ and $u = F$.

Note that, if instead we take as generalized coordinates $q_1 = \theta$ and $q_2 = x$, we can write the system as a lower-actuated system

$$\begin{aligned} m_p \ddot{q}_1 + m_p \ell \cos(q_1)\ddot{q}_2 - m_p \ell g \sin(q_1) &= 0 & (13.14) \\ m_p \ell \cos(q_1)\ddot{q}_1 + (m_p + m_c)\ddot{q}_2 - m_c \dot{q}_1^2 \sin(q_1) &= u & (13.15) \end{aligned}$$

showing that we may use the upper-actuated or lower-actuated models interchangeably as convenient.

13.3.2 The Acrobot

The **Acrobot**, short for **Acro**batic Ro**bot**, is a two-link RR robot with actuation at the second link. The Acrobot is representative of a gymnast on a high bar where q_2, u_2 represent a hip angle and hip torque, respectively. There is no actuator at the first joint where the hands grasp the bar. Referring to Figure 6.9, the dynamic equations of the Acrobot are identical to the two-link RR robot given by Equation (6.90) with the torque at the first joint set to zero.

$$\begin{aligned} m_{11}\ddot{q}_1 + m_{12}\ddot{q}_2 + c_1 + g_1 &= 0 & (13.16) \\ m_{21}\ddot{q}_1 + m_{22}\ddot{q}_2 + c_2 + g_2 &= u & (13.17) \end{aligned}$$

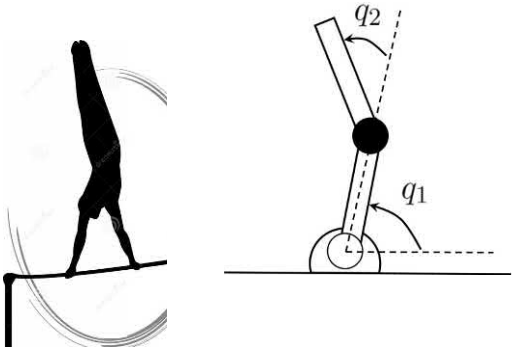


Figure 13.6: The Acrobot as a gymnastic robot.

where

$$\begin{aligned} m_{11} &= m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2 + 2\ell_1\ell_{c2}\cos q_2) + I_1 + I_2 \\ m_{12} &= m_{21} = m_2(\ell_{c2}^2 + \ell_1\ell_{c2}\cos q_2) + I_2 \\ m_{22} &= m_2\ell_{c2}^2 + I_2 \end{aligned}$$

$$\begin{aligned} c_1 &= -m_2\ell_1\ell_{c2}\sin(q_2)(2\dot{q}_1\dot{q}_2 + \dot{q}_1^2) \\ c_2 &= m_2\ell_1\ell_{c2}\dot{q}_1^2 \end{aligned}$$

$$\begin{aligned} g_1 &= (m_1\ell_{c1} + m_2\ell_1)g\cos q_1 + m_2\ell_{c2}g\cos(q_1 + q_2) \\ g_2 &= m_2\ell_{c2}g\cos(q_1 + q_2) \end{aligned}$$

with all parameters defined as in Chapter 6.

13.3.3 The Pendubot

The **Pendubot**, or **Pendu**lum Ro**bot**, is likewise a two-link RR robot. In this case only the first link is actuated. The Pendubot is a variation of the cart-pole system where the rotational first link plays the role of the cart and is used to balance the passive second link, which plays the role of the pendulum. The dynamic equations are of the form

$$m_{11}\ddot{q}_1 + m_{12}\ddot{q}_2 + c_1 + g_1 = u \tag{13.18}$$

$$m_{21}\ddot{q}_1 + m_{22}\ddot{q}_2 + c_2 + g_2 = 0 \tag{13.19}$$

where the left-hand side is identical to the Acrobot dynamics and u is the input torque at the first joint.

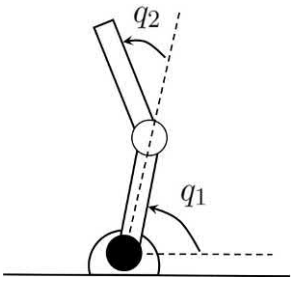


Figure 13.7: The Pendubot.

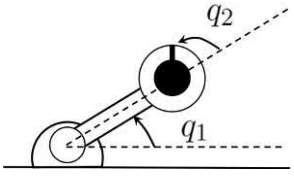


Figure 13.8: The Reaction-Wheel Pendulum.

13.3.4 The Reaction-Wheel Pendulum

The **Reaction-Wheel Pendulum** is a simple pendulum with a rotating disk at the distal end. Actuating the disk results in a reaction torque to move the pendulum.

The dynamic equations of the Reaction-Wheel Pendulum are the simplest of the various examples considered so far. In order to derive the equations of motion for the Reaction-Wheel Pendulum we may observe that if the second link of the Acrobot is counterbalanced to place its center of mass at the axis of the second joint, so that $\ell_2 = \ell_{c2} = 0$, then the equations of motion of the Acrobot reduce to those of the Reaction-Wheel Pendulum. Therefore, we leave it as an exercise (Problem 13–2) to show that the equations of motion of the Reaction-Wheel Pendulum can be expressed in the form of a lower-actuated system as

$$\begin{aligned} m_{11}\ddot{q}_1 + m_{12}\ddot{q}_2 + g_1(q_1) &= 0 && (13.20)\\ m_{21}\ddot{q}_1 + m_{22}\ddot{q}_2 &= u && (13.21) \end{aligned}$$

where

$$\begin{aligned} m_{11} &= m_1\ell_{c1}^2 + I_1 + I_2\\ m_{12} &= m_{21} = m_{22} = I_2\\ g_1(q_1) &= (m_1\ell_{c_1} + m_2\ell_1)g\cos(q_1) \end{aligned}$$

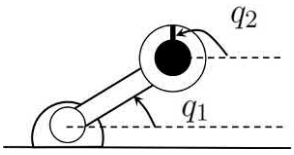


Figure 13.9: The Reaction Wheel Pendulum.

Remark 13.1. *Instead of using the DH convention for the joint angles, if we define the angle of the reaction-wheel relative to the horizontal as shown in Figure 13.9, then it is left as an exercise (Problem 13–3) to show that the equations of motion simplify to*

$$J_1\ddot{q}_1 + (m_1\ell_{c_1} + m_2\ell_1)g\cos(q_1) = -u \tag{13.22}$$
$$J_2\ddot{q}_2 = u \tag{13.23}$$

where we define $J_1 = m_1\ell_{c_1}^2 + m_2\ell_1^2$ *and* $J_2 = I_2$*. An additional simplification results if we assume that the mass of the first link is concentrated at joint* 2*, i.e.* $\ell c_1 = \ell_1 = \ell$*. In this case we can write the system as*

$$J_1\ddot{q}_1 + mg\ell\cos(q_1) = -u \tag{13.24}$$
$$J_2\ddot{q}_2 = u \tag{13.25}$$

where $m = m_1 + m_2$*. In this form, the equations of motion of the Reaction-Wheel Pendulum can be seen as the parallel combination of a simple pendulum and a double integrator. Parallel in this context means that Equations (13.24) and (13.25) have the same input* u*. We will make this more precise in Section 13.7.2 when we discuss passivity-based control.*

13.4 Equilibria and Linear Controllability

For a general Lagrangian mechanical system (13.1) with n degrees of freedom we define the state of the system, $x(t) \in \mathbb{R}^N$, $N = 2n$, in terms of the generalized coordinates and generalized velocities as

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix}$$

The state equations are then given by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -M^{-1}(x_1)(C(x_1, x_2)x_2 + \phi(x_1) - Bu) \end{aligned}$$

which can be written as

$$\dot{x} = \mathcal{F}(x, u) = f(x) + g(x)u \tag{13.26}$$

with

$$f(x) = \begin{pmatrix} x_2 \\ -M^{-1}(x_1)(C(x_1, x_2)x_2 + \phi(x_1)) \end{pmatrix} \tag{13.27}$$

$$g(x) = \begin{pmatrix} 0 \\ M^{-1}(x_1)Bu \end{pmatrix} \tag{13.28}$$

Definition 13.2. *An* **equilibrium** *of a dynamical system $\dot{x} = \mathcal{F}(x, u)$ is a constant vector (x_e, u_e) satisfying*

$$\mathcal{F}(x_e, u_e) = 0 \tag{13.29}$$

Examining Equations (13.27) and (13.28) we see that $x_2 = 0$ at an equilibrium and therefore the equilibrium configurations are given as solutions of the equation

$$\phi(x_{1e}) = Bu_e \tag{13.30}$$

In particular with $u_e = 0$, Equation (13.30) shows that the equilibrium points are local extrema (minima or maxima) of the potential energy, since ϕ is the gradient of the potential energy.

The equilibrium points may be isolated fixed points for each u_e as shown below in the case of the Acrobot and Pendubot or they may be non-isolated as happens for systems without potential energy terms. For example, in the absence of potential energy (gravity or elasticity), and with $u_e = 0$, Equation (13.30) shows that every configuration $x_1 = q$ in the configuration space Q corresponds to an equilibrium point $(q, 0)$ in the state space. The nature of the equilibrium configurations of the system (13.1) is closely related to its controllability properties.

Example 13.2. *Consider the Acrobot/Pendubot with $u_1 = u_2 = 0$. It is easy to show (Problem 13–4) that the only equilibrium points belong to the following set:*

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \in \left\{ \begin{bmatrix} -\pi/2 \\ 0 \end{bmatrix}, \begin{bmatrix} -\pi/2 \\ \pi \end{bmatrix}, \begin{bmatrix} \pi/2 \\ \pi \end{bmatrix}, \begin{bmatrix} \pi/2 \\ 0 \end{bmatrix} \right\}$$

as shown in Figure 13.10.

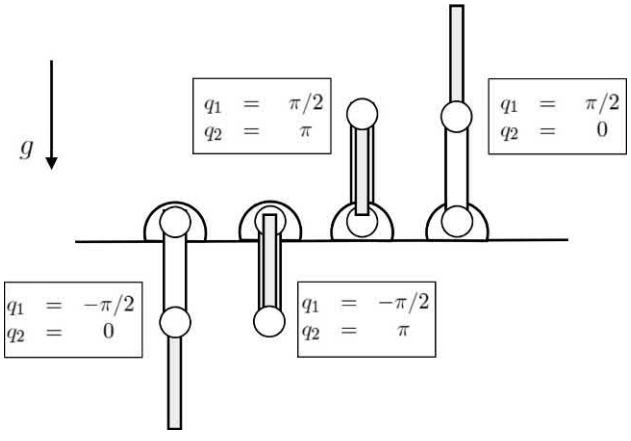


Figure 13.10: Equilibrium configurations of the Acrobot and Pendubot under gravity with zero input torque.

13.4.1 Linear Controllability

We next discuss the notion of **linear controllability**, which refers to controllability of the linear approximation of a nonlinear system about an equilibrium.

Definition 13.3. *Given a nonlinear dynamical system*

$$\dot{x} \;=\; \mathcal{F}(x,u),\; x \in \mathbb{R}^N,\; u \in \mathbb{R}^m \tag{13.31}$$

suppose that x_e, u_e defines an equilibrium of the system, i.e., $\mathcal{F}(x_e, u_e) = 0$, and let

$$\dot{\tilde{x}} = F\tilde{x} + G\tilde{u} \tag{13.32}$$

with $\tilde{x} = x - x_e$, $\tilde{u} = u - u_e$ be the linear approximation of (13.31) at x_e, u_e. Recall that this means

$$F = \frac{\partial \mathcal{F}}{\partial x}(x_e, u_e),\; G = \frac{\partial \mathcal{F}}{\partial u}(x_e, u_e) \tag{13.33}$$

where $\frac{\partial \mathcal{F}}{\partial x}(x_e, u_e) \in \mathbb{R}^{N\times N}$ and $\frac{\partial \mathcal{F}}{\partial u}(x_e, u_e) \in \mathbb{R}^{N\times m}$ are Jacobian matrices of $F(x,u)$ evaluated at x_e, u_e. Then the nonlinear system (13.31) is said to be **linearly controllable at** *x_e, u_e if the linear system (13.32) is a controllable linear system, which is equivalent to the statement that*

$$\text{rank } [G, FG, F^2G, \ldots, F^{N-1}G] = N \tag{13.34}$$

The property of linear controllability allows the design of linear control laws for local exponential stabilization around equilibrium points. In addition, linear controllability is also useful in the context of switching control to achieve global or almost global stability. Systems that are not linearly controllable, such as the nonholonomic systems considered in Chapter 14, require fundamentally different design approaches even for local stabilization as we shall see.

Computation of the Linearization

To compute the linear approximation about an equilibrium of a Lagrangian mechanical system

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \phi(q) = Bu$$

we write the above system in the state space form (13.26) and suppose that $x_e = (x_{1e}, x_{2e})$ and u_e define an equilibrium. Note that $x_{1e} = q_e$ and $x_{2e} = \dot{q}_e = 0$. The Jacobians given by (13.33) are then

$$F = \frac{\partial \mathcal{F}}{\partial x}(x_e, u_e) = \begin{bmatrix} 0 & I \\ -M^{-1}(x_{1e})\dfrac{\partial \phi}{\partial x}(x_{1e}) & 0 \end{bmatrix} \tag{13.35}$$

$$G = \frac{\partial \mathcal{F}}{\partial u}(x_e, u_e) = \begin{bmatrix} 0 \\ M^{-1}(x_{1e})B \end{bmatrix} \tag{13.36}$$

To see why this is true, note that the Coriolis and centrifugal terms $C(q,\dot{q})\dot{q}$ are quadratic in the velocities and hence their partial derivatives vanish at $\dot{q} = 0$. Likewise, the partial derivative of M^{-1} is multiplied by $\phi - Bu$, which also vanishes at the equilibrium. The details are left as an exercise (Problem 13–5).

A Necessary Condition for Linear Controllability

We can therefore write the linear approximation of the system in state space using (13.35) and (13.36) as

$$\dot{\tilde{x}}_1 = \tilde{x}_2 \tag{13.37}$$

$$\dot{\tilde{x}}_2 = M^{-1}(x_{1e})(-\frac{\partial \phi}{\partial x}(x_{1e})\tilde{x}_1 + B\tilde{u}) \tag{13.38}$$

Since $M(q_e)$ has full rank n, it follows that $B\tilde{u} - \dfrac{\partial \phi}{\partial q}(q_e)\tilde{x}_1$ must have full row rank in order for the linearization to be controllable. For a lower-actuated

system, i.e. $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$, we can express Equations (13.37) and (13.38) as

$$M_{11}\ddot{\tilde{q}}_1 + M_{12}\ddot{\tilde{q}}_2 = -\frac{\partial \phi_1}{\partial q}(q_e)\tilde{q}_1 \tag{13.39}$$

$$M_{21}\ddot{\tilde{q}}_1 + M_{22}\ddot{\tilde{q}}_2 = -\frac{\partial \phi_2}{\partial q}(q_e)\tilde{q}_1 + \tilde{u} \tag{13.40}$$

It follows that $\frac{\partial \phi_1}{\partial q}(q_e)$ must have full row rank. Since $\frac{\partial \phi_1}{\partial q}(q_e)$ has dimension $\ell \times m$, it is necessary that $m \geq \ell$. Therefore, we can state the following

Proposition 13.1. *A lower-actuated system is linearly controllable at an equilibrium $q = q_e$, $\dot{q} = 0$, $u = u_e$ only if*

- *$m \geq \ell$, that is, the number of active joints is at least as great as the number of passive joints, and*
- *$\frac{\partial \phi_1}{\partial q}(q_e)$ has full row rank.*

Remark 13.2. *An identical argument shows that an upper-actuated system is linearly controllable only if*

- *$m \geq \ell$, that is, the number of active joints is at least as great as the number of passive joints, and*
- *$\frac{\partial \phi_2}{\partial q}(q_e)$ has full row rank.*

An important implication of Proposition 13.1 is that each passive joint axis must have a non-zero potential force, such as a gravitational or an elastic force, at a given equilibrium configuration in order to be linearly controllable at that equilibrium. In particular, serial-link robots without gravitational or elastic forces at the passive joints are never linearly controllable.

Example 13.3. *Consider the Reaction-Wheel Pendulum*

$$\begin{aligned} J_1\ddot{q}_1 + mg\ell\cos(q_1) &= -u \\ J_2\ddot{q}_2 &= u \end{aligned}$$

which is equivalent to the lower-actuated system

$$\begin{aligned} J_1\ddot{q}_1 + J_2\ddot{q}_2 + mg\ell\cos(q_1) &= 0 \\ J_2\ddot{q}_2 &= u \end{aligned}$$

It follows immediately from Proposition 13.1 that the system is linearly controllable only if $mg\ell$ is nonzero. In this case, the condition turns out to be sufficient as well. With state vector $x = (q_1, q_2, \dot{q}_1, \dot{q}_2)^T$ we can write this system in state space as

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= -\frac{mg\ell}{J_1}\cos(x_1) - \frac{1}{J_1}u \\ \dot{x}_4 &= \frac{1}{J_2}u \end{aligned}$$

It is easily computed that $x_e = (\pm\pi/2, 0, 0, 0)$ are equilibrium points and that the linear approximations about these equilibrium points are

$$\dot{x} = Fx + Gu$$

where

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \pm\frac{mg\ell}{J_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad G = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{J_1} \\ \frac{1}{J_2} \end{bmatrix}$$

It is left as an exercise (Problem 13–6) to show that the linearized systems at each equilibrium are controllable if and only if $\frac{mg\ell}{J_1}$ is nonzero.

Example 13.4. *Let's consider the problem of designing a control law to balance the Reaction-Wheel Pendulum about the inverted equilibrium $q_1 = \pi/2$, $q_2 = 0$ (with zero velocity). For simplicity we take $m = \ell = J_1 = J_2 = 1$. Therefore, with $g = 9.8$, the linear approximation at the inverted equilibrium is*

$$\dot{x} = Fx + Gu \tag{13.41}$$

with

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 9.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

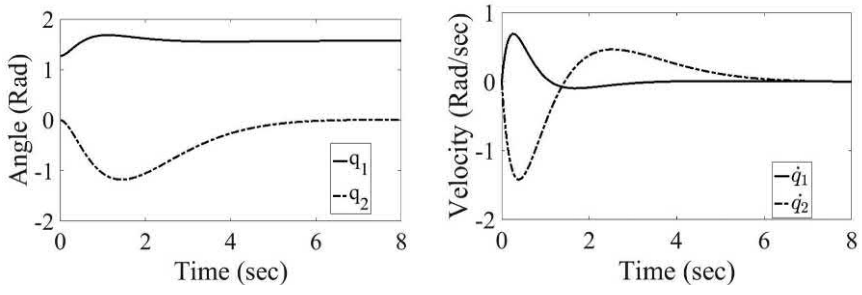


Figure 13.11: Local stabilization of the Reaction-Wheel Pendulum at the inverted position $q_1 = \pi/2$, $q_2 = 0$.

A stabilizing controller $u = -k^T x$ for this linear system can be found using Matlab's **lqr** *function that computes the optimal control minimizing*

$$J = \int_0^\infty (x^T Q x + R u^2) dt$$

subject to (13.41). With Q as the 4×4 identity matrix and $r = 1$, the optimal gain turns out to be

$$k^T = [-117.38, -30.38, -43.58, -13.43]$$

A particular response of the system with this controller is shown in Figure 13.11.

Example 13.5. *We next give a more detailed example using the Pendubot that gives further insight into the property of linear controllability. Figure 13.10 showed the equilibrium configurations of the Pendubot with zero input torque. A nonzero constant torque u_e can hold the first link at a fixed value q_{1e}. Specifically, with the gravitational torque at link 1 given by Equation (5.85)*

$$g_1 = (m_1 \ell_{c1} + m_2 \ell_1) g \cos(q_1) + m g \ell_{c2} g \cos(q_1 + q_2)$$

let q_{1e} be any desired angle and take q_{2e} so that $q_{1e} + q_{2e} = \pi/2$ or $3\pi/2$. Then since $\cos(q_{1e} + q_{2e}) = \pm 1$ the constant torque input

$$u_e = (m_1 \ell_{c1} + m_2 \ell_1) g \cos(q_{1e}) \pm m g \ell_{c2} g$$

corresponds to equilibrium configurations of the type shown in Figure 13.12. The Pendubot thus has a rich set of configurations to balance the second link.

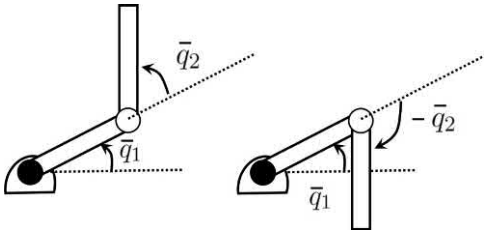


Figure 13.12: Equilibrium configurations of the Pendubot for u_e nonzero.

Since the gravitational torque at the second link is given by

$$g_2 = m_2 \ell_{c2} g \cos(q_1 + q_2)$$

it follows from Proposition 13.1 that the system is linearly controllable only if $\sin(q_{1e} + q_{2e}) \neq 0$, *which is satisfied at each equilibrium configuration* $q_{1e} + q_{2e} = \pm\pi/2$. *In fact, the Pendubot is linearly controllable at each such equilibrium except when the first link is horizontal, as we show next.*

Since the Pendubot is upper-actuated, a straightforward calculation shows that the linear approximation at any equilibrium x_e *is*

$$\dot{\tilde{x}} = F\tilde{x} + G\tilde{u}, \text{with } F \in \mathbb{R}^{4\times 4} \text{ and } G \in \mathbb{R}^{4\times 1}$$

where

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ f_{31}(x_e) & f_{32}(x_e) & 0 & 0 \\ f_{41}(x_e) & f_{42}(x_e) & 0 & 0 \end{bmatrix}, \qquad G = \begin{bmatrix} 0 \\ 0 \\ -\frac{m_{22}(x_e)}{\Delta(x_e)} \\ \frac{m_{21}(x_e)}{\Delta(x_e)} \end{bmatrix}$$

with

$$\begin{aligned} f_{31} &= \frac{1}{\Delta}(-m_{22}\frac{\partial \phi_1}{\partial x_1} + m_{12}\frac{\partial \phi_2}{\partial x_1}) \\ f_{32} &= \frac{1}{\Delta}(-m_{22}\frac{\partial \phi_1}{\partial x_2} + m_{12}\frac{\partial \phi_2}{\partial x_2}) \\ f_{41} &= \frac{1}{\Delta}(m_{21}\frac{\partial \phi_1}{\partial x_1} - m_{11}\frac{\partial \phi_2}{\partial x_1}) \\ f_{42} &= \frac{1}{\Delta}(m_{21}\frac{\partial \phi_1}{\partial x_2} - m_{11}\frac{\partial \phi_2}{\partial x_2}) \end{aligned}$$

and $\Delta = m_{11}m_{22} - m_{12}m_{21}$.

m_1	m_2	ℓ_1	ℓ_{c1}	ℓ_{c2}	I_1	I_2
1	1	2	1	1	1	1

Table 13.1: Example Pendubot inertia parameters.

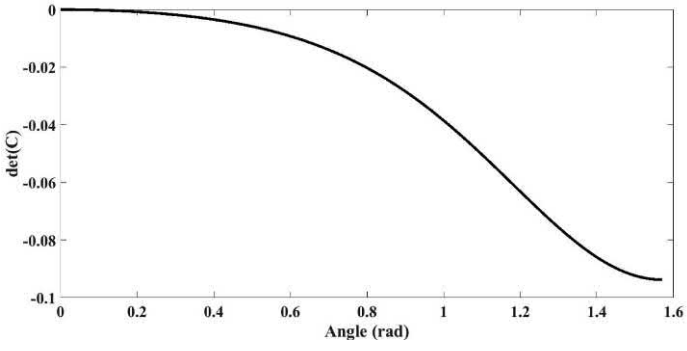


Figure 13.13: The determinant of the controllability matrix for equilibrium positions $(0, \pi/2)$ to $(\pi/2, 0)$. The Pendubot is not linearly controllable at $q_{1e} = 0, q_{2e} = \pi/2$, but is linearly controllable at all other equilibria.

Using the parameters in Table 13.1 we can compute the linear approximation for each $q_{1e} \in [0, \pi/2]$ *with* $q_{2e} = \pi/2 - q_{1e}$*, and we denote by* $\mathcal{C}$ *the* 4×4 *controllability matrix,* $\mathcal{C} = [G, FG, F^2G, F^3G]$.

Figure 13.13 shows a plot of the determinant of the controllability matrix $\mathcal{C}$ *in the interval* $q_{1e} \in [0, \pi/2]$ *showing that the linear system is uncontrollable at* $q_{1e} = 0$ *and controllable at all other equilibria in this interval.*

13.5 Partial Feedback Linearization

In this section we introduce the notions of **collocated** and **noncollocated partial feedback linearization** for underactuated robots. By collocated partial feedback linearization we mean using nonlinear feedback to create a linear relationship between the accelerations of the active joints and their respective inputs. Noncollocated partial feedback linearization means establishing a linear relationship between the accelerations of the passive joints and the inputs to the active joints. In both cases, we obtain systems of double integrator equations, of the form

$$\ddot{q}_i = a_i, \ i = 1 \text{ or } 2$$

where a_i is an outer-loop control, as in the case of the inverse dynamics in Chapter 9. Both the collocated and noncollocated partial feedback lineariza-

tion approaches lead to **normal forms** that are useful to design control laws in a host of applications, including the control of gymnastic robots, bipedal walking robots, snake robots, and others.

13.5.1 Collocated Partial Feedback Linearization

Consider the lower-actuated system[1]

$$\begin{aligned} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 &= 0 \qquad (13.42)\\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 &= u \qquad (13.43)\end{aligned}$$

Let us examine in more detail the first equation (13.42) above

$$M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 = 0 \qquad (13.44)$$

The term $M_{11} \in \mathbb{R}^{\ell\times\ell}$ is nonsingular as a result of the uniform positive definiteness of the robot inertia matrix M. Therefore, we may solve for $\ddot{q}_1$ in (13.44) as

$$\ddot{q}_1 = -M_{11}^{-1}(M_{12}\ddot{q}_2 + c_1 + \phi_1) \qquad (13.45)$$

and substitute the resulting expression (13.45) into (13.43) to obtain

$$\bar{M}_{22}\ddot{q}_2 + \bar{c}_2 + \bar{\phi}_2 = u \qquad (13.46)$$

where the terms $\bar{M}_{22}$, $\bar{c}_2$ and $\bar{\phi}_2$ are given by (Problem 13–7)

$$\begin{aligned} \bar{M}_{22} &= M_{22} - M_{21}M_{11}^{-1}M_{12}\\ \bar{c}_2 &= c_2 - M_{21}M_{11}^{-1}c_1 \qquad (13.47)\\ \bar{\phi}_2 &= \phi_2 - M_{21}M_{11}^{-1}\phi_1\end{aligned}$$

Proposition 13.2. *The $m \times m$ matrix $\bar{M}_{22}$ is symmetric and positive definite at each $q \in \mathbb{R}^m$.*

Proof: To see this, it is left as an exercise (Problem 13–8) to show that

$$\bar{M}_{22} = S^T M S \qquad (13.48)$$

where S is the $n \times m$ matrix

$$S = \begin{bmatrix} -M_{11}^{-1}M_{12} \\ I_{m\times m} \end{bmatrix} \qquad (13.49)$$

[1] As noted before, we could just as easily choose to work with the upper-actuated system.

with $I_{m\times m}$ denoting the $m \times m$ identity matrix. Since the matrix S has rank m for all q and M is symmetric, positive definite, it follows that $\bar{M}_{22}$ is likewise symmetric and positive definite.

Referring to Appendix B, we see that the matrix $\bar{M}_{22}$ is the **Schur complement** of M_{22} in M. Now, by inspection, we can see that the control law

$$u = \bar{M}_{22}a_2 + \bar{c}_2 + \bar{\phi}_2 \tag{13.50}$$

where $a_2 \in \mathbb{R}^m$ is an additional outer-loop control term, results in

$$\ddot{q}_2 = a_2 \tag{13.51}$$

The complete system up to this point may be written as

$$\begin{aligned} M_{11}\ddot{q}_1 + c_1 + \phi_1 &= -M_{12}a_2 && (13.52)\\ \ddot{q}_2 &= a_2 && (13.53) \end{aligned}$$

Definition 13.4. *The system (13.52)–(13.53) is called a* **second-order normal form with input-driven internal dynamics***, or simply* **second-order normal form***. Equation (13.52) is called the* **internal dynamics***.*

Since the system (13.52)–(13.53) is **feedback equivalent** to the original system it can be used as a starting point for subsequent control analysis and design.

Example 13.6. *Consider the cart-pole system given by Equations (13.14)–(13.15) and let us normalize all constants to unity for simplicity*

$$\begin{aligned} \ddot{q}_1 + \cos(q_1)\ddot{q}_2 - \sin(q_1) &= 0 && (13.54)\\ \cos(q_1)\ddot{q}_1 + 2\ddot{q}_2 - \dot{q}_1^2\sin(q_1) &= u && (13.55) \end{aligned}$$

It is easy to show (Problem 13–10) that the collocated partial feedback linearization control

$$u = (2 - \cos^2(q_1))a_2 + \cos(q_1)\sin(q_1) - \dot{q}_1^2\sin(q_1) \tag{13.56}$$

results in the normal form

$$\begin{aligned} \ddot{q}_1 - \sin(q_1) &= -\cos(q_1)a_2 && (13.57)\\ \ddot{q}_2 &= a_2 && (13.58) \end{aligned}$$

13.5.2 Noncollocated Partial Feedback Linearization

In the previous section we showed that the dynamics of the active degrees of freedom can be globally linearized by nonlinear feedback. In this section we show, under a condition regarding the degree of coupling among the active and passive degrees of freedom, that a similar partial feedback linearizing control can linearize the dynamics of the passive degrees of freedom. This is an interesting and, at first glance, somewhat surprising result. In this case, the linearization may hold either locally or globally.

Consider again the lower-actuated system

$$M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 = 0 \tag{13.59}$$

$$M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 = u \tag{13.60}$$

The inertia matrix terms M_{12} and $M_{21} = M_{12}^T$ generate coupling generalized forces among the degrees of freedom. For example, a control input torque u in (13.60) will not only result in an acceleration of the active degrees of freedom q_2 but also an acceleration of the passive degrees of freedom q_1, and the latter acceleration will depend on these off-diagonal terms in the inertia matrix $M(q)$. Since M_{12} is an $\ell \times m$ matrix, we make the following definition:

Definition 13.5. *Let $\mathcal{U} \subset \mathbb{R}^n$ be an open subset of the configuration space $\mathbb{R}^n$. The system (13.59)–(13.60) is said to be* **strongly inertially coupled** *in $\mathcal{U}$ if and only if*

$$\text{rank } M_{12}(q) = \ell \text{ for all } q \in \mathcal{U} \tag{13.61}$$

Note that since M_{12} is an $\ell \times m$ matrix and there are m control inputs, the condition of strong inertial coupling requires that $m \geq \ell$, i.e. that the number of active degrees of freedom be at least as great as the number of passive degrees of freedom.

If M_{12} has full rank ℓ, it follows that the $\ell \times \ell$ matrix $M_{12}M_{12}^T$ has rank ℓ and is therefore invertible. Thus, under the assumption of strong inertial coupling, we let

$$M_{12}^\dagger = M_{12}^T(M_{12}M_{12}^T)^{-1} \tag{13.62}$$

be the right pseudoinverse of M_{12} as defined in Appendix B. We may therefore write $\ddot{q}_2$ in Equation (13.59) as

$$\ddot{q}_2 = -M_{12}^\dagger(M_{11}\ddot{q}_1 + c_1 + \phi_1)$$

and substitute this expression for $\ddot{q}_2$ into Equation (13.60) to obtain

$$\bar{M}_{21}\ddot{q}_1 + \bar{c}_2 + \bar{\phi}_2 = u$$

where

$$\begin{aligned} \bar{M}_{21} &= M_{21} - M_{22}M_{12}^{\dagger}M_{11} \\ \bar{c}_2 &= c_2 - M_{22}M_{12}^{\dagger}c_1 \\ \bar{\phi}_2 &= \phi_2 - M_{22}M_{12}^{\dagger}\phi_1 \end{aligned} \tag{13.63}$$

A calculation similar to that previously given for $\bar{M}_{22}$ shows that $\bar{M}_{21}$ has full rank ℓ since

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} I_{\ell\times\ell} \\ -M_{12}^{\dagger}M_{11} \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{M}_{21} \end{bmatrix} \tag{13.64}$$

Thus, with the control input

$$u = \bar{M}_{21}a_1 + \bar{c}_2 + \bar{\phi}_2 \tag{13.65}$$

we obtain

$$M_{12}\ddot{q}_2 + c_1 + \phi_1 = -M_{11}a_1 \tag{13.66}$$

$$\ddot{q}_1 = a_1 \tag{13.67}$$

The system (13.66)–(13.67) is also in second-order normal form and Equation (13.66) represents the input-driven internal dynamics.

Example 13.7. *The cart-pole system (13.54)–(13.55) satisfies the strong inertial coupling condition in the interval $-\frac{\pi}{2} < q_1 < \frac{\pi}{2}$. It can therefore be shown (Problem 13–11) that the noncollocated control law*

$$u = 2\tan(q_1) - \dot{q}_1^2\sin(q_1) - \frac{1+\sin^2(q_1)}{\cos(q_1)}a_1$$

results in the feedback equivalent system

$$\begin{aligned} \cos(q_1)\ddot{q}_2 - \sin(q_1) &= -a_1 \\ \ddot{q}_1 &= a_1 \end{aligned}$$

which is valid in the interval $-\frac{\pi}{2} < q_1 < \frac{\pi}{2}$.

Example 13.8. *Consider next the Reaction-Wheel Pendulum in the collocated second-order normal form*

$$\begin{aligned} J_1\ddot{q}_1 + J_2\ddot{q}_2 + mg\ell\cos(q_1) &= 0 \\ J_2\ddot{q}_2 &= u \end{aligned}$$

Since $J_2 \neq 0$ is constant, the strong inertial coupling condition is satisfied globally. It is easy to see that the control input

$$u = -J_1 a_1 - mg\ell \cos(q_1)$$

results in the noncollocated second-order normal form

$$\begin{aligned} J_2 \ddot{q}_2 + mg\ell \cos(q_1) &= -J_1 a_1 \\ \ddot{q}_1 &= a_1 \end{aligned}$$

13.6 Output Feedback Linearization

In this section we introduce the notions of **output feedback linearization**, **relative degree**, and **zero dynamics** for underactuated mechanical systems. The goal of output feedback linearization is to create a linear input/output relationship using feedback control and is related to both the partial feedback linearization considered in Section 13.5 and the state feedback linearization problem considered in Chapter 12. In fact, the partial feedback linearization in Section 13.5 is a special case of output feedback linearization as we shall see.

We also introduce the notion of **virtual holonomic constraints**. Virtual holonomic constraints (VHCs) are constraints that are maintained by feedback control, using the active degrees of freedom, rather than being imposed by the natural dynamics of the robot or the environment and are useful to generate coordinated motion among the active and passive joints. VHCs are particularly useful in locomotion, for example for control of walking robots, snake robots, brachiation robots, and gymnastic robots.

Consider again a lower-actuated system in second-order normal form and suppose that we have a p-dimensional output $y = h(q_1, q_2) : \mathbb{R}^n \to \mathbb{R}^p$ defined as a smooth function of the configuration $q = (q_1, q_2)$

$$\begin{aligned} M_{11}\ddot{q}_1 + c_1 + \phi_1 &= -M_{12} a_2 && (13.68) \\ \ddot{q}_2 &= a_2 && (13.69) \\ y &= h(q_1, q_2) && (13.70) \end{aligned}$$

Differentiating the output y yields

$$\dot{y} = \frac{\partial h}{\partial q_1}\dot{q}_1 + \frac{\partial h}{\partial q_2}\dot{q}_2 = J_1(q)\dot{q}_1 + J_2(q)\dot{q}_2$$

where $J_i := \frac{\partial h}{\partial q_i}$. Computing the second derivative $\ddot{y}$ of y, and substituting for $\ddot{q}_1$ and $\ddot{q}_2$ from (13.68) and (13.69) yields

$$\begin{aligned} \ddot{y} &= J_1\ddot{q}_1 + J_2\ddot{q}_2 + \dot{J}_1\dot{q}_1 + \dot{J}_2\dot{q}_2 \\ &= (J_2 - J_1 M_{11}^{-1} M_{12})a_2 + \eta(q,\dot{q}) \\ &= \bar{J}a_2 + \eta(q,\dot{q}) \end{aligned} \tag{13.71}$$

where $\bar{J} = J_2 - J_1 M_{11}^{-1} M_{12}$ is a $p \times m$ matrix, called the **decoupling matrix** and $\eta = \dot{J}_1\dot{q}_1 + \dot{J}_2\dot{q}_2 - J_1 M_{11}^{-1}(c_1 + \phi_1)$.

The system (13.68)–(13.70) is said to have **vector relative degree two** provided the decoupling matrix $\bar{J}$ has full rank. The relative degree can be interpreted as the number of times the output $y(t)$ must be differentiated before the input a_2 appears. Since $\bar{J}$ is a $p \times m$ matrix, the relative degree is well defined provided the rank of $\bar{J}$ is equal to p at each configuration q. Note that the relative degree can be well-defined globally or locally for q in a subset of the configuration space. Note also that for the relative degree to be well defined it is necessary that $m \geq p$, i.e., that the number of outputs does not exceed the number of active degrees of freedom.

Under the assumption that $\bar{J}$ has full rank, we can then define the control input a_2, using the right pseudo-inverse $\bar{J}^\dagger = \bar{J}^T(\bar{J}\bar{J}^T)^{-1}$ of $\bar{J}$, as

$$a_2 = \bar{J}^\dagger(\bar{a}_2 - \eta) \tag{13.72}$$

to obtain the linearized and decoupled output equation

$$\ddot{y} = \bar{a}_2 \tag{13.73}$$

and we note that an outer-loop control $\bar{a}_2$ can easily be designed to stabilize the equilibrium $y = 0$ or to track an arbitrary reference trajectory $y^d(t)$ in (13.73).

Definition 13.6. *With output $y = h(q_1, q_2)$, let $\Gamma = \{(q,\dot{q}) : h(q) = 0, \dot{h}(q) = \frac{\partial h}{\partial q}(q)\dot{q} = 0\}$. Γ is called the* **zero-dynamics manifold**. *An outer-loop control a_2 that asymptotically stabilizes the equilibrium $y = 0$ in (13.73) makes Γ an invariant manifold for the system (13.68)–(13.70). In this case, Γ is called a* **controlled-invariant manifold**. *The reduced-order dynamics on Γ are called the* **zero dynamics**.

Remark 13.3. *Returning to the general lower-actuated system*

$$\begin{aligned} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 &= 0 \\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 &= u \end{aligned}$$

it is straightforward to show (Problem 13–12) that if we take as output $y = q_2$, then the control input given by Equation (13.50) achieves both output linearization and places the system in the collocated second-order normal form.

Likewise, the noncollocated second-order normal form is achieved via output feedback linearization with the choice of output $y = q_1$ (Problem 13–13).

13.6.1 Computation of the Zero Dynamics

For a general output function $h(q_1, q_2)$, it is not easy to characterize the reduced-order dynamics on the zero-dynamics manifold Γ. In the special cases $y = q_i$, $i = 1$ or 2, the zero dynamics can be easily characterized and has a nice physical interpretation as we show in this section.

Let's first take as output $y = q_2$. Therefore, $p = m$ and the decoupling matrix is just $I_{m\times m}$, the $m \times m$ identity matrix. With this output, we have

$$M_{11}\ddot{q}_1 + c_1 + \phi_1 = -M_{12}a_2 \tag{13.74}$$

$$\ddot{q}_2 = a_2 \tag{13.75}$$

$$y = q_2 \tag{13.76}$$

The zero dynamics are found by setting the output y identically zero, which implies that $q_2 = 0$, $\dot{q}_2 = 0$, and $a_2 = 0$ in Equation (13.74). Setting

$$\tilde{M}_{11} = M_{11}(q_1),\ \tilde{c}_1 = c_1(q_1, \dot{q}_1, 0, 0),\ \tilde{\phi}_1 = \phi_1(q_1) \tag{13.77}$$

the zero dynamics are given by the system

$$\tilde{M}_{11}\ddot{q}_1 + \ \tilde{h}_1 + \tilde{\phi}_1 = 0 \tag{13.78}$$

The reduced-order model (13.78) represents the dynamics of a robot with ℓ passive joints where the m active joints are fixed, at $q_2 = 0$, and is therefore a (reduced-order) Lagrangian mechanical system.

Let E be the total energy of the reduced-order system (13.78).

$$E = \frac{1}{2}\dot{q}_1^T \tilde{M}_{11}(q_1)\dot{q}_1 + \tilde{P}(q_1) \tag{13.79}$$

Then, from the standard properties of Lagrangian dynamics, we know that $\dot{E} = 0$ along trajectories of the system (13.78). The implication is that trajectories of the zero dynamics are constant energy levels of the reduced-order system.

Example 13.9. *Consider the Acrobot model in second-order normal form*

$$\begin{aligned} m_{11}\ddot{q}_1 + c_1 + g_1 &= -m_{12}a_2 && (13.80)\\ \ddot{q}_2 &= a_2 && (13.81)\\ y &= q_2 && (13.82) \end{aligned}$$

The zero dynamics are found by setting $q_2 = 0$, $\dot{q}_2 = 0$, and $a_2 = 0$ in (13.80). From Equation (5.83) we have

$$\begin{aligned} \tilde{m}_{11} &= m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2 + 2\ell_1\ell_{c2}) + I_1 + I_2\\ \tilde{h}_1 &= 0\\ \tilde{\phi}_1 &= (m_1\ell_{c1} + m_2\ell_1 + m_2\ell_{c2})g\cos(q_1) \end{aligned}$$

Substituting these expressions into (13.80) we end up with

$$\tilde{m}_{11}\ddot{q}_1 + n_1 g\cos(q_1) = 0$$

where

$$n_1 = m_1\ell_{c1} + m_2\ell_1 + m_2\ell_{c2}$$

Note that these zero dynamics are just the dynamics of a simple pendulum.

Since almost all trajectories on the above zero dynamics manifold are periodic orbits the equilibrium solutions are not asymptotically stable. Such systems are called **nonminimum phase systems**.

In the case of noncollocated partial feedback linearization, consider again the normal form equations

$$\begin{aligned} M_{12}\ddot{q}_2 + c_1 + \phi_1 &= -M_{11}a_1 && (13.83)\\ \ddot{q}_1 &= a_1 && (13.84)\\ y &= q_1 && (13.85) \end{aligned}$$

with output $y = q_1$. Note that the strong inertial coupling condition that is necessary for the existence of the above normal form ensures that the number of outputs is less than the number of active joints. In this case, the zero dynamics are found by setting $q_1 = 0$, $\dot{q}_1 = 0$, and $a_1 = 0$ in the above system, which leads to

$$\tilde{M}_{12}\ddot{q}_2 + \tilde{h}_1 + \tilde{\phi}_1 = 0 \qquad (13.86)$$

with $\tilde{M}_{12}(0, q_2)$, $\tilde{h}_1(0, 0, q_2, \dot{q}_2)$, $\tilde{\phi}_1(0, q_2)$. Equation (13.86) need not, in general, represent a Lagrangian system since the $\ell \times m$ matrix M_{12} is not guaranteed to be symmetric or positive definite, even in the case $\ell = m$.

Feedback Linearization of the Reaction-Wheel Pendulum

As we noted in the introduction to this chapter, underactuated robots are generally not fully feedback linearizable. The best one can achieve in most cases is partial feedback linearization, either collocated or noncollocated. It is interesting, therefore, that the Reaction-Wheel Pendulum is an example of a robot that is fully feedback linearizable. In order to see this, let's return to the model for the Reaction-Wheel Pendulum

$$\begin{aligned} J_1\ddot{q}_1 + mg\ell\cos(q_1) &= -u \\ J_2\ddot{q}_2 &= u \end{aligned}$$

and choose the output equation

$$y_1 = J_1 q_1 + J_2 q_2 \tag{13.87}$$

Then computing successive derivatives of y_1 we get

$$y_2 = \dot{y}_1 = J_1\dot{q}_1 + J_2\dot{q}_2 \tag{13.88}$$

$$y_3 = \dot{y}_2 = J_1\ddot{q}_1 + J_2\ddot{q}_2 = -\frac{mg\ell}{J_1}\cos(q_1) \tag{13.89}$$

$$y_4 = \dot{y}_3 = \frac{mg\ell}{J_1}\sin(q_1)\dot{q}_1 \tag{13.90}$$

Then $\dot{y}_4$ satisfies

$$\begin{aligned} \dot{y}_4 &= \frac{mg\ell}{J_1}\sin(q_1)\ddot{q}_1 + \frac{mg\ell}{J_1}\cos(q_1)\dot{q}_1^2 \\ &= -\frac{mg\ell}{J_1}\sin(q_1)u - \frac{mg\ell^2}{J_1}\sin(q_1)\cos(q_1) + mg\ell\cos(q_1)\dot{q}_1^2 \end{aligned}$$

Therefore, the control input

$$u = -\frac{J_1}{mg\ell\sin(q_1)}(a + \frac{mg\ell^2}{J_1}\sin(q_1)\cos(q_1) - mg\ell\cos(q_1)\dot{q}_1^2) \tag{13.91}$$

results in the linear system in Brunovsky canonical form

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= y_3 \\ \dot{y}_3 &= y_4 \\ \dot{y}_4 &= a \end{aligned}$$

with output $y = y_1$. We note therefore that the output (13.87) has relative degree four in the region $0 < q_1 < \pi$ and the control input (13.91) is valid in this same region. Thus the Reaction-Wheel Pendulum is locally output feedback linearizable. As a result the inverted equilibrium can be stabilized with the above feedback linearizable control law provided that the initial orientation of the pendulum is above the horizontal position where $0 < q_1 < \pi$. However, one must be careful that the transient response does not violate this constraint, which may happen if there is a large initial velocity or if the control results in undershoot. Problem 13–14 deals with the design of the outer-loop control term a in (13.91).

13.6.2 Virtual Holonomic Constraints

We next introduce the notion of **virtual holonomic constraints** for underactuated robots and discuss the relation to output feedback linearization.

Definition 13.7. *Let $h(q_1, q_2) : \mathbb{R}^n \to \mathbb{R}^p$ be a smooth function of the configuration variables with rank$(dh_q) = p$ for all $q \in h^{-1}(0)$. The function h is said to define a* **virtual holonomic constraint** *for a given underactuated system if there exists a feedback control such that*

$$\Gamma = \{(q, \dot{q}) : h(q) = 0, \dot{h}(q) = \frac{\partial h}{\partial q}(q)\dot{q} = 0\}$$

is a controlled-invariant manifold for the system.

The term **virtual constraint** arises from the fact that, if the system is initialized on Γ, i.e, $h(q(0)) = 0$, then the solution trajectory $q(t)$ remains on Γ for all $t > 0$. We can immediately see that a useful way to enforce a given set of virtual holonomic constraints is to define an output function $y = h(q_1, q_2)$ and design the control u to achieve output feedback linearization. This will work provided the constraint function h yields an output function with vector relative degree two.

Example 13.10. *Suppose that we wish to constrain the motion of the Acrobot such that $q_1(t) + 0.5q_2(t) = 0$. This motion simulates a so-called* **continuous contact brachiation**. *Figure 13.14 shows the response with the output $y = h(q_1, q_2) = q_1 + 0.5q_2$ as a virtual holonomic constraint and an output-linearizing control.*

13.7 Passivity-Based Control

In this section we discuss the use of **Energy** and **Passivity** methods for control of underactuated robots. We recall from Chapter 6 that the total

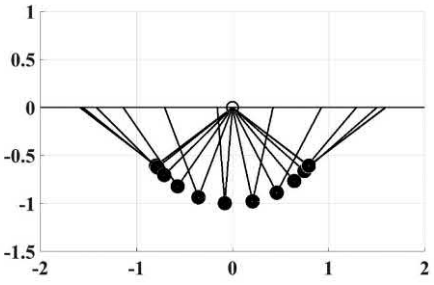


Figure 13.14: Brachiation motion of the Acrobot with virtual holonomic constraint $q_1 + 0.5q_2 = 0$.

energy E for a Lagrangian mechanical system

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \phi(q) = Bu \tag{13.92}$$

satisfies

$$\dot{E} = \dot{q}^T Bu \tag{13.93}$$

which means that the system defines a passive map from input Bu to output $\dot{q}$. Thus energy and passivity are intimately related for the class of mechanical systems that we consider. In Chapter 6 we used the passivity property to derive robust and adaptive control laws for fully-actuated n-link manipulators. We will show in this section that passivity, combined with **switching control** and **saturation**, provides elegant solutions for control of underactuated robots. Specifically, we will focus on the problem of **swingup** and **balance**; for example, in the case of the Acrobot, swingup and balance mimics the motion of a gymnast performing a handstand on a high bar.

As we shall see, the problem of swingup and balance is readily accomplished using energy/passivity methods combined with switching control. The balance control problem is essentially the problem of stabilizing the equilibrium at the inverted configuration and is solvable locally with linear feedback control as we have previously illustrated. The swingup control problem then becomes one of controlling the state of the system so that the trajectory enters the region of attraction of the balance controller, at which point control can be switched to the balance control.

13.7.1 The Simple Pendulum

To motivate the subsequent treatment, consider a simple pendulum of length ℓ and mass m as shown in Figure 13.15. Assume that a force F acts on the

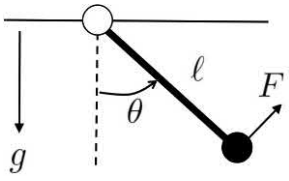


Figure 13.15: A simple pendulum with a force F acting at the bob.

end of the pendulum as shown. We can think of this pendulum as a simplified model of a passive link in a larger system where the force F arises from the motion of active links, for example, by actively swinging the second link in the case of the Acrobot. Note that the force F induces a torque $\tau = \ell F$ at the pendulum pivot. The equation of motion of this system is therefore given by

$$m\ell^2\ddot{\theta} + mg\ell\sin(\theta) \quad = \quad \tau \tag{13.94}$$

and the total energy is

$$E \quad = \quad \frac{1}{2}m\ell^2\dot{\theta}^2 + mg\ell(1 - \cos(\theta)) \tag{13.95}$$

With τ equal to zero, the pendulum energy is constant along solution trajectories of (13.94). Stated another way, the set

$$\Sigma_c = \{(\theta, \dot{\theta}) \mid E(\theta, \dot{\theta}) = c\}$$

defines a trajectory of the system in the sense that, if $(\theta(0), \dot{\theta}(0)) \in \Sigma_c$, the solution of (13.94) with $F = 0$ satisfies $(\theta(t), \dot{\theta}(t)) \in \Sigma_c$ for $t > 0$. The set Σ_c is an invariant manifold called a **first integral of motion** for the simple pendulum.

Figure (13.16) shows a portion of the **phase portrait** of the unforced pendulum where each trajectory corresponds to a particular energy level. Each trajectory of the simple pendulum is periodic with the exception of the equilibrium configurations $(0, 0)$ and $(\pm\pi, 0)$, and the so-called **homoclinic orbit**. We have seen previously that the simple pendulum is relevant for more complicated underactuated systems, for example, appearing as the zero dynamics manifold in Example 13.9.

Definition 13.8. *A* **homoclinic orbit** *of a dynamical system is a trajectory that connects a saddle-point equilibrium to itself. A homoclinic orbit lies in the intersection of the stable and unstable manifolds of the saddle point.*

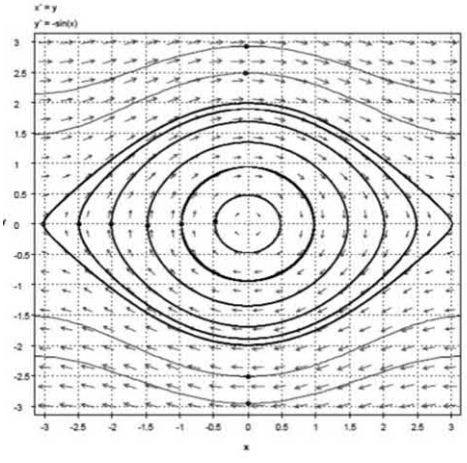


Figure 13.16: Phase portrait of the simple pendulum. The constant energy curves are solution trajectories. Figure generated by pplane, courtesy of John C. Polking, Rice University.

With regard to the phase portrait of the simple pendulum in Figure 13.16, the trajectory that connects the saddle-point equilibria $(-\pi, 0)$ and $(+\pi, 0)$ is a homoclinic orbit if we identify $(-\pi, 0)$ and $(+\pi, 0)$.

Let us now consider the problem of using the force F as a feedback control law to control the energy of the pendulum. In other words, given a constant $c > 0$, we wish to design the input $\tau = \ell F$ so that the energy $E(t)$ converges to c. In doing so, the motion of the pendulum will converge to the particular periodic solution defined by $E = c$. With this in mind let V be a Lyapunov function candidate defined as

$$V = \frac{1}{2}(E - E_r)^2 \tag{13.96}$$

where $E_r = c$ is chosen as a reference energy. Then $\dot{V}$ is given by

$$\dot{V} = (E - E_r)\dot{\theta}\tau \tag{13.97}$$

Note that the above expression means that the system is passive from input τ to output $y = (E - E_r)\dot{\theta}$. If we take the input τ as

$$\tau = -ky = -k(E - E_r)\dot{\theta}, \ k > 0 \tag{13.98}$$

we end up with

$$\dot{V} = -ky^2 \leq 0 \tag{13.99}$$

An elementary application of LaSalle's theorem (Problem 13–15) shows that all trajectories converge either to a trajectory with energy $E_r = c$ or to $\dot{\theta} = 0$.

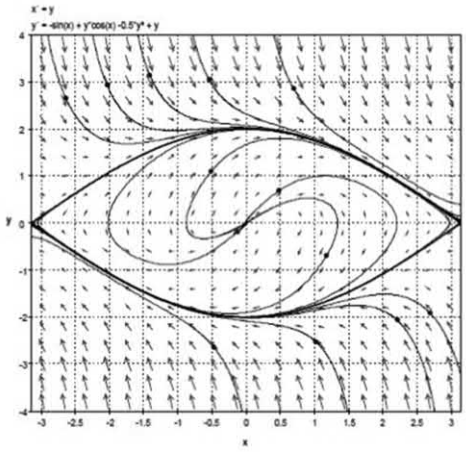


Figure 13.17: Phase portrait of the closed-loop system. Figure generated by pplane, courtesy of John C. Polking, Rice University.

Remark 13.4. *The condition,* $\dot{\theta} = 0$*, cannot be ruled out since the open-loop equilibrium solutions* $(0, 0)$ *and* $(\pm\pi, 0)$*, remain equilibrium points for the closed-loop system since the control input is zero if* $\dot{\theta} = 0$*. However, the equilibrium* $(0, 0)$ *is now unstable for the closed-loop system (Problem 13–16).*

With E_c as the energy on the homoclinic orbit, Figure 13.17 shows the phase portrait of the closed loop system.

Saturation

An important property of the passivity-based control approach is that bounds on the available control effort (i.e., saturation) are easily handled. To see this, suppose that the above input τ constrained as $|\tau| \leq m$. We can choose the control input as

$$\tau = -sat_m(ky) = -sat_m(k(E - E_r)\dot{\theta}),\ k > 0 \tag{13.100}$$

where $\text{sat}_m(\cdot)$ is the saturation function

$$\text{sat}_m(\sigma) = \begin{cases} m & \text{if} \quad \sigma > m \\ \sigma & \text{if} \quad -m \leq \sigma \leq m \\ -m & \text{if} \quad \sigma < -m \end{cases}$$

The saturation function is a so-called **first and third quadrant** nonlinearity which means that

$$\sigma \text{sat}_m(\sigma) \geq 0 \quad \text{for all} \quad \sigma$$

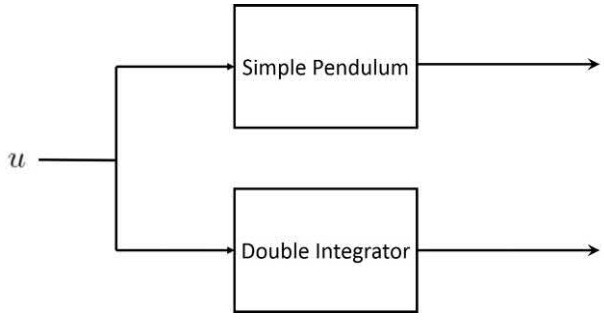


Figure 13.18: The Reaction-Wheel Pendulum as a parallel interconnection of passive systems.

Therefore, using the control (13.100) in place of (13.98) we have

$$\dot{V} = -ysat_m(ky) \leq 0$$

We leave it as an exercise (Problem 13–17) to show that the conclusions from the application of LaSalle's theorem remain the same with the above saturation control.

13.7.2 The Reaction-Wheel Pendulum

In this section we consider the swingup control problem for the Reaction-Wheel Pendulum using the tools derived above. Consider again the Reaction-Wheel Pendulum dynamics in the form

$$J_1\ddot{q}_1 + mg\ell\cos(q_1) = -u \quad (13.101)$$

$$J_2\ddot{q}_2 = u \quad (13.102)$$

Here we see that the Reaction-Wheel Pendulum dynamics are described by a parallel connection of a simple pendulum and a double integrator both of which satisfy a passivity property from input torque to output velocity. Specifically, with

$$E_1 = \frac{1}{2}J_1\dot{q}_1^2 + mg\ell\sin(q_1) \quad (13.103)$$

the usual energy of the pendulum and

$$E_2 = \frac{1}{2}J_2\dot{q}_2^2 \quad (13.104)$$

the energy of the reaction wheel, we have

$$\dot{E}_1 = -\dot{q}_1 u \tag{13.105}$$
$$\dot{E}_2 = \dot{q}_2 u \tag{13.106}$$

Since the parallel interconnection of passive systems is passive (Problem 13–18), it remains to define a suitable output y for the parallel interconnection. Following the previous example of the simple pendulum, we can define a Lyapunov function V as

$$V = \frac{1}{2}(E_1 - E_r)^2 + E_2 \tag{13.107}$$

where E_r is a constant reference energy for the pendulum. A straightforward calculation then gives

$$\dot{V} = -(E_1 - E_r)\dot{q}_1 u + \dot{q}_2 u = (\dot{q}_2 - (E_1 - E_r)\dot{q}_1)u$$

Thus if we define $y = \dot{q}_2 - (E_1 - E_r)\dot{q}_1$ as a new output we get

$$\dot{V} = yu$$

and therefore the system is passive from input u to output y. We can then choose the control input u as

$$u = -\text{sat}_m(ky) = -\text{sat}_m(k(\dot{q}_2 - (E_1 - E_r)\dot{q}_1)), \quad k > 0 \tag{13.108}$$

Proposition 13.3. *Let $E_r > 0$ be a constant reference value for the Reaction-Wheel Pendulum energy E_1. Choose the control input u in (13.101)–(13.102) according to (13.108). Then all trajectories of the closed-loop system converge to the set*

$$\mathcal{M} = \{(q, \dot{q}) \mid (E_1 - E_r)\cos q_1 = 0\} \tag{13.109}$$

Proof: The proof is a straightforward calculation using LaSalle's theorem and is left as an exercise (Problem 13–19).

Therefore, all trajectories of the closed loop system will converge either to $E_1 = E_r$ or to $\cos(q_1) = 0$. In the first case, that $E_1 = E_r$, it follows from (13.108) that the velocity of the reaction wheel $\dot{q}_2 = 0$. In the second case, that $\cos(q_1) = 0$, it follows that $q_1 = n\pi$ and $\dot{q}_2 = 0$.

Figures 13.19 and 13.20 show the simulation with E_r equal to the energy of the homoclinic orbit of the pendulum with a switch to a linear balancing controller at $t = 8$ seconds.

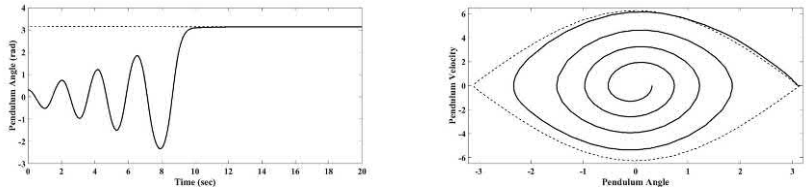


Figure 13.19: Swingup and balance of the Reaction-Wheel Pendulum (left) and phase plane trajectory of the pendulum (right).

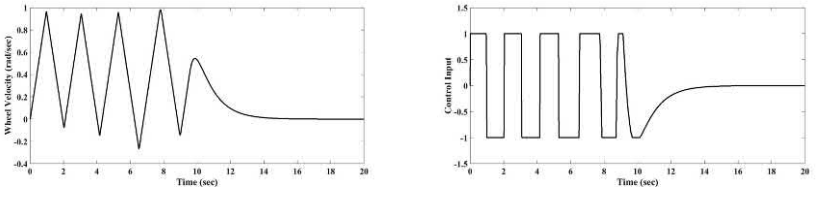


Figure 13.20: Reaction-wheel velocity (left) and saturated control input (right).

13.7.3 Swingup and Balance of The Acrobot

We next consider the swingup and balance for the Acrobot, beginning in the collocated second-order normal form in Example (13.9).

$$m_{11}\ddot{q}_1 + c_1 + \phi_1 = -a_2 \tag{13.110}$$

$$\ddot{q}_1 = a_2 \tag{13.111}$$

It is important that the internal dynamics in the second-order normal form is driven by the outer-loop control term a_2 so that we can use this term both to stabilize the linearized subsystem of the system and modify the internal dynamics.

The inverted equilibrium for the Acrobot model is $q_1 = +\pi/2$, $\dot{q}_1 = 0$, $q_2 = 0$, $\dot{q}_2 = 0$. Suppose that we define the outer-loop control a_2

$$a_2 = -k_p q_2 - k_d \dot{q}_2 + k_3 \text{sat}_m((E - E_c)\dot{q}_1) \tag{13.112}$$

where E is the total energy of the Acrobot and E_c is the energy at the inverted equilibrium. A successful swingup and balance motion with this strategy is shown in Figure 13.21, where control is switched to an LQR balance control at approximately 7.5 seconds.

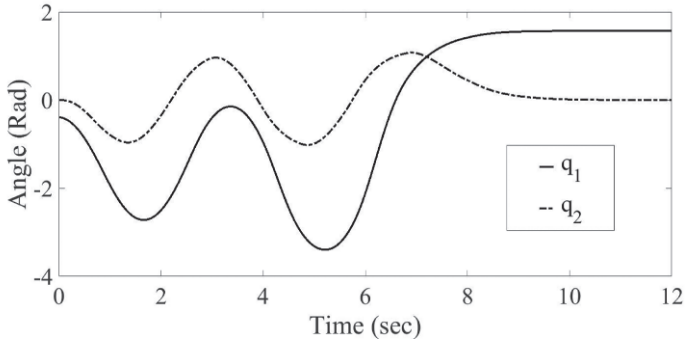


Figure 13.21: Swingup and balance of the Acrobot using switching control

13.8 Chapter Summary

In this chapter we discussed the control of underactuated mechanical systems with a focus on underactuated serial-link mechanisms. Many of the control techniques for fully-actuated systems that we discussed in previous chapters do not apply without modification to the class of underactuated systems. One of the primary obstructions to control of this class of systems is the presence of non-minimum phase zero dynamics.

Upper and Lower-Actuated Models

The class of robots that we treated in this chapter is characterized as n-degree-of-freedom Lagrangian dynamical systems with $m < n$ control inputs, and thus, m actuated degrees of freedom and $\ell = n - m$ unactuated degrees of freedom. The difference $\ell = n - m$ is the **degree of underactuation of the system**. We showed that any such system may be represented either as an **upper-actuated system**

$$\begin{aligned} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 &= u \\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 &= 0 \end{aligned}$$

with $q_1 \in \mathbb{R}^m$ and $q_2 \in \mathbb{R}^\ell$, or as a **lower-actuated system**

$$\begin{aligned} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 &= 0 \\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 &= u \end{aligned}$$

with $q_1 \in \mathbb{R}^\ell$ and $q_2 \in \mathbb{R}^m$.

Linear Controllability

Classifying underactuated systems by whether or not they are linearly controllable is useful for determining the global controllability properties. We showed that a necessary condition for linear controllability is that each passive degree of freedom must have a nonzero potential force such as elasticity or gravitational force. The property of linear controllability allows one to apply switching control methods that combine nonlinear control laws far from the equilibrium and linear control laws close to the equilibrium. We illustrated this idea for problems of swingup and balance of the Acrobot and the Reaction-Wheel Pendulum.

Collocated and Noncollated Partial Feedback Linearization

We introduced the notions of collocated and noncollocated partial feedback linearization, which transforms a given underactuated system into a **second-order normal form** that is important for subsequent analysis and controller design. The second-order normal form in the collocated case is

$$\begin{aligned} M_{11}\ddot{q}_1 + c_1 + g_1 &= -M_{12}a_2 \\ \ddot{q}_2 &= a_2 \end{aligned}$$

The second-order normal form in the noncollocated case is

$$\begin{aligned} M_{12}\ddot{q}_2 + c_1 + \phi_1 &= -M_{11}a_1 \\ \ddot{q}_1 &= a_1 \end{aligned}$$

where a_2, respectively a_1, are additional (outer-loop) controls.

Output Feedback Linearization and Virtual Holonomic Constraints

A **virtual holonomic constraint** is a relation of the form $h(q) = 0$, where $h : \mathcal{Q} \to \mathbb{R}^p$ is a smooth function from the configuration space $\mathcal{Q}$ to $\mathbb{R}^p$ that is enforced by the feedback control. Virtual constraints may be chosen as an output function to achieve a desired task, such as coordinated motion among the degrees of freedom. Enforcing the virtual constraints leads to the notion of **zero dynamics**, which are the dynamics of the system restricted to a reduced-order manifold in the state space of the full system.

Switching Control and Passivity

Starting with the second-order normal form we showed how stabilization to fixed points can be accomplished by energy-based methods and switching

control. Energy shaping methods have the advantage of not relying on the need to plan time-based trajectories for tracking.

Problems

13–1 Complete the derivation of the Euler–Lagrange equations (13.12)–(13.13) for the cart-pole system from the expressions for the kinetic and potential energy given in (13.10)–(13.11).

13–2 Show that the dynamic equations of the Acrobot reduce to those of the Reaction Wheel Pendulum if $\ell_{c2} = 0$.

13–3 Verify the claims of Remark 13.1 by deriving Equations (13.22)–(13.23) and (13.24)–(13.25).

13–4 Show that, with $u_e = 0$, the only equilibrium points of the Acrobot and Pendubot are shown in Figure 13.10.

13–5 Verify the expressions for the matrices F and G in Equations (13.35)–(13.36).

13–6 Show by direct calculation that the linearized equations of motion of the Reaction-Wheel Pendulum about the origin are controllable if and only if the constant term $\frac{mg\ell}{J_1}$ is nonzero.

13–7 Verify the expressions for the terms $\bar{M}_{22}$, $\bar{h}_2$ and $\bar{g}_2$ in Equations (13.47).

13–8 Complete the proof of Proposition 13.2.

13–9 Verify the expression for the control law (13.65).

13–10 Complete the calculations to verify the collocated control law for the cart-pole system in Example 13.4.

13–11 Complete the calculations to verify the noncollocated control law for the cart-pole system in Example 13.4.

13–12 Consider the lower-actuated system

$$\begin{aligned} M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 + c_1 + \phi_1 &= 0 \\ M_{21}\ddot{q}_1 + M_{22}\ddot{q}_2 + c_2 + \phi_2 &= u \end{aligned}$$

Show that if we take as an output $y = q_2$ for this system, where q_2^r is a constant reference vector for q_2, then the control input u that

achieves the linear system (13.73) is identical to the control u given by Equation (13.50) and therefore achieves both output linearization and places the system in normal form.

13–13 Show for the system of Problem 12 that the noncollocated second-order normal form is achieved via output feedback linearization with the choice of output $y = q_1$.

13–14 Design a linear outer-loop control a in equation (13.91) to stabilize the Reaction-Wheel Pendulum at the inverted position. Investigate the response for various initial conditions.

13–15 Using the expression for $\dot{V}$ in Equation (13.99), show that all trajectories of the simple pendulum converge to those with energy E_r or to $\omega = 0$.

13–16 Consider the simple pendulum with control law (13.98). Show that the equilibrium $(0, 0)$ is unstable for $E_r = 2$ and stable (but not asymptotically stable) for $0 \leq E_r < 2$.

13–17 Use LaSalle's theorem to show that applying the control (13.100) in place of (13.98) for swingup of the simple pendulum yields the same asymptotic behavior.

13–18 Show that the parallel interconnection of passive systems is passive.

13–19 Complete the proof of Proposition 13.3 using LaSalle's theorem.

13–20 Using the cart-pole system, derive and simulate a swingup and balance control using any of the methods in this chapter.

Notes and References

Research in the control of underactuated mechanical systems is an active area and there is a large body of literature devoted to the subject. Research monographs specifically for control of underactuated systems are [186, 41, 179]. Proposition 13.1 is taken from [98]. Most of the material on collocated and noncollocated partial feedback linearization presented here is taken from [162]. Related work followed in [157, 159, 163]. The second-order normal forms for both the collocated and noncollocated case is attributed to [162]. Several treatments of second-order nonholonomic constraints are

found in [130, 183, 123, 151]. The classical inverted pendulum has been studied extensively in the control literature [127, 62, 117, 184]. The Acrobot first appeared in [122], where it was shown that the Acrobot dynamics are not feedback linearizable. The swingup problem for the Acrobot was first solved in [158]. The Pendubot [160] and the Reaction-Wheel Pendulum [164] both came out of the University of Illinois College of Engineering Control Systems Laboratory. A rather complete monograph devoted entirely to the Reaction-Wheel Pendulum is [14], which includes the passivity-based control approach presented here. The concept of virtual constraints and it's application to bipedal locomotion is due to [59].

Chapter 14

MOBILE ROBOTS

In this chapter we discuss the control of nonholonomic systems with a focus on the control of **mobile robots**. Mobile robots are becoming increasingly important in many applications, such as factory and warehouse automation, service and cleaning, household and medical assistance, military applications, agricultural and forestry applications, and search and rescue. Autonomous highway vehicles (cars and trucks) can also be considered mobile robots.

A distinguishing feature of mobile robots that impacts the control problem is the presence of **nonholonomic constraints**. Nonholonomic constraints arise in two primary ways.

1. In **rolling without slipping** constraints. For example, the translation and rotation of a wheel are not independent if the wheel rolls without slipping. Examples include:
    - A wheeled mobile robot, unicycle, automobile, or tractor/trailer
    - Manipulation of rigid objects, for example, fingers in rolling contact with an object
2. In systems where **angular momentum** is conserved. Examples include:
    - Satellites and space robots
    - Gymnastic, diving, and running robots

We will mainly focus in this chapter on the control of wheeled mobile robots, which are part of a particular class of systems known as **driftless systems**, which we define later. These systems are examples of underactuated systems that are not linearly controllable according to Definition 13.3 in Chapter 13.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

We first discuss the notion of **nonholonomic constraints** and **nonholonomic systems**, which we introduced briefly in Chapter 6. Understanding the difference between holonomic and nonholomic constraints is fundamental to the understanding of every concept introduced in this chapter. We then discuss the problems of controllability and stabilizability for mobile robots and outline both open-loop and closed-loop control methods for the problems of set-point regulation and trajectory tracking. The main theoretical control results we will discuss are **Chow's theorem** and **Brockett's theorem**. Chow's theorem gives necessary and sufficient conditions for a driftless system to be controllable. Brockett's theorem gives a necessary condition for stabilization of such systems by smooth feedback control.

14.1 Nonholonomic Constraints

We introduced **holonomic constraints** in Chapter 6 when we derived the Euler–Lagrange equations of motion for rigid manipulators. We saw that interconnecting n rigid bodies, each of which independently has 6 degrees of freedom, with revolute or prismatic joints, introduces **holonomic constraints** that reduce the number of degrees of freedom from $6n$ to just n. Our treatment of force control in Chapter 10 dealt with **unilateral constraints**, which are imposed by contact between the robot and the environment. In this section we expand upon the notion of systems subject to constraints and discuss **nonholonomic constraints**.

Let $(q_1, \ldots, q_n)$ denote a coordinate representation of the configuration q in the configuration space $\mathcal{Q}$ of a given system. We recall the following definition from Section (6.1.2).

Definition 14.1 (Holonomic constraint)**.** *An equality constraint of the form*

$$h(q_1, \ldots, q_n) = 0 \tag{14.1}$$

where h is a mapping from $\mathcal{Q} \mapsto \mathbb{R}$ is called a **holonomic constraint**. *We assume that the function h is continuously differentiable and that the differential $dh = \frac{\partial h}{\partial q} = (\frac{\partial h}{\partial q_1}, \ldots, \frac{\partial h}{\partial q_n})$ is nonzero at each $q \in \mathcal{Q}$.*

As a result of this holonomic constraint, the configuration of the system, $q = (q_1, \ldots, q_n)$, is restricted to an $n-1$ dimensional manifold in the configuration space defined by Equation (14.1).

Example 14.1. *A simple example of a system subject to a holonomic constraint is a mass m attached to the end of a rigid rod of length ℓ that is*

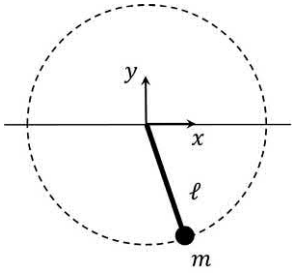


Figure 14.1: Mass m connected to a rigid rod.

pinned to the origin and free to rotate in the plane. The (x, y) coordinates of the mass therefore satisfy the holonomic constraint

$$x^2 + y^2 - \ell^2 = 0 \tag{14.2}$$

which constrains the motion of the mass to the circle of radius ℓ in $\mathbb{R}^2$.

We often have constraints that are expressed as functions of both the configurations and velocities as $a(q, \dot{q}) = 0$, where the function $a(q, \dot{q})$ is continuously differentiable in both arguments. We will limit our discussion to so-called **Pfaffian constraints**, as defined next

Definition 14.2 (Pfaffian constraint)**.** *Constraints of the form*

$$w_1(q)\dot{q}_1 + \cdots + w_n(q)\dot{q}_n = w(q)\dot{q} = 0 \tag{14.3}$$

where the row vector $w(q) = (w_1(q), \ldots, w_n(q))$ is a **covector** *in the cotangent space $T_q^*(\mathcal{Q})$, are called* **Pfaffian constraints**.

Note that the constraint given by Equation (14.3) constrains the velocity vector $\dot{q}$ at each q but does not necessarily constrain the configuration q to a lower dimensional manifold. Stated another way, at each configuration q, Equation (14.3) specifies the allowable velocities.

If the configuration $q(t)$ is constrained by the holonomic constraint in Equation (14.1) for all time $t \geq 0$ it follows that

$$\dot{h} = \frac{\partial h}{\partial q}\dot{q} = dh\dot{q} = 0 \tag{14.4}$$

Thus we can say that a Pfaffian constraint $w(q)\dot{q} = 0$ is holonomic if there exists a function $h : \mathbb{R}^n \to \mathbb{R}$ such that

$$\sum_{j=i}^{n} w_j(q)\dot{q}_j = \sum_{j=i}^{n} \frac{\partial h}{\partial q_j}\dot{q}_j = 0$$

This, in turn, implies the existence of a function $\gamma(q)$, called an **integrating factor** such that

$$\gamma(q)w_j(q) = \frac{\partial h}{\partial q_j}(q), \quad j = 1, \ldots, n \tag{14.5}$$

In this case, the constraint $w(q)\dot{q} = 0$ is referred to as **integrable**. Note that if the constraint is integrable, it follows, since h is continuously differentiable, that

$$\frac{\partial(\gamma w_i)}{\partial q_j} = \frac{\partial^2 h}{\partial q_j \partial q_i} = \frac{\partial^2 h}{\partial q_i \partial q_j} = \frac{\partial(\gamma w_j)}{\partial q_i}$$

This leads to the following necessary and sufficient conditions for a Pfaffian constraint to be integrable.

Proposition 14.1. *Equation (14.3) defines an integrable (holonomic) constraint, if and only if there exists a nonzero function $\gamma(q)$ such that the components w_i of w satisfy*

$$\frac{\partial(\gamma w_i)}{\partial q_j} = \frac{\partial(\gamma w_j)}{\partial q_i}, \text{ for all } i, j \tag{14.6}$$

Example 14.2. *Consider the Pfaffian constraint on $\mathbb{R}^3$*

$$\dot{q}_2 - q_3\dot{q}_1 = 0 \tag{14.7}$$

which is of the form $w(q)\dot{q}$ with $w = (-q_3, 1, 0)$. A simple calculation, using Equation (14.6), shows that

$$\begin{aligned} \frac{\partial \gamma}{\partial q_1} + q_3 \frac{\partial \gamma}{\partial q_2} &= 0 \\ q_3 \frac{\partial \gamma}{\partial q_3} + \gamma &= 0 \\ \frac{\partial \gamma}{\partial q_3} &= 0 \end{aligned}$$

It is easy to see from the above that the only possible solution is $\gamma = 0$ (Problem 14–1). Therefore the constraint is nonholonomic.

Multiple Constraints

Suppose now that we have $k \geq 2$ Pfaffian constraints

$$w_i(q)\dot{q} = 0 \ , \ i = 1, \ldots, k \leq n \tag{14.8}$$

where each covector $w_i(q)$ is given by $w_i(q) = (w_{i1}(q), \ldots, w_{in}(q))$.

Example 14.3. *We note that checking the holonomy of each constraint separately does not give us sufficient information about the constraints on the system configuration. Consider the two Pfaffian constraints*

$$w_1(q)\dot{q} = \dot{q}_2 - q_3\dot{q}_1 = 0 \tag{14.9}$$
$$w_2(q)\dot{q} = \dot{q}_3 - q_1\dot{q}_2 = 0 \tag{14.10}$$

One can check that individually these constraints are nonholonomic. In fact, we showed the nonholonomy of the constraint given by Equation (14.9) in Example 14.2 above. Note, however, that the linear combination $q_1w_1 + w_2$ satisfies

$$(q_1w_1 + w_2)\dot{q} = \dot{q}_3 - q_1q_3\dot{q}_1 = 0 \tag{14.11}$$

which can be integrated to $q_3 = ke^{q_1^2/2}$ and $q_2 = \int ke^{q_1^2/2}dq_1$ (Problem 14–2). Therefore, the constraints are holonomic. In fact, it will follow from Theorem 14.1 below that any set of $n-1$ Pfaffian constraints in $\mathbb{R}^n$ is holonomic.

We can express a set of multiple Pfaffian constraints, as in Equation (14.8) as a matrix equation

$$A(q)\dot{q} = 0 \tag{14.12}$$

where A is the $k \times n$ matrix

$$A = \begin{pmatrix} w_{11} & \dots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{k1} & \dots & w_{kn} \end{pmatrix} \tag{14.13}$$

Note that $A(q)\dot{q} = 0$ means that the velocity $\dot{q}$ lies in the null space of $A(q)$ for each q. If the rank of $A(q)$ is equal to k, implying that the constraints are independent, then there will be an $m = n - k$-dimensional null space of A. Let $\{g_1(q), \dots, g_m(q)\}$ be a basis for the null space of $A(q)$. Then each velocity vector $\dot{q}$ satisfying Equation (14.8) can be expressed as

$$\dot{q} = g_1(q)u_1 + \cdots + g_mu_m \tag{14.14}$$

for suitable coefficients $u_1, \dots, u_m$. Equation (14.14) specifies the allowable velocity vectors as a linear combination of the basis vectors $g_1, \dots, g_m$.

Example 14.4. *Consider again the $k = 2$ constraints in Equations (14.9)–(14.10) on $\mathbb{R}^3$. We can write these as $A(q)\dot{q} = 0$ where*

$$A(q) = \begin{bmatrix} -q_3 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \ \dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

It is now easy to show by direct calculation that

$$g_1(q) = \begin{bmatrix} 1 \\ q_3 \\ q_3 \end{bmatrix}$$

spans the one-dimensional null space of A.

Remark 14.1. *Equation (14.14) is called a* **driftless system** *because* $\dot{q} = 0$ *when the control inputs* $u_1, \ldots, u_m$ *are zero. In many systems of interest the coefficients* u_i *in Equation (14.14) have the interpretation of control inputs. We will use this model in later sections for control design.*

14.2 Involutivity and Holonomy

In this section we refine the notion of nonholonomic constraints that will be useful later when we discuss controllability of nonholonomic systems and Chow's theorem.

Definition 14.3. *Given a set of Pfaffian constraints, as in Equation (14.8), let* Ω *be the codistribution spanned by the covectors* $\{w_1, \ldots, w_k\}$ *and let* $\{g_1, \ldots, g_m\}$*, for* $m = n - k$*, be a basis for a distribution* Δ *satisfying*

$$w_i(q)g_j(q) = 0 \quad i = 1, \ldots, k, \ j = 1, \ldots, m \quad \text{for } q \in \mathcal{Q} \tag{14.15}$$

The vectors $g_1, \ldots, g_m$ *then span the null space the matrix A defined previously. The distribution* Δ *is called the* **annihilator** *or* **annihilating distribution** *of* Ω *and is denoted* $\Delta = \Omega^{\perp}$*. Conversely,* Ω *is the* **annihilating codistribution** *of* Δ*.*

Theorem 14.1. *Suppose that* Ω *is the codistribution spanned by the covectors* $\{w_1, \ldots, w_k\}$*. Then the set of constraints* $\{w_i(q)\dot{q} = 0, i = 1, \ldots, k\}$ *is holonomic if and only if the distribution* $\Delta = \Omega^{\perp}$ *is involutive.*

Proof: Using the Lie derivative notation from Chapter 12, we can say that there exist functions $h_i, \ldots, h_k$ such that

$$w_i g_j = \frac{\partial h_i}{\partial q} g_j = L_{g_j} h_i = 0, \ i = 1, \ldots, k, \ j = 1, \ldots, m \tag{14.16}$$

if and only if the constraints $w_i\dot{q} = 0, \ i = 1, \ldots, k$ are holonomic. From the Frobenius theorem the set of partial differential equations

$$L_{g_j} h_i = 0, \quad i = 1, \ldots, k, \ j = 1, \ldots, m$$

has a solution for $h_1, \ldots, h_k$ if and only if the distribution Δ is involutive, and the result follows.

Remark 14.2. *In the case of a single constraint $w(q)\dot{q}$, the distribution $\Delta = \text{span}\ \{g_1, \ldots, g_{n-1}\}$ and involutivity of Δ is equivalent to Equation (14.5). At the other extreme, a set of $n-1$ Pfaffian constraints for an n-dimensional system is always holonomic, since, in this case, $\Delta = \text{span}\ \{g_1\}$, which is always involutive.*

Example 14.5. *Consider again the constraint Equation (14.7) with $w = (-q_3, 1, 0)$, which was shown to be nonholonomic. With $\Omega = \text{span}\{w\}$, it is easy to see that a basis for the annihilating distribution $\Delta = \Omega^{\perp}$ is*

$$g_1 = \begin{bmatrix} 1 \\ q_3 \\ 0 \end{bmatrix}, \ g_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{14.17}$$

Computing the Lie Bracket $[g_1, g_2]$ gives

$$[g_1, g_2] = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{14.18}$$

which is not a linear combination of g_1 and g_1. Hence, the distribution Δ is not involutive showing again that the Pfaffian constraint defined by w is nonholonomic.

Filtrations

Suppose now that the distribution Δ is not involutive. Then there is at least one pair of vector fields g_i and g_j in Δ such that the Lie Bracket $[g_i, g_j]$ does not belong to Δ. Thus, with $\Delta_1 = \Delta = \text{span}\{g_1, \ldots, g_m\}$, we define Δ_2 as the distribution

$$\Delta_2 = \Delta_1 + [\Delta_1, \Delta_1]$$

where

$$[\Delta_1, \Delta_1] = \text{span}\{[g_i, g_j], \text{ for all } g_i, g_j \in \Delta_1\}$$

where $[g_i, g_j]$ is the Lie Bracket of the vector fields g_i and g_j. The distribution Δ_2 is spanned by all vectors $g_i \in \Delta_1$ together with all possible vectors generated by taking Lie Brackets of pairs of vectors in Δ_1. Thus, if $\Delta_2 = \Delta_1$ it simply means that the original distribution Δ is involutive.

If Δ_1 is not involutive, then $\Delta_2 \neq \Delta_1$ and its dimension is strictly larger than the dimension of Δ_1. We can continue this process and define a sequence of distributions Δ_i, with $\Delta_1 = \Delta$ according to

$$\Delta_i = \Delta_{i-1} + [\Delta_1, \Delta_{i-1}]$$

where

$$[\Delta_1, \Delta_{i-1}] = \text{span}\{[g, \bar{g}] : g \in \Delta_1, \bar{g} \in \Delta_{i-1}\}$$

This sequence of distributions $\{\Delta_i\}$ is called a **filtration** associated with Δ.

Definition 14.4. *A filtration is said to be* **regular** *in a neighborhood U of q_0 if*

$$\text{rank } \Delta_i(q) = \text{rank } \Delta_i(q_0), \forall i \text{ and for all } q \in U$$

In other words, Δ_i has constant rank for all q.

If the filtration is regular then the dimension of Δ_i is greater than or equal to the dimension of Δ_{i-1} and cannot be greater than n, the dimension of the configuration space itself. If at some point there exists κ such that rank $\Delta_{\kappa+1}$ = rank Δ_κ then the above construction terminates. This leads us to

Definition 14.5. *Let $\{\Delta_i\}$ be a regular filtration associated with a distribution Δ. The smallest integer κ such* rank $\Delta_{\kappa+1}$ = rank Δ_κ *is called the* **degree of nonholonomy** *of the distribution Δ.*

Note that $m \leq \text{rank } \Delta_\kappa \leq n$ and $1 \leq \kappa \leq n - m + 1$. The degree of nonholonomy is related to the difficulty (or ease) of the motion planning and control problems. Based on the above derivation we can refine our definition of holonomic constraints as follows

Definition 14.6. *The set of k Pfaffian constraints, given by Equation (14.8), is*

1. *Holonomic if $\kappa = 1$ or, in other words, if the dimension* dim $\Delta_1 = n$
2. *Partially Nonholonomic if $m <$* dim $\Delta_\kappa < n$, *i.e. $1 < \kappa < n - m + 1$*
3. *Completely Nonholonomic if* dim $\Delta_\kappa = n$, *i.e. $\kappa = n - m + 1$*

The distribution Δ_κ is called the **involutive closure** of Δ, denoted $\bar{\Delta}$, and by construction, is the smallest involutive distribution containing Δ, in the sense that if Δ' is an involutive distribution such that $\Delta \subset \Delta'$, then $\bar{\Delta} \subset \Delta'$.

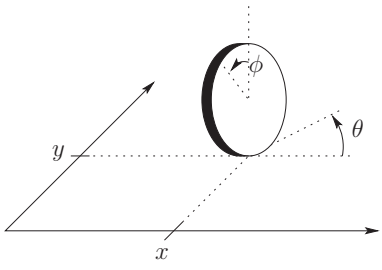


Figure 14.2: The rolling disk.

14.3 Examples of Nonholonomic Systems

In this section we give examples of systems subject to nonholonomic constraints.

Example 14.6 (The Rolling Disk)**.** *Figure 14.2 shows a disk rolling on a plane with its rolling axis parallel to the plane. The configuration of the disk can be defined by the variables* $q = (x, y, \theta, r\phi)$*, where* x *and* y *denote the Cartesian position of the ground contact point,* θ *denotes the heading angle, and* ϕ *denotes the angle of the wheel measured from the vertical. The rolling without slipping condition means that the instantaneous velocity of the ground contact point is zero, which implies*

$$\begin{array}{rcl} \dot{x} - r\dot{\phi}\cos\theta & = & 0 \\ \dot{y} - r\dot{\phi}\sin\theta & = & 0 \end{array} \tag{14.19}$$

where r *is the radius of the wheel. These constraints can be written as Pfaffian constraints in the form of Equation (14.3) with* $q = [x, y, \theta, r\phi]^T$ *and*

$$w_1 = \left[\begin{array}{cccc} 0 & 1 & 0 & -\sin\theta \end{array}\right], \; w_2 = \left[\begin{array}{cccc} 1 & 0 & 0 & -\cos\theta \end{array}\right]$$

Since the dimension of the configuration space is $n = 4$ *and there are two constraint equations, a basis for* $\Omega^{\perp}$ *will consist of two function* g_1*,* g_2 *orthogonal to* w_1*,* w_2*. It is easy to see that*

$$g_1 = \left[\begin{array}{c} \cos\theta \\ \sin\theta \\ 0 \\ 1 \end{array}\right] \; ; \; g_2 = \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \end{array}\right] \tag{14.20}$$

are both orthogonal to w_1 *and* w_2*. Thus, we can write a velocity vector* $\dot{q}$ *as*

$$\dot{q} = g_1(q)u_1 + g_2(q)u_2 \tag{14.21}$$

where $u_1 = v$ is the rate of rolling and $u_2 = \omega$ is the turning rate.

We can now check to see if rolling without slipping constraints on the disk are holonomic or nonholonomic using Theorem 14.1. Computing the Lie brackets (Problem 14–3)

$$g_3 = [g_1, g_2] = \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \\ 0 \end{bmatrix}, \quad g_4 = [g_2, g_3] = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \\ 0 \end{bmatrix} \tag{14.22}$$

we see that

$$\begin{aligned} \Delta = \Delta_1 &= \text{span } \{g_1, g_2\} \text{ has rank 2} \\ \Delta_2 &= \text{span } \{g_1, g_2, g_3\} \text{ has rank 3} \\ \Delta_3 &= \text{span } \{g_1, g_2, g_3, g_4\} \text{ has rank 4} \end{aligned}$$

Therefore $\dim \bar{\Delta} = \dim \Delta_3 = 4$. Thus the degree of nonholonomy is 3 and the constraints are completely nonholonomic.

Remark 14.3. *If we do not care about the orientation of the disk about the horizontal axis, i.e., if we ignore the last equation $r\dot{\phi} = u_1$, then the constraints in Equation (14.19) can be reduced to the single constraint*

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = 0$$

and we have a reduced-order model

$$\begin{aligned} \dot{x} &= \cos(\theta)u_1 \\ \dot{y} &= \sin(\theta)u_1 \\ \dot{\theta} &= u_2 \end{aligned} \tag{14.23}$$

where

$$g_1 = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \; ; \; g_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The system in Equation (14.23) is more commonly known as the **unicyle model**.

Example 14.7 (The Kinematic Car)**.** *Figure 14.3 shows a simple representation of a car, or mobile robot, with steerable front wheels. The configuration can be described by $q = (x, y, \theta, \phi)$, where x and y are the point at the*

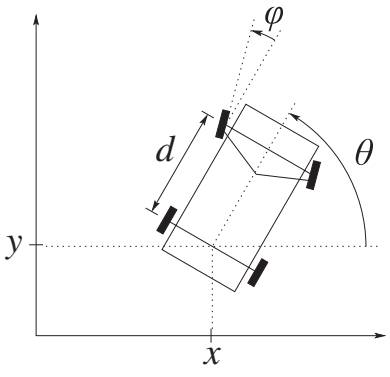


Figure 14.3: The kinematic car.

center of the rear axle, θ is the heading angle, and ϕ is the steering angle as shown in the figure. The rolling without slipping constraints are found by setting the sideways velocity of the front and rear wheels to zero. This leads to

$$\begin{aligned} \sin\theta\ \dot{x} - \cos\theta\ \dot{y} &= 0 \\ \sin(\theta+\phi)\ \dot{x} - \cos(\theta+\phi)\ \dot{y} - d\cos\phi\ \dot{\theta} &= 0 \end{aligned} \tag{14.24}$$

which can be written as

$$\begin{aligned} \left[\ \sin\theta,\quad \cos\theta,\quad 0,\quad 0\ \right]\dot{q} &= w_1\dot{q} = 0 \\ \left[\ \sin(\theta+\phi),\quad -\cos(\theta+\phi),\quad -d\cos\phi,\quad 0\ \right]\dot{q} &= w_2,\dot{q} = 0 \end{aligned} \tag{14.25}$$

where the covectors w_1 and w_2 are

$$w_1 = [\sin\theta,\ \cos\theta,\ 0,\ 0],\ w_2 = [\sin(\theta+\phi), -\cos(\theta+\phi),\ -d\cos\phi,\ 0]$$

It is now straightforward to find vectors g_1 and g_2 orthogonal to w_1 and w_2 and write the corresponding system in the form of Equation (14.14). For example, we may take

$$g_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}\ ;\ g_2 = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{d}\tan\phi \\ 0 \end{bmatrix} \tag{14.26}$$

Computing the Lie Brackets $[g_1, g_2]$ and $[g_2, g_3]$ gives

$$\begin{aligned} g_3 &= [g_1, g_2] = [0,\ 0,\ \frac{1}{d\cos^2(\phi)},\ 0]^T \\ g_4 &= [g_2, g_3] = [\frac{\sin(\theta)}{d\cos^2(\phi)},\ \frac{-\cos(\theta)}{d\cos^2(\phi)},\ 0,\ 0]^T \end{aligned}$$

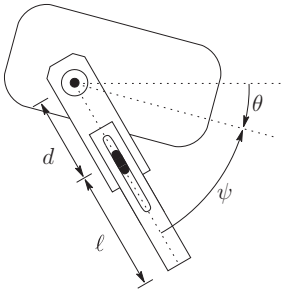


Figure 14.4: A hopping robot.

We have that

$$\Delta_4 = \text{span } \{g_1, g_2, g_3, g_4\}$$

is involutive of rank 4. *Hence the system has degree of nonholonomy* $\kappa = 3$.

Example 14.8 (A Hopping Robot)**.** *Consider the hopping robot in Figure 14.4. The configuration of this robot is defined by* $q = (\psi, \ell, \theta)$, *where*

ψ = *the leg angle*
θ = *the body angle*
ℓ = *the leg extension*

During its flight phase the hopping robot's angular momentum is conserved. Letting I *and* m *denote the body moment of inertia and leg mass, respectively, conservation of angular momentum leads to the expression*

$$I\dot{\theta} + m(\ell + d)^2(\dot{\theta} + \dot{\psi}) = 0 \tag{14.27}$$

assuming the initial angular momentum is zero. This constraint may be written as $w(q)\dot{q} = 0$ *where* $w = \left[\ m(\ell+d)^2, \quad 0, \quad I + m(\ell+d)^2\ \right]$. *Since the dimension of the configuration space is three and there is one constraint, we need to find two independent vectors,* g_1 *and* g_2, *spanning the annihilating distribution* $\Delta = \Omega^{\perp}$. *It is easy to see that*

$$g_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad g_2 = \begin{bmatrix} 1 \\ 0 \\ -\frac{m(\ell+d)^2}{I+m(\ell+d)^2} \end{bmatrix} \tag{14.28}$$

are linearly independent at each point and orthogonal to w. *Checking involutivity of* Δ *we find that*

$$[g_1, g_2] = \begin{bmatrix} 0 \\ 0 \\ \frac{-2Im(\ell+d)}{[I+m(\ell+d)^2]^2} \end{bmatrix} \tag{14.29}$$

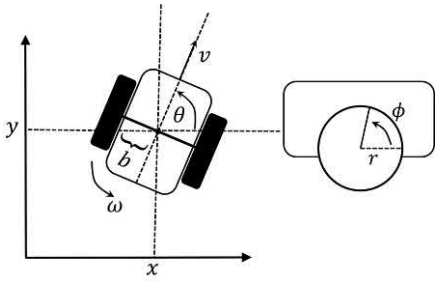


Figure 14.5: The differential drive robot: top view (left), side view (right).

which is not a linear combination of g_1 and g_2. It follows that the constraint is nonholonomic and the degree of nonholonomy is $\kappa = 2$.

Example 14.9 (Differential Drive Robot)**.** *The differential drive robot (henceforth, DDR) shown in Figure 14.5 consists of a chassis and two independently controlled wheels, giving it three degrees of freedom, x, y, and θ. The basic kinematic equations of the DDR are identical to those of the unicycle*

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= \omega \end{aligned} \tag{14.30}$$

where v and ω are the linear and angular velocities of the chassis, respectively. Assume that each wheel has radius r and that the distance between the wheels is $2b$. Assuming no wheel slipping, we can relate the chassis velocities v and ω to the left and right wheel velocities $\dot{\phi}_L$ and $\dot{\phi}_R$ as

$$\begin{aligned} v + b\omega &= r\dot{\phi}_R \\ v - b\omega &= r\dot{\phi}_L \end{aligned}$$

Solving for v and ω gives

$$\begin{aligned} v &= \frac{r}{2}(\dot{\phi}_R + \dot{\phi}_L) \\ \omega &= \frac{r}{2b}(\dot{\phi}_R - \dot{\phi}_L) \end{aligned}$$

Note that actuating both wheels in the same direction with speed $\dot{\phi}$ results in a pure translation of the chassis with speed $r\dot{\phi}$ while actuating both wheels in opposite directions with speed $\dot{\phi}$ results in a pure rotation with angular speed $\frac{r}{b}\dot{\phi}$.

Typically the wheels are actuated by DC-electric motors. Setting τ_R and τ_L as the left and right motor torques, respectively, we can express the dynamics of the DDR as

$$\begin{aligned} m\dot{v} &= \frac{1}{r}\tau_R + \frac{1}{r}\tau_L \\ J\dot{\omega} &= \frac{b}{r}\tau_R - \frac{b}{r}\tau_L \end{aligned}$$

where m is the total mass of the DDR and J is its moment of inertia about the center of mass, which we can take as the midpoint of the wheel axle. With

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix} \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix}$$

and

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{mr} & \frac{1}{mr} \\ \frac{b}{Jr} & -\frac{b}{Jr} \end{bmatrix} \begin{bmatrix} \tau_R \\ \tau_L \end{bmatrix}$$

we have

$$\begin{aligned} \begin{bmatrix} \tau_R \\ \tau_L \end{bmatrix} &= \begin{bmatrix} \frac{1}{mr} & \frac{1}{mr} \\ \frac{b}{Jr} & -\frac{b}{Jr} \end{bmatrix}^{-1} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix} \begin{bmatrix} \ddot{\phi}_R \\ \ddot{\phi}_L \end{bmatrix} \\ &= \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} \ddot{\phi}_R \\ \ddot{\phi}_L \end{bmatrix} \end{aligned}$$

where $D_{11} = D_{22} = \frac{r^2}{4}m + \frac{r^2}{4b^2}J$ and $D_{12} = D_{21} = \frac{r^2}{4}m - \frac{r^2}{4b^2}J$. If we include damping terms $-B\dot{\phi} = (-b_R\dot{\phi}_R, -b_L\dot{\phi}_L)$ we can write the dynamic equations relating the motor torque to wheel acceleration as

$$D\ddot{\phi} + B\dot{\phi} = \tau \tag{14.31}$$

A typical DDR will be equipped with a low-level controller (usually PID control) to control the motor speed and position, which allows control of the wheel speed and position from Equation (14.31) assuming no wheel slipping.

Thus, given desired linear and angular speeds, v^d and ω^d, of the chassis, one can assume that $v = v^d$ and $\omega = \omega^d$ after a short transient. Therefore, it is common just to use the kinematic equations (14.30) as the model for motion planning and control.

14.4 Dynamic Extension

Dynamic models for nonholonomic systems can be obtained in a straightforward way as an extension of the previous kinematic models. These extended models typically take the form

$$\dot{q} = g_1(q)u_1 + \cdots + g_m(q)u_m \tag{14.32}$$

$$\dot{u}_i = v_i, \quad i = 1, \ldots, m \tag{14.33}$$

Note that the system in Equations (14.32)–(14.33) is now a system with drift of the form

$$\begin{bmatrix} \dot{q} \\ \dot{u} \end{bmatrix} = \begin{bmatrix} G(q)u \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} v$$

The above model can be derived from the Euler–Lagrange equations in the form

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = A^T(q)\lambda + B(q)\tau \tag{14.34}$$

$$A(q)\dot{q} = 0 \tag{14.35}$$

where $A(q)\dot{q}$ is defined using the nonholonomic Pfaffian constraints and λ is a vector of Lagrange multipliers. This model relates to the kinematic model

$$\dot{q} = g_1(q)u_1 + \cdots + g_m(q)u_m = G(q)u \tag{14.36}$$

if we take, as before, $g_1, \ldots, g_m$ as a basis for the null space of $A(q)$. Differentiating Equation (14.36) gives

$$\ddot{q} = G(q)\dot{u} + \dot{G}(q)u \tag{14.37}$$

Since $G^T(q)A^T(q) = 0$ by choice of basis vectors g_i, if we substitute the expression for $\ddot{q}$ from Equation (14.37) in into the Euler–Lagrange equations and multiply by $G^T(q)$ we obtain

$$G^T(q)M(q)G(q)\dot{u} + F(q, \dot{q}, u) = G^T(q)B(q)\tau \tag{14.38}$$

where

$$F(q, \dot{q}, u) = G^T(q)(M(q)\dot{G}(q)u + C(q, \dot{q})\dot{q} + g(q)) \tag{14.39}$$

Assuming $G^T(q)B(q)$ is invertible, the control input τ can be chosen as a feedback linearizing control

$$\tau = -(G^T(q)B(q))^{-1}(G^T(q)M(q)G(q)v + F(q,\dot{q},u)) \tag{14.40}$$

to obtain

$$\dot{u} = v \tag{14.41}$$

Note that τ depends on the control input u since $F = F(q,\dot{q},u)$. We can compute u from Equation (14.36) as

$$u = G^\dagger \dot{q} = G(GG^T)^{-1}\dot{q}$$

where $G^\dagger$ is the right pseudoinverse of G, and substitute this expression into Equation (14.39) to complete the computation of the control τ.

Example 14.10. *For the unicycle model,*

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= \omega \end{aligned}$$

let τ_1 be the driving force and τ_2 be the steering torque so that

$$\begin{aligned} m\dot{v} &= \tau_1 \\ J\dot{\omega} &= \tau_2 \end{aligned}$$

where m is the unicycle mass and J the moment of inertia about the vertical axis. Then a simple dynamic extension is given by

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= \omega \\ \dot{v} &= \nu_1 \\ \dot{\omega} &= \nu_2 \end{aligned}$$

with $\nu_1 = \tau_1/m$ and $\nu_2 = \tau_2/J$. To see how this model is obtained from the general theory, we can compute, using the above equations,

$$\begin{aligned} m\ddot{x} &= -\sin(\theta)mv + \cos(\theta)m\dot{v} \\ m\ddot{y} &= \cos(\theta)mv + \sin(\theta)m\dot{v} \\ J\ddot{\theta} &= J\dot{\omega} \end{aligned}$$

which we can write as

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \\ 0 \end{bmatrix} \lambda + \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = 0$$

where the second equation above describes the nonholonomic constraint. Thus, with A and G given by

$$A = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \\ 0 \end{bmatrix}, \qquad G = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix}$$

the above equations are of the form

$$\begin{aligned} M\ddot{q} &= A(q)\lambda + G(q)\tau \\ A^T(q)\dot{q} &= 0 \end{aligned}$$

and a straightforward calculation (Problem 14–10) gives

$$G^T M G = \begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix}, \quad G^T M \dot{G} u + G^T(C\dot{q} + g) = 0, \quad G^T G = I$$

and therefore we obtain

$$\begin{aligned} m\dot{v} &= \tau_1 \\ J\dot{\omega} &= \tau_2 \end{aligned}$$

as above.

14.5 Controllability of Driftless Systems

In this section we consider the controllability of driftless systems of the form Equation (14.14). There are several notions of controllability for general nonlinear systems, all of which are equivalent for linear systems. The examples of wheeled mobile robots that we consider in this chapter all satisfy the strongest notion of controllability defined next and so we will not have need for other (weaker) notions of controllability.

We assume that the vector fields $g_1(q), \ldots, g_m(q)$ are smooth, complete,[1] and linearly independent at each $q \in \mathbb{R}^n$. Under these conditions we can state the following:

[1] A complete vector field is one for which the solution of the associated differential equation exists for all time t.

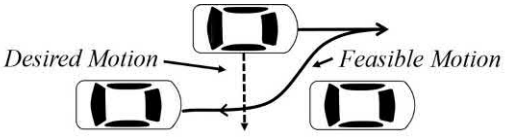


Figure 14.6: The car parking problem.

Definition 14.7. *The driftless nonlinear system in Equation (14.14) is* **controllable** *if, for any q_0 and $q_1 \in \mathbb{R}^n$, there exists a time $T > 0$ and a control input*

$$u(t) = (u_1(t), \ldots, u_m(t))^T \ : \ [0, T] \times \mathcal{U} \to \mathbb{R}^m$$

such that the solution $q(t)$ of Equation (14.14) satisfies $q(0) = q_0$ and $q(T) = q_1$.

Let us digress for a moment to gain an intuitive understanding of the notion of nonlinear controllability and its relation to nonholonomy. Consider the familiar problem of parking a car. Referring to Figure (14.6) the desired motion is to move the car laterally, which is prohibited by the rolling constraints on the wheels. However, by sequentially applying the two control inputs, namely 1) turning the wheels left and right, and 2) driving forward and backward, one is able to park the car. With this example in mind, let the system

$$\dot{q} = g_1(q)u_1 + g_2(q)u_2 \tag{14.42}$$

be defined by the two vector fields g_1 and g_2 and consider the following thought experiment consisting of sequentially applying the control u_1 and u_2:

- Set $u_1 = 1$ and $u_2 = 0$ and follow the vector field g_1 for a time $t = \epsilon$.
- Then set $u_1 = 0$ and $u_2 = 1$ to follow the vector field g_2 for time $\epsilon < t \leq 2\epsilon$.
- Reverse the above by setting $u_1 = -1$, $u_2 = 0$ for time $2\epsilon < t \leq 3\epsilon$.
- Finally, set $u_2 = -1$ and $u_1 = 0$ for time $3\epsilon < t \leq 4\epsilon$.

Denote by $\phi_t^g(q_0)$ the state of the system $\dot{q} = g(q)$ at time t starting at q_0 at time $t = 0$. The function ϕ is called the **flow** of the vector field g. With this

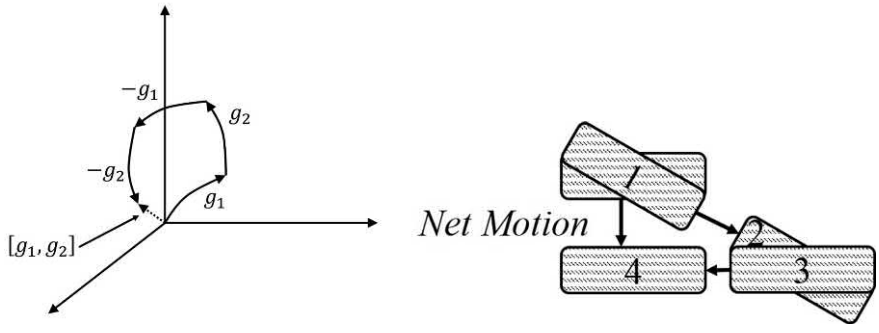


Figure 14.7: Illustrating the notion of Lie bracket direction.

notation, the state $q(4\epsilon)$ that is reached after the above thought experiment is given by

$$q(4\epsilon) = \phi_\epsilon^{-g_2}(\phi_\epsilon^{-g_1}(\phi_\epsilon^{g_2}(\phi_\epsilon^{g_1}(q_0))))$$

It is shown in most basic texts in differential geometry using a Taylor series expansion of the flow that the resulting state reached after the four sequential moves is given by

$$q(4\epsilon) = q_0 + \epsilon^2 \left(\frac{\partial g_2}{\partial q} g_1(q_0) - \frac{\partial g_1}{\partial q} g_2(q_0) \right) + O(\epsilon^3)$$

In other words, the motion is equivalent, for small ϵ, to moving in the Lie bracket direction $[g_1, g_2]$. We will show later that the lateral motion of the car is precisely the direction given by the Lie bracket of the 'turning' and 'driving' input vectors.

The above discussion suggests that the possible directions of motion, i.e., the velocity directions, include not only the directions given by $g_1, \ldots, g_m$ but also motions in the direction $[g_i, g_j]$, $[g_k, [g_i, g_j]]$, and so on. Thus, it is important to know if the vectors $[g_i, g_j]$ are simply linear combinations of the original vectors $g_1, \ldots, g_m$ or if they are vectors defining new and independent directions.

The next result, known as **Chow's theorem**, formalizes this notion and gives a necessary and sufficient condition for a system given by Equation (14.43) to be controllable.

Theorem 14.2 (Chow)**.** *The driftless system*

$$\dot{q} = g_1(q)u_1 + \cdots + g_m(q)u_m \tag{14.43}$$

is controllable if and only if $\operatorname{rank} \bar{\Delta}(q) = n$ *at each* $q \in \mathbb{R}^n$*, where* $\bar{\Delta}(q)$ *is the involutive closure of* $\Delta(q) = \operatorname{span}\{g_1(q), \ldots, g_m(q)\}$.

Example	n	m	κ
rolling disk	4	2	3
unicycle	3	2	2
kinematic car	4	2	3
hopper	3	2	2

Table 14.1: Examples of controllable driftless systems.

The condition rank $\bar{\Delta}(q) = n$ is called the **controllability rank condition**. The involutive closure $\bar{\Delta}$ of Δ is also called the **control Lie algebra** for the system in Equation (14.43).

Example 14.11. *Consider the following system on* $\mathbb{R}^3$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} q_3 \\ 1 - q_3^2 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2 = g_1(q)u_1 + g_2(q)u_2$$

with three states q_1, q_2, q_3, *and two control inputs,* u_1 *and* u_2. *It is easy to see that the distribution* $\Delta = \text{span}\{g_1, g_2\}$ *has rank two for all values of* $x \in \mathbb{R}^3$. *The Lie bracket* $[g_1, g_2]$ *is*

$$[g_1, g_2] = \begin{bmatrix} -1 \\ 2q_3 \\ 0 \end{bmatrix}$$

and, therefore, we have

$$\text{rank } \{g_1, g_2, [g_1, g_2]\} = \text{rank } \begin{bmatrix} q_3 & 0 & -1 \\ 1 - q_3^2 & 0 & 2q_3 \\ 0 & 1 & 0 \end{bmatrix} = 3$$

for all values of $q \in \mathbb{R}^3$. *Therefore, by Chow's theorem, the system is controllable on* $\mathbb{R}^3$.

Remark 14.4. *It is apparent from Chow's theorem and our definition of nonholonomy that a driftless system with* $q \in \mathbb{R}^n$ *and* $u \in \mathbb{R}^m$ *is controllable if and only if the constraints are completely nonholonomic, which means that the degree of nonholonomy* κ *should be* $\kappa = n - m + 1$. *The reader should now check that each of the examples that we presented in Section 14.3 is controllable according to Chow's theorem. Table 14.1 summarizes this result.*

14.6 Motion Planning

We have given several examples of nonholonomic mobile robots that are controllable according to Chow's theorem. Chow's theorem tells us when a driftless system is controllable but does not tell us how to find the control input to steer the system from a given initial state x_0 to a desired final state x_1. In this section we present several ideas for planning feasible trajectories for these systems.

14.6.1 Conversion to Chained Forms

We have seen the usefulness of transforming systems into canonical or normal forms through the use of nonlinear coordinate transformations and nonlinear feedback, as in the feedback linearization in Chapter 12 and partial feedback linearization in Chapter 13. In this section we introduce the so-called **chained form** which gives a useful transformation of the kinematics of many nonholonomic systems. We limit the discussion to systems with two inputs, such as the unicycle, DDR, and kinematic car.

Example 14.12. *To motivate the subsequent development, consider again the unicycle model*

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= \omega \end{aligned}$$

Define a coordinate transformation

$$\begin{aligned} z_1 &= \theta \\ z_2 &= x\cos(\theta) + y\sin(\theta) \\ z_3 &= -x\sin(\theta) + y\cos(\theta) \end{aligned}$$

Note that the coordinates (z_2, z_3) represent a rotational transformation of (x, y) through the angle θ. In terms of the transformed coordinates, z_1, z_2, and z_3, the input transformation

$$\begin{aligned} v_1 &= \omega \\ v_2 &= v - z_3\omega \end{aligned}$$

results in the system (Problem 14–5)

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \end{aligned} \tag{14.44}$$

which is referred to as a $(2,3)$**-chained system** *since it has two inputs and three states.*

We note that, like the unicycle model it derives from, Equation (14.44) defines a driftless nonlinear system, and can be shown to be controllable using Chow's theorem (Problem 14–12).

In the system of equations (14.44) the variables z_1 and z_2 are called **base variables** and can be independently controlled with inputs v_1 and v_2, respectively. The third coordinate z_3 is called a **fiber variable** and evolves according to the input $z_2 v_1$ and therefore cannot be independently controlled.

Steering Using Sinusoids

The chain form representation is useful for deriving trajectories in configuration space using several methods. One such method is to use sinusoidal functions at different frequencies for the inputs v_1 and v_2 to move the shape variables through periodic motions. This will move the fiber variables to new values that are functions of the amplitudes and frequencies of the input functions. The motion planning problem then becomes one of choosing appropriate values for the amplitudes and frequencies of v_1 and v_2. We illustrate this idea below for the case of the unicycle model.

Example 14.13. *Consider the chained-form system derived above from the unicycle model*

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \end{aligned} \tag{14.45}$$

and let us choose the control inputs v_1 *and* v_2 *to minimize the quadratic cost function*

$$J = \frac{1}{2}\int_0^{t_f} (v_1^2(t) + v_2^2(t))dt$$

To solve this problem we form the Hamiltonian function

$$\mathcal{H}(z,v) = \frac{1}{2}v_1^2(t) + \frac{1}{2}v_2^2(t) + \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 z_2 v_1 \tag{14.46}$$

where the Lagrange multiplier vector $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ *satisfies the adjoint equations* $\dot{\lambda} = -\partial\mathcal{H}/\partial z$. *Since the Hamiltonian* $\mathcal{H}$ *is independent of* z_1 *and*

z_3, the adjoint equations are easily computed as

$$\begin{aligned} \dot{\lambda}_1 &= 0 \\ \dot{\lambda}_2 &= -\lambda_3 v_1 \\ \dot{\lambda}_3 &= 0 \end{aligned} \tag{14.47}$$

The necessary condition for optimality, $\partial \mathcal{H}/\partial v = 0$, leads to

$$\partial \mathcal{H}/\partial v_1 = v_1 + \lambda_1 + \lambda_3 z_2 = 0 \tag{14.48}$$

$$\partial \mathcal{H}/\partial v_2 = v_2 + \lambda_2 = 0 \tag{14.49}$$

from which we get

$$v_1 = -\lambda_1 - \lambda_3 z_2, \qquad v_2 = -\lambda_2 \tag{14.50}$$

Differentiating v_1 and v_2 in Equation (14.50) and using Equation (14.47) yields

$$\dot{v}_1 = -\lambda_3 v_2 \tag{14.51}$$

$$\dot{v}_2 = \lambda_3 v_1 \tag{14.52}$$

which has the general solution

$$v_1(t) = v_1(0)\cos(\lambda_3 t) - v_2(0)\sin(\lambda_3 t) \tag{14.53}$$

$$v_2(t) = v_1(0)\sin(\lambda_3 t) + v_2(0)\cos(\lambda_3 t) \tag{14.54}$$

Thus, we have shown that the control for the chained-form system that minimizes the cost function (14.46) consists of sinusoidal inputs.

This result leads to the following motion planning strategy for the above system:

- First, use any method to move the base variables z_1 and z_2 to their desired final values, z_1^d and z_2^d, respectively. Since the base variables are independently controlled, this step is simple and should be completed in finite time t_1.
- Next, execute a periodic motion of the base variables over one period T. This motion will bring the base variables back to their desired final value and move the fiber variable z_3 from $z_3(t_1)$ to $z_3(t_1 + T)$. Choose the periodic motion of the base variables so that $z_3(t_1 + T) = z_3^d$, the desired value for z_3.

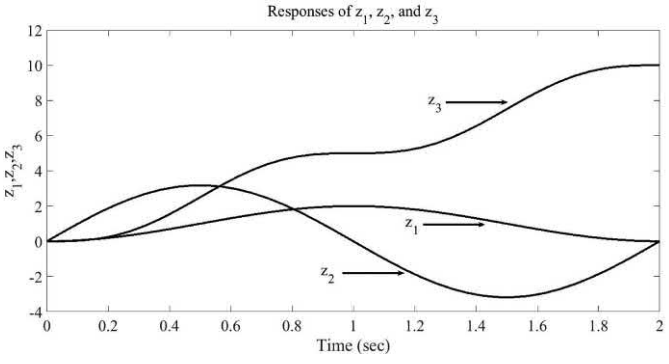


Figure 14.8: Response of z_1, z_2, and z_3: Note that z_1 and z_2 return to their original values after 2 seconds while z_3 moves from the origin to $z_3 = 10$ as desired.

The strategy is suboptimal since the control of the shape variables in the first step is not necessarily optimal. If we now choose the controls v_1 and v_2 as

$$\begin{aligned} v_1 &= a\sin(\omega t) \\ v_2 &= b\cos(\omega t) \end{aligned}$$

it is easily shown (Problem 14–8) that after $2\pi/\omega$ seconds, z_1 and z_2 return to their initial values whereas the change in z_3 is $ab\pi/\omega^2$.

Example 14.14. *Suppose that we wish to move the unicycle from the origin* $(z_1, z_2, z_3) = (0, 0, 0)$ *to* $(x_1, x_2, x_3) = (0, 0, 10)$ *in two seconds. Using the above controls with* $a = \pi$, $\omega = \pi$, $b = 10$ *results in the response shown in Figure 14.8.*

Chained Form for Higher Dimensional Systems

We can generalize the previous idea of conversion to chained form to higher dimensional $(2, n)$-chained form systems as follows.

Definition 14.8. *A* $(2, n)$ **chained-form** *is a driftless system with* 2 *inputs,*

v_1 and v_2, and n configuration variables $z_1, \ldots, z_n$ in the form

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \\ &\vdots \\ \dot{z}_n &= z_{n-1} v_1 \end{aligned} \tag{14.55}$$

In this case, z_1 and z_2 are the base variables and $z_3, \ldots, z_n$ are the fiber variables.

The system in Equation (14.55) can be written in the form of a driftless system as

$$\dot{z} = a_1(z)v_1 + a_2(z)v_2$$

where

$$a_1(z) = \begin{bmatrix} 1 \\ 0 \\ z_2 \\ \vdots \\ z_{n-1} \end{bmatrix}, \quad a_2(z) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{14.56}$$

It is not too difficult to show that the chained-form system, Equation (14.55), is controllable according to Chow's theorem. In order to see this, one can show by induction (Problem 14–6) that

$$ad_{a_1}^k(a_2) = \begin{bmatrix} 0 \\ \vdots \\ (-1)^k \\ \vdots \\ 0 \end{bmatrix}$$

where the non-zero entry $(-1)^k$ appears in the $k+2$ position. It follows by direct calculation that the distribution

$$\Delta = \text{span}\{a_1, a_2, ad_{a_1}(a_2), \ldots, ad_{a_1}^{n-2} a_2\}$$

has dimension n and controllability follows from Chow's theorem.

We state without proof a sufficient condition for the two-input case for conversion to chained form. Specifically, given a driftless system of the form

$$\dot{q} = g_1(q)u_1 + g_2(q)u_2 \tag{14.57}$$

there exist a diffeomorphic change of coordinates $z = T(q) : \mathbb{R}^n \to \mathbb{R}^n$ and a control input $v = \beta(u)$ that transforms Equation (14.57) into the $(2, n)$ chain form Equation (14.55) if the following distributions are constant rank and involutive

$$\begin{aligned} \Delta_0 &= \text{span}\{g_1, g_2, ad_{g_1}(g_2), \ldots, ad_{g_1}^{n-2}(g_2)\} \\ \Delta_1 &= \text{span}\{g_2, ad_{g_1}(g_2), \ldots, ad_{g_1}^{n-2}(g_2)\} \\ \Delta_2 &= \text{span}\{g_2, ad_{g_1}(g_2), \ldots, ad_{g_1}^{n-3}(g_2)\} \end{aligned} \tag{14.58}$$

and there exists a function $h_1(q)$ such that

$$L_{\Delta_1} h_1 = 0 \text{ and } L_{g_1} h_1 = 1 \tag{14.59}$$

and a function h_2, independent of h_1, such that

$$L_{\Delta_2} h_2 = 0 \tag{14.60}$$

Note that Equations (14.59) and (14.60) are a set of partial differential equations that must be solved to find h_1 and h_2. The required change of coordinates in the state and control inputs are then given by

$$\begin{aligned} z_1 &= h_1 \\ z_2 &= L_{g_1}^{n-2} h_2 \\ &\vdots \\ z_{n-1} &= L_{g_1} h_2 \\ z_n &= h_2 \end{aligned}$$

$$\begin{aligned} v_1 &= u_1 \\ v_2 &= (L_{g_1}^{n-1} h_2) u_1 + (L_{g_2} L_{g_1}^{n-2} h_2) u_2 \end{aligned}$$

Example 14.15. *We can apply the above transformation to the kinematic car*

$$\begin{aligned} \dot{q}_1 &= u_1 \\ \dot{q}_2 &= \tan(q_3) u_1 \\ \dot{q}_3 &= \frac{1}{d} \tan(q_4) u_1 \\ \dot{q}_4 &= u_2 \end{aligned} \tag{14.61}$$

It is straightforward (Problem 14–7) to compute the distributions Δ_0, Δ_1, *and* Δ_2 *and verify that they are of constant rank and involutive. With* $h_1 = q_1$ *and* $h_2 = q_2$ *the state transformation*

$$\begin{aligned} z_1 &= q_1 \\ z_2 &= \frac{1}{d}\sec^2(q_3)\tan q_4 \\ z_3 &= \tan(q_3) \\ z_4 &= q_2 \end{aligned}$$

and input transformation

$$\begin{aligned} u_1 &= v_1 \\ u_2 &= -\frac{2}{d}\sin^2(q_4)\tan(q_3)v_1 + d\cos^2(q_3)\cos^2(q_4)v_2 \end{aligned}$$

yields the required $(2,4)$ *chain system.*

For the $(2,n)$ chained-form system, Equation (14.55), a similar procedure to the above can be applied to sequentially move each variable z_k to its desired value as follows:

- First choose v_1 and v_2 to move the base variables z_1 and z_2 to their desired values in time t_1. As before, the fiber variables $z_3, \ldots, z_n$ will move to new values $z_3(t_1), \ldots, z_n(t_1)$.
- For $3 \leq k \leq n$, steer z_k to its desired value z_k^d with controls

  $$v_1 = a\sin(\omega t), \qquad v_2 = b\cos(k\omega t)$$

  over one period, $T = 2\pi/\omega$, where a and b are chosen to satisfy

  $$z_k^d - z_k(t_1 + (k-1)T) = (\frac{a}{2\omega})^k \frac{b}{k!}$$

Carrying out the sequential steering controls moves the chained-form system from z_0 to z^d in time $t = t_1 + (n-2)T$.

Example 14.16. *Consider the* $(2,4)$ *chained-form system*

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \\ \dot{z}_4 &= z_3 v_1 \end{aligned}$$

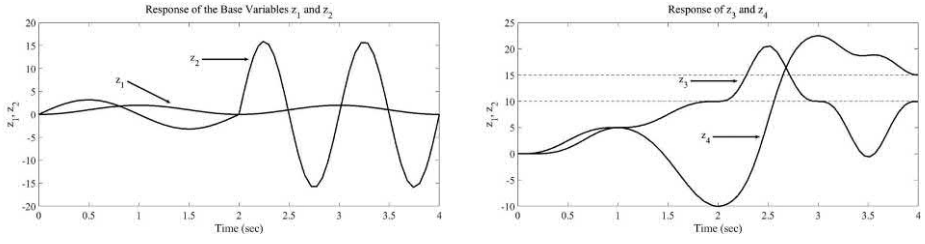


Figure 14.9: Response of z_1, z_2 and z_3.

and suppose we wish to move the states from $(0,0,0,0)$ *to* $(0,0,10,15)$ *in* 4 *seconds. In the first stage we take* v_1 *and* v_2 *as in the previous example as*

$$v_1 = a\sin(\omega t), \quad v_2 = b\cos(\omega t), \quad \text{with } ab/\omega = 10$$

After 2 *seconds,* z_3 *reaches its desired value of* 10 *and* $z_4(2) = -10$*. In the second stage we take*

$$v_1 = a\sin(\omega t), \quad v_2 = b\cos(2\omega t)$$

with $z_4^d - z_4(2) = (\frac{a}{2\omega})^2 \frac{b}{2}$*. With* $z_4^d = 15$ *and* $a = \omega = \pi$*, this gives* $b = 100$*. The responses are shown in Figure 14.9.*

14.6.2 Differential Flatness

In this section we introduce the notion of differential flatness and its application to the motion planning and control problems. The property of flatness simplifies the process of determining feasible trajectories for mechanical systems. This is particularly useful for underactuated systems and systems with nonholonomic constraints since, as we have seen, not all trajectories are feasible. It turns out that all of the systems considered in the chapter, indeed all driftless systems that can be converted to chained form, are differentially flat.

The basic idea of differential flatness is, given a system with n states and m inputs, find a set of m outputs so that the states and inputs can be expressed as functions of these m outputs and their derivatives. More precisely, we have

Definition 14.9. *The nonlinear system*

$$\dot{x} = f(x,u),\ x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

is said to be **differentially flat** *if there exists an output* $y = (y_1, \ldots, y_m)$*, with the same dimension as the control input, satisfying the following conditions:*

- *There exists a nonnegative integer* α *such* $y_i = h_i(x, u, \dot{u}, \ldots, u^{(\alpha)})$ *for each* $i = 1, \ldots, m$*, for some smooth functions* $h_1, \ldots h_m$*.*
- *There are smooth functions,* ϕ*, and* ψ*, and a nonnegative integer* $r \geq 0$ *such that*

$$\begin{aligned} x &= \phi(y, \dot{y}, \ldots, y^{(r)}) \\ u &= \psi(y, \dot{y}, \ldots, y^{(r+1)}) \end{aligned}$$

- *The components* y_i *of* y *are differentially independent in the sense that they satisfy no differential equation of the form* $\Phi(y, \dot{y}, y^{(k)}) = 0$*.*

The components of the vector y in the above definition are called **flat outputs**. In other words, once a suitable set of flat outputs is identified, the state x and input u of the system are completely determined as functions of the flat outputs and their derivatives. Specifically, for every sufficiently differentiable curve $t \to y(t)$, there corresponds a trajectory

$$t \to \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} = \begin{pmatrix} \phi(y(t), \dot{y}(t), \ldots, y^{(r)}(t)) \\ \psi(u(t), \dot{y}(t), \ldots, y^{(r+1)}(t)) \end{pmatrix}$$

The proof of this property is beyond the scope of this text. In the control field, the notion of differential flatness grew out of work on static and dynamic feedback linearization and is intimately related to these latter concepts.

Remark 14.5. *It is routine to show that any system that can be transformed to the Brunovsky form in Equation (12.35) is flat. This includes controllable linear systems as well as nonlinear systems that are feedback linearizable (Problem 14–13).*

For the problem of trajectory tracking, suppose that we want to find a feasible trajectory with initial and final conditions given as $(t_0, x(t_0), u(t_0))$, and $(t_f, x(t_f), u(t_f))$, respectively. Then, we can calculate corresponding values $(y(t_0), \dot{y}(t_0), \ldots, y^{(s+1)}(t_0))$ and $(y(t_f), \dot{y}(t_f), \ldots, y^{(s+1)}(t_f))$ for the flat outputs with $s \leq r$. We can then interpolate a smooth curve $t \to y(t)$ between these initial and final values, which then determines the trajectory $t \to (x(t), u(t))$ between t_0 and t_f. By construction, the computed trajectory automatically satisfies the system equations and hence any constraints, nonholonomic or otherwise.

Example 14.17. *For the DDR system, a set of flat outputs is given by* $y_1 = x$, $y_2 = y$. *The remaining variables are then*

$$\begin{aligned} \theta &= \text{Atan2}(\dot{y}_1, \dot{y}_2) + k\pi \\ v &= \pm\sqrt{\dot{y}_1^2 + \dot{y}_2^2} \\ \omega &= \frac{\ddot{y}_2\dot{y}_1 - \ddot{y}_1\dot{y}_2}{\dot{y}_1^2 + \dot{y}_2^2} \end{aligned}$$

Suppose that we want to move the DDR from configuration $(x, y, \theta) = (0, 0, 0)$ *at time* $t_0 = 0$ *to* $(x, y, \theta) = (10, 10, 0)$ *at time* $t_f = 1$. *Then, in terms of the flat outputs, the constraints are*

$$\begin{array}{ll} y_1(0) = 0 & y_2(0) = 0 \\ y_1(1) = 10 & y_2(1) = 10 \\ & \dot{y}_2(0) = 0 \\ & \dot{y}_2(1) = 0 \end{array}$$

where the last two constraints on $\dot{y}_2$ *indicate that the orientation* θ *of the DDR is horizontal at the beginning and end of the trajectory. Since there are two constraints on* y_1 *and four constraints on* y_2, *we can choose reference trajectories as polynomials of the form*

$$\begin{aligned} y_1(t) &= a_0 + a_1 t \\ y_2(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^3 \end{aligned}$$

Solving for the polynomial coefficients using the given constraints yields

$$\begin{aligned} y_1(t) &= t \\ y_2(t) &= 30t^2 - 20t^3 \end{aligned}$$

Figure 14.10 shows the resulting trajectory and control inputs.

In general, determining when a given system is flat and finding a set of flat outputs is difficult. However, it is known that the $(2, n)$ chain-form system is always flat with flat outputs $y_1 = z_1$ and $y_2 = z_n$.

Example 14.18. *For the chain-form system below, a set of flat outputs*

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \end{aligned}$$

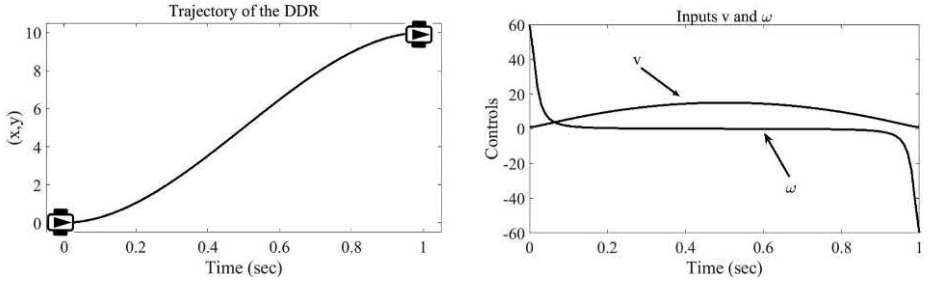


Figure 14.10: Trajectory and control inputs of the DDR computed from the flat outputs.

is $y_1 = z_1$ *and* $y_2 = z_3$*. Then it is straightforward to show that the remaining variables* z_2, v_1, v_2 *are given by*

$$\begin{aligned} z_2 &= \frac{\dot{y}_2}{\dot{y}_1} \\ v_1 &= \dot{y}_1 \\ v_2 &= \frac{\dot{y}_1 \ddot{y}_2 - \dot{y}_2 \ddot{y}_1}{\dot{y}_1^2} \end{aligned}$$

14.7 Feedback Control of Driftless Systems

In this section we consider the problem of finding feedback control laws, as opposed to the purely time-based, open-loop controllers that we derived in the previous section. We consider both the problem of pose regulation, i.e., point-to-point motion, and the problem of trajectory tracking. The first result below, known as Brockett's theorem, gives a necessary condition for the existence of a smooth stabilizing control.

14.7.1 Stabilizability

Theorem 14.3 (Brockett)**.** *Given the nonlinear system*

$$\dot{x} = f(x, u), \;\; f : \mathbb{R}^n \times U \to \mathbb{R}^n \tag{14.62}$$

with $f(x, u)$ *continuously differentiable and* $f(0, 0) = 0$*, suppose that there exists a continuously differentiable (smooth) feedback control law* $u = u(x)$ *that makes the origin* $x = 0$ *asymptotically stable. Then the image of the map* f *contains a neighborhood of the origin.*

Example 14.19. *For the system*

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= u_1 x_2 \end{aligned}$$

It is easily seen that, given any $a \neq 0$, there does not exist (x, u) satisfying

$$\begin{pmatrix} u_1 \\ u_2 \\ u_2 x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ a \end{pmatrix}$$

Thus, the image (range) of the function f does not contain a neighborhood of the origin and so there does not exist a smooth C^1 feedback control law $u(x)$ that stabilizes the origin.

Most nonholonomic systems, including the examples considered in this chapter, do not satisfy Brockett's necessary condition for the existence of a smooth stabilizing feedback control. As a result, Brockett's theorem has motivated a great deal of research into finding alternative control schemes for more general nonholonomic systems. These include

- Open-loop control laws
- Time-varying control laws $u = u(x, t)$
- Discontinuous and switching control laws

Example 14.20. *Consider the unicycle system. It is easy to see that for any initial condition x_0, y_0, θ_0, the following switching control scheme drives the system to the origin, i.e. is a stabilizing control:*

1. *Rotate the unicycle in place to line up the orientation with the origin*
2. *Translate until $(x, y) = (0, 0)$*
3. *Rotate in place until $\theta = 0$*

We note that if the nonlinear system $\dot{x} = f(x, u)$ is linearly controllable, then there is always a smooth feedback control law, in fact, of the form $u = -kx$, that stabilizes the origin locally.

Example 14.21. *Consider the Acrobot model*

$$\begin{aligned} m_{11}\ddot{q}_1 + m_{12}\ddot{q}_2 + h_1 + g_1 &= 0 \\ m_{21}\ddot{q}_1 + m_{22}\ddot{q}_2 + h_2 + g_2 &= u \end{aligned}$$

and suppose that $g_1 = 0$. From Proposition (13.1) we know that this system is not linearly controllable about the origin. It is left as an exercise (Problem 14–9) to show that there is no smooth feedback control that stabilizes the origin.

14.7.2 Nonsmooth Control

One way to circumvent the obstruction due to Brockett's theorem is to use discontinuous or switching control.

Sliding-Mode Control

The possibility of using discontinuous control to circumvent the obstruction imposed by Brockett's theorem suggests the application of **sliding-mode control**. Consider the chain-form system

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \end{aligned}$$

Define $\sigma = z_3 - z_1 z_2/2$ and let the controls v_1 and v_2 be given as

$$v_1 = -z_1 - z_2 \text{sign}(\sigma) \tag{14.63}$$
$$v_2 = -z_2 + z_1 \text{sign}(\sigma) \tag{14.64}$$

We want the manifold $\sigma = 0$ to serve as a sliding surface for the system trajectory. Let V be given as

$$V = \frac{1}{2} z_1^2 + \frac{1}{2} z_2^2 \tag{14.65}$$

Then

$$\dot{V} = z_1(-z_1 - z_2 \text{sign}(\sigma)) + z_2(-z_2 + z_1 \text{sign}(\sigma)) \tag{14.66}$$
$$= -z_1^2 - z_2^2 = -2V \tag{14.67}$$

Thus, $V(t) = V(0)e^{-2t} \to 0$ as $t \to \infty$ and therefore $z_1(t)$ and $z_2(t)$ converge asymptotically to zero. With the given choice of σ a straightforward calculation shows

$$\dot{\sigma} = -V \text{sign}(\sigma) \tag{14.68}$$

Therefore, $|\sigma(t)|$ is nonincreasing. If $\sigma(t)$ reaches the origin at some time $t = T$ it will remain at the origin and slide along the surface $z_3 = z_1 z_2/2$ toward the origin. Figures (14.11) and (14.12) shows a simulation of this controller.

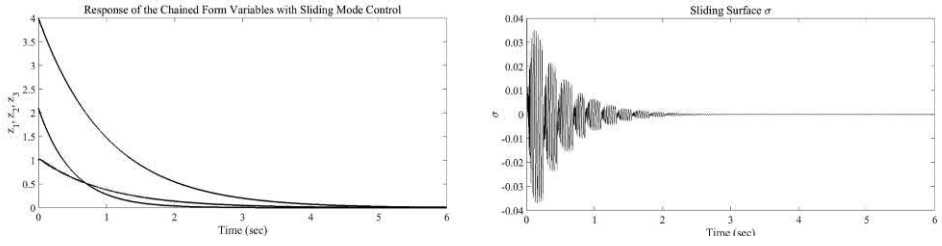


Figure 14.11: Sliding-mode control of the DDR in chain form: response of the chain variables and the sliding variable σ.

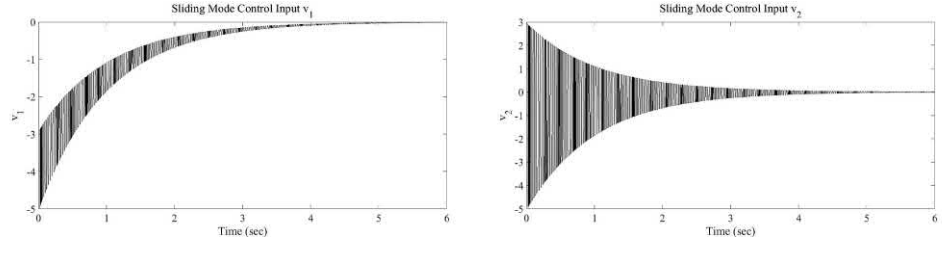


Figure 14.12: Sliding-mode control of the DDR in chain form: control inputs.

Dynamic Extension

For systems like the unicycle or DDR, it may be difficult to implement such a sliding-mode controller that requires rapid switching of the velocities v and ω. For this reason we may consider the dynamic extension

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \\ \dot{v}_1 &= u_1 \\ \dot{v}_2 &= u_2 \end{aligned}$$

For example, u_1 and u_2 could represent the torque inputs, which we can take as

$$u_1 = k(v_1^d - v_1); \qquad u_2 = k(v_2^d - v_2)$$

where we choose

$$\begin{aligned} v_1^d &= -z_1 - z_2 \text{sign}(\sigma) \\ v_2^d &= -z_2 + z_1 \text{sign}(\sigma) \end{aligned}$$

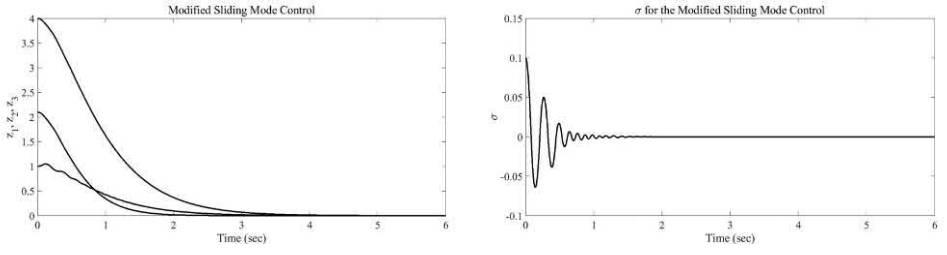


Figure 14.13: Modified sliding-mode control of the DDR in chain form: response of the chain variables and the sliding variable σ

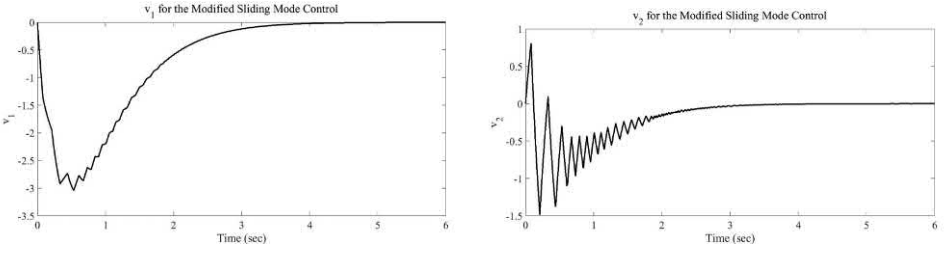


Figure 14.14: Modified sliding-mode control of the DDR in chain form: control inputs.

as reference trajectories for v_1 and v_2. This will have the effect of smoothing the control signals and is a more practical control from the standpoint of implementation. Figures (14.13) and (14.14) show the responses with the modified controls.

14.7.3 Trajectory Tracking

Lyapunov Design

In this section we illustrate the use of Lyapunov-like design for pose regulation of the DDR robot. Considering the DDR model

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= \omega \end{aligned} \tag{14.69}$$

let us first change to polar coordinates (r, ϕ) by setting

$$\begin{aligned} x &= r\cos(\phi) \\ y &= r\sin(\phi) \end{aligned}$$

and define $\beta = \phi - \theta$. In terms of coordinates r, β, and ϕ a straightforward computation gives

$$\begin{aligned} \dot{r} &= \cos(\beta)v \\ \dot{\beta} &= -\omega + \frac{1}{r}\sin(\beta)v \\ \dot{\phi} &= \frac{1}{r}\sin(\beta)v \end{aligned} \tag{14.70}$$

Note that Equation (14.70) is only valid on $R^3 - \{0\}$ because of the singularity at $r = 0$. Thus, the assumptions of Brockett's prohibiting the design of a smooth stabilizing control law do not hold. Define a candidate Lyapunov function

$$V = \frac{1}{2}r^2 + \frac{1}{2}\beta^2 + \frac{1}{2}\phi^2 \tag{14.71}$$

Then, $\dot{V}$ satisfies

$$\begin{aligned} \dot{V} &= r\dot{r} + \beta\dot{\beta} + \phi\dot{\phi} \\ &= r\cos(\beta)v + \beta(-\omega + \frac{1}{r}\sin(\beta)v) + \phi\frac{1}{r}\sin(\beta)v \\ &= r\cos(\beta)v + \beta(-\omega + \frac{1}{r}\sin(\beta)(\frac{\beta+\phi}{\beta}))v \end{aligned}$$

If we set the controls v and ω as

$$v = -\gamma\cos(\beta)r \tag{14.72}$$

$$\omega = k\beta - \gamma\frac{\cos(\beta)\sin(\beta)}{\beta}(\beta + \phi) \tag{14.73}$$

a straightforward calculation shows that

$$\dot{V} = -\gamma\cos^2(\beta)r^2 - k\beta^2 \leq 0 \tag{14.74}$$

Due to the singularity at $r = 0$, some additional care must be taken to conclude that the trajectory of the system converges asymptotically to the origin. We note that Equations (14.71) and (14.74) together imply that all solutions are bounded and that $\dot{V} \to 0$ asymptotically. Moreover, r does not reach zero in finite time due to the boundedness of β and the expression for $\dot{\beta}$. An application of Barbalat's Lemma (see Appendix C) now implies that all trajectories asymptotically converge to the origin.

Example 14.22. *Suppose we want to move the DDR from the initial pose* $(x, y, \theta) = (-1, 1, 3\pi/4)$ *to the origin. The corresponding initial conditions for the transformed coordinates* (r, β, ϕ) *are* $(\sqrt{2}, -\pi, -\pi/4)$. *Figure 14.15 shows the path computed using the above procedure.*

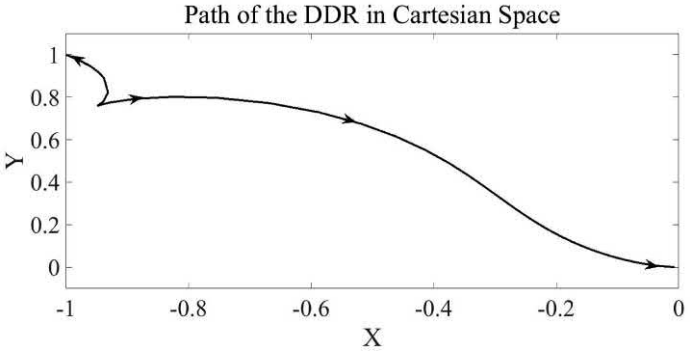


Figure 14.15: DDR pose regulation.

14.7.4 Feedback Linearization

In this section we present the application of feedback linearization for the problem of trajectory tracking for mobile robots. Thus, instead of pose regulation, we want the system configuration to follow a time-varying trajectory. With regard to static feedback linearization, the driftless system

$$\dot{q} = g_1(q)u_1 + \cdots + g_m(q)u_m = G(q)u,\ q \in \mathbb{R}^n \tag{14.75}$$

is feedback linearizable in a neighborhood U of q_0 if and only if rank $G(q) = n$ in U. This requires the system to be fully actuated, i.e. $m = n$, in which case the distribution

$$\Delta = \text{span}\{g_1(q), \ldots, g_n(q)\} \tag{14.76}$$

is trivially involutive and a feedback linearizing control is given by

$$u = G^{-1}(q)v \tag{14.77}$$

In the more interesting underactuated case $m < n$ we may apply the method of partial or input/output linearization to try to linearize m degrees of freedom. In this case, the problem is to determine a suitable output with respect to which a linear input/output relation can be established.

Example 14.23. *Consider again the differential drive robot*

$$\begin{aligned} \dot{x} &= \cos(\theta)v \\ \dot{y} &= \sin(\theta)v \\ \dot{\theta} &= \omega \end{aligned} \tag{14.78}$$

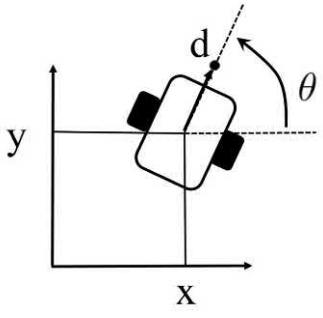


Figure 14.16: Differential drive robot showing the location of the output located d units ahead of the wheel axle.

and suppose that we choose the outputs

$$\begin{aligned} y_1 &= x + d\cos(\theta) && (14.79)\\ y_2 &= y + d\sin(\theta) && (14.80) \end{aligned}$$

with $d \neq 0$. The outputs are the coordinates of the point d shown in Figure 14.16. We can take d positive if it is located ahead of the wheel axle and negative otherwise. Differentiating the outputs gives

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = A(\theta) \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad (14.81)$$

The matrix A is invertible for $d \neq 0$ and the input transformation

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (14.82)$$

results in

$$\begin{aligned} \dot{y}_1 &= u_1 \\ \dot{y}_2 &= u_2 \end{aligned}$$

Using the expression for ω from Equation (14.82) yields the 3-dimensional normal form

$$\begin{aligned} \dot{y}_1 &= u_1 \\ \dot{y}_2 &= u_2 \\ \dot{\theta} &= (u_2\cos(\theta) - u_1\sin(\theta))/d \end{aligned} \qquad (14.83)$$

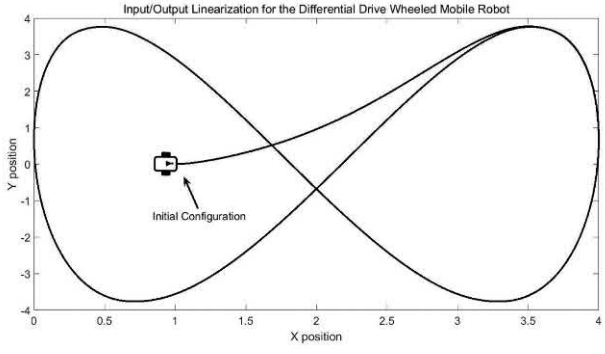


Figure 14.17: Trajectory of the differential drive robot with partial feedback linearization control.

Thus, given a desired output trajectory $(y_1^d(t), y_2^d(t))$*, which is really a desired trajectory for the point* d *to follow, the control input*

$$u_1 = \dot{y}_1^d + k_1(y_1^d - y_1) \tag{14.84}$$
$$u_2 = \dot{y}_2^d + k_2(y_2^d - y_2) \tag{14.85}$$

results in exponential tracking of the output trajectory. Note that the zero dynamics are given by the equation

$$\dot{\theta} = (\dot{y}_2^d \cos(\theta) - \dot{y}_1^d \sin(\theta))/d \tag{14.86}$$

Figure 14.17 shows a simulation of the above controller with reference trajectory

$$y_1 = 2 + 2\sin(\pi/8t)$$
$$y_2 = 2 + 2\sin(2\pi/8t)$$

The initial conditions were chosen as $(0, 0, 0)$ *with* $d = 0.1$*.*

Dynamic Feedback Linearization

The above method of input/output linearization fails for $d = 0$. To have the (x, y) coordinates track desired reference trajectories, we will use the method of **dynamic feedback linearization**, which we have so far not considered in any applications. To understand this design method, we consider the differential drive robot

$$\dot{x} = \cos(\theta)v$$
$$\dot{y} = \sin(\theta)v$$
$$\dot{\theta} = \omega$$

with output equations

$$\begin{aligned} y_1 &= x \\ y_2 &= y \end{aligned}$$

Differentiating the outputs gives

$$\dot{y}_1 = \cos(\theta)v \tag{14.87}$$
$$\dot{y}_2 = \sin(\theta)v \tag{14.88}$$

Differentiating the outputs a second time would result in the derivative of v appearing in the equations, which we want to avoid. Therefore, we let v be represented as a new state variable $\xi = v$. Then $\dot{\xi} = a$, the linear acceleration. Now, if we differentiate Equations (14.87)–(14.88) with respect to time we get

$$\begin{aligned} \begin{pmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} &= \begin{pmatrix} \dot{\xi}\cos(\theta) - \xi\dot{\theta}\sin(\theta) \\ \dot{\xi}\sin(\theta) + \xi\dot{\theta}\cos(\theta) \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta) & -\xi\sin(\theta) \\ \sin(\theta) & \xi\cos(\theta) \end{pmatrix} \begin{pmatrix} a \\ \omega \end{pmatrix} \end{aligned}$$

Thus, we can take

$$\begin{pmatrix} a \\ \omega \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\xi\sin(\theta) \\ \sin(\theta) & \xi\cos(\theta) \end{pmatrix}^{-1} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

which results in

$$\begin{aligned} \ddot{y}_1 &= u_1 \\ \ddot{y}_2 &= u_2 \end{aligned}$$

as long as $\xi \neq 0$. Given a desired output trajectory $(y_1^d(t), y_2^d(t))$ we can set

$$\begin{aligned} u_1 &= \ddot{y}_1^d + k_{p1}(y_1^d - y_1) + k_{d1}(\dot{y}_1^d - \dot{y}_1) \\ u_2 &= \ddot{y}_2^d + k_{p2}(y_2^d - y_2) + k_{d2}(\dot{y}_2^d - \dot{y}_2) \end{aligned}$$

and the tracking errors $e_i = y_i - y_i^d$ satisfy

$$\begin{aligned} \ddot{e}_1 + k_{d1}\dot{e}_1 + k_{p1}e_1 &= 0 \\ \ddot{e}_2 + k_{d2}\dot{e}_2 + k_{p2}e_2 &= 0 \end{aligned}$$

and exponentially convergence to zero. The resulting dynamic compensator is given by

$$\begin{aligned} \dot{\xi} &= u_1 \cos(\theta) + u_2 \sin(\theta) \\ v &= \xi \\ \omega &= (u_2 \cos(\theta) - u_1 \sin(\theta))/\xi \end{aligned} \tag{14.89}$$

Note that the angular velocity ω in Equation (14.89) is unbounded as the linear velocity ξ tends to zero. This means that the dynamic feedback linearization approach requires that the robot never stop moving and, hence, this method cannot be used for pose regulation.

14.8 Chapter Summary

This chapter provided an introduction to the control of nonholonomic systems with particular emphasis on wheeled mobile robots. We provided examples and case studies for several models including the **unicycle**, **differential drive robot** and a **car-like robot**. All of these systems are modeled as **driftless systems** of the form

$$\dot{q} = g_1(q)u_1 + \cdots + g_m(q)u_m = G(q)u$$

with n **configuration variables** and $m < n$ **control inputs**.

Nonholonomic Systems

We introduced the notion of nonholonomic systems, which has applications in mobile robots, hopping robots, gymnastic robots, and other systems that are subject to either rolling without slipping constraints or conservation of momentum constraints. The importance of nonholomomic constraints is that **controllability** is possible even though the systems are underactuated and not linearly controllable. The concept of **involutive closure** of the distribution $\Delta = \text{span } \{g_1, \ldots, g_m\}$ is the key concept to understand controllability of driftless systems.

Controllability and Stabilizability

We introduced two important control results, namely **Chow's theorem** and **Brockett's theorem**. Chow's theorem states that a driftless system is completely controllable if and only if the involutive closure $\bar{\Delta}$ of the above

distribution Δ has full rank n. Brockett's theorem states that, for the driftless systems considered here, there does not exist a continuously differentiable, time-invariant feedback control law to steer the system to the origin. Thus, one has to consider alternative control strategies, such as open-loop control, time-varying feedback control, or switching control.

Motion Planning

With regard to open-loop strategies to control driftless systems, we showed that most of the systems of interest can be transformed into **chained form**, which is a particularly useful canonical form for motion planning. We illustrated the idea of steering using sinusoids to generate feasible motions for stabilization. We also showed that the systems considered in this chapter are examples of systems that are **differentially flat**, which also simplifies the motion planning and trajectory generation problems.

Feedback Control

We presented several feedback control strategies for regulation and tracking, in particular **Lyapunov-based design**, **sliding-mode control**, **input/output linearization**, and **dynamic feedback linearization**.

Problems

14-1 Verify that the only solution for γ to the equations in Example 14.2 is $\gamma = 0$.

14-2 Verify that the constraint $\dot{q}_3 - q_1 q_3 \dot{q}_1 = 0$ in Equation (14.11) can be integrated to $q_3 = ke^{q_1^2/2}$ and $q_2 = \int ke^{q_1^2/2} dq_1$.

14-3 Fill in the details in Example 14.6 showing that the constraints are nonholonomic.

14-4 Fill in the details in Example 14.7 necessary to derive the vector fields g_1 and g_2 and show that the constraints are nonholonomic.

14-5 Verify the calculations given in Example 14.12 that transform the DDR model into chain form.

14-6 Show by induction that the $(2, n)$ chained-form system in Equation (14.55) is controllable.

14-7 Compute the distributions Δ_0, Δ_1, and Δ_2 in Example 14.15 and verify that they are of constant rank and involutive.

14-8 Carry out the details to verify the claim in Example 14.14 that the variables z_1 and z_2 return to their initial values after $2\pi/\omega$ seconds and the change in z_3 is $ab\pi/\omega^2$.

14-9 Show, using Brockett's theorem, that there does not exist a smooth stabilizing feedback control for the Acrobot model in Example 14.21 if $g_1 = 0$.

14-10 Verify the calculations in Example 14.10.

14-11 Consider the two-link planar RR robot

$$\begin{aligned} m_{11}\ddot{q}_1 + m_{12}\ddot{q}_2 + h_1 + g_1 &= u_1 \\ m_{21}\ddot{q}_1 + m_{22}\ddot{q}_2 + h_2 + g_2 &= u_2 \end{aligned}$$

Investigate the existence of a smooth stabilizing control law under various scenarios regarding u_1, u_2, g_1, g_2.

14-12 Use Chow's theorem to show that the system defined by Equation (14.44) is controllable.

14-13 Let $\dot{x} = f(x) + g(x)u$ be a single-input nonlinear system and suppose that the system is feedback linearizable with a nonlinear change of coordinates and nonlinear feedback. Show that the system is differentially flat.

Notes and References

A more complete treatment of the control of nonholonomic systems, including mobile robots, can be found in [118] and [32]. An excellent introduction to the topics here is [100], which gives examples of underactuated planar manipulators in addition to wheeled mobile robots. Another good tutorial on the topic of nonholonomic control is [80]. Additional references on nonholonomic behavior and control of planar manipulators are [130, 101, 102].

Our treatment of systems in chained form follows closely the development in [118] and [100]. The treatment of steering using sinusoids follows [121] and [118]. A proof of Brockett's theorem is in [18]. An excellent treatment of switching control is [95]. Additional details on differential flatness and its application to nonholonomic motion planning is in [120, 141]. The

Lyapunov-like approach to control of the unicycle using polar coordinates is taken from [3]. More details on the concept of dynamic extension can be found in [13, 80]. Dynamic feedback linearization is treated in [99, 101, 22].

Appendix A

TRIGONOMETRY

A.1 The Two-Argument Arctangent Function

The usual inverse tangent function returns an angle in the range $(-\pi/2, \pi/2)$. In order to express the full range of angles we will find it useful to define the so-called **two-argument arctangent function**, $\text{Atan2}(x, y)$, which is defined for all $(x, y) \neq (0, 0)$ and equals the unique angle $\theta \in [-\pi, \pi]$ such that

$$\cos\theta = \frac{x}{(x^2 + y^2)^{\frac{1}{2}}} , \qquad \sin\theta = \frac{y}{(x^2 + y^2)^{\frac{1}{2}}} \tag{A.1}$$

This function uses the signs of x and y to select the appropriate quadrant for the angle θ. For example, $\text{Atan2}(1, -1) = -\frac{\pi}{4}$, while $\text{Atan2}(-1, 1) = +\frac{3\pi}{4}$. If both x and y are zero, then Atan2 is undefined.

A.2 Useful Trigonometric Formulas

Below is a list of trigonometric identities that are used to derive various expressions related to forward, inverse, and velocity kinematics.

Pythagorean Identities

$$\sin^2\theta + \cos^2\theta = 1 , \quad 1 + \tan^2\theta = \sec^2\theta , \quad 1 + \cot^2\theta = \csc^2\theta$$

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

Reduction Formulas

$$\begin{array}{ll} \sin(-\theta) = -\sin\theta, & \sin\theta = \cos(\frac{\pi}{2} - \theta) \\ \cos(-\theta) = \cos\theta, & \cos\theta = \sin(\frac{\pi}{2} - \theta) \\ \tan(-\theta) = -\tan\theta, & \tan\theta = \cot(\frac{\pi}{2} - \theta) \end{array}$$

Sum-Difference Identities

$$\begin{aligned} \sin(\alpha \pm \beta) &= \sin\alpha\cos\beta \pm \cos\alpha\sin\beta \\ \cos(\alpha \pm \beta) &= \cos\alpha\cos\beta \mp \sin\alpha\sin\beta \\ \tan(\alpha \pm \beta) &= \frac{\tan\alpha \pm \tan\beta}{1 \mp \tan\alpha\tan\beta} \end{aligned}$$

Double-Angle Identities

$$\sin 2\theta = 2\sin\theta\cos\theta, \qquad \cos 2\theta = 2\cos^2\theta - 1, \qquad \tan 2\theta = \frac{2\tan\theta}{1 - \tan^2\theta}$$

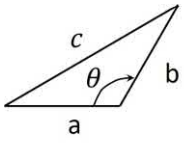
Half-Angle Identities

$$\sin\frac{\theta}{2} = \pm\sqrt{\frac{1-\cos\theta}{2}}, \quad \cos\frac{\theta}{2} = \pm\sqrt{\frac{1+\cos\theta}{2}}, \quad \tan\frac{\theta}{2} = \pm\sqrt{\frac{1-\cos\theta}{1+\cos\theta}}$$

Law of Cosines

If a triangle has sides of length a, b, and c, and θ is the angle opposite the side of length c, then

$$c^2 = a^2 + b^2 - 2ab\cos\theta$$



Appendix B

LINEAR ALGEBRA

We assume that the reader has some familiarity with basic properties of vectors and matrices, such as matrix addition, multiplication, and determinants. For additional background, see [9, 60].

The symbol $\mathbb{R}$ denotes the set of real numbers and $\mathbb{R}^n$ denotes the set of n-tuples of real numbers. We use lower case letters a, b, c, x, y, etc., to denote scalars in $\mathbb{R}$ and vectors in $\mathbb{R}^n$. Uppercase letters A, B, C, M, R, etc., denote matrices.

B.1 Vectors

Vectors in $\mathbb{R}^n$ are defined as column vectors

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \text{ with } x_i \in \mathbb{R},\ i = 1, \ldots, n$$

The entries $x_1, \ldots, x_n$ are called **components** or **coordinates** of x. The **transpose** of a vector x, denoted x^T, is the row vector

$$x^T = [x_1, \ldots, x_n]$$

In this text we will often drop the superscript T when the vector x is not part of a displayed equation and simply write

$$x = [x_1, \ldots, x_n] \quad \text{or} \quad x = (x_1, \ldots, x_n)$$

to denote x as a row instead of a column. This is done simply to improve the readability of the text where convenient and the reader should bear in

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

mind that vectors are, by definition, columns. Also, since we do not use bold font or arrows to denote vectors, the difference between, for example, x_i as a vector and x_i as a component of a vector x should be clear from the context.

Linear Independence

A set of vectors $\{x_1, \ldots, x_m\}$ is said to be **linearly independent** if and only if, for arbitrary scalars $\alpha_i \in \mathbb{R}$,

$$\sum_{i=1}^{m} \alpha_i x_i = 0 \text{ implies } \alpha_i = 0 \text{ for all } i$$

Otherwise, the vectors $x_1, \ldots, x_m$ are said to be **linearly dependent**.

Remark B.1. *It is easy to show that a set of vectors $x_1, \ldots, x_m$ is linearly dependent if and only if some x_k, for $2 \leq k \leq m$, is a linear combination of the preceding vectors $x_1, \ldots, x_{k-1}$.*

A **basis** of a real vector space is a linearly independent set of vectors $\{e_1, \ldots, e_m\}$ such that every vector x can be written as a linear combination

$$x = x_1 e_1 + \cdots + x_m e_m, \quad x_i \in \mathbb{R}, \ i = 1, \ldots, m$$

The coordinate representation $x = (x_1, \ldots, x_m)$ is uniquely determined by the particular basis $\{e_1, \ldots, e_m\}$. The **dimension** of the vector space is the number of vectors in any basis.

Subspaces

A set of vectors $\mathcal{M}$ contained in a vector space $\mathcal{V}$ is a **subspace** of $\mathcal{V}$ if $\alpha x + \beta y$ belongs to $\mathcal{M}$ for every x and y in $\mathcal{M}$ and every α and β in $\mathbb{R}$. In other words $\mathcal{M}$ is closed under the operations of addition and scalar multiplication. A subspace $\mathcal{M}$ is itself a vector space. The dimension of $\mathcal{M}$ is less than or equal to the dimension of $\mathcal{V}$.

B.2 Inner Product Spaces

An **inner product** in a (real) vector space is a scalar-valued, bilinear function $< \cdot, \cdot >$ of pairs of vectors x and y such that

- $< x, y >=< y, x >$

- $< \alpha_1 x + \alpha_2 y, z >= \alpha_1 < x, z > + \alpha_2 < y, z >$ where a_1 and a_2 are scalars
- $< x, x > \geq 0$ and $< x, x > = 0$ if and only if $x = 0$

An **inner product space** is a vector space with an inner product. An inner product induces a **norm** $\|x\|$ on an inner product space via the formula

$$\|x\| = \sqrt{< x, x >}$$

The norm generalizes the usual notion of **length** of a vector. Some useful properties of the inner product and norm include

Cauchy–Schwartz inequality: $| < x, y > | \leq \|x\| \ \|y\|$ with equality if and only if x and y are linearly dependent.

homogeneity: $\|\alpha x\| = |\alpha| \|x\|$ for x a vector and α a scalar.

triangle inequality: $\|x + y\| \leq \|x\| \ + \|y\|$

parallelogram law: $\|x + y\|^2 + \|x - y\|^2 = 2\|x\|^2 + 2\|y\|^2$

Euclidean Space

The Euclidean space $\mathbb{R}^n$ is an inner product space with inner product defined as

$$< x, y >= x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n \tag{B.1}$$

The inner product (B.1) on $\mathbb{R}^n$ is also denoted by $x \cdot y$ and called the **dot product** or **scalar product**. For vectors in $\mathbb{R}^2$ or $\mathbb{R}^3$ the scalar product can be expressed as

$$x^T y = \|x\| \ \|y\| \cos\theta$$

where θ is the angle between the vectors x and y.
We will use i, j, and k to denote the standard unit vectors in $\mathbb{R}^3$

$$i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad j = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Using this notation, a vector $x = (x_1, x_2, x_3)$ may be written as

$$x \ = \ x_1 i + x_2 j + x_3 k$$

and the components of x are equal to the dot products

$$x_1 = x \cdot i, \quad x_2 = x \cdot j, \quad x_3 = x \cdot k$$

Orthogonal Complement

If W is a subspace of $\mathbb{R}^n$, the **orthogonal complement** of W in $\mathbb{R}^n$ is the subspace

$$W^\perp = \{v \in \mathbb{R}^n \mid v^T w = 0 \text{ for all } w \in W\}$$

The symbol $W^\perp$ is read "W perp". Thus $W^\perp$ is the set of all vectors $v \in \mathbb{R}^n$ that are orthogonal to all vectors $w \in W$. The orthogonal complement of a subspace W in $\mathbb{R}^n$ satisfies the following:

- $W^\perp$ is also a subspace of $\mathbb{R}^n$
- $(W^\perp)^\perp = W$
- $\dim(W) + \dim(W^\perp) = n$

B.3 Matrices

Matrices and matrix representations of linear transformations are fundamental to most of the concepts in this text, from kinematics and dynamics, to control. A **linear transformation** A on a vector space $\mathcal{V}$ is a function $A : \mathcal{V} \to \mathcal{V}$ such that

$$A(\alpha x_1 + \beta x_2) = \alpha A(x_1) + \beta A(x_2), \text{ for } x_1, x_2 \in \mathcal{V} \text{ and } \alpha, \beta \in \mathbb{R}$$

We will denote $A(x)$ by Ax. We represent linear transformations in coordinate form as matrices by taking a particular basis for $\mathcal{V}$.

An $m \times n$ matrix $A = (a_{ij})$ over $\mathbb{R}$ is an ordered array of real numbers with m row vectors $(a_{i1}, \ldots, a_{in})$ for $i = 1, \ldots, m$, and n column vectors $(a_{1j}, \ldots, a_{mj})$, for $j = 1, \ldots, n$ written as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{B.2}$$

The matrix A is a representation of a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$. If $n = m$, i.e., if the number of rows and the number of columns are equal, then the matrix A is said to be **square**. The set of $n \times n$ square matrices, which we denote by $\mathbb{R}^{n \times n}$, is itself a vector space of dimension n^2.

The **rank** of a matrix A is the largest number of linearly independent rows (or columns) of A. Thus, the rank of an $m \times n$ matrix can be no greater than the minimum of m and n.

The **transpose** of a matrix A is denoted A^T and is formed by interchanging rows and columns of A.

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} \tag{B.3}$$

Thus A^T is an $n \times m$ matrix. Some properties of the matrix transpose are

- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A + B)^T = A^T + B^T$

A square $n \times n$ matrix A is said to be

- **symmetric** if $A^T = A$
- **skew-symmetric** if $A^T = -A$
- **orthogonal** if $A^T A = AA^T = I$, where I is the $n \times n$ identity matrix

Matrix Trace

The **trace** of an $n \times n$ matrix A, denoted $Tr(A)$ is the sum of the diagonal entries of A. Thus, if $A = (a_{ij})$, then $Tr(A) = \sum_{i=1}^{n} a_{ii} = a_{11} + \cdots + a_{nn}$.

The Determinant

A **determinant** is a function that assigns a particular scalar to a linear transformation (matrix) $A \in \mathbb{R}^{n \times n}$. For our purposes we may define the determinant, $\det(A)$, of a square matrix A via the recursive formula that holds for any $i = 1, \ldots, n$

$$\det(A) = \sum_{j=1}^{n} a_{ij}(-1)^{i+j} \det(C_{ij})$$

where C_{ij} is the $(n-1) \times (n-1)$ matrix formed by deleting the i-th row and j-th column of A. The matrix determinant satisfies the following properties

- $\det(AB) = \det(A) \det(B)$
- $\det(\alpha A) = \alpha^n \det(A)$
- $\det(A^T) = \det(A)$

If $\det(A) = 0$, the matrix A is called **singular**; otherwise A is **nonsingular** or **invertible**. The **inverse** of a square matrix $A \in \mathbb{R}^{n\times n}$ is a matrix $B \in \mathbb{R}^{n\times n}$ satisfying

$$AB = BA = I$$

where I is the $n \times n$ identity matrix. We denote the inverse of A by A^{-1}. The inverse of a matrix A exists and is unique if and only if A has rank n, equivalently, if and only if the determinant $\det(A)$ is nonzero. The matrix inverse satisfies the following

- $(A^{-1})^{-1} = A$
- $(AB)^{-1} = B^{-1}A^{-1}$

The set of $n \times n$ nonsingular matrices over $\mathbb{R}$ is denoted $GL(n)$, the **general linear group** of order n. Note that $GL(n)$ is not a vector subspace of $\mathbb{R}^{n\times n}$ since, for example, the sum of two invertible matrices is not necessarily invertible.

B.4 Eigenvalues and Eigenvectors

The **eigenvalues** of a matrix A are the solutions in λ of the equation

$$\det(\lambda I - A) \quad = \quad 0$$

The function $\det(\lambda I - A)$ is a polynomial of degree n in λ, called the **characteristic polynomial** of A. If λ_e is an eigenvalue of A, an eigenvector of A corresponding to λ_e is a nonzero vector x_e satisfying the system of linear equations

$$Ax_e \quad = \quad \lambda_e x_e$$

Similarity Transformation

If T is an $n \times n$ nonsingular matrix, then

$$\bar{A} = T^{-1}AT \tag{B.4}$$

is called a similarity transformation. Since T is nonsingular, the column vectors of T are linearly independent, and hence form a basis of $R^{n\times n}$. For this reason, (B.4) is also called a **change of basis**. The matrix $\bar{A}$ represents the same linear transformation as A in the basis defined by T.

Diagonalizing a Symmetric Matrix

If A is a symmetric matrix, then

- its eigenvalues $\lambda_1, \ldots, \lambda_n$ are real
- eigenvectors corresponding to distinct eigenvalues are orthogonal
- there exists an $n \times n$ orthogonal matrix T such that

$$\bar{A} = T^{-1}AT = \text{diag}[\lambda_1, \ldots, \lambda_n]$$

  where diag[ ] is a diagonal matrix.

The columns of the matrix T consist of (a basis of) eigenvectors corresponding to the respective eigenvalues $\lambda_1, \ldots, \lambda_n$.

Quadratic Forms

Definition B.1. *A* **quadratic form** *on $\mathbb{R}^n$ is a scalar function*

$$V(x) = x^T Px = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}$$

where $P = (p_{ij})$ is an $n \times n$ symmetric matrix. The function V, equivalently, the matrix P, is **positive definite** *if*

- *$x^T Px > 0$ for all $x \neq 0$*
- *all eigenvalues of P are positive*
- *all principal minors or principal minor determinants of P are positive.*

If P is positive definite, then we have the useful bounds

$$\lambda_{min}(P)\|x\|^2 \leq x^T Px \leq \lambda_{max}(P)\|x\|^2 \tag{B.5}$$

where $\lambda_{min}(P)$ and $\lambda_{max}(P)$ are, respectively, the minimum and maximum eigenvalues of P.

Range and Null Space

The **range space**, $\mathcal{R}(A)$, of an $m \times n$ matrix A is the subspace of $\mathbb{R}^m$ defined by

$$\mathcal{R}(A) = \{y \in \mathbb{R}^m : \ y = Ax, \ \text{for some } x \in \mathbb{R}^n\}$$

The **null space**, $\mathcal{N}(A)$, of an $m \times n$ matrix A is the subspace of $\mathbb{R}^n$ defined as

$$\mathcal{N}(A) = \{x \in \mathbb{R}^n : Ax = 0\}$$

The range and null spaces of a matrix are related according to

$$\begin{aligned} \mathcal{N}(A) &= \mathcal{R}(A^T)^\perp \\ \mathcal{R}(A) &= \mathcal{N}(A^T)^\perp \end{aligned}$$

An important property of the null space is that

$$\dim\mathcal{R}(A) + \dim\mathcal{N}(A) = n$$

An $n \times n$ matrix is invertible if and only if the nullspace consists of only the zero vector, that is, $Ax = 0$ implies $x = 0$.

Vector Product

The **vector product** or **cross product** $x \times y$ of two vectors x and y belonging to $\mathbb{R}^3$ is a vector c defined by

$$\begin{aligned} c &= x \times y = \det \begin{bmatrix} i & j & k \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \\ &= (x_2y_3 - x_3y_2)i + (x_3y_1 - x_1y_3)j + (x_1y_2 - x_2y_1)k \end{aligned}$$

The cross product is a vector whose magnitude is

$$\|c\| = \|x\|\,\|y\|\sin(\theta)$$

where $0 \leq \theta \leq \pi$ is the angle between x and y in the plane containing them, and whose direction is given by the right hand rule shown in Figure B.1.

A right-handed coordinate frame x–y–z is a coordinate frame with axes mutually perpendicular and that also satisfies the right hand rule, in the sense that $k = i \times j$, where i, j, and k are unit vectors along the x, y, and z-axes, respectively. We can remember the right hand rule as being the direction of advancement of a right-handed screw rotated from the positive x-axis into the positive y-axis through the smallest angle between the axes. The cross product has the properties

$$\begin{aligned} x \times y &= -y \times x \\ x \times (y + z) &= (x \times y) + (x \times z) \\ \alpha(x \times y) &= (\alpha x) \times y = x \times (\alpha y) \end{aligned}$$

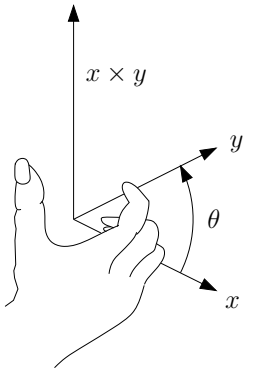


Figure B.1: The right hand rule.

The cross product is not associative, but satisfies the **Jacobi identity**

$$x \times (y \times z) + y \times (z \times x) + z \times (x \times y) = 0$$

Lagrange's formula relates the cross product and inner product according to

$$x \times (y \times z) = (x \cdot z)y - (x \cdot y)z$$

The **outer product** of two vectors x and y belonging to $\mathbb{R}^n$ is an $n \times n$ matrix defined by

$$xy^T = \begin{bmatrix} x_1y_1 & \dots & x_1y_n \\ x_2y_1 & \dots & x_2y_n \\ \vdots & \dots & \vdots \\ x_ny_1 & \dots & x_ny_n \end{bmatrix}$$

The scalar product and the outer product are related by

$$x^Ty = Tr(xy^T)$$

B.5 Differentiation of Vectors

Suppose that the vector $x(t) = (x_1(t), \dots, x_n(t))$ is a function of time. Then the time derivative $\dot{x} = \frac{dx}{dt}$ of x is the vector

$$\dot{x} = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix}$$

Similarly, the derivative $\frac{dA}{dt}$ of a matrix $A = (a_{ij})$ is the matrix $(\frac{da_{ij}}{dt})$. Similar statements hold for integration of vectors and matrices. The scalar and vector products satisfy the following product rules for differentiation similar to the product rule for differentiation of ordinary functions:

$$\begin{aligned} \frac{d}{dt}x^T y &= \frac{dx}{dt}^T y + x^T \frac{dy}{dt} \\ \frac{d}{dt}(x \times y) &= \frac{dx}{dt} \times y + x \times \frac{dy}{dt} \end{aligned}$$

B.6 The Matrix Exponential

Given an $n \times n$ matrix M, we define the **exponential**, e^M, of M using the series expansion

$$e^M = I + M + \frac{1}{2!}M^2 + \frac{1}{3!}M^3 + \cdots \tag{B.6}$$

The above series expansion converges for any square matrix M and so is well defined. The matrix exponential satisfies the following properties:

- $e^0 = I$ where 0 is the $n \times n$ zero matrix and I is the $n \times n$ identity matrix
- $(e^M)^T = e^{M^T}$
- $e^M e^N = e^{M+N}$ if $MN = NM$, i.e., if M and N commute
- If T is a nonsingular $n \times n$ matrix, then $T^{-1}e^M T = e^{T^{-1}MT}$
- $\det(e^M) = e^{Tr(M)}$

B.7 Lie Groups and Lie Algebras

Definition B.2 (Lie Group)**.** *A Lie group is a group that is also a differentiable manifold, such that the group operations are smooth.*

The most relevant examples of Lie groups for our purposes are:

- $GL_n(R)$, the **general linear group** over $\mathbb{R}$
- $SO(3)$, the **rotation group**
- $SE(3)$, the **Euclidean group of rigid motions**

Definition B.3 (Lie Algebra)**.** *A Lie algebra is a vector space together with a non-associative, alternating bilinear map, $(x, y) \mapsto [x, y]$, called the* **Lie bracket***, satisfying the following*

- $[x, x] = 0$ — *Alternativity*
- $[x_1, [x_2, x_3]] + [x_3, [x_1, x_2]] + [x_2, [x_3, x_1]]$ — *The Jacobi identity*

Note that bilinearity and alternatively together imply

- $[x_1, x_2] = -[x_2, x_1]$ — *anticommutativity*

Examples of Lie algebras include:

- $\mathbb{R}^3$ with $[x_1, x_2] = x_1 \times x_2$, the vector cross product
- quaternions, with $[i, j] = ij - ji = 2k; [j, k] = jk - kj = 2i; [k, i] = ki - ik = 2j$
- $so(3)$, the vector space of skew-symmetric matrices with $[S_1, S_2] = S_1S_2 - S_2S_1$, the matrix commutator. Note that it is straightforward to show that $[S_1, S_2]$ is again a skew-symmetric matrix.
- vector fields on a differentiable manifold with $[f, g](x) = \frac{\partial g}{\partial x} f(x) - \frac{\partial f}{\partial x} g(x)$, where $\frac{\partial f}{\partial x}$ and $\frac{\partial g}{\partial x}$ are the Jacobians of f and g, respectively.
- $se(3)$, the vector space of **twists**.

We note that a twist is defined as follows. If

$$H(t) = \begin{bmatrix} R(t) & d(t) \\ 0 & 1 \end{bmatrix}$$

is a homogeneous transformation matrix defining a rigid motion in $\mathbb{R}^3$, the **twist** associated with H is the 4×4 matrix

$$\mathcal{S} = \begin{bmatrix} S(\omega) & d \times \omega + v \\ 0 & 0 \end{bmatrix}$$

where v and ω represent the linear and angular velocities associated with the rigid motion.

The Lie Algebra of a Lie Group

Lie groups and Lie algebras are related by the exponential map. The relation between Lie groups and Lie algebras allows one to study geometric properties of Lie groups via algebraic properties of Lie algebras.

- $SO(3)$ and $so(3)$: every rotation matrix R in $SO(3)$ can be expressed as e^S for some skew-symmetric matrix S in $so(3)$.
- $SE(3)$ and $se(3)$: every homogeneous matrix H in $SE(3)$ can be expressed as $e^{\mathcal{S}}$, where $\mathcal{S}$ is a twist in $se(3)$.

B.8 Matrix Pseudoinverse

The inverse of a matrix A is only defined if A is a square matrix. If A is an $m \times n$ matrix, with $n \neq m$, we may define a so-called **pseudoinverse** of A. We assume that A has full rank $r = \min(m, n)$ and we distinguish between the two cases, $m < n$ and $m > n$.

If $m < n$, then A is a **fat matrix**, meaning it has more columns than rows. The **right pseudoinverse** of A is defined as the $n \times m$ matrix

$$A^\dagger = A^T(AA^T)^{-1}$$

In this case AA^T is $m \times m$ with full rank m and $AA^\dagger = AA^T(AA^T)^{-1} = I_{m\times m}$. Given a vector $b \in \mathbb{R}^m$, the general solution of the equation

$$Ax = b \tag{B.7}$$

is given by $x = A^\dagger b + (I - A^\dagger A)w$, where $w \in \mathbb{R}^m$ is an arbitrary vector. The vector $x = A^\dagger b$, i.e., with $w = 0$, gives the minimum norm solution $\|x\|$ of Equation (B.7).

In the case $m > n$, then A is a **tall matrix**, meaning it has more rows than columns. The **left pseudoinverse** of A is defined as the $n \times m$

$$A^\dagger = (A^T A)^{-1}A^T$$

In this case $A^T A$ is $n \times n$ with full rank n and $A^\dagger A = (A^T A)^{-1}A^T A = I_{n\times n}$. In this case $A^\dagger$ is the **least squares solution** of $y = Ax$, i.e., the solution that minimizes the norm $\|Ax - y\|$.

B.9 Schur Complement

Let M be a $(p + q) \times (p + q)$ matrix partitioned into sub-blocks as

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

in which A, B, C, and D are, respectively, $p \times p$, $p \times q$, $q \times p$, and $q \times q$ sub-matrices. Assuming that D is invertible, the **Schur complement** of the block D in the matrix M is the $p \times p$ matrix

$$M/D = A - BD^{-1}C$$

In the case that D above is singular, a generalized inverse of D can be used in place of the inverse to define a generalized Schur complement. Some properties of the Schur complement include:

- If M is a symmetric, positive definite matrix, then so is the Schur complement of D in M
- The determinant of M is $\det(D)\det(A - BD^{-1}C)$
- The rank of M is equal to $\text{rank}(D) + \text{rank}(A - BD^{-1}C)$

The Schur complement is useful for solving systems of equations of the form

$$\begin{aligned} Ax + By &= a \\ Cx + Dy &= b \end{aligned}$$

and will be useful for the study of underactuated systems in Chapter 13.

B.10 Singular Value Decomposition (SVD)

For square matrices, we can use tools such as the determinant, eigenvalues, and eigenvectors to analyze their properties. However, for nonsquare matrices these tools simply do not apply. Their generalizations are captured by the **singular value decomposition (SVD)**.

As we described above, for $A \in \mathbb{R}^{m\times n}$, we have $AA^T \in \mathbb{R}^{m\times m}$. It is easily seen that AA^T is symmetric and positive semi-definite, and, therefore, has real and nonnegative eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$. The **singular values** for the matrix A are given by the square roots of the eigenvalues of AA^T,

$$\sigma_i = \sqrt{\lambda_i},\ i = 1, \ldots, m \tag{B.8}$$

The **singular value decomposition** (SVD) of the matrix A is then given by

$$A = U\Sigma V^T \tag{B.9}$$

in which

$$U = [u_1, u_2, \ldots, u_m],\ V = [v_1, v_2, \ldots, v_n] \tag{B.10}$$

are orthogonal matrices of dimensions $m \times m$ and $n \times n$, respectively, and $\Sigma \in R^{m\times n}$ is given by

$$\Sigma = \left[\begin{array}{cccc|c} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & 0 \\ & & & \sigma_m & \end{array}\right] \tag{B.11}$$

We can compute the singular value decomposition of the matrix A as follows. We begin by finding the singular values σ_i of A, which can then be used to find eigenvectors $u_1, \cdots, u_m$ that satisfy

$$AA^T u_i = \sigma_i^2 u_i \tag{B.12}$$

These eigenvectors comprise the columns of the matrix $U = [u_1, u_2, \ldots, u_m]$. The system of equations (B.12) can be written as

$$AA^T U = U\Sigma_m^2 \tag{B.13}$$

where the matrix Σ_m is defined as

$$\Sigma_m = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \end{bmatrix}$$

Now, define

$$V_m = A^T U \Sigma_m^{-1} \tag{B.14}$$

and let V be any orthogonal matrix that satisfies $V = [V_m \mid V_{n-m}]$ (note that here V_{n-m} contains just enough columns so that the matrix V is an $n \times n$ matrix). It is a simple matter to combine the above equations to verify Equation (B.9).

Appendix C

LYAPUNOV STABILITY

We give here some basic results on stability theory for nonlinear systems. For simplicity we treat only time-invariant systems. For a more general treatment of the subject the reader is referred to [180]. We first need a few definitions from real analysis regarding continuity and differentiability of functions.

C.1 Continuity and Differentiability

Definition C.1 (Continuous function). *Let $\mathcal{U} \in \mathbb{R}$ be an open subset of $\mathbb{R}$. A function $f : \mathcal{U} \to \mathbb{R}$ is continuous at a point $x_0 \in \mathcal{U}$ if, for all $\epsilon > 0$, there exists a $\delta > 0$ such that, if $|x - x_0| < \delta$ (and $x \in \mathcal{U}$), then $|f(x) - f(x_0)| < \epsilon$.*

An alternative characterization of continuity is that $f : \mathbb{R} \to \mathbb{R}$ is continuous at $x = x_0$ if $f(x) \to f(x_0)$ as $x \to x_0$.

Definition C.2 (Uniform continuity). *A function $f : \mathcal{U} \to \mathbb{R}$ is uniformly continuous on $\mathcal{U}$ if for every $\epsilon > 0$ there exists a $\delta > 0$ such that $|x - y| < \delta$ implies $|f(x) - f(y)| < \epsilon$.*

Continuity of a function is a property that holds at a point. Uniform continuity, on the other hand, is a global property on $\mathcal{U}$.

Definition C.3 (Derivative). *A function $f : \mathcal{U} \to \mathbb{R}$ is differentiable at $x \in \mathcal{U}$ if*

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

exists and is finite.

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

A function of several variables, $f : \mathcal{U} \to \mathbb{R}$, where $\mathcal{U} \in \mathbb{R}^n$, is differentiable if the **partial derivatives** *exist and are finite, where the partial derivatives are defined for $k = 1, \ldots, n$ as*

$$\frac{\partial f(x)}{\partial x_k} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_k + h, \ldots, x_n) - f(x_1, \ldots, x_n)}{h}$$

We say that a function is **continuously differentiable** if the derivative function exists and is continuous.

Since the derivative $\frac{df}{dx}$ is itself a function, we can ask if the derivative function is continuous, differentiable, and so on. Higher derivatives are denoted as $\frac{d^k f}{dk^k}, k = 2, 3, \ldots$, for functions of a single variable.

For partial derivatives, the order in which higher derivatives are computed is important. For example, if f is a function of 3-variables, we may compute a second partial derivative with respect to any of the variables, in any order, i.e.

$$\frac{\partial^2 f}{\partial x_1^2}, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2}, \quad \frac{\partial^2 f}{\partial x_3 \partial x_2}$$

and so on. These are called **mixed partials**. Note that equality of the mixed partials, such as,

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_1}$$

is true if the derivatives themselves are continuous, i.e. if f is at least twice continuously differentiable.

In order not to have to specify the precise continuity properties of a function each time we use it, we will often use the term **smooth function** to mean a function that is continuously differentiable as many times as needed in a particular context.

Gradient, Hessian, and Jacobian

Given a smooth scalar function $f : \mathbb{R}^n \to \mathbb{R}$, we define the **differential** of f, denoted by df, as

$$df = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

The vector function

$$\nabla f = df^T = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

is called the **gradient** of f. The gradient of a scalar function is orthogonal to the level curves of the function and points in the direction of maximum increase of the function.

The $n \times n$ matrix of second derivatives of the function f

$$H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

is called the **Hessian** of f. The Hessian matrix is symmetric if the entries are continuous and describes the local curvature of the function f.

Let $f(x) = (f_1(x), \ldots, f_m(x)) : \mathbb{R}^n \to \mathbb{R}^m$ be a smooth function from $\mathbb{R}^n$ to $\mathbb{R}^m$. The $m \times n$ matrix

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

is called the **Jacobian** of f. We also use the notation $J(x)$ to mean a Jacobian.

C.2 Vector Fields and Equilibria

Definition C.4 (Vector Field)**.** *A* **vector field** f *on* $\mathbb{R}^n$ *is a smooth function* $f : \mathbb{R}^n \to \mathbb{R}^n$*. A vector field* f *is* **linear** *if* $f(x) = Ax$*, where* A *is an* $n \times n$ *matrix of real numbers.*

We can think of a differential equation

$$\dot{x}(t) \quad = \quad f(x(t)) \tag{C.1}$$

as being defined by a vector field f on $\mathbb{R}^n$. A **solution** or **trajectory** $t \to x(t)$ of Equation (C.1), with $x(t_0) = x_0$, is then a curve C in $\mathbb{R}^n$, beginning at x_0, parameterized by t, such that, at each point of C, the vector field $f(x(t))$ is tangent to C. $\mathbb{R}^n$ is then called the **state space** of the system given by Equation (C.1).

Definition C.5 (Equilibrium Point)**.** *A vector $x^* \in \mathbb{R}^n$ is an* **equilibrium point** *or* **fixed point** *for the system (C.1) if and only if $f(x^*) = 0$. For autonomous, or time-invariant vector fields we may take $x^* = 0$ without loss of generality.*

If $x(t_0) = x^*$ is an equilibrium point for (C.1), then the function $x(t) \equiv x^*$ for $t > t_0$ is a solution of Equation (C.1), called the **null** or **equilibrium** solution. In other words, if the system represented by Equation (C.1) starts initially at an equilibrium, then it remains at the equilibrium thereafter.

Stability

The question of stability deals with the solutions of Equation (C.1) for initial conditions away from the equilibrium point. Intuitively, the null solution should be called stable if, for initial conditions close to the equilibrium, the solution remains close thereafter. We can formalize this notion into the following.

Definition C.6. *Given the nonlinear system defined by Equation (C.1), suppose that $x = 0 \in \mathbb{R}^n$ is an equilibrium. Then the null solution $x(t) = 0$ is said to be*

- **stable** *if and only if, for any $\epsilon > 0$ there exist $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(t_0)\| \quad < \quad \delta \text{ implies } \|x(t)\| < \epsilon \text{ for all } t > t_0 \tag{C.2}$$

- **asymptotically stable** *if $x = 0$ is stable and, in addition,*

$$\|x(t_0)\| \quad < \quad \delta \text{ implies } \|x(t)\| \to 0 \text{ as } t \to \infty. \tag{C.3}$$

- **unstable** *if it is not stable.*

The stability, respectively, asymptotic stability, is said to be **global** *if the corresponding conditions hold for every initial condition $x(t_0) \in \mathbb{R}^n$.*

This situation is illustrated by Figure C.1 and says that the system is stable if the solution remains within a ball of radius ϵ around the equilibrium, provided that the initial condition lies in a ball of radius δ around the

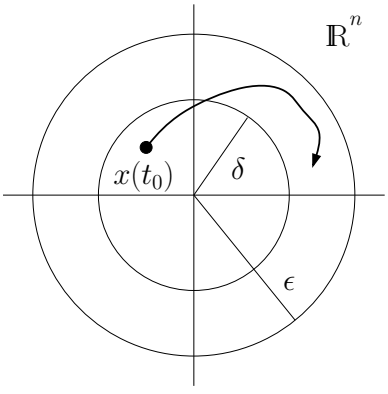


Figure C.1: Illustrating the definition of stability.

equilibrium. To put it another way, a system is stable if "small" perturbations in the initial conditions result in "small" perturbations from the null solution.

If the equilibrium is asymptotically stable, then the solutions return to the equilibrium as $t \to \infty$. If the equilibrium is unstable, then, given $\epsilon > 0$, there is no value of δ satisfying (C.3).

The above notions of stability are local in nature, that is, they may hold for initial conditions "sufficiently near" the equilibrium point but may fail for initial conditions farther away from the equilibrium. Stability (respectively, asymptotic stability) is said to be **global** if it holds for all initial conditions. A stronger notion than asymptotic stability is that of **exponential stability** defined next.

Definition C.7 (Exponential Stability)**.** *The equilibrium $x = 0$ of the system Equation (C.1) is* **exponentially stable** *if there are positive constants α and γ such that*

$$\|x(t)\| \leq \alpha \|x(t_0)\| e^{-\lambda t} \qquad \text{for all } t > 0 \tag{C.4}$$

The exponential stability is local or global depending on whether or not the inequality (C.4) holds for all initial conditions $x(t_0) \in \mathbb{R}^n$.

For a linear system

$$\dot{x} \;=\; Ax$$

the null solution is globally exponentially stable if and only if all eigenvalues of the matrix A lie in the open left half of the complex plane. Such a matrix

is called a **Hurwitz matrix**. For nonlinear systems, global stability cannot be so easily determined. However, local stability of the null solution of a nonlinear system $\dot{x} = f(x)$ can sometimes be determined by examining the eigenvalues of the Jacobian of the vector field $f(x)$.

Given the system (C.1) and suppose $x = 0$ is an equilibrium point. Let A be the $n \times n$ Jacobian matrix of $f(x)$, evaluated at $x = 0$. In other words

$$A = (a_{ij}), \text{ where } a_{ij} = \frac{\partial f_i}{\partial x_j}|_{x=0}$$

The system

$$\dot{x} = Ax \tag{C.5}$$

is called the **linear approximation** about the equilibrium of the nonlinear system (C.1).

Theorem C.1.

1. *Suppose A in (C.5) is a Hurwitz matrix so that $x = 0$ is a globally exponentially stable equilibrium point for the linearized system. Then $x = 0$ is locally exponentially stable for the nonlinear system (C.1).*

2. *Suppose A has one or more eigenvalues in the open right half plane so that $x = 0$ is an unstable equilibrium point for the linear system (C.5). Then $x = 0$ is unstable for the nonlinear system (C.1).*

3. *Suppose A has no eigenvalues in the open right half plane but one or more eigenvalues on the $j\omega$-axis. Then the stability properties of the equilibrium $x = 0$ for the nonlinear system (C.1) cannot be determined from A alone.*

Eigenvalues on the $j\omega$-axis are called **critical eigenvalues**. Examining the eigenvalues of the linear approximation of a nonlinear system in order to determine its stability properties is referred to as **Lyapunov's indirect method**. We see that local stability of the equilibrium of the nonlinear system (C.1) can be determined provided the matrix A of the linear approximation has no critical eigenvalues. If A has critical eigenvalues, or if one desires to determine the global stability properties of the nonlinear system (C.1) then Lyapunov's indirect method is inconclusive and other methods must be used. **Lyapunov's direct method**, also called the **second method of Lyapunov** introduced below, addresses these latter issues.

C.3 Lyapunov Functions

Definition C.8 (Positive Definite Functions)**.** *Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable scalar function, such that $V(0) = 0$ and $V(x) > 0$ for $x > 0$. Then V is said to be* **positive definite***.*

We say that $V(x)$ is **positive semi-definite** or **nonnegative definite** if $V(x) \geq 0$ for all x and we say that $V(x)$ is **negative (semi-)definite** if $-V(x)$ is positive (semi-)definite.

A particularly useful class of positive definite functions are **quadratic forms**

$$V(x) = x^T P x = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} > 0$$

where $P = (p_{ij})$ is a symmetric positive definite matrix.

The level surfaces of a positive definite quadratic form, given as solutions of $x^T P x =$ constant, are ellipsoids in $\mathbb{R}^n$. A positive definite quadratic form defines a norm on $\mathbb{R}^n$. In fact, given the usual norm $\|x\|$ on R^n, the function V given as

$$V(x) \quad = \quad x^T x = \|x\|^2$$

is a positive definite quadratic form corresponding to the choice $P = I$, the $n \times n$ identity matrix.

C.4 Stability Criteria

Definition C.9 (Lyapunov function candidate)**.** *Let $V(x) : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable scalar function on $\mathbb{R}^n$. Furthermore, suppose that V is positive definite. Then V is called a* **Lyapunov function candidate***.*

The power of Lyapunov stability theory comes from the fact that any function may be used in an attempt to show stability of a given system provided it is a Lyapunov function candidate according to the above definition.

Definition C.10. *By the derivative of V along trajectories of Equation (C.1), or the derivative of V in the direction of the vector field $f(x)$ defining Equation (C.1), we mean*

$$\dot{V}(t) \quad = \quad \frac{\partial V}{\partial x} f(x) = \frac{\partial V}{\partial x_1} f_1(x) + \cdots + \frac{\partial V}{\partial x_n} f_n(x)$$

Suppose that we evaluate the Lyapunov function candidate V at points along a solution trajectory $x(t)$ of Equation (C.1) and find that $V(t)$ is decreasing for increasing t. Intuitively, since V acts like a norm, this must mean that the given solution trajectory is converging toward the origin. This is the idea of Lyapunov stability theory.

Theorem C.2. *The null solution of Equation (C.1) is stable if there exists a Lyapunov function candidate V such that $\dot{V}$ is negative semi-definite along solution trajectories of Equation (C.1), that is, if*

$$\dot{V} = \frac{\partial V}{\partial x} f(x) \leq 0 \tag{C.6}$$

The inequality (C.6) says that the derivative of V computed along solutions of Equation (C.1) is nonpositive, which says that V itself is nonincreasing along solutions. Since V is a measure of how far the solution is from the origin, (C.6) says that a solution starting sufficiently near the origin must remain near the origin. If a Lyapunov function candidate V can be found satisfying (C.6) then V is called a **Lyapunov function** for the system given by Equation (C.1).

Note that Theorem C.2 gives only a sufficient condition for stability of Equation (C.1). If one is unable to find a Lyapunov function satisfying the inequality (C.6) it does not mean that the system is unstable. However, an easy sufficient condition for instability of Equation (C.1) is for there to exist a Lyapunov function candidate V such that $\dot{V} > 0$ along at least one solution of the system.

Theorem C.3. *The null solution of Equation (C.1) is asymptotically stable if there exists a Lyapunov function candidate V such that $\dot{V}$ is strictly negative definite along solutions of Equation (C.1), that is,*

$$\dot{V}(x) < 0 \tag{C.7}$$

The strict inequality in Equation (C.7) means that V is actually decreasing along solution trajectories of Equation (C.1) and hence, the trajectories must be converging to the equilibrium point.

C.5 Global and Exponential Stability

The condition $\dot{V} < 0$ along solution trajectories of a given system guarantees only local asymptotic stability even if the condition holds globally. In order to show global (asymptotic) stability, the Lyapunov function V must satisfy an additional condition, known as **radial unboundedness**.

Definition C.11. *Suppose $V : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. $V(x)$ is said to be* **radially unbounded** *if*

$$V(x) \to \infty \quad \text{as} \quad \|x\| \to \infty$$

With this additional property for V we can state the following:

Theorem C.4. *Let $V : \mathbb{R}^n \to \mathbb{R}$ be Lyapunov function candidate for the system given by Equation (C.1) and suppose that V is radially unbounded. Then $\dot{V} < 0$ implies that $x = 0$ is globally asymptotically stable.*

A sufficient condition for exponential stability is the following:

Theorem C.5. *Suppose that V is a Lyapunov function candidate for the system given by Equation (C.1) such that*

$$\begin{aligned} K_1\|x\|^p \leq V(x) \leq K_2\|x\|^p \text{ and} \\ \dot{V} \leq -K_3\|x\|^p \end{aligned} \tag{C.8}$$

where K_1, K_2, K_3, and p are positive constants. Then the origin $x = 0$ is exponentially stable. Moreover, if the inequalities (C.8) hold globally, then $x = 0$ is globally exponentially stable.

C.6 Stability of Linear Systems

Consider the linear system given by Equation (C.5) and let

$$V(x) \;=\; x^T P x \tag{C.9}$$

be a Lyapunov function candidate, where P is symmetric and positive definite. Computing $\dot{V}$ along solutions of Equation (C.5) yields

$$\begin{aligned} \dot{V} &= \dot{x}^T P x + x^T P \dot{x} \\ &= x^T (A^T P + PA) x \\ &= -x^T Q x \end{aligned}$$

where we have defined Q as

$$A^T P + PA \;=\; -Q \tag{C.10}$$

Theorem C.2 says that if Q given by Equation (C.10) is positive definite (it is automatically symmetric since P is symmetric), then the linear system

given by Equation (C.5) is globally exponentially stable. One approach that we can take is to first fix Q to be symmetric, positive definite and solve Equation (C.10), which is called a **matrix Lyapunov equation**, for P. If a symmetric positive definite solution P can be found to this equation, then the matrix A in Equation (C.5) is Hurwitz and $x^T Px$ is a Lyapunov function for the linear system (C.5). The converse to this statement also holds. In fact, we can summarize these statements as

Theorem C.6. *Given an $n \times n$ matrix A, then all eigenvalues of A have negative real part if and only if for every symmetric positive definite $n \times n$ matrix Q, the matrix Lyapunov equation (C.10) has a unique positive definite solution P.*

Thus, we can reduce the determination of stability of a linear system to the solution of a system of linear equations.

C.7 LaSalle's Theorem

The main difficulty in the use of Lyapunov stability theory is finding a suitable Lyapunov function satisfying $\dot{V} < 0$ in order to prove asymptotic stability. **LaSalle's invariance principle**, or **LaSalle's theorem** gives us a tool to determine the asymptotic properties of a system in the weaker case that V is only negative semidefinite, that is, when $\dot{V} \leq 0$.

The version of LaSalle's Theorem here follows closely the development in [76]. Consider the nonlinear system

$$\dot{x} = f(x), \quad x \in \mathbb{R}^n \tag{C.11}$$

where f is a smooth vector field on $\mathbb{R}^n$ with $f(0) = 0$.

Definition C.12 (Invariant Set)**.** *A set M is* **invariant** *or* **positively invariant***, with respect to the system (C.11) if*

$$x(t_0) \in M \implies x(t) \in M, \text{ for all } t > 0$$

Theorem C.7. (LaSalle's Theorem)

Let D be a region in $\mathbb{R}^n$ and let $\Omega \subset D$ be a compact set that is positively invariant with respect to the nonlinear system (C.11). Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that $\dot{V} \leq 0$ in Ω. Let E be the set of all points in Ω where $\dot{V} = 0$. Let M be the largest invariant set in E. Then every solution starting in Ω approaches M as $t \to \infty$.

As a corollary to LaSalle's Theorem it follows that the equilibrium solution $x = 0$ of Equation (C.11) is asymptotically stable if V does not vanish identically along any solution of Equation (C.11) other than the null solution, that is, if the only solution of Equation (C.11) satisfying $\dot{V} \equiv 0$ is the null solution.

Note that, in the statement of LaSalle's theorem, the function V need not be positive definite, i.e., it need not be a Lyapunov function. The key is to find a compact, positively invariant set Ω so that $\dot{V} \leq 0$ in Ω. In the case that V is a valid Lyapunov function, then such a set Ω may be determined from the level sets of V. We state this as the following proposition, whose proof is immediate from the definition of a Lyapunov function.

Proposition C.1. *Let V be a Lyapunov function candidate and let $V^{-1}(c_0)$ be any level surface of V, that is,*

$$V^{-1}(c_0) \quad = \quad \{x \in \mathbb{R}^n | V(x) = c_0\}$$

for some constant $c_0 > 0$. If

$$\dot{V} \quad = \quad \frac{\partial V}{\partial x} f(x) \leq 0$$

for $x \in V^{-1}(c_0)$. Then the set $\Omega = \{x \in \mathbb{R}^n | V(x) <= c_0\}$ is positively invariant for the system (C.1).

C.8 Barbalat's Lemma

LaSalle's theorem is valid only for autonomous, or time-invariant, systems. The next result, known as Barbalat's lemma, provides an additional tool for showing stability and asymptotic stability for nonlinear systems, including time-varying systems. Barbalat's lemma will prove useful for analyzing the trajectory tracking performance of robust and adaptive nonlinear controllers.

Lemma C.1. (Barbalat)

If $f : \mathbb{R}^n \to \mathbb{R}$ is a square integrable function and $\frac{df}{dx}$ is uniformly continuous, then $\frac{df}{dx} \to 0$ as $t \to \infty$.

Remark C.1.
1: A function $f : \mathbb{R}^n \to \mathbb{R}$ is **square integrable** *if*

$$\int_0^\infty f^T(x) f(x) dx < \infty$$

2: The statement that $\dfrac{df}{dx}$ *is uniformly continuous can be replaced by the statement* $\dfrac{d^2f}{dx^2}$ *is bounded.*

Another important notion related to stability is the notion of **uniform ultimate boundedness** of solutions.

Definition C.13. *A solution* $x(t) : [t_0, \infty) \to \mathbb{R}^n$ *for Equation (C.1) is said to be* **uniformly ultimately bounded** *(u.u.b.), with ultimate bound* b *if there exist positive constants* a *and* b *and a time* $T = T(a, b)$ *such that*

$$\|x(t_0)\| \leq a \Rightarrow \|x(t)\| \leq b, \text{for all } t \geq t_0 + T \tag{C.12}$$

The ultimate boundedness is global if (C.12) hold for arbitrarily large a*.*

Uniform ultimate boundedness says that the solution trajectory of Equation (C.1) will ultimately enter the ball of radius b and remain there for $t > t_0 + T$. If b defines a small region about the equilibrium, then uniform ultimate boundedness is a practical notion of stability that is useful in control system design.

Uniform ultimate boundedness is often established via Lyapunov theory as follows: Suppose V is a Lyapunov function candidate for the system (C.1) and suppose that the set $\Omega_\epsilon = \{0 \leq V(x) \leq \epsilon\}$ is compact. If $\dot{V} < 0$ outside the set Ω_ϵ, then Ω_ϵ is positively invariant for (C.1) and, moreover, trajectories starting outside of Ω_ϵ will eventually enter and, therefore, remain in Ω_ϵ. Therefore, the system is uniformly ultimately bounded with bound $b = \max_{x \in \Omega_\epsilon} \|x\|$.

Appendix D

OPTIMIZATION

In this Appendix we present some terminology and results from optimization theory that are used in the text. We will state the various results without proof. The reader should consult any introductory textbook on optimization for details.

A general optimization problem may be stated as follows:

$$\begin{aligned} \text{minimize} \quad & f(x),\ x \in \mathbb{R}^n && \text{(D.1)} \\ \text{subject to} \quad & h_i(x) = 0, \quad i = 1, \ldots, m && \text{(D.2)} \\ & g_j(x) \leq 0, \quad j = 1, \ldots, p && \text{(D.3)} \end{aligned}$$

The function $f : \mathbb{R}^n \to \mathbb{R}$ is called the **objective function** or **cost function**. The functions $h_i : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$ are equality and inequality constraints, respectively.

A vector $x \in \mathbb{R}^n$ is called a **feasible point** if it satisfies the constraints. For $x \in \mathbb{R}^n$, if $g_j(x) = 0$, the constraint g_j is called an **active constraint**, while, if $g_j(x) < 0$, the constraint is called **inactive**. The equality constraints $h_i = 0$ are, therefore, considered always active.

D.1 Unconstrained Optimization

In the case that there are no constraints on the objective function $f(x)$ we may give simple conditions for local optimality. For the problem

$$\text{minimize } f(x) \quad x \in \mathbb{R}^n$$

an **extreme point** or **extremum** of $f(x)$ is a vector x^* satisfying

$$\nabla f(x^*) \;=\; 0 \qquad \text{(D.4)}$$

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

A necessary condition for a vector $x^* \in \mathbb{R}^n$ to be a local minimizer of $f(x)$ is that x^* be a extremum of $f(x)$.

If x^* is an extremum of $f(x)$ then a sufficient condition for x^* to be a local minimizer of $f(x)$ is that the Hessian matrix of $f(x)$ is positive definite at $x = x^*$.

Gradient Descent

Since the gradient of a scalar function points in the direction of maximum increase of the function, a common approach to find the minimum value of an unconstrained objective function $f(x)$ is to choose an initial guess x^0 for the value of x^* and compute iteratively a sequence of vectors $x^1, x^2, \ldots$, such that

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k),\ k = 0, 1, \ldots$$

In other words, x^{k+1} points in the direction of the negative gradient of f at the value x^k. The scalar α_k is called the **step size** and influences the rate of convergence of the sequence to the minimizer x^*. This is known as the method of **Gradient Descent**. The method of **Steepest Descent** is a gradient descent method where the step size α_k is chosen at each iterative k according to

$$\alpha_k = \arg\min f(x^k - \alpha \nabla f(x^k)) \quad \alpha \geq 0$$

In other problems, such as the artificial potential field approach for motion planning considered in Chapter 7, the step size may be taken as a small constant value to avoid the case where a trajectory contacts an obstacle.

D.2 Constrained Optimization

In the case that there are no inequality constraints we have

$$\text{minimize} \quad f(x),\ x \in \mathbb{R}^n \tag{D.5}$$

$$\text{subject to} \quad h_i(x) = 0, \quad i = 1, \ldots, m \tag{D.6}$$

A vector x^* satisfying the constraints $h_i(x^*) = 0$ is called a **regular point** if the gradient vectors $\nabla h_1(x^*), \ldots, \nabla h_m(x^*)$ are linearly independent at x^*.

Lagrange Multipliers

Define the **Lagrangian** function

$$\ell(x, \lambda) = f(x) + \lambda_1 h_1(x) + \cdots + \lambda_m h_m(x) = f(x) + \lambda^T h(x)$$

where the $\lambda = (\lambda_1, \ldots, \lambda_m)$ is a vector of **Lagrange multipliers**. The following theorem, known as Lagrange's theorem, gives a necessary condition for a solution to the above minimization problem.

Theorem D.1. *Let x^* be a regular point of the optimization problem (D.5)-(D.6). If x^* is a minimizer of $f : \mathbb{R}^n \to \mathbb{R}$ subject to the constraints $h_i(x) = 0$, $i = 1, \ldots, m$, then there is a vector $\lambda^* = [\lambda_1^*, \ldots, \lambda_m^*]$ such that*

$$\frac{\partial \ell}{\partial x}(x^*, \lambda^*) = \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* h(x^*) = 0 \tag{D.7}$$

$$\frac{\partial \ell}{\partial \lambda}(x^*, \lambda^*) = h(x^*) = 0 \tag{D.8}$$

Lagrange's theorem says that, at a local minimum x^*, the gradient of the objective function can be expressed as a linear combination of the equality constraints.

Next, define the matrix function

$$L(x, \lambda) = F(x) + \lambda_1 H_1(x) + \cdots + \lambda_m H_m(x)$$

Where $F(x)$ and $H_i(x)$ are the Hessian matrices of $f(x)$ and $h_i(x)$, respectively. Then, if x^*, λ^* satisfy Equations (D.7) and (D.8), a sufficient condition for x^* to be a strict local minimum is that the matrix $L(x^*, \lambda^*)$ is positive definite.

Example D.1. *Consider the problem*

$$\begin{array}{ll} \text{maximize} & x^T Q x, \ x \in \mathbb{R}^n \\ \text{subject to} & x^T P x = 1 \end{array}$$

where Q and P are $n \times n$ symmetric positive definite matrices. We first note that maximizing an objective $f(x)$ is equivalent to minimizing $-f(x)$. Therefore, we form the Lagrangian function

$$\ell(x, \lambda) = -x^T Q x + \lambda (x^T P x - 1)$$

The necessary conditions for optimality are then

$$\begin{aligned} \frac{\partial \ell}{\partial x} &= -2x^T Q + 2\lambda x^T P \\ \frac{\partial \ell}{\partial \lambda} &= x^T P x - 1 = 0 \end{aligned}$$

Any feasible point for this problem is regular. From the first equation above we have

$$(\lambda P - Q)x = 0$$

which, since P is symmetric and positive definite, can be written as

$$(\lambda I - P^{-1}Q)x = 0$$

Therefore, a solution (x^*, λ^*), if it exists, is such that x^* is an eigenvector of $P^{-1}Q$ with λ^* the corresponding eigenvector.

Appendix E

CAMERA CALIBRATION

As described in Section 11.2, a digital image is a discrete array of gray level values. The objective of camera calibration is to determine all of the parameters that are necessary to relate the pixel coordinates (r, c) to the world coordinates (x, y, z) of a point in the camera's field of view. In other words, given the coordinates of a point P relative to the world coordinate frame, after we have calibrated the camera we will be able to compute (r, c), the image pixel coordinates for the projection of this point.

Camera calibration is a ubiquitous problem in computer vision. Numerous solution methods have been developed, many of which have been implemented in open-source software libraries (e.g., OpenCV [17] and Matlab's Computer Vision Toolbox [26]). Here, we present an approach that is conceptually straightforward and relatively easy to implement.

E.1 The Image Plane and the Sensor Array

In order to relate digital images to the 3D world, we must first determine the relationship between the image-plane coordinates (u, v) and the pixel coordinates (r, c). We typically define the origin of the pixel array to be located at a corner of the image rather than at the center of the image. Let the pixel array coordinates of the pixel that contains the principal point be given by (o_r, o_c). In general, the sensing elements in the camera will not be of unit size, nor will they necessarily be square. Denote by s_x and s_y the horizontal and vertical dimensions, respectively, of a pixel. Finally, it is often the case that the horizontal and vertical axes of the pixel array coordinate system point in opposite directions from the horizontal and vertical axes of

Robot Modeling and Control, Second Edition.
Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

the camera coordinate frame.[1] Combining these, we obtain the following relationship between image-plane coordinates and pixel array coordinates

$$-\frac{u}{s_x} = (r - o_r), \quad -\frac{v}{s_y} = (c - o_c) \tag{E.1}$$

Note that the coordinates (r, c) will be integers, since they are the discrete indices into an array that is stored in computer memory. Therefore, this relationship is only an approximation. In practice, the value of (r, c) can be obtained by truncating or rounding the ratio on the left-hand side of these equations.

E.2 Extrinsic Camera Parameters

In Section 11.2.1, we considered only the case in which coordinates are expressed relative to the camera frame. In typical robotics applications, tasks are expressed in terms of the world coordinate frame. If we know the position and orientation of the camera frame relative to the world coordinate frame (i.e., if we know O_c^w and R_c^w, respectively), we can write

$$x^w = R_c^w x^c + O_c^w$$

or, if we know x^w and wish to solve for x^c,

$$x^c = R_w^c (x^w - O_c^w)$$

In the remainder of this section, to simplify notation, we will define

$$R = R_w^c, \quad T = -R_w^c O_c^w$$

and we write

$$x^c = R x^w + T$$

Together, R and T are called the **extrinsic camera parameters**.

Cameras are typically mounted on tripods or on mechanical positioning units. In the latter case, a popular configuration is the pan/tilt head. A pan/tilt head has two degrees of freedom: a rotation about the world z-axis and a rotation about the pan/tilt head's x-axis. These two degrees of freedom are analogous to those of a human head, which can easily look up

[1] This is an artifact of our choice to place the center of projection behind the image plane. The directions of the pixel array axes may vary, depending on the particular software drivers used to acquire digital images from the camera.

or down, and can turn from side to side. In this case, the rotation matrix R is given by

$$R = R_{z,\theta} R_{x,\alpha}$$

where θ is the pan angle and α is the tilt angle. More precisely, θ is the angle between the world x-axis and the camera x-axis, about the world z-axis, while α is the angle between the world z-axis and the camera z-axis, about the camera x-axis.

E.3 Intrinsic Camera Parameters

The mapping from 3D world coordinates to pixel coordinates is obtained by combining Equations (11.4) and (E.1) to obtain

$$r = -\frac{\lambda}{s_x}\frac{x}{z} + o_r, \qquad c = -\frac{\lambda}{s_y}\frac{y}{z} + o_c \tag{E.2}$$

Thus, once we know the values of the parameters $\lambda, s_x, o_r, s_y, o_c$ we can determine (r, c) from (x, y, z), where (x, y, z) are coordinates relative to the camera frame. In fact, we don't need to know all of λ, s_x, s_y; it is sufficient to know the ratios

$$f_x = \frac{\lambda}{s_x} \qquad f_y = \frac{\lambda}{s_y}$$

These parameters f_x, o_r, f_y, o_c are known as the **intrinsic camera parameters**. They are constant for a given camera and do not change when the camera moves.

E.4 Determining the Camera Parameters

Of all the camera parameters, o_r, o_c (the image pixel coordinates of the principal point) are the easiest to determine. This can be done by using the idea of vanishing points, which was introduced in Example 11.2. The vanishing points for three mutually orthogonal sets of parallel lines in the world will define a triangle in the image. The orthocenter of this triangle (that is, the point at which the three altitudes intersect) is the image principal point. Thus, a simple way to compute the principal point is to position a cube in the workspace, find the edges of the cube in the image (this will produce the three sets of mutually orthogonal parallel lines), compute the intersections of the image lines that correspond to each set of parallel lines in the world (this will produce three points in the image), and determine the orthocenter for the resulting triangle.

To determine the remaining camera parameters, we construct a system of equations in terms of the known coordinates of points in the world and the pixel coordinates of their projections in the image. The unknowns in this system are the camera parameters. The first step is to acquire a data set of the form $\{r_i, c_i, x_i, y_i, z_i\}$ for $i = 1 \cdots N$, in which r_i, c_i are the image pixel coordinates of the projection of a point in the world with coordinates x_i, y_i, z_i relative to the world coordinate frame. This acquisition is often done manually, for example, by placing a small bright light at known (x, y, z) coordinates in the world and then hand selecting the corresponding image point.

Once we have acquired the data set, we proceed to set up a linear system of equations. The extrinsic parameters of the camera are given by

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

With respect to the camera frame, the coordinates of a point in the world are thus given by

$$\begin{aligned} x^c &= r_{11}x + r_{12}y + r_{13}z + T_x \\ y^c &= r_{21}x + r_{22}y + r_{23}z + T_y \\ z^c &= r_{31}x + r_{32}y + r_{33}z + T_z \end{aligned}$$

Combining these three equations with Equation (E.2) we obtain

$$r - o_r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{E.3}$$

$$c - o_c = -f_y \frac{y^c}{z^c} = -f_y \frac{r_{21}x + r_{22}y + r_{23}z + T_y}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{E.4}$$

Since we know the coordinates of the principal point, we can simplify these equations by using the coordinate transformation

$$r \leftarrow r - o_r, \qquad c \leftarrow c - o_c$$

We now write the two transformed projection equations as functions of the unknown variables $r_{ij}, T_x, T_y, T_z, f_x, f_y$. This is done by solving Equations (E.3) and (E.4) for z^c, and setting the resulting equations to be equal to one another. In particular, r_i, c_i, x_i, y_i, z_i we have

$$r_i f_y (r_{21}x_i + r_{22}y_i + r_{23}z_i + T_y) = c_i f_x (r_{11}x_i + r_{12}y_i + r_{13}z_i + T_x)$$

Defining $\alpha = f_x/f_y$, we can rewrite this as

$$r_i r_{21} x_i + r_i r_{22} y_i + r_i r_{23} z_i + r_i T_y - \alpha c_i r_{11} x_i - \alpha c_i r_{12} y_i - \alpha c_i r_{13} z_i - \alpha c_i T_x = 0$$

We can combine the N such equations into the matrix equation

$$Ax = 0 \tag{E.5}$$

in which

$$A = \begin{bmatrix} r_1 x_1 & r_1 y_1 & r_1 z_1 & r_1 & -c_1 x_1 & -c_1 y_1 & -c_1 z_1 & -c_1 \\ r_2 x_2 & r_2 y_2 & r_2 z_2 & r_2 & -c_2 x_2 & -c_2 y_2 & -c_2 z_2 & -c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ r_N x_N & r_N y_N & r_N z_N & r_N & -c_N x_N & -c_N y_N & -c_N z_N & -c_N \end{bmatrix}$$

and

$$x = [r_{21}, r_{22}, r_{23}, T_y, \alpha r_{11}, \alpha r_{12}, \alpha r_{13}, \alpha T_x]^T$$

If $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_8]^T$ is a solution for Equation (E.5) we only know that this solution is some scalar multiple of the desired solution x, namely,

$$\bar{x} = k[r_{21}, r_{22}, r_{23}, T_y, \alpha r_{11}, \alpha r_{12}, \alpha r_{13}, \alpha T_x]^T$$

in which k is an unknown scale factor. In order to solve for the true values of the camera parameters, we can exploit constraints that arise from the fact that R is a rotation matrix. In particular,

$$(\bar{x}_1^2 + \bar{x}_2^2 + \bar{x}_3^2)^{\frac{1}{2}} = (k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = |k|$$

and likewise

$$(\bar{x}_5^2 + \bar{x}_6^2 + \bar{x}_7^2)^{\frac{1}{2}} = (\alpha^2 k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = \alpha|k|$$

Note that by definition, $\alpha > 0$.

Our next task is to determine the sign of k. Using Equation (E.2) we see that $rx^c < 0$ (recall that we have used the coordinate transformation $r \leftarrow r - o_r$). Therefore, we choose k such that $r(r_{11}x + r_{12}y + r_{13}z + T_x) < 0$.

At this point we know the values for $k, \alpha, r_{21}, r_{22}, r_{23}, r_{11}, r_{12}, r_{13}, T_x, T_y$, and all that remains is to determine T_z, f_x, f_y, since the third column of

R can be determined as the vector cross product of its first two columns. Since $\alpha = f_x/f_y$, we need only determine T_z and f_x. Returning again to the projection equations, we can write

$$r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z}$$

Using an approach similar to that used above to solve for the first eight parameters, we can write this as the linear system

$$r(r_{31}x + r_{32}y + r_{33}z + T_z) = -f_x(r_{11}x + r_{12}y + r_{13}z + T_x)$$

which can easily be solved for T_z and f_x.

Bibliography

[1] C. Abdallah, D.M. Dawson, P. Dorato, and M. Jamshidi. A survey of robust control of rigid robots. *IEEE Control Systems Magazine*, 11(2):24–30, February 1991.

[2] R. Abraham and J. E. Marsden. *Foundations of Mechanics*. The Benjamin/Cummings Pub. Co., Inc., London, 1978.

[3] M. Aicardi, G. Cassalino, A. Bicchi, and A. Balestrino. Closed loop steering of unicycle-like vehicles via lyapunov techniques. *IEEE Robotics and Automation Magazine*, pages 27–35, March 1995.

[4] R. J. Anderson and M. W. Spong. Hybrid impedance control of robots. *IEEE Journal of Robotics and Automation*, 4(5):549–556, October 1988.

[5] Russell L. Anderson. Dynamic sensing in a ping-pong playing robot. *IEEE Transaction on Robotics and Automation*, 5(6):723–739, 1989.

[6] R. L. Andersson. *A Robot Ping-Pong Player. Experiment in Real-Time Intelligent Control*. MIT Press, Cambridge, MA, 1988.

[7] H. Asada and J.A. Cro-Granito. Kinematic and static characterization of wrist joints and their optimal design. In *Proc. IEEE Conf. on Robotics and Automation*, St. Louis, MO, 1985.

[8] K.J. Åstrom and Tore Hagglund. *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, 1995.

[9] S. Barnett. *Matrix Methods for Engineers and Scientists*. McGraw-Hill, London, 1979.

[10] Jerome Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, December 1991.

[11] A. K. Bejczy. Robot arm dynamics and control. *JPL Technical Memorandum*, pages 33–69, February 1974.

[12] H. Berghuis and H. Nijmeijer. A passivity approach to controller-observer design for robots. *IEEE Trans. on Robotics and Automation*, 9:740–754, 1993.

[13] A.M. Bloch, M. Reyhanoglu, and N.H. McClamroch. Control and stabilization of nonholonomic dynamic systems. *IEEE Transactions on Automatic Control*, 37:1746–1757, 1992.

[14] Daniel J. Block, Karl J. Åström, and Mark W. Spong. *The Reaction Wheel Pendulum*. Morgan & Claypool Publishers, 2007.

[15] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press, San Diego, CA, 2nd edition, 1986.

[16] O. Botema and B. Roth. *Theoretical Kinematics*. North Holland, Amsterdam, 1979.

[17] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[18] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. S. Millman and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191. Birkhauser, 1983.

[19] B. Brogliato, I. D. Landau, and R. Lozano. Adaptive motion control of robot manipulators: A unified approach based on passivity. *Int. J. of Robust and Nonlinear Control*, 1:187–202, 1991.

[20] R. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. Int. Joint Conf. on Art. Intell.*, pages 799–806, 1983.

[21] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[22] B. Charlet, J. Levine, and R. Marino. On dynamic feedback linearization. *Systems and Control Letters*, 13:143–152, 1989.

[23] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman,

G. Hager, and S. Morse, editors, *The confluence of vision and control*, volume 237 of *Lecture Notes in Control and Information Sciences*, pages 66–78. Springer Verlag, 1998.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.

[25] J.C. Colson and N. D. Perreira. Kinematic arrangements used in industrial robots. In *Proc. 13th International Symposium on Industrial Robots*, 1983.

[26] P.I. Corke. *Robotics, Vision and Control*. Springer, 2011.

[27] P.I. Corke and S. A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Trans. on Robotics and Automation*, 17(4):507–515, August 2001.

[28] M. Corless and G. Leitmann. Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems. *IEEE Transactions on Automatic Control*, 26:1139–1144, 1981.

[29] J.J. Craig. *Adaptive Control of Mechanical Manipulators*. Addison-Wesley, Reading, MA, 1988.

[30] M. L. Curtiss. *Matrix Groups*. Springer Verlag, New York, NY, second edition, 1984.

[31] V. S. Cvetković and M. Vukobratović. One robust, dynamic control algorithm for manipulation systems. *International Journal of Robotics Research*, 1(4):15–28, winter 1982.

[32] C. Canudas de Wit et al. *Theory of Robot Control*. Springer Verlag, Berlin, 1996.

[33] K. Deguchi. Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proc. Int. Conf. Intelligent Robots and Systems*, pages 705–711, October 1998.

[34] A. DeLuca. Dynamic control properties of robot arms with joint elasticity. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 1574–1580, Philadelphia, PA, 1988.

[35] J. Denavit and R. S. Hartenberg. A kinematic notation for lower pair mechanisms. *Applied Mechanics*, 22:215–221, 1955.

[36] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback Control Theory.* Macmillan Publishing Company, New York, NY, 1992.

[37] J. Duffy. *Analysis of Mechanisms and Robot Manipulators.* John Wiley and Sons, Inc., New York, NY, 1980.

[38] J. Duffy. The fallacy of modern hybrid control theory that is based on orthogonal complements of twist and wrench spaces. *J. Robot Syst.*, 7:139–144, 1990.

[39] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8:313–326, 1992.

[40] S. E. Fahlman. A planning system for robot construction tasks. *Artificial Intelligence*, 5:1–49, 1974.

[41] I. Fantoni and R. Lozano. *Nonlinear Control for Underactuated Mechanical Systems.* Springer Verlag, Berlin, 2001.

[42] O.D. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, Cambridge, MA, 1993.

[43] J. T. Feddema, C. S. George Lee, and O. R. Mitchell. Weighted Selection of Image Features for Resolved Rate Visual Feedback Control. *IEEE Transactions on Robotics and Automation*, 7:31–47, 1991.

[44] J.T. Feddema and O.R. Mitchell. Vision-guided servoing with feature-based trajectory generation. *IEEE Trans. on Robotics and Automation*, 5(5):691–700, October 1989.

[45] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[46] A.F. Filippov. *Differential Equations with Discontinuous Right Hand Sides.* Kluwer, Dordrecht, 1988.

[47] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, 2003.

[48] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems.* Addison-Wesley, 2nd edition, 1990.

[49] S. H. Friedberg, A. J. Insel, and L. E. Spence. *Linear Algebra.* Prentice-Hall, Englewood Cliffs, NJ, 1979.

[50] M. Gautier and W. Khalil. On the identification of the inertial parameters of robots. In *IEEE Conf. on Decision and Control*, pages 2264–2269, 1988.

[51] M. Gautier and W. Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Trans. on Robotics and Automation*, 6:368–373, 1990.

[52] S.S. Ge, T.H. Lee, and C.J. Harris. *Adaptive Neural Network Control of Robotic Manipulators.* World Scientific, Singapore, 1998.

[53] F. Ghorbel, B. Srinivasan, and M.W. Spong. On the uniform boundedness of the inertia matrix of serial robot manipulators. *Journal of Robotic Systems*, 15(1):17–28, January 1998.

[54] A. A. Goldenberg, B. Benhabib, and R. G. Fenton. A complete generalized solution to the inverse kinematics of robots. *IEEE J. Robotics and Automation*, RA-1(1):14–20, 1985.

[55] H. Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, MA, 1974.

[56] G.H. Golub and C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1983.

[57] Irving M. Gottlieb. *Electric Motors and Control Techniques.* TAB Books, New York, NY, 1994.

[58] W. M. Grimm. Robustness analysis of nonlinear decoupling for elastic-joint robots. *IEEE Trans. on Robotics and Automation*, 6:373–377, 1990.

[59] J.W. Grizzle, E.R. Westervelt, and C. Canudas de Wit. Event-based PI control of an underactuated biped walker. In *Proc. 42nd IEEE Conference on Decision and Control*, pages 3091–3096, December 2003.

[60] Paul R. Halmos. *Finite-dimensional vector spaces.* Springer Verlag, New York, 1974.

[61] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision, Vols. I and II.* Addison-Wesley Pub. Co., Inc., Reading, MA, 1993.

[62] H. Hemami, F. Weimer, and S. Koozekanani. Some aspects of the inverted pendulum problem for modeling of locomotion systems. *IEEE Transactions on Automatic Control*, 18(6):658–661, 1973.

[63] N. Hogan. Impedance control: An approach to manipulation: Parts I–III. *ASME J. of Dynamic Systems, Measurement, and Control*, 107:1–24, 1985.

[64] J.M. Hollerbach. A recursive formulation of Lagrangian manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-10(11):730–736, Nov 1980.

[65] J.M. Hollerbach and S. Gideon. Wrist-partitioned inverse kinematic accelerations and manipulator dynamics. *International Journal of Robotics Research*, 4:61–76, 1983.

[66] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, 1986.

[67] R. Horowitz and M. Tomizuka. An adaptive control scheme for mechanical manipulators - compensation of nonlinearities and decoupling control. *ASME Journal of Dynamic Systems, Meas. and Control*, 108:127–135, 1986.

[68] K. Hunt. *Kinematic Geometry of Mechanisms*. Oxford University Press, London, 1978.

[69] A. Isidori. *Nonlinear Control Systems*. Springer Verlag, New York, NY, 1999.

[70] J.J. Uicker Jr., J. Denavit, and R. S. Hartenberg. An iterative method for the displacement analysis of spatial mechanisms. *Trans. Applied Mechanics*, 31 Series E:309–314, 1964.

[71] T. Kailath. *Linear Systems*. Prentice Hall, Englewood Cliffs, NJ, 1980.

[72] R. Kalman. Contributions to the theory of optimal control. *Boletin de la Sociedad Matemática Mexicana*, 5:102–119, 1960.

[73] Subbarao Kambhampati and Larry S. Davis. Multiresolution path planning for mobile robots. *IEEE Journal of Robotics and Automation*, 2(3):135–145, September 1986.

[74] Lydia E. Kavraki. *Random Networks in Configuration Space for Fast Path Planning*. PhD thesis, Stanford University, Stanford, CA, 1994.

[75] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4):566–580, August 1996.

[76] H.K. Khalil. *Nonlinear Systems, 2nd Ed.* Prentice Hall, Englewood Cliffs, NJ, 1996.

[77] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[78] D.E. Koditschek. The application of total energy as a Lyapunov function for mechanical control systems. In J.E. Marsden, et al., editors, *Dynamics and Control of Multibody Systems*, volume 97, pages 131–157. AMS, 1989.

[79] D.E. Koditschek. Robot planning and control via potential functions. In *The Robotics Review 1*, pages 349–367. MIT Press, 1989.

[80] I. Kolmanovsky and N. H. McClamroch. Developments in nonholonomic control problems. *IEEE Control Magazine*, pages 20–36, December 1995.

[81] A.J. Krener and I. Isidori. Linearization by output injection and nonlinear observers. *Syst. Control Lett.*, 3:47–52, 1983.

[82] A.J. Krener and W. Respondek. Nonlinear observers with linearizable error dynamics. *SIAM J. Control and Optimization*, 23(2):197–216, 1985.

[83] K. Kreutz. On manipulator control by exact linearization. *IEEE Transactions on Automatic Control*, 34(7):763–767, July 1989.

[84] B.C. Kuo. *Control Systems.* Prentice Hall, Englewood Cliffs, NJ, 1982.

[85] C. Lanczos. *The Variational Principles of Mechanics.* University of Toronto Press, Toronto, CA, 4th edition, 1970.

[86] I.D. Landau and R. Horowitz. Synthesis of adaptive controllers for robotic manipulators using a passive feedback systems approach. *Int. J. of Adaptive Control and Signal Processing*, 3:23–38, 1989.

[87] J. C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[88] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, MA, 2006.

[89] S. M. LaValle. Motion planning. *IEEE Robotics Automation Magazine*, 18(1):79–89, March 2011.

[90] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conference on Robotics and Automation*, pages 473–479, 1999.

[91] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *New Directions in Algorithmic and Computational Robotics*, pages 293–308. AK Peters, 2001.

[92] C.S.G. Lee, R. C. Gonzales, and K. S. Fu. *Tutorial on Robotics*. IEEE Computer Society Press, Silver Spring, MD, 1983.

[93] C.S.G. Lee and M. Ziegler. A geometric approach in solving the inverse kinematics of PUMA robots. *IEEE Trans. Aero. and Elect. Sys.*, AES-20(6):695–706, 1984.

[94] F.L. Lewis, S. Jagannathan, and A. Yesildirek. *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor and Francis, London, 1999.

[95] Daniel Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, 2003.

[96] Ming Lin and Dinesh Manocha. Efficient contact determination in dynamic environments. *International Journal of Computational Geometry and Applications*, 7(1):123–151, 1997.

[97] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, February 1983.

[98] A. De Luca, S. Iannitti, R. Mattone, and G. Oriolo. Underactuated manipulators: Control properties and techniques. *Machine Intelligence and Robotic Control*, 4(3):113–125, 2002.

[99] A. De Luca and P. Lucibello. A general algorithm for dynamic feedback linearization of robots with elastic joints. In *1998 IEEE Int. Conf. on Robotics and Automation*, pages 504–510, Leuven, 1998.

[100] A. De Luca and G. Oriolo. *Modelling and control of nonholonomic mechanical systems*, volume 360 of *CISM Courses and Lectures*, chapter Kinematics and Dynamics of Multi-Body Systems, pages 277–342. Springer Verlag, Wien, 1995.

[101] A. De Luca and G. Oriolo. Motion planning and trajectory control of an underactuated three-link robot via dynamic feedback linearization.

In *2000 IEEE Int. Conf. on Robotics and Automation*, pages 2789–2795, San Francisco, CA, 2000.

[102] A. De Luca and G. Oriolo. Trajectory planning and control for planar robots with passive last joint. *Int. J. of Robotics Research*, 21(5–6):575–590, 2002.

[103] D.G. Luenberger. Observing the state of a linear system. *IEEE Trans. Mil. Electronics*, MIL-8:74–80, 1964.

[104] J.Y.S. Luh, M. W. Walker, and R.P.C. Paul. On-line computational scheme for mechanical manipulators. *ASME J. of Dynamic Systems*, 102:69–76, 1980.

[105] K.M. Lynch and F.C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[106] Yi Ma, Stefano Soatto, Jana Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, New York, NY, 2003.

[107] E. Malis, F. Chaumette, and S. Boudet. 2-1/2-D visual servoing. *IEEE Trans. on Robotics and Automation*, 15(2):238–250, April 1999.

[108] R. Manseur and K.L. Doty. A robot manipulator with 16 real inverse kinematic solutions. *IJRR*, 8:75–79, 1989.

[109] R. Marino and M.W. Spong. Nonlinear control techniques for flexible joint manipulators: A single link case study. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1030–1026, San Francisco, CA, 1986.

[110] B.R. Markiewicz. Analysis of the computed torque drive method and comparison with conventional position servo for a computer-controlled manipulator. Technical Report TM 33-601, Jet Propulsion Laboratory, Pasadena, CA, March 1973.

[111] D. Marr. *Vision*. Freeman, San Francisco, CA, 1982.

[112] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer Verlag, New York, NY, second edition, 1999.

[113] M.T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, 14:418–432, 1981.

[114] R.H. Middleton and G. Goodwin. Adaptive computed torque control of robot manipulators. *Systems and Control Letters*, 1987.

[115] Brian Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, MA 02139, June 1997.

[116] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot. Explicit incorporation of 2D constraints in vision based control of robot manipulators. In Peter Corke and James Trevelyan, editors, *Experimental Robotics VI*, volume 250 of *Lecture Notes in Control and Information Sciences*, pages 99–108. Springer Verlag, 2000. ISBN: 1 85233 210 7.

[117] S. Mori, H. Nishihara, and K. Furuta. Control of unstable mechanical systems: Control of pendulum. *Int. J. Control*, 23:673–692, 1976.

[118] R. Murray, X. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[119] R. M. Murray, Z. X. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.

[120] R. M. Murray, M. Rathinam, and W. Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *1995 ASME Int. Mechanical Engineering Congress and Exposition*, San Francisco, CA, 1995.

[121] R. M. Murray and S. S. Sastry. Nonholonomic motion planning steering using sinusoids. *IEEE Trans. on Automatic Control*, 38:700–713, 1993.

[122] R.M. Murray and J. Hauser. A case study in approximate linearization: The Acrobot example. In *Proc. American Control Conference*, 1990.

[123] Y. Nakamura, T. Suzuki, and M. Koinuma. Nonlinear behavior and control of nonholonomic free-joint manipulator. *IEEE Trans. on Robotics and Automation*, 13(6):853–862, 1997.

[124] B. Nelson and P. K. Khosla. Integrating sensor placement and visual tracking strategies. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1351–1356, 1994.

[125] B. J. Nelson and P. K. Khosla. The resolvability ellipsoid for visual servoing. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 829–832, 1994.

[126] J. Nethery and M.W. Spong. Robotica. *IEEE Robotics Magazine*, 1(1), 1995.

[127] H. L. Newkirk, W. R. Haseltine, and A. V. Pratt. Stability of rotating space vehicles. *Proceedings of the IRE*, 48(4):74–750, 1960.

[128] S. Nicosia, F. Nicolo, and D. Lentini. Dynamical control of industrial robots with elastic and dissipative joints. In *Proceedings of the IFAC 8th Triennial World Congress*, pages 1933–1939, Kyoto, Japan, 1981.

[129] N. Nilsson. A mobile automaton: An application of artificial intellgience techniques. In *Proc. Int. Joint Conf. on Art. Intell.*, 1969.

[130] G. Oriolo and Y. Nakamura. Control of mechanical systems with second order nonholonomic constraints: Underactuated manipulators. In *Proceedings of the 30th IEEE Conf. on Dec. and Contr.*, pages 306–308, Brighton, England, 1991.

[131] R. Ortega and M.W. Spong. Adaptive control of robot manipulators: A tutorial. *Automatica*, 25(6):877–888, 1989.

[132] Mark H. Overmars and Petr Švestka. A probabilistic learning approach to motion planning. In *Proceedings of Workshop on Algorithmic Foundations of Robotics*, pages 19–37, 1994.

[133] K. Passino and S. Yurkovich. *Fuzzy Control.* Addison Wesley, Menlo Park, CA, 1998.

[134] R. Paul. *Robot Manipulators: Mathematics, Programming and Control.* MIT Press, Cambridge, MA, 1982.

[135] R. P. Paul, B. E. Shimano, and G. Mayer. Kinematic control equations for simple manipulators. *IEEE Trans. Systems, Man., and Cybernetics*, SMC-ll(6):339–455, 1981.

[136] R.P. Paul. Modeling, trajectory calculation, and servoing of a computer controlled arm. Technical Report AIM 177, Stanford University Artificial Intelligence Laboratory, Palo Alto, CA, Nov 1972.

[137] D. L. Pieper. *The Kinematics of Manipulators under Computer Control.* PhD thesis, Stanford University, 1968.

[138] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[139] E.J.F. Primrose. On the input-output equation of the general 7R mechanism. *Mechanisms and Machine Theory*, 21:509–510, 1986.

[140] M.H. Raibert and J.J. Craig. Hybrid position/force control of manipulators. *ASME*, 102:126–133, 1981.

[141] M. Rathinam and R. M. Murray. Configuration flatness of Lagrangian systems underactuated by one control. *SIAM J. of Control and Optimization*, 36(1):164–179, 1998.

[142] Mark C. Readman. *Flexible Joint Robots*. CRC Press, Boca Raton, FL, 1994.

[143] J. N. Reddy and M.L. Rasmussen. *Advanced Engineering Analysis*. John Wiley and Sons, Inc., New York, NY, 1982.

[144] Azriel Rosenfeld and Avi Kak. *Digital Picture Processing*. Academic Press, New York, NY, 1982.

[145] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: The Task Function Approach*. Clarendon Press, Oxford, England, 1992.

[146] A. C. Sanderson, L. E. Weiss, and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Trans. on Robotics and Automation*, RA-3(5):404–417, October 1987.

[147] J. T. Schwartz, M. Sharir, and J. Hopcroft, editors. *Planning, Geometry, and Complexity of Robot Motion*. Ablex, Norwood, NJ, 1987.

[148] M. Shahinpoor. The exact inverse kinematics solutions for the Rhino XR-2 robot. *Robotics Age*, 7(8):6–14, 1985.

[149] R. Sharma and S. Hutchinson. Motion perceptibility and its application to active vision-based servo control. *IEEE Trans. on Robotics and Automation*, 13(4):607–617, August 1997.

[150] R. Sharma and S. A. Hutchinson. On the observability of robot motion under active camera control. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 162–167, May 1994.

[151] N. Shiroma, H. Arai, and K. Tanie. Nonholonomic motion planning for coupled planar rigid bodies with passive revolute joints. *Int. J. of Robotics Research*, 21(5-6):563–574, 2002.

[152] D.B. Silver. On the equivalence of Lagrangian and Newton–Euler dynamics for manipulators. *International Journal of Robotics Research*, 1(2), 1982.

[153] J.-J. E. Slotine and W. Li. Adaptive manipulator control: A case study. *IEEE Trans. on Automatic Control*, 33:995–1003, 1988.

[154] J.J.-E. Slotine. The robust control of robot manipulators. *Int. J. Robotics Research*, 4(2):49–64, 1985.

[155] I. S. Sokolnikoff and R. M. Redheffer. *Mathematical Methods of Physics and Modern Engineering*. McGraw-Hill, New York, NY, 1958.

[156] M. Spivak. *A comprehensive introduction to differential geometry*. Publish or Perish, Inc., Berkeley, CA, second edition, 1979.

[157] M. W. Spong. The control of underactuated mechanical systems. In *First International Conference on Mecatronics*, Mexico City, 1994.

[158] M. W. Spong. The swingup control problem for the acrobot. *IEEE Control Systems*, 15(1):49–55, February 1995.

[159] M. W. Spong. Energy based control of a class of underactuated mechanical system. In *1996 IFAC World Congress*, volume F, pages 431–436, San Francisco, CA, July 1996.

[160] M. W. Spong and D. Block. The pendubot: A mechatronic systems for control research and education. In *IEEE Conference on Decision and Control*, pages 555–557, New Orleans, LA, December 1995.

[161] M. W. Spong, R. Ortega, and R. Kelly. Comments on 'adaptive manipulator control: A case study'. *IEEE Trans. on Automatic Control*, 35:761–762, 1990.

[162] Mark W. Spong. Partial feedback linearization of underactuated mechanical systems. In *IROS'94*, Munich, Germany, September 1994.

[163] Mark W. Spong. Underactuated mechanical systems. In B. Siciliano and K. P. Valavanis, editors, *Control Problems in Robotics and Automation*, volume 230, pages 135–150. Springer Verlag, 1998.

[164] Mark W. Spong, Daniel J. Block, and Karl J. Astrom. The mechatronics control kit for education and research. In *IEEE Conference on Control Applications*, pages 105–110, Mexico City, September, 5-7 2001.

[165] M.W. Spong. Modeling and control of elastic joint robots. *Transactions of the ASME, J. Dynamic Systems, Measurement and Control*, 109:310–319, December 1987.

[166] M.W. Spong. On the robust control of robot manipulators. *IEEE Transactions on Automatic Control*, AC-37(11):1782–1786, November 1992.

[167] M.W. Spong, F.L. Lewis, and C.T. Abdallah. *Robot Control: Dynamics, Motion Planning, and Analysis*. IEEE Press, 1992.

[168] M.W. Spong, J.S. Thorp, and J. Kleinwaks. The control of robot manipulators with bounded input part II: robustness and disturbance rejection. In *IEEE Conf. on Decision and Control*, Las Vegas, December 1984.

[169] M.W. Spong, J.S. Thorp, and J. Kleinwaks. The control of robot manipulators with bounded input. *IEEE Transactions on Automatic Control*, 31(3):483–490, March 1986.

[170] M.W. Spong and M. Vidyasagar. Robust linear compensator design for nonlinear robotic control. *IEEE Trans. on Robotics and Automation*, RA-3(4):345–351, August 1987.

[171] R.J. Su. On the linear equivalents of nonlinear systems. *Syst. Control Lett.*, 2, 1981.

[172] L. Sweet and M.C. Good. Re-definition of the robot motion control problem: Effects of plant dynamics, drive system constraints, and user requirements. In *Proc. 23rd IEEE Conf. on Decision and Control*, pages 724–731, Las Vegas, December 1984.

[173] L. M. Sweet and M. C. Good. Redefinition of the robot motion control problem. *IEEE Control Systems Magazine*, 5(3):18–24, 1985.

[174] M. Takegaki and S. Arimoto. A new feedback method for dynamic control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102:119–125, 1981.

[175] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, Upper Saddle River, NJ, 1998.

[176] L. Tsai and A. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. In *Proc. ASME Mechanisms Conference*, Boston, October 1984.

[177] Gino van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999.

[178] Gino van den Bergen. *User's Guide to the SOLID Interference Detection Library.* Eindhoven University of Technology, Eindhoven, The Netherlands, 1999.

[179] Ravi N. Vanavar and Velupillai Sankaranarayanan. *Switched Finite Time Control of a Class of Underactuated Systems.* Springer Verlag, Berlin, 2006.

[180] M. Vidyasagar. *Nonlinear Systems Analysis, 2nd Ed.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[181] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *J. Dyn. Sys., Meas. Cont.*, December 1972.

[182] E. T. Whittaker. *Dynamics of Particles and Rigid Bodies.* Cambridge University Press, London, 1904.

[183] K. Y. Wichlund, O.J. Sordalen, and O. Egeland. Control of vehicles with second-order nonholonomic constraints: Underactuated vehicles. In *Proceedings of the European Control Conference*, pages 3086–3091, Rome, Italy, 1995.

[184] M. Wiklund, A. Kristenson, and K. J. Astrom. A new strategy for swinging up an inverted pendulum. In *Proceedings of the IFAC Symposium*, Sydney, Australia, 1993.

[185] W. Wonham. On pole assignment in multi-input controllable linear systems. *IEEE Trans. Aut. Cont*, 12(6):660–665, December 1967.

[186] Xin Xin. *Control Design and Analysis for Underactuated Robotic Systems.* Springer Verlag, London, 2014.

[187] T. Yoshikawa. Manipulability of robotic mechanisms. *International Journal of Robotics Research*, 4(2):3–9, 1985.

Index

Čapek, Karel, 1

accuracy, 10
Acrobot, 439, 445, 510
across variable, 353
adaptive control, 311, 327
adaptive inverse dynamics, 327
analytic approach, 36
angle convention, 443
angular momentum, 479, 490
angular velocity, 107, 108
annihilating codistribution, 484
anthropomorphic, 13
anti-windup, 308
apparent damping, 357
apparent inertia, 357
apparent stiffness, 357
application area, 9
approach vector, 85
arm singularity, 124
armature, 273
armature current, 274
armature inductance, 274
armature resistance, 274
armature voltage, 274
articulated (RRR), 9
artificial potential field, *see also* potential field
assembly robots, 9
asymptotic stability, 542
Atan2, 523
axis-angle representation, 53, 58

back emf, 273
back emf constant, 275
backlash, 272
balance control, 473
bang-bang trajectory, 260
Barbalat's lemma, 333, 514, 549
base frame, 20
base variables, 500
baseline distance, 402
basic homogeneous transformation, 63
basic rotation matrix, 42
basis, 526
    change of, 530
    reciprocal, 348
bilinear form, 349
blend time, 257
body-attached frame, 177
Brockett's theorem, 480, 509
Brunovsky canonical form, 420

camera calibration, 371, 555
    extrinsic camera parameters, 556
    intrinsic camera parameters, 557
camera coordinate frame, 369
capacitive environment, 354
capacitive impedance, 354
cart-pole system, 443, 460
Cartesian (PPP), 9
center of projection, 369
central limit theorem, 235
centrifugal force, 183
chained form, 499
characteristic polynomial, 418, 530

Chow's theorem, 480, 497
Christoffel symbols, 183
closed-loop, 8
codistribution, 412, 484
commutator, 274
compensator, 272
completely integrable, 416
compliance, 272
compliance frame, 349
computed torque, 25
computer interface, 10
computer vision, 1
configuration, 5, 217
configuration space, 5, 217
configuration space obstacle, 218
configuration space roadmap, 246
connectivity graph, 227
constraint, 347
    artificial, 349, 351
    holonomic, 165, 171, 461, 480
    natural, 349, 350
    nonholonomic, 171, 439, 479
    Pfaffian, 481
    unilateral, 439
    virtual, 461, 466
constraint frame, 349
continuous function, 539
continuous path control, 9
continuous path tracking, 271
control, 1
    adaptive, 311, 327, 332
    chattering, 326
    discontinuous, 325
    independent joint, 272
    inner loop, 320, 409
    inner-loop/outer-loop, 418
    inverse dynamics, 318
    linear quadratic optimal, 301
    multivariable, 311
    optimal, 334
    outer loop, 320, 356, 409, 418
    passivity-based, 329
    pointwise optimal, 335
    robust, 311, 331
    state feedback, 299
control architecture, 320
controllability, 299, 496
controllability matrix, 299
controllability rank condition, 498
controllable system, 299
controlled-invariant manifold, 466
controller resolution, 11
convex spatial decomposition, 227
Coriolis force, 183
cotangent space, 411
covector field, 411
critical eigenvalues, 544
cross product, 532
cross product of inertia, 179
current frame, 49, 52
cylindrical (RPP), 9

D'Alembert's principle, 174
DC motor, 167, 272
decoupling matrix, 462
degree of nonholonomy, 486
degree of underactuation, 440
degrees of freedom, 6
Denavit–Hartenberg convention, 21, 79
dexterous workspace, 7
DH convention, 79
diffeomorphism, 410, 420, 432
differential flatness, 506
differential drive robot, 491
direct method of Lyapunov, 26
direct-drive robot, 30, 272
directional derivative, 410, 413
distribution, 412
    annihilating, 484
    involutive, 416, 484

involutive closure of, 486
disturbance, 272
disturbance rejection, 272
dot product, 527
double integrator, 318, 456
driftless system, 479, 484
controllability of, 495
dual space, 411
dynamic extension, 493, 512
dynamic feedback linearization, 517
dynamics, 1, 24, 165
Newton–Euler formulation, 165

effort, 353
eigenvalues, 530
elbow, 14
elbow manipulator, 13
end effector, 12
end-of-arm tooling, 10
energy, 353
kinetic, 165
potential, 165, 167
environment, 345
capacitive, 354
classification of, 358
inertial, 354
resistive, 354
environment stiffness, 352
equilibrium, 449
equilibrium point, 542
estimation error, 302
Euler angles, 53, 90, 172
Euler–Lagrange equations, 165
Euler-Lagrange equations, 166
exponential coordinates, 59
exponential stability, 543, 546
external power source, 10
eye-in-hand configuration, 366

feedback linearizable, 419
feedback linearization, 27, 317, 409, 426, 461, 515
feedforward control, 25, 288
fiber variables, 500
Filippov solution, 326
filtration, 486
final value theorem, 285
five-bar linkage, 191
fixed frame, 50, 52
fixed point, 449, 542
fixed-camera configuration, 366
flat output, 507
flexibility, 292
flow, 353
focal center, 369
focal length, 370
force control, 26, 345
forward kinematics, 19, 21, 75
forward kinematics problem, 20
frame, 76
base, 76
current, 52
end-effector, 85
fixed, 52
inertial, 76
tool, 85
world, 76
friction, 272
Frobenius theorem, 27, 414, 422, 484

gain
derivative, 282
integral, 281
proportional, 281
gear
harmonic, 292
strain wave, 292
gear ratio, 276
gear train, 167
generalized coordinates, 167, 171
generalized force, 168, 175

generalized Voronoi diagram, 224
geometric nonlinear control, 409
global stability, 542, 543, 546
gradient, 449, 541
gravitational force, 167
group, 67
gyroscopic force, 200

Hamilton's principle, 166
Hamiltonian, 500
hand, 12
hardware/software trade-off, 271
harmonic gear, 292, 293
Hessian, 541
HOG feature detector, 377
home position, 95
homoclinic orbit, 468
homogeneous coordinates, 21
homogeneous representation, 63
homogeneous transformation, 21, 63
hopping robot, 490
Hurwitz matrix, 325, 544
Hurwitz polynomial, 289
hybrid control, 26
hybrid impedance control, 358

image feature, 373
image feature velocity, 378
image features, 369
image Jacobian, 379
image-plane coordinates, 370
impedance, 353, 354
    dual, 358
impedance control, 26, 356
impedance control, hybrid, 358
impedance operator, 353
implicit function theorem, 416
independent joint control, 272
industrial manipulator, 1
inertia
    apparent, 357
inertia matrix, 180
inertia tensor, 178
inertial environment, 354
inertial impedance, 354
integral manifold, 415
integrating factor, 482
integrator windup, 308
interaction matrix, 379, 380, 385
internal dynamics, 458, 460
inverse dynamics, 26, 317, 356, 409
inverse dynamics control, 318
inverse kinematics, 22, 141
inverse orientation kinematics, 144
inverse position kinematics, 144
inverted pendulum, 443
involutivity, 416

Jacobi identity, 533
Jacobian, 21, 101, 111, 130, 541
Jacobian matrix, 413
jerk, 256
joint
    prismatic, 5
    revolute, 5
joint angle, 79
joint elasticity, 292
joint flexibility, 292, 409, 423
joint variable, 5

Killing form, 349
kinematic car, 488
kinematic chain, 5, 75
kinematically redundant, 6
kinematics, 1
kinetic energy, 165, 166
Kinova robot, 6
Klein form, 349
KUKA 500 FORTEC robot, 2

Lagrange multiplier, 336, 493, 500
Lagrangian, 165, 167, 177

Lagrangian dynamics, 24
Laplace domain, 354
Laplace transform, 354
LaSalle's theorem, 469, 548
law of cosines, 23
Lie bracket, 412, 417
Lie derivative, 413
linear approximation, 544
linear controllability, 450
linearity-in-the-parameters, 197
linearly independent, 526
link length, 79
link offset, 79
link twist, 79
lower-actuated, 441
LSPB, 257
Lyapunov
    direct method, 544
Lyapunov design, 513
Lyapunov function, 545, 546
Lyapunov function candidate., 545
Lyapunov redesign, 323
Lyapunov's direct method, 544
Lyapunov's indirect method, 544

magnetic flux, 273
manifold, 410, 466
    integral, 415
manipulability ellipsoid, 133
manipulator
    cylindrical, 89, 91
    elbow, 87
    industrial, 1
    SCARA, 95
    spherical, 14
    Stanford, 93
matrix
    range, 531
matrix inverse, 530
matrix Lyapunov equation, 548
mechanical impedance, 353
mechatronics, 4
minimum phase, 290
mobile robot, 4, 479
mobility tensor, 356
motion perceptibility, 397
motion planning, 1
multivariable control, 311
multivariable system, 311

narrow passage problem, 248
network model, 353
Newton–Euler formulation, 198
nonassembly robots, 9
nonholonomic constraint, 439
nonlinear control, 25
nonservo controlled robots, 8
normal form, 458, 460
normal vector, 85
Norton equivalent, 355
Norton network, 355
null solution, 542
numerically controlled
        milling machines, 2
nutation, 122

observability, 302
observability matrix, 302
observable, 302
observer, 301
one-port network, 353
open-loop, 8
orthogonal complement, 528
orthogonal matrix, 40
orthonormal basis, 350
outer product, 533
output linearization, 461
overhead crane, 443

parallel manipulator, 9
parameter drift, 344
partial feedback linearization, 456

collocated, 456, 457
noncollocated, 456, 459
partitioned methods, 394
passivity, 329, 466
passivity property, 195
passivity-based adaptive control, 332
passivity-based control, 26, 329, 466
path planning, 24
PD control, 313
peg in hole, 9
Pendubot, 439, 446, 449, 454
perspective projection, 371
phase portrait, 468
pinhole lens model, 369
pitch, 56
pixel, 368
pixel coordinates, 371
point-to-point, 252
point-to-point control, 8
point-to-point motion, 271
port variable, 353
position control, 25
positive definite, 545
positive definite matrix, 181
positively invariant set, 548
potential energy, 165, 167, 449
potential field, 229
attractive potential, 230
conic well potential, 230
parabolic well potential, 230
repulsive potential, 231
workspace potential, 229
power, 353
precession, 122
primal-dual method, 336
principal moment of inertia, 179
principal point, 370
principle of virtual work, 166, 173
prismatic joint, 5
PRM, 246
probabilistic roadmap, 246
product of exponentials, 100
programmable robot, 30
projection ray, 385
proper rational function, 290
pseudoinverse, 459

quadratic form, 545
quadratic programming, 334
quaternion, 70

radially unbounded, 546
randomized methods, 234
range space, 531
rank, 528
rapidly-exploring random tree, 246, 250
reachable workspace, 7
Reaction-Wheel Pendulum, 447, 452, 460
reciprocity, 348
reference trajectory, 25
regressor, 197
relative degree, 461, 462
repeatability, 10
representational singularities, 122
resistive environment, 354
resistive impedance, 354
resolvability, 407
revolute, 13
revolute joint, 5
Riccati equation, 301
rigid motion, 19
robot, 1
definition of, 2
direct-drive, 272
flexible joint, 439
industrial, 1
mobile, 4
parallel, 18
underactuated, 440

robot geometry, 9
Robot Institute of America, 2
robota, 1
robotic system, 10
robust control, 311, 331
Rodrigues' formula, 61
roll, 56
roll-pitch-yaw angles, 53
root locus, 295
rotation matrix, 20, 39
rotor, 273
Routh-Hurwitz criterion, 288
RRT, 246, 250

sampling-based planner, 245
saturation, 286, 334, 470
scalar product, 527
SCARA (RRP), 9
SCARA robot, 14
Schur complement, 458
second moment matrix, 376
second-order normal form, 458, 460
sensor
    force, 346
    tactile, 346
    torque, 346
separation principle, 303
serial manipulator, 9
servo controlled robots, 8
set-point tracking, 284
shoulder elevation, 14
single-input/single-output system, 25
singular configuration, 22, 102, 123
singular value decomposition, 537
singular values, 537
singularity, 123
sinusoids, 500
SISO system, 25, 272
skew symmetry property, 194
skew-symmetric matrix, 59, 103
sliding vector, 85
sliding-mode control, 511
spherical (RRP), 9
spherical manipulator, 14
spherical wrist, 12, 90
spin, 122
square integrable function, 549
stability
    asymptotic, 542
    exponential, 543
    global, 546
stable equilibrium, 542
Stanford arm, 30
state space, 6, 542
stator, 273
steering, 500
stereo cameras, 402
strain gauge, 346
strain wave gear, 292
strong inertial coupling, 459
structure tensor, 376
superposition, 283
swingup control, 473
switching control, 511
switching time, 260
symmetric matrix, 181
synthetic approach, 36
system
    multivariable, 311
system type number, 285

tangent plane, 415
tangent space, 410
task space, 355
teach and playback mode, 252
teach pendant, 9, 11
teleoperator, 2
through variable, 353
Thévenin equivalent, 355
Thévenin network, 355
tool, 12
torque constant, 275

torque optimization, 333
tracking, 272
tracking and disturbance rejection, 25
tracking error, 281
trajectory, 252
trajectory generation, 24
trajectory planning, 24
trajectory tracking, 513
transformation
    basic homogeneous, 63
    homogeneous, 63
    matrix, 77
transformation matrix, 77
trapezoidal decomposition, 227
trapezoidal velocity profile, 257
two-argument arctangent function, 54, 523
type-1 system, 285

ultimate boundedness, 323
uncertainty, 322
underactuated mechanical system, 439
underactuated robot, 439
underactuation, 439
unicycle, 479, 487, 510
uniform continuity, 539
uniform ultimate boundedness, 332, 550
unilateral constraint, 439
Unimate, 30
unstable equilibrium, 542
upper-actuated, 440

vanishing point, 373, 557
vector field, 411, 541
    complete, 495
vector product, 532
velocity
    angular, 107
velocity kinematics, 21
via points, 252, 261
virtual displacement, 130, 172
virtual holonomic constraint, 466
virtual work, 130, 165, 166
visibility graph, 222
vision-based control, 26, 365
visual servo control
    image-based, 366, 368, 386
    partitioned methods, 368
    position-based, 367

waist rotation, 14
workspace, 7
    dexterous, 7
    reachable, 7
world frame, 20
wrist, 12
wrist center, 90, 144
wrist center point, 12
wrist singularity, 124

yaw, 56

zero dynamics, 461, 462, 517
zero dynamics manifold, 462

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.