

# *Robot Modeling and Control*

First Edition

**Mark W. Spong, Seth Hutchinson, and M. Vidyasagar**

**JOHN WILEY & SONS, INC.**

New York / Chichester / Weinheim / Brisbane / Singapore / Toronto



# *Preface*

TO APPEAR



# *Contents*

|  |            |
|--|------------|
| <i>Preface</i>   | <i>i</i>   |
| <i>TABLE OF CONTENTS</i>                                 | <i>ii</i>  |
| <b>1 INTRODUCTION</b>                                    | <b>1</b>   |
| <b>1.1 Mathematical Modeling of Robots</b>               | <b>3</b>   |
| <b>1.1.1 Symbolic Representation of Robots</b>           | <b>3</b>   |
| <b>1.1.2 The Configuration Space</b>                     | <b>4</b>   |
| <b>1.1.3 The State Space</b>                             | <b>5</b>   |
| <b>1.1.4 The Workspace</b>                               | <b>5</b>   |
| <b>1.2 Robots as Mechanical Devices</b>                  | <b>5</b>   |
| <b>1.2.1 Classification of Robotic Manipulators</b>      | <b>5</b>   |
| <b>1.2.2 Robotic Systems</b>                             | <b>7</b>   |
| <b>1.2.3 Accuracy and Repeatability</b>                  | <b>7</b>   |
| <b>1.2.4 Wrists and End-Effectors</b>                    | <b>8</b>   |
| <b>1.3 Common Kinematic Arrangements of Manipulators</b> | <b>9</b>   |
| <b>1.3.1 Articulated manipulator (RRR)</b>               | <b>10</b>  |
| <b>1.3.2 Spherical Manipulator (RRP)</b>                 | <b>11</b>  |
| <b>1.3.3 SCARA Manipulator (RRP)</b>                     | <b>12</b>  |
|  | <i>iii</i> |

|       |  |    |
|-------|--|----|
| 1.3.4 | <i>Cylindrical Manipulator (RPP)</i>                         | 13 |
| 1.3.5 | <i>Cartesian manipulator (PPP)</i>                           | 14 |
| 1.3.6 | <i>Parallel Manipulator</i>                                  | 15 |
| 1.4   | <i>Outline of the Text</i>                                   | 16 |
| 1.5   | <i>Chapter Summary</i>                                       | 24 |
|       | <i>Problems</i>  | 26 |
| 2     | <i>RIGID MOTIONS AND HOMOGENEOUS TRANSFORMATIONS</i>         |    |
|       |  | 29 |
| 2.1   | <i>Representing Positions</i>                                | 30 |
| 2.2   | <i>Representing Rotations</i>                                | 32 |
| 2.2.1 | <i>Rotation in the plane</i>                                 | 32 |
| 2.2.2 | <i>Rotations in three dimensions</i>                         | 35 |
| 2.3   | <i>Rotational Transformations</i>                            | 37 |
| 2.3.1 | <i>Similarity Transformations</i>                            | 41 |
| 2.4   | <i>Composition of Rotations</i>                              | 42 |
| 2.4.1 | <i>Rotation with respect to the current frame</i>            | 42 |
| 2.4.2 | <i>Rotation with respect to the fixed frame</i>              | 44 |
| 2.5   | <i>Parameterizations of Rotations</i>                        | 46 |
| 2.5.1 | <i>Euler Angles</i>  | 47 |
| 2.5.2 | <i>Roll, Pitch, Yaw Angles</i>                               | 49 |
| 2.5.3 | <i>Axis/Angle Representation</i>                             | 50 |
| 2.6   | <i>Rigid Motions</i>   | 53 |
| 2.7   | <i>Homogeneous Transformations</i>                           | 54 |
| 2.8   | <i>Chapter Summary</i>                                       | 57 |
| 3     | <i>FORWARD AND INVERSE KINEMATICS</i>                        |    |
| 3.1   | <i>Kinematic Chains</i>                                      | 65 |
| 3.2   | <i>Forward Kinematics: The Denavit-Hartenberg Convention</i> | 68 |
| 3.2.1 | <i>Existence and uniqueness issues</i>                       | 69 |
| 3.2.2 | <i>Assigning the coordinate frames</i>                       | 72 |
| 3.2.3 | <i>Examples</i>  | 75 |
| 3.3   | <i>Inverse Kinematics</i>                                    | 85 |
| 3.3.1 | <i>The General Inverse Kinematics Problem</i>                | 85 |
| 3.3.2 | <i>Kinematic Decoupling</i>                                  | 87 |
| 3.3.3 | <i>Inverse Position: A Geometric Approach</i>                | 89 |
| 3.3.4 | <i>Inverse Orientation</i>                                   | 97 |
| 3.3.5 | <i>Examples</i>  | 98 |

|   |            |
|---|------------|
| <i>3.4 Chapter Summary</i>  | 100        |
| <i>3.5 Notes and References</i>                                     | 102        |
| <i>Problems</i>   | 103        |
| <br>  |            |
| <b>4 VELOCITY KINEMATICS – THE MANIPULATOR JACOBIAN</b>             | <b>113</b> |
| <i>4.1 Angular Velocity: The Fixed Axis Case</i>                    | 114        |
| <i>4.2 Skew Symmetric Matrices</i>                                  | 115        |
| <i>4.2.1 Properties of Skew Symmetric Matrices</i>                  | 116        |
| <i>4.2.2 The Derivative of a Rotation Matrix</i>                    | 117        |
| <i>4.3 Angular Velocity: The General Case</i>                       | 118        |
| <i>4.4 Addition of Angular Velocities</i>                           | 119        |
| <i>4.5 Linear Velocity of a Point Attached to a Moving Frame</i>    | 121        |
| <i>4.6 Derivation of the Jacobian</i>                               | 122        |
| <i>4.6.1 Angular Velocity</i>                                       | 123        |
| <i>4.6.2 Linear Velocity</i>  | 124        |
| <i>4.6.3 Combining the Angular and Linear Jacobians</i>             | 126        |
| <i>4.7 Examples</i>   | 127        |
| <i>4.8 The Analytical Jacobian</i>                                  | 131        |
| <i>4.9 Singularities</i>  | 132        |
| <i>4.9.1 Decoupling of Singularities</i>                            | 133        |
| <i>4.9.2 Wrist Singularities</i>                                    | 134        |
| <i>4.9.3 Arm Singularities</i>                                      | 134        |
| <i>4.10 Inverse Velocity and Acceleration</i>                       | 139        |
| <i>4.11 Manipulability</i>  | 141        |
| <i>4.12 Chapter Summary</i>   | 144        |
| <i>Problems</i>   | 146        |
| <br>  |            |
| <b>5 PATH AND TRAJECTORY PLANNING</b>                               | <b>149</b> |
| <i>5.1 The Configuration Space</i>                                  | 150        |
| <i>5.2 Path Planning Using Configuration Space Potential Fields</i> | 154        |
| <i>5.2.1 The Attractive Field</i>                                   | 154        |
| <i>5.2.2 The Repulsive field</i>                                    | 156        |
| <i>5.2.3 Gradient Descent Planning</i>                              | 157        |
| <i>5.3 Planning Using Workspace Potential Fields</i>                | 158        |
| <i>5.3.1 Defining Workspace Potential Fields</i>                    | 159        |

|       |   |     |
|-------|---|-----|
| 5.3.2 | <i>Mapping workspace forces to joint forces and torques</i> | 161 |
| 5.3.3 | <i>Motion Planning Algorithm</i>                            | 165 |
| 5.4   | <i>Using Random Motions to Escape Local Minima</i>          | 166 |
| 5.5   | <i>Probabilistic Roadmap Methods</i>                        | 167 |
| 5.5.1 | <i>Sampling the configuration space</i>                     | 169 |
| 5.5.2 | <i>Connecting Pairs of Configurations</i>                   | 169 |
| 5.5.3 | <i>Enhancement</i>  | 170 |
| 5.5.4 | <i>Path Smoothing</i>                                       | 170 |
| 5.6   | <i>trajectory planning</i>                                  | 171 |
| 5.6.1 | <i>Trajectories for Point to Point Motion</i>               | 173 |
| 5.6.2 | <i>Trajectories for Paths Specified by Via Points</i>       | 182 |
| 5.7   | <i>Historical Perspective</i>                               | 184 |
|       | <i>Problems</i>   | 186 |
| 6     | <i>DYNAMICS</i>   | 187 |
| 6.1   | <i>The Euler-Lagrange Equations</i>                         | 188 |
| 6.1.1 | <i>One Dimensional System</i>                               | 188 |
| 6.1.2 | <i>The General Case</i>                                     | 190 |
| 6.2   | <i>General Expressions for Kinetic and Potential Energy</i> | 196 |
| 6.2.1 | <i>The Inertia Tensor</i>                                   | 197 |
| 6.2.2 | <i>Kinetic Energy for an <math>n</math>-Link Robot</i>      | 199 |
| 6.2.3 | <i>Potential Energy for an <math>n</math>-Link Robot</i>    | 200 |
| 6.3   | <i>Equations of Motion</i>                                  | 200 |
| 6.4   | <i>Some Common Configurations</i>                           | 202 |
| 6.5   | <i>Properties of Robot Dynamic Equations</i>                | 211 |
| 6.5.1 | <i>The Skew Symmetry and Passivity Properties</i>           | 212 |
| 6.5.2 | <i>Bounds on the Inertia Matrix</i>                         | 213 |
| 6.5.3 | <i>Linearity in the Parameters</i>                          | 214 |
| 6.6   | <i>Newton-Euler Formulation</i>                             | 215 |
| 6.7   | <i>Planar Elbow Manipulator Revisited</i>                   | 222 |
|       | <i>Problems</i>   | 225 |
| 7     | <i>INDEPENDENT JOINT CONTROL</i>                            | 229 |
| 7.1   | <i>Introduction</i>   | 229 |
| 7.2   | <i>Actuator Dynamics</i>                                    | 231 |

|       |  |     |
|-------|--|-----|
| 7.3   | <i>Set-Point Tracking</i>                      | 237 |
| 7.3.1 | <i>PD Compensator</i>                          | 238 |
| 7.3.2 | <i>Performance of PD Compensators</i>          | 239 |
| 7.3.3 | <i>PID Compensator</i>                         | 240 |
| 7.3.4 | <i>Saturation</i>                              | 242 |
| 7.4   | <i>Feedforward Control and Computed Torque</i> | 244 |
| 7.5   | <i>Drive Train Dynamics</i>                    | 248 |
| 7.6   | <i>State Space Design</i>                      | 251 |
| 7.6.1 | <i>State Feedback Compensator</i>              | 254 |
| 7.6.2 | <i>Observers</i>                               | 256 |
|       | <i>Problems</i>                                | 259 |
| 8     | <i>MULTIVARIABLE CONTROL</i>                   | 263 |
| 8.1   | <i>Introduction</i>                            | 263 |
| 8.2   | <i>PD Control Revisited</i>                    | 264 |
| 8.3   | <i>Inverse Dynamics</i>                        | 266 |
| 8.3.1 | <i>Task Space Inverse Dynamics</i>             | 269 |
| 8.4   | <i>Robust and Adaptive Motion Control</i>      | 271 |
| 8.4.1 | <i>Robust Feedback Linearization</i>           | 271 |
| 8.4.2 | <i>Passivity Based Robust Control</i>          | 275 |
| 8.4.3 | <i>Passivity Based Adaptive Control</i>        | 277 |
|       | <i>Problems</i>                                | 279 |
| 9     | <i>FORCE CONTROL</i>                           | 281 |
| 9.1   | <i>Introduction</i>                            | 281 |
| 9.2   | <i>Coordinate Frames and Constraints</i>       | 282 |
| 9.2.1 | <i>Natural and Artificial Constraints</i>      | 284 |
| 9.3   | <i>Network Models and Impedance</i>            | 285 |
| 9.3.1 | <i>Impedance Operators</i>                     | 288 |
| 9.3.2 | <i>Classification of Impedance Operators</i>   | 288 |
| 9.3.3 | <i>Thévenin and Norton Equivalents</i>         | 289 |
| 9.4   | <i>Task Space Dynamics and Control</i>         | 290 |
| 9.4.1 | <i>Static Force/Torque Relationships</i>       | 290 |
| 9.4.2 | <i>Task Space Dynamics</i>                     | 291 |
| 9.4.3 | <i>Impedance Control</i>                       | 292 |
| 9.4.4 | <i>Hybrid Impedance Control</i>                | 293 |
|       | <i>Problems</i>                                | 297 |
| 10    | <i>GEOMETRIC NONLINEAR CONTROL</i>             | 299 |

|        |  |     |
|--------|--|-----|
| 10.1   | <i>Introduction</i>  | 299 |
| 10.2   | <i>Background</i>  | 300 |
| 10.2.1 | <i>The Frobenius Theorem</i>                                   | 304 |
| 10.3   | <i>Feedback Linearization</i>                                  | 306 |
| 10.4   | <i>Single-Input Systems</i>                                    | 308 |
| 10.5   | <i>Feedback Linearization for <math>n</math>-Link Robots</i>   | 315 |
| 10.6   | <i>Nonholonomic Systems</i>                                    | 318 |
| 10.6.1 | <i>Involutivity and Holonomy</i>                               | 319 |
| 10.6.2 | <i>Driftless Control Systems</i>                               | 320 |
| 10.6.3 | <i>Examples of Nonholonomic Systems</i>                        | 320 |
| 10.7   | <i>Chow's Theorem and Controllability of Driftless Systems</i> | 324 |
|        | <i>Problems</i>  | 328 |
| 11     | <i>COMPUTER VISION</i>   | 331 |
| 11.1   | <i>The Geometry of Image Formation</i>                         | 332 |
| 11.1.1 | <i>The Camera Coordinate Frame</i>                             | 332 |
| 11.1.2 | <i>Perspective Projection</i>                                  | 333 |
| 11.1.3 | <i>The Image Plane and the Sensor Array</i>                    | 334 |
| 11.2   | <i>Camera Calibration</i>                                      | 334 |
| 11.2.1 | <i>Extrinsic Camera Parameters</i>                             | 335 |
| 11.2.2 | <i>Intrinsic Camera Parameters</i>                             | 335 |
| 11.2.3 | <i>Determining the Camera Parameters</i>                       | 336 |
| 11.3   | <i>Segmentation by Thresholding</i>                            | 338 |
| 11.3.1 | <i>A Brief Statistics Review</i>                               | 339 |
| 11.3.2 | <i>Automatic Threshold Selection</i>                           | 341 |
| 11.4   | <i>Connected Components</i>                                    | 346 |
| 11.5   | <i>Position and Orientation</i>                                | 348 |
| 11.5.1 | <i>Moments</i>   | 349 |
| 11.5.2 | <i>The Centroid of an Object</i>                               | 349 |
| 11.5.3 | <i>The Orientation of an Object</i>                            | 350 |
|        | <i>Problems</i>  | 353 |
| 12     | <i>VISION-BASED CONTROL</i>                                    | 355 |
| 12.1   | <i>Approaches to vision based-control</i>                      | 356 |
| 12.1.1 | <i>Where to put the camera</i>                                 | 356 |
| 12.1.2 | <i>How to use the image data</i>                               | 357 |
| 12.2   | <i>Camera Motion and Interaction Matrix</i>                    | 357 |
| 12.2.1 | <i>Interaction matrix vs. Image Jacobian</i>                   | 358 |

|  |     |
|--|-----|
| <i>12.3 The interaction matrix for points</i>                        | 359 |
| <i>12.3.1 Velocity of a fixed point relative to a moving camera</i>  | 360 |
| <i>12.3.2 Constructing the Interaction Matrix</i>                    | 361 |
| <i>12.3.3 Properties of the Interaction Matrix for Points</i>        | 363 |
| <i>12.3.4 The Interaction Matrix for Multiple Points</i>             | 363 |
| <i>12.4 Image-Based Control Laws</i>                                 | 364 |
| <i>12.4.1 Computing Camera Motion</i>                                | 365 |
| <i>12.4.2 Proportional Control Schemes</i>                           | 366 |
| <i>12.5 The relationship between end effector and camera motions</i> | 367 |
| <i>12.6 Partitioned Approaches</i>                                   | 369 |
| <i>12.7 Motion Perceptibility</i>                                    | 372 |
| <i>12.8 Chapter Summary</i>  | 374 |
| <i>Problems</i>  | 375 |
| <br>   |     |
| <i>Appendix A Geometry and Trigonometry</i>                          | 377 |
| <i>A.1 Trigonometry</i>  | 377 |
| <i>A.1.1 Atan2</i>   | 377 |
| <i>A.1.2 Reduction formulas</i>                                      | 378 |
| <i>A.1.3 Double angle identitites</i>                                | 378 |
| <i>A.1.4 Law of cosines</i>  | 378 |
| <br>   |     |
| <i>Appendix B Linear Algebra</i>                                     | 379 |
| <i>B.1 Differentiation of Vectors</i>                                | 381 |
| <i>B.2 Linear Independence</i>                                       | 382 |
| <i>B.3 Change of Coordinates</i>                                     | 383 |
| <i>B.4 Eigenvalues and Eigenvectors</i>                              | 383 |
| <i>B.5 Singular Value Decomposition (SVD)</i>                        | 383 |
| <br>   |     |
| <i>Appendix C Lyapunov Stability</i>                                 | 387 |
| <i>C.0.1 Quadratic Forms and Lyapunov Functions</i>                  | 389 |
| <i>C.0.2 Lyapunov Stability</i>                                      | 390 |
| <i>C.0.3 Lyapunov Stability for Linear Systems</i>                   | 391 |
| <i>C.0.4 LaSalle's Theorem</i>                                       | 392 |
| <br>   |     |
| <i>Appendix D State Space Theory of Dynamical Systems</i>            | 393 |

x CONTENTS

|   |            |
|---|------------|
| <i>D.0.5 State Space Representation of Linear Systems</i> | <i>395</i> |
| <i>References</i>   | <i>397</i> |
| <i>Index</i>  | <i>403</i> |

# 1

---

## INTRODUCTION

**R**obotics is a relatively young field of modern technology that crosses traditional engineering boundaries. Understanding the complexity of robots and their applications 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.

This book is concerned with fundamentals of robotics, including **kinematics**, **dynamics**, **motion planning**, **computer vision**, and **control**. Our goal is to provide a complete introduction to the most important concepts in these subjects as applied to industrial robot manipulators, mobile robots, and other mechanical systems. A complete treatment of the discipline of robotics would require several volumes. Nevertheless, at the present time, the majority of robot applications deal with industrial robot arms operating in structured factory environments so that a first introduction to the subject of robotics must include a rigorous treatment of the topics in this text.

The term **robot** was first introduced into our vocabulary by the Czech playwright Karel Capek in his 1920 play *Rossum's Universal Robots*, the word *robota* being the Czech word for work. Since then the term has been applied to a great variety of mechanical devices, such as teleoperators, underwater vehicles, autonomous land rovers, etc. Virtually anything that operates with some degree of autonomy, usually under computer control, has at some point been called a robot. In this text the term robot will mean a computer controlled industrial manipulator of the type shown in Figure 1.1. This type of robot is



*Fig. 1.1* The ABB IRB6600 Robot. Photo courtesy of ABB.

essentially a mechanical arm operating under computer control. Such devices, though far from the robots of science fiction, are nevertheless extremely complex electro-mechanical 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 of robots. It is the computer brain that gives the robot its utility and adaptability. The so-called robotics revolution is, in fact, part of the larger computer revolution.

Even this restricted version 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 robot, as we have defined it, 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 machining of certain items, such as components of high performance aircraft. The first

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, where the robot merely attends 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.

It should be pointed out that the important applications of robots are by no means limited to those industrial jobs where the robot is directly replacing a human worker. 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.

## 1.1 MATHEMATICAL MODELING OF ROBOTS

While robots are themselves mechanical systems, in this text we will be primarily concerned with developing and manipulating mathematical models for robots. In particular, we will develop methods to represent basic geometric aspects of robotic manipulation, dynamic aspects of manipulation, and the various sensors available in modern robotic systems. Equipped with these mathematical models, we will be able to develop methods for planning and controlling robot motions to perform specified tasks. Here we describe some of the basic ideas that are common in developing mathematical models for robot manipulators.

### 1.1.1 Symbolic Representation of Robots

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 1.2. For example, a three-link arm with three revolute joints is 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

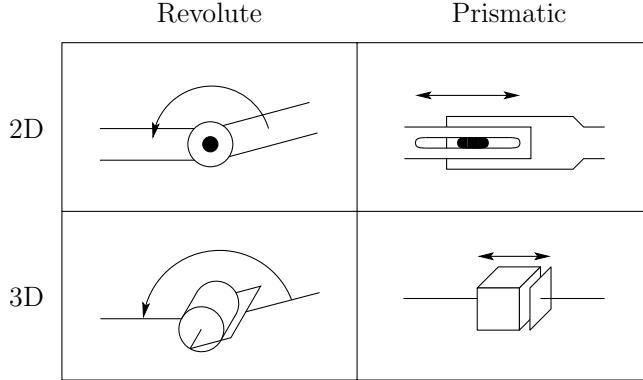


Fig. 1.2 Symbolic representation of robot joints.

**joint variables**, denoted by  $\theta$  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 possible configurations is called the **configuration space**. In our case, 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 assumed to be fixed. Therefore, in this text, 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  $q_1 \dots q_n$ , 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 DOF. A rigid object in three-dimensional space has six DOF: three for **positioning** and three for **orientation** (e.g., roll, pitch and yaw angles). 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 environment with arbitrary orientation. Certain applications such as reaching around or behind obstacles may require more than six DOF. A manipulator having more than six links is referred to as a **kinematically redundant** manipulator. The difficulty of controlling a manipulator increases rapidly with the number of links.

### 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 input, are sufficient to determine any future state of the manipulator. The **state space** is 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 joint velocities  $\dot{q}$  (acceleration is related to the derivative of joint velocities). We typically represent the state as a vector  $x = (q, \dot{q})^T$ . 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  $360^\circ$  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 implemented), 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**, or way in which the joints are actuated, their **geometry**, or kinematic structure, their intended **application area**, or their **method of control**. Such classification is useful primarily in order to determine which robot is right for a given task. For example, a hydraulic robot would not be suitable for food handling or clean room applications. We explain this in more detail below.

**Power Source.** Typically, 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-servo 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.

**Application Area.** Robots are often classified by application into **assembly** and **non-assembly robots**. Assembly robots tend to be small, electrically driven and either revolute or SCARA (described below) in design. The main nonassembly application areas to date have been in welding, spray painting, material handling, and machine loading and unloading.

**Method of Control.** Robots are classified by control method into **servo** and **non-servo** robots. The earliest robots were non-servo robots. These robots are essentially open-loop devices whose movement is limited to predetermined mechanical stops, and they are useful primarily for materials transfer. In fact, according to the definition given previously, 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.

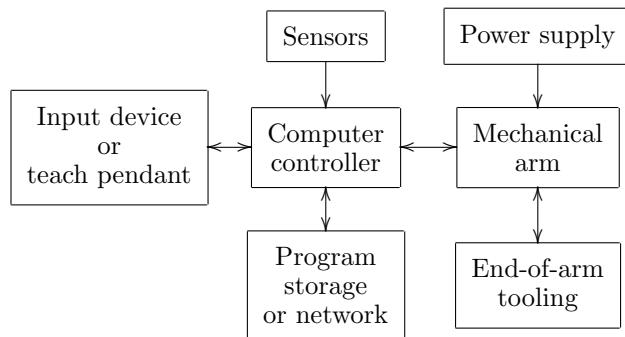
Servo controlled robots are further classified according to the method that the controller uses to guide the end-effector. The simplest type of robot in this class is the **point-to-point** robot. A point-to-point robot can be taught a discrete set of points but there is no control on the path of the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

**Geometry.** Most industrial manipulators at the present time have six or fewer degrees-of-freedom. 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.

Each of these five manipulator arms are **serial link** robots. 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.

### 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 1.3, which consists of the **arm**, **external power**



*Fig. 1.3 Components of a robotic system.*

**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.

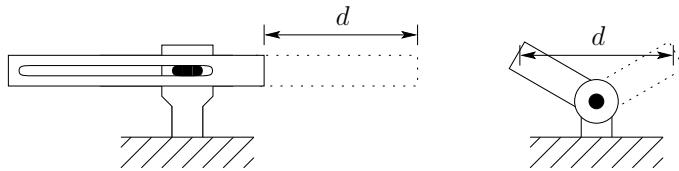
### 1.2.3 Accuracy and Repeatability

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 in most cases 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 must rely on the assumed geometry of the manipulator and its rigidity to infer (i.e., 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 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 by the tip 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 1.4 shows that for the same range of motion,



*Fig. 1.4 Linear vs. rotational link 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.

#### 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. The spherical wrist is represented symbolically in Figure 1.5.

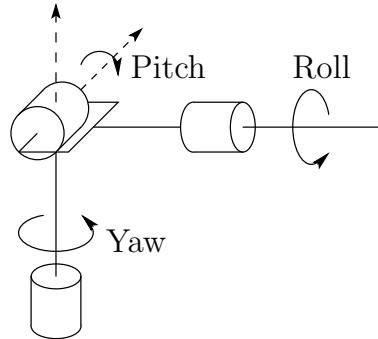


Fig. 1.5 Structure of a spherical wrist.

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

It has been said that a robot is only as good as its **hand** or **end-effector**. The arm and wrist assemblies of a robot are used primarily for positioning the end-effector and any tool it may carry. It is the end-effector or tool that actually performs the work. The simplest type of end-effectors are grippers, which usually are 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 to tools that can be rapidly changed as the task dictates. There is also much research on the development of anthropomorphic hands. Such hands have been developed both for prosthetic use and for use in manufacturing. Since we are concerned with the analysis and control of the manipulator itself and not in the particular application or end-effector, we will not discuss end-effector design or the study of grasping and manipulation.

### 1.3 COMMON KINEMATIC ARRANGEMENTS OF MANIPULATORS

Although there are many possible ways use prismatic and revolute joints to construct kinematic chains, in practice only a few of these are commonly used. Here we briefly describe several arrangements that are most typical.

### 1.3.1 Articulated manipulator (RRR)

The articulated manipulator is also called a **revolute**, or **anthropomorphic** manipulator. The ABB IRB1400 articulated arm is shown in Figure 1.6. A



*Fig. 1.6* The ABB IRB1400 Robot. Photo courtesy of ABB.

common revolute joint design is the **parallelogram linkage** such as the Motoman SK16, shown in Figure 1.7. In both of these arrangements joint axis



*Fig. 1.7* The Motoman SK16 manipulator.

$z_2$  is parallel to  $z_1$  and both  $z_1$  and  $z_2$  are perpendicular to  $z_0$ . This kind of manipulator is known as an elbow manipulator. The structure and terminology associated with the elbow manipulator are shown in Figure 1.8. Its workspace is shown in Figure 1.9.

The revolute manipulator provides for relatively large freedom of movement in a compact space. The parallelogram linkage, although typically less dexterous than the elbow manipulator manipulator, nevertheless has several advantages that make it an attractive and popular design. The most notable feature of the

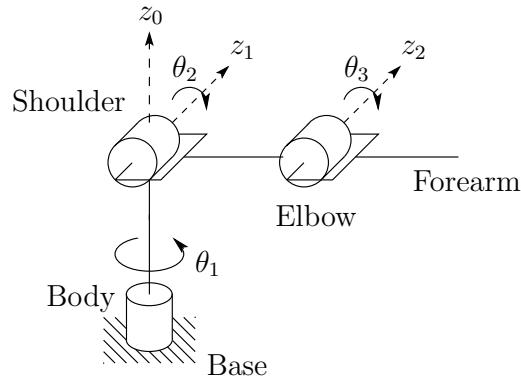


Fig. 1.8 Structure of the elbow manipulator.

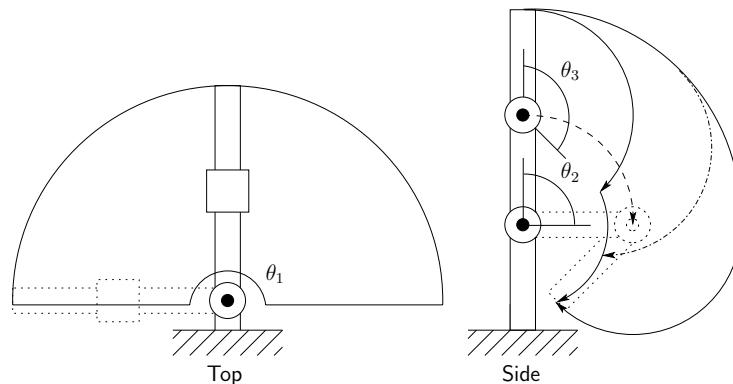
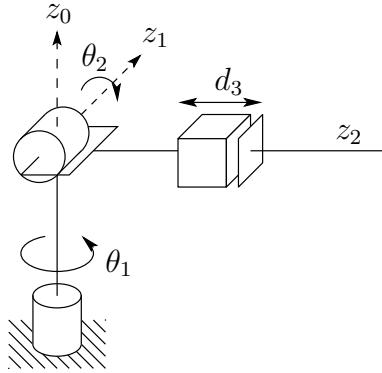


Fig. 1.9 Workspace of the elbow manipulator.

parallelogram linkage manipulator is that the actuator for joint 3 is located on link 1. Since the weight of the motor is born by link 1, links 2 and 3 can be made more lightweight and the motors themselves can be less powerful. Also the dynamics of the parallelogram manipulator are simpler than those of the elbow manipulator, thus making it easier to control.

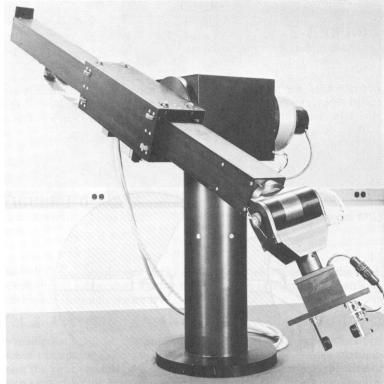
### 1.3.2 Spherical Manipulator (RRP)

By replacing the third or elbow joint in the revolute manipulator by a prismatic joint one obtains the spherical manipulator shown in Figure 1.10. The term **spherical manipulator** derives from the fact that the spherical coordinates defining the position of the end-effector with respect to a frame whose origin lies at the intersection of the three  $z$  axes are the same as the first three joint variables. Figure 1.11 shows the Stanford Arm, one of the most well-



*Fig. 1.10* The spherical manipulator.

known spherical robots. The workspace of a spherical manipulator is shown in

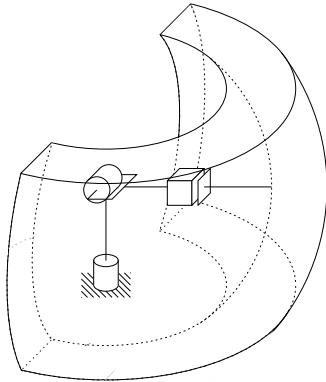


*Fig. 1.11* The Stanford Arm. Photo courtesy of the Coordinated Science Lab, University of Illinois at Urbana-Champaign.

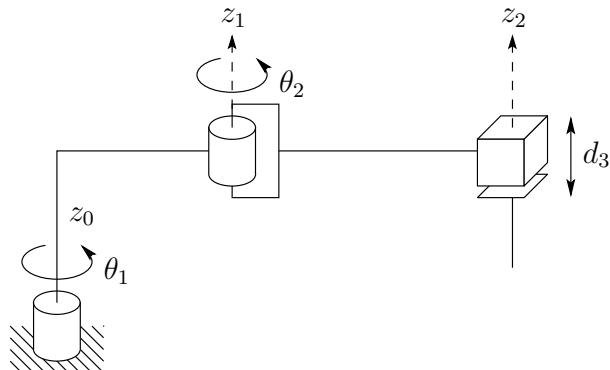
Figure 1.12.

### 1.3.3 SCARA Manipulator (RRP)

The **SCARA** arm (for Selective Compliant Articulated Robot for Assembly) shown in Figure 1.13 is a popular manipulator, which, as its name suggests, is tailored for 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 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 Epson E2L653S, a manipulator of this type. The SCARA manipulator workspace is shown in Figure 1.15.



*Fig. 1.12* Workspace of the spherical manipulator.



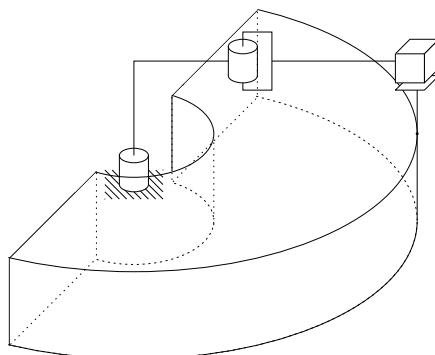
*Fig. 1.13* The SCARA (Selective Compliant Articulated Robot for Assembly).

#### 1.3.4 Cylindrical Manipulator (RPP)

The cylindrical manipulator is shown in Figure 1.16. 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. A cylindrical robot, the Seiko RT3300, is shown in Figure 1.17, with its workspace shown in Figure 1.18.



*Fig. 1.14* The Epson E2L653S SCARA Robot. Photo Courtesy of Epson.

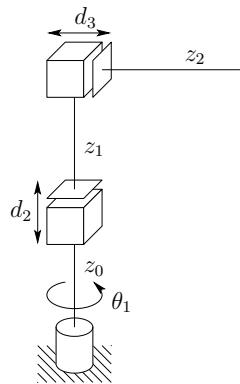


*Fig. 1.15* Workspace of the SCARA manipulator.

### 1.3.5 Cartesian manipulator (PPP)

A manipulator whose first three joints are prismatic is known as a Cartesian manipulator, shown in Figure 1.19.

For the Cartesian manipulator the joint variables 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. An example of a Cartesian robot, from Epson-Seiko, is shown in Figure 1.20. The workspace of a Cartesian manipulator is shown in Figure 1.21.



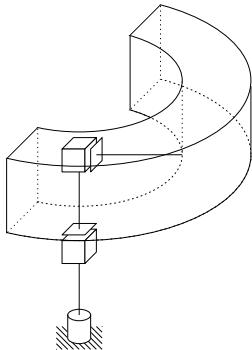
*Fig. 1.16* The cylindrical manipulator.



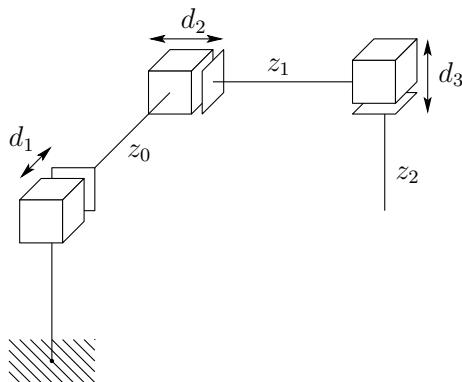
*Fig. 1.17* The Seiko RT3300 Robot. Photo courtesy of Seiko.

### 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 independent kinematic chains connecting the base to the end-effector. Figure 1.22 shows the ABB IRB 940 Tricept 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.



*Fig. 1.18* Workspace of the cylindrical manipulator.



*Fig. 1.19* The Cartesian manipulator.

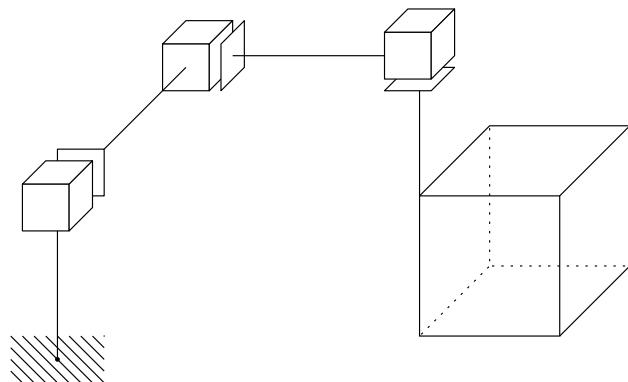
#### 1.4 OUTLINE OF THE TEXT

A typical application involving an industrial manipulator is shown in Figure 1.23. The manipulator is shown with a grinding tool that it must use to remove a certain amount of metal from a surface. In the present text we are concerned with the following question: *What are the basic issues to be resolved and what must we learn in order to be able to program a robot to perform such tasks?*

The ability to answer this question for a full six degree-of-freedom manipulator represents the goal of the present text. The answer is too complicated to be presented at this point. We can, however, use the simple two-link planar mechanism to illustrate some of the major issues involved and to preview the topics covered in this text.



*Fig. 1.20* The Epson Cartesian Robot. Photo courtesy of Epson.



*Fig. 1.21* Workspace of the Cartesian manipulator.

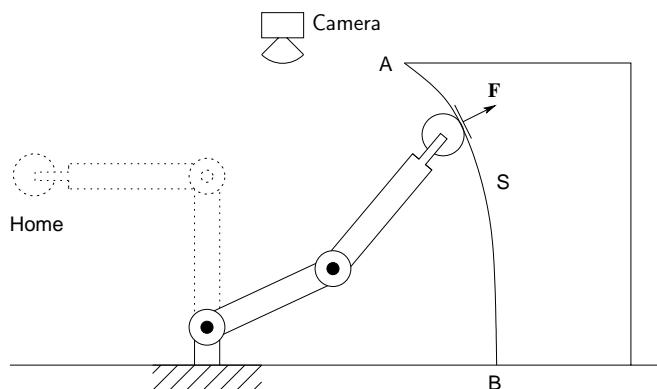
Suppose we wish to move the manipulator from its **home** position to position *A*, from which point the robot is to follow the contour of the surface *S* to the point *B*, at constant velocity, while maintaining a prescribed force *F* normal to the surface. In so doing the robot will cut or grind the 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 remainder of the text.

### Forward Kinematics

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 give some background on repre-



*Fig. 1.22* The ABB IRB940 Tricept Parallel Robot. Photo courtesy of ABB.



*Fig. 1.23* Two-link planar robot example.

sentations of coordinate systems and transformations among various coordinate systems.

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  $\theta_1$  and  $\theta_2$ . We also need therefore to express the positions  $A$  and  $B$  in terms of these joint angles. This leads 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 1.24. The coordinates  $(x, y)$  of the tool are expressed in this

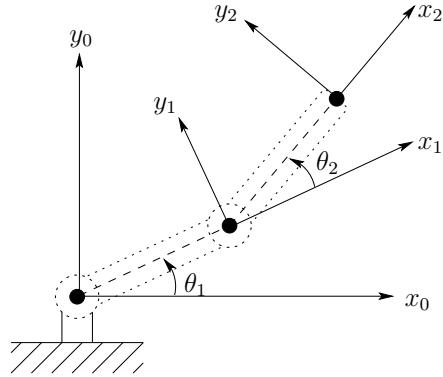


Fig. 1.24 Coordinate frames for two-link planar robot.

coordinate frame as

$$x = x_2 = \alpha_1 \cos \theta_1 + \alpha_2 \cos(\theta_1 + \theta_2) \quad (1.1)$$

$$y = y_2 = \alpha_1 \sin \theta_1 + \alpha_2 \sin(\theta_1 + \theta_2) \quad (1.2)$$

in which  $\alpha_1$  and  $\alpha_2$  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,

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

which we may combine into an **orientation matrix**

$$\begin{bmatrix} x_2 \cdot x_0 & y_2 \cdot x_0 \\ x_2 \cdot y_0 & y_2 \cdot y_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix} \quad (1.3)$$

Equations (1.1), (1.2) and (1.3) are called the **forward kinematic equations** for this arm. For a six degree-of-freedom 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** to simplify the transformation among coordinate frames.

### Inverse Kinematics

Now, given the joint angles  $\theta_1, \theta_2$  we can determine the end-effector coordinates  $x$  and  $y$ . In order to command the robot to move to location  $A$  we need the

inverse; that is, we need the joint variables  $\theta_1, \theta_2$  in terms of the  $x$  and  $y$  coordinates of  $A$ . This is the problem of **inverse kinematics**. In other words, given  $x$  and  $y$  in the forward kinematic Equations (1.1) and (1.2), we wish to solve for the joint angles. 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 1.25, the so-called **elbow up**

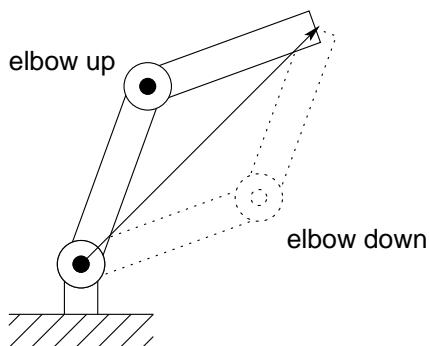


Fig. 1.25 Multiple inverse kinematic solutions.

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 1-25).

Consider the diagram of Figure 1.26. Using the **Law of Cosines** we see that

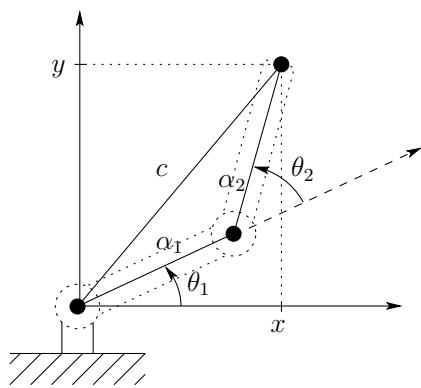


Fig. 1.26 Solving for the joint angles of a two-link planar arm.

the angle  $\theta_2$  is given by

$$\cos \theta_2 = \frac{x^2 + y^2 - \alpha_1^2 - \alpha_2^2}{2\alpha_1\alpha_2} := D \quad (1.4)$$

We could now determine  $\theta_2$  as

$$\theta_2 = \cos^{-1}(D) \quad (1.5)$$

However, a better way to find  $\theta_2$  is to notice that if  $\cos(\theta_2)$  is given by Equation (1.4) then  $\sin(\theta_2)$  is given as

$$\sin(\theta_2) = \pm\sqrt{1 - D^2} \quad (1.6)$$

and, hence,  $\theta_2$  can be found by

$$\theta_2 = \tan^{-1} \frac{\pm\sqrt{1 - D^2}}{D} \quad (1.7)$$

The advantage of this latter approach is that both the elbow-up and elbow-down solutions are recovered by choosing the positive and negative signs in Equation (1.7), respectively.

It is left as an exercise (Problem 1-19) to show that  $\theta_1$  is now given as

$$\theta_1 = \tan^{-1}(y/x) - \tan^{-1} \left( \frac{\alpha_2 \sin \theta_2}{\alpha_1 + \alpha_2 \cos \theta_2} \right) \quad (1.8)$$

Notice that the angle  $\theta_1$  depends on  $\theta_2$ . This makes sense physically since we would expect to require a different value for  $\theta_1$ , depending on which solution is chosen for  $\theta_2$ .

### Velocity Kinematics

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

$$\begin{aligned} \dot{x} &= -\alpha_1 \sin \theta_1 \cdot \dot{\theta}_1 - \alpha_2 \sin(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{y} &= \alpha_1 \cos \theta_1 \cdot \dot{\theta}_1 + \alpha_2 \cos(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \end{aligned} \quad (1.9)$$

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

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -\alpha_1 \sin \theta_1 - \alpha_2 \sin(\theta_1 + \theta_2) & -\alpha_2 \sin(\theta_1 + \theta_2) \\ \alpha_1 \cos \theta_1 + \alpha_2 \cos(\theta_1 + \theta_2) & \alpha_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \dot{\theta} \\ &= J\dot{\theta} \end{aligned} \quad (1.10)$$

The matrix  $J$  defined by Equation (1.10) 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 Jacobian for any manipulator in the so-called **cross-product form**.

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

$$\dot{\theta} = J^{-1}\dot{x} \quad (1.11)$$

where  $J^{-1}$  is given by

$$J^{-1} = \frac{1}{\alpha_1 \alpha_2 s_{\theta_2}} \begin{bmatrix} \alpha_2 c_{\theta_1+\theta_2} & \alpha_2 s_{\theta_1+\theta_2} \\ -\alpha_1 c_{\theta_1} - \alpha_2 c_{\theta_1+\theta_2} & -\alpha_1 s_{\theta_1} - \alpha_2 s_{\theta_1+\theta_2} \end{bmatrix} \quad (1.12)$$

in which  $c_\theta$  and  $s_\theta$  denote respectively  $\cos \theta$  and  $\sin \theta$ . The determinant of the Jacobian in Equation (1.10) is  $\alpha_1 \alpha_2 \sin \theta_2$ . The Jacobian does not have an inverse, therefore, when  $\theta_2 = 0$  or  $\pi$ , in which case the manipulator is said to be in a **singular configuration**, such as shown in Figure 1.27 for  $\theta_2 = 0$ . The

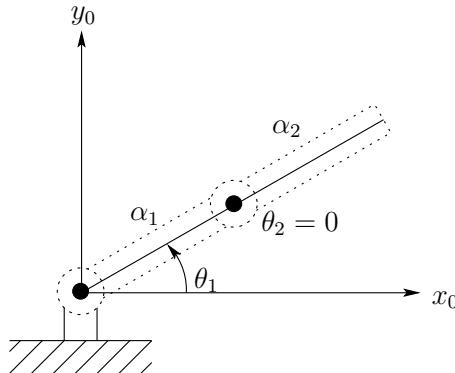


Fig. 1.27 A singular configuration.

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 cases 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, 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 configuration 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.

### **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 5, 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, i.e. without considering velocities and accelerations along the planned paths. The trajectory generation problem, also considered in Chapter 5, is to generate reference trajectories that determine the time history of the manipulator along a given path or between initial and final configurations.

### **Dynamics**

A robot manipulator is primarily a positioning device. To control the position we must know the dynamic properties of the manipulator in order to know how much force to exert on it to cause it to move: too little force and the manipulator is slow to react; too much force and the arm may crash into objects or oscillate about its desired position.

Deriving the dynamic equations of motion for robots is not a simple task due to the large number of degrees of freedom and nonlinearities present in the system. In Chapter 6 we develop techniques based on Lagrangian dynamics for systematically deriving the equations of motion of such a system. In addition to the rigid links, the complete description of robot dynamics includes the dynamics of the actuators that produce the forces and torques to drive the robot, and the dynamics of the drive trains that transmit the power from the actuators to the links. Thus, in Chapter 7 we also discuss actuator and drive train dynamics and their effects on the control problem.

### **Position Control**

In Chapters 7 and 8 we discuss the design of control algorithms for the execution of programmed tasks. The motion control problem consists of the **Tracking and Disturbance Rejection Problem**, which is the problem of determining the control inputs necessary to follow, or **track**, a desired trajectory that has been planned for the manipulator, while simultaneously **rejecting** disturbances due to unmodeled dynamic effects such as friction and noise. We detail the standard approaches to robot control based on frequency domain techniques. We also introduce the notion of **feedforward control** and the techniques of **computed torque** and **inverse dynamics** as a means for compensating the complex nonlinear interaction forces among the links of the manipulator. Robust and adaptive control are introduced in Chapter 8 using the **Second Method of Lyapunov**. Chapter 10 provides some additional advanced techniques from nonlinear control theory that are useful for controlling high performance robots.

### Force Control

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 9 we discuss force control and compliance along with common approaches to force control, namely **hybrid control** and **impedance control**.

### Vision

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 external to the robot in its workspace. In Chapter 11 we discuss the use of computer vision to determine position and orientation of objects.

### Vision-based Control

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 12. There are several approaches to vision-based control, but we will focus on the method of Image-Based Visual Servo (IBVS). This method has become very popular in recent years, and it relies on mathematical development analogous to that given in Chapter 4.

## 1.5 CHAPTER SUMMARY

In this chapter, we have given an introductory overview of some of the basic concepts required to develop mathematical models for robot arms. We have also discussed a few of the relevant mechanical aspects of robotic systems. In the remainder of the text, we will address the basic problems confronted in sensor-based robotic manipulation.

Many books have been written about these and more advance topics, including [1][3] [6][10][16][17][21] [23][30][33][41] [44][49][50][51] [59][63][67][70][77] [42][13]. There is a great deal of ongoing research in robotics. Current research

results can be found in journals such as *IEEE Transactions on Robotics* (previously *IEEE Transactions on Robotics and Automation*), *IEEE Robotics and Automation Magazine*, *International Journal of Robotics Research*, *Robotics and Autonomous Systems*, *Journal of Robotic Systems*, *Robotica*, *Journal of Intelligent and Robotic Systems*, *Autonomous Robots*, *Advanced Robotics*. and in proceedings from conferences such as *IEEE International Conference on Robotics and Automation*, *IEEE International Conference on Intelligent Robots and Systems*, *Workshop on the Algorithmic Foundations of Robotics*, *International Symposium on Experimental Robotics*, and *International Symposium on Robotics Research*.

---

## Problems

---

- 1-1 What are the key features that distinguish robots from other forms of automation such as CNC milling machines?
- 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 robotics related magazines and journals carried by the university library.
- 1-5 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.
- 1-6 List several applications for non-servo robots; for point-to point robots, for continuous path robots.
- 1-7 List five applications that a continuous path robot could do that a point-to-point robot could not do.
- 1-8 List five applications where computer vision would be useful in robotics.
- 1-9 List five applications where either tactile sensing or force feedback control would be useful in robotics.
- 1-10 Find out how many industrial robots are currently in operation in the United States. How many are in operation in Japan? What country ranks third in the number of industrial robots in use?
- 1-11 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?
- 1-12 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?
- 1-13 Discuss possible applications where redundant manipulators would be useful.
- 1-14 Referring to Figure 1.28, suppose that the tip of a single link travels a distance  $d$  between two points. A linear axis would travel the distance  $d$

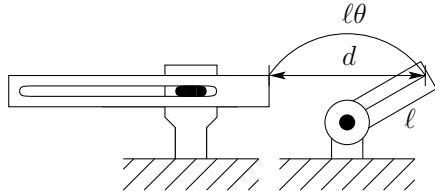


Fig. 1.28 Diagram for Problem 1-15

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 and  $\ell = 1\text{m}$ ,  $\theta = 90^\circ$  what is the resolution of the linear link? of the rotational link?

- 1-15 A single-link revolute arm is shown in Figure 1.28. If the length of the link is 50 cm and the arm travels 180° what is the control resolution obtained with an 8-bit encoder?
- 1-16 Repeat Problem 1.15 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.
- 1-17 Why is accuracy generally less than repeatability?
- 1-18 How could manipulator accuracy be improved using direct endpoint sensing? What other difficulties might direct endpoint sensing introduce into the control problem?
- 1-19 Derive Equation (1.8).
- 1-20 For the two-link manipulator of Figure 1.24 suppose  $\alpha_1 = \alpha_2 = 1$ . Find the coordinates of the tool when  $\theta_1 = \frac{\pi}{6}$  and  $\theta_2 = \frac{\pi}{2}$ .
- 1-21 Find the joint angles  $\theta_1, \theta_2$  when the tool is located at coordinates  $(\frac{1}{2}, \frac{1}{2})$ .
- 1-22 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  $\theta_1 = \theta_2 = \frac{\pi}{4}$ ?
- 1-23 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.
- 1-24 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.

- 1-25 For the two-link planar manipulator of Figure 1.24 is it possible for there to be an infinite number of solutions to the inverse kinematic equations? If so, explain how this can occur.

# 2

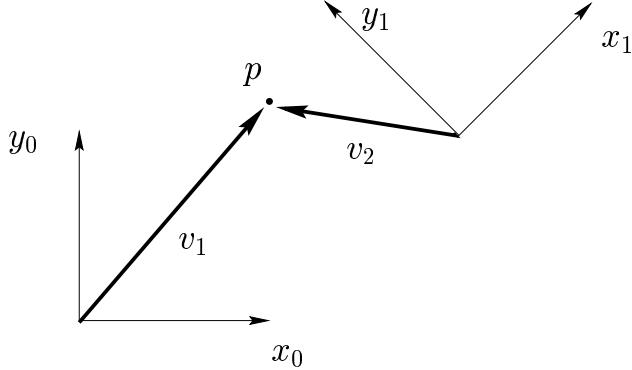
---

## *RIGID MOTIONS AND HOMOGENEOUS TRANSFORMATIONS*

**A** large part of robot kinematics is concerned with the establishment of various coordinate systems to represent the positions and orientations of rigid objects, and with transformations among these coordinate systems. 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.<sup>1</sup> 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.

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. Then we 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.

<sup>1</sup>Since we make extensive use of elementary matrix theory, the reader may wish to review Appendix B before beginning this chapter.



*Fig. 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.

Consider Figure 2.1. This figure shows two coordinate frames that differ in orientation by an angle of  $45^\circ$ . 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 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  $(5, 6)^T$ , and in the latter case  $(-2.8, 4.2)^T$ . 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 would write

$$p^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \quad p^1 = \begin{bmatrix} -2.8 \\ 4.2 \end{bmatrix}$$

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.

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

$$o_1^0 = \begin{bmatrix} 10 \\ 5 \end{bmatrix}, \quad o_0^1 = \begin{bmatrix} -10.6 \\ 3.5 \end{bmatrix}$$

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 systems, 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*, i.e., 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 vector can be moved to any location in space. Under this convention, two vectors are equal if they have the same direction and the same magnitude.

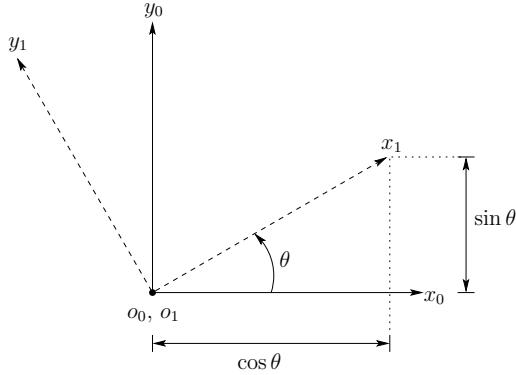
When assigning coordinates to vectors, we use the same notational convention that we used when assigning coordinates to points. Thus,  $v_1$  and  $v_2$  are geometric entities that are invariant with respect to the choice of coordinate systems, but the representation by coordinates of these vectors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

$$v_1^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \quad v_1^1 = \begin{bmatrix} 7.77 \\ 0.8 \end{bmatrix}, \quad v_2^0 = \begin{bmatrix} -5.1 \\ 1 \end{bmatrix}, \quad v_2^1 = \begin{bmatrix} -2.89 \\ 4.2 \end{bmatrix}$$

### Coordinate Convention

*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, i.e. 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 systems, but also for a mechanism that allows us to transform the coordinates of points that are expressed in one coordinate system into the appropriate coordinates with



*Fig. 2.2* Coordinate frame  $o_1x_1y_1$  is oriented at an angle  $\theta$  with respect to  $o_0x_0y_0$ .

respect to some other coordinate frame. Such coordinate transformations and their derivations 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 will rigidly attach coordinate frames to each body, and then specify the geometric relationships 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 orientations in a three dimensional space.

### 2.2.1 Rotation in the plane

Figure 2.2 shows two coordinate frames, with frame  $o_1x_1y_1$  being obtained by rotating frame  $o_0x_0y_0$  by an angle  $\theta$ . Perhaps the most obvious way to represent the relative orientation of these two frames is to merely specify the angle of rotation,  $\theta$ . There are two immediate disadvantages to such a representation. First, there is a discontinuity in the mapping from relative orientation to the value of  $\theta$  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  $\theta$  (i.e., a rotation by  $\epsilon$  causes  $\theta$  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 frame

$o_0x_0y_0$ <sup>2</sup>:

$$R_1^0 = [x_1^0 | y_1^0]$$

where  $x_1^0$  and  $y_1^0$  are the coordinates in frame  $o_0x_0y_0$  of unit vectors  $x_1$  and  $y_1$ , respectively. 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,

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

which gives

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

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  $\theta$ , it is not necessary that we do so. An alternative approach, and one that 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

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

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  $(x_1 \cdot x_0, x_1 \cdot y_0)^T$  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 was obtained in Equation (2.1).

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

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

<sup>2</sup>We will use  $x_i$ ,  $y_i$  to denote both coordinate axes and unit vectors along the coordinate axes depending on the context.

**Table 2.2.1: Properties of the Matrix Group  $SO(n)$** 

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

Since the inner product is commutative, (i.e.  $x_i \cdot y_j = y_j \cdot x_i$ ), 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 always mutually orthogonal, it can readily be seen that

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

The column vectors of  $R_1^0$  are of unit length and mutually orthogonal (Problem 2-4). Such a matrix is said to be **orthogonal**. It can also be shown (Problem 2-5) that  $\det R_1^0 = \pm 1$ . If we restrict ourselves to right-handed coordinate systems, as defined in Appendix B, then  $\det R_1^0 = +1$  (Problem 2-5). It is customary to refer to the set of all such  $n \times n$  matrices by the symbol  $SO(n)$ , which denotes the **Special Orthogonal group of order  $n$** . The properties of such matrices are summarized in Table 2.2.1.

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  $\theta$  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  $o_1x_1y_1z_1$  is projected onto coordinate frame  $o_0x_0y_0z_0$ . The resulting rotation matrix is given by

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

As was the case for rotation matrices in two dimensions, matrices in this form are orthogonal, with determinant equal to 1. In this case,  $3 \times 3$  rotation matrices belong to the group  $SO(3)$ . The properties listed in Table 2.2.1 also apply to rotation matrices in  $SO(3)$ .

#### Example 2.1

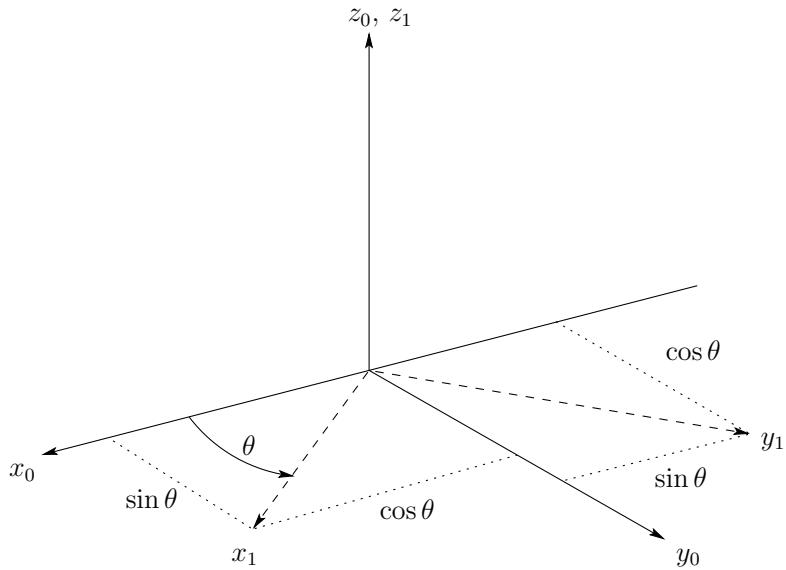


Fig. 2.3 Rotation about  $z_0$  by an angle  $\theta$ .

Suppose the frame  $o_1x_1y_1z_1$  is rotated through an angle  $\theta$  about the  $z_0$ -axis, and it is desired to find the resulting transformation matrix  $R_1^0$ . Note that by convention the positive sense for the angle  $\theta$  is given by the right hand rule; that is, a positive rotation by angle  $\theta$  about the  $z$ -axis would advance a right-hand

threaded screw along the positive  $z$ -axis<sup>3</sup>. From Figure 2.3 we see that

$$\begin{aligned} x_1 \cdot x_0 &= \cos \theta, & y_1 \cdot x_0 &= -\sin \theta, \\ x_1 \cdot y_0 &= \sin \theta, & y_1 \cdot y_0 &= \cos \theta \end{aligned}$$

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

$$R_1^0 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

◊

### The Basic Rotation Matrices

The rotation matrix given in Equation (2.2) 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

$$R_{z,0} = I \quad (2.3)$$

$$R_{z,\theta} R_{z,\phi} = R_{z,\theta+\phi} \quad (2.4)$$

which together imply

$$(R_{z,\theta})^{-1} = R_{z,-\theta} \quad (2.5)$$

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

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad (2.6)$$

$$R_{y,\theta} = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2.7)$$

which also satisfy properties analogous to Equations (2.3)-(2.5).

### Example 2.2

<sup>3</sup>See also Appendix B.

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. We see that the coordinates of  $x_1$  are  $\left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)^T$ , the coordinates of  $y_1$  are  $\left(\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}\right)^T$  and the coordinates of  $z_1$  are  $(0, 1, 0)^T$ . The rotation matrix  $R_1^0$  specifying the orientation of  $o_1x_1y_1z_1$  relative to  $o_0x_0y_0z_0$  has these as its column vectors, that is,

$$R_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \quad (2.8)$$

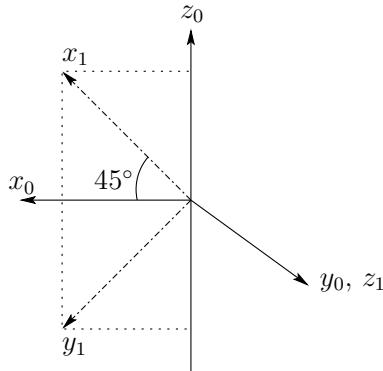


Fig. 2.4 Defining the relative orientation of two frames.

◊

### 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  $p$  (i.e., 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  $p^1 = (u, v, w)^T$  satisfy the equation

$$p = ux_1 + vy_1 + wz_1$$

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

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

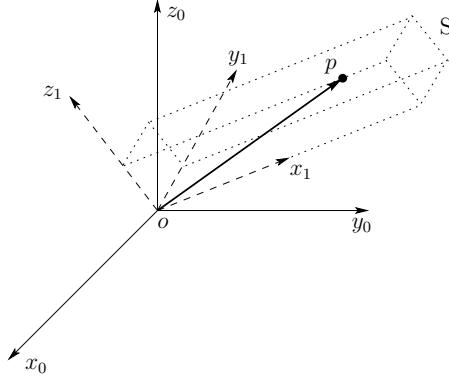


Fig. 2.5 Coordinate frame attached to a rigid body.

Combining these two equations we obtain

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

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

$$p^0 = R_1^0 p^1 \quad (2.9)$$

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  $R_1^0 p^1$  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. Consider Figure 2.6. One corner of the block in Figure 2.6(a) 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  $\pi$ . In Figure 2.6(b), 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

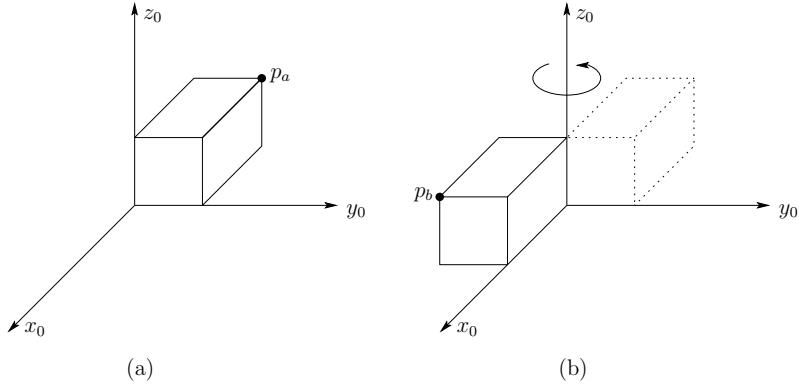


Fig. 2.6 The block in (b) is obtained by rotating the block in (a) by  $\pi$  about  $z_0$ .

it is coincident with the frame  $o_0x_0y_0z_0$ . After the rotation by  $\pi$ , the block's coordinate frame, which is rigidly attached to the block, is also rotated by  $\pi$ . 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

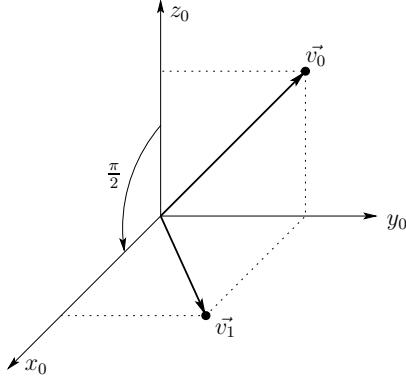
$$p_b^0 = R_{z,\pi} p_b^1$$

The key thing to notice is 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  $o_0x_0y_0z_0$  (i.e., before the rotation is performed), the coordinates  $p_b^1 = p_a^0$ , 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 us 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

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

Fig. 2.7 Rotating a vector about axis  $y_0$ .

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

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

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (2.11)$$

◊

Thus, as we have now seen, 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.

### Summary

We have seen that 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 a new vector in the same coordinate system.

The particular interpretation of a given rotation matrix  $R$  that is being used must then be made clear by the context.

### 2.3.1 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**<sup>4</sup>. 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

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

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

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

as shown in Figure 2.4. 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

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

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.

◊

<sup>4</sup>See Appendix B.

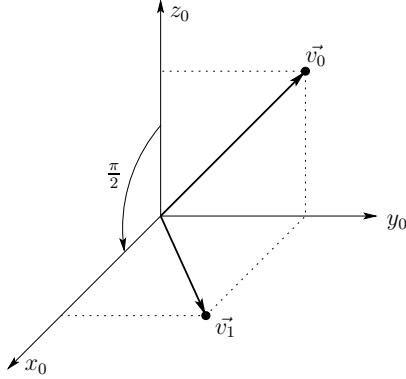


Fig. 2.8 Coordinate Frames for Example 2.4.

## 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

$$p^0 = R_1^0 p^1 \quad (2.13)$$

$$p^1 = R_2^1 p^2 \quad (2.14)$$

$$p^0 = R_2^0 p^2 \quad (2.15)$$

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

$$p^0 = R_1^0 R_2^1 p^2 \quad (2.16)$$

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

$$R_2^0 = R_1^0 R_2^1 \quad (2.17)$$

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

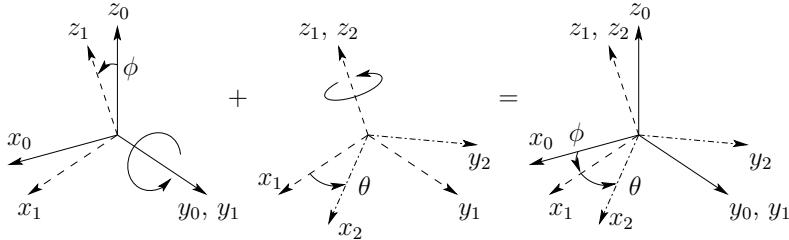


Fig. 2.9 Composition of rotations about current axes.

$p^2$  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 initially that all three of the coordinate frames coincide. We first rotate the frame  $o_2x_2y_2z_2$  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  $R_2^1$ . In each case 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  $\phi$  about the current  $y$ -axis followed by a rotation of angle  $\theta$  about the current  $z$ -axis. Refer to Figure 2.9. Then the matrix  $R$  is given by

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

◊

It is important to remember that the order in which a sequence of rotations are carried out, 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

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

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

To see why this is so, 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 2.3.1 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

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

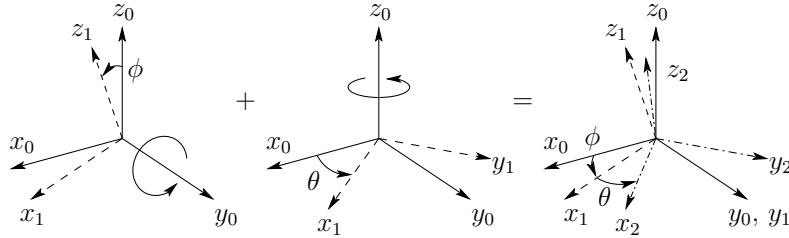


Fig. 2.10 Composition of rotations about fixed axes.

### Example 2.7

Referring to Figure 2.10, suppose that a rotation matrix  $R$  represents a rotation of angle  $\phi$  about  $y_0$  followed by a rotation of angle  $\theta$  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

$$\begin{aligned} p^0 &= R_{y,\phi}p^1 \\ &= R_{y,\phi} [R_{y,-\phi}R_{z,\theta}R_{y,\phi}] p^2 \\ &= R_{z,\theta}R_{y,\phi}p^2 \end{aligned} \quad (2.21)$$

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.

◇

**Summary**

We can summarize the rule of composition of rotational transformations by the following recipe. Given a fixed frame  $o_0x_0y_0z_0$  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 **post-multiply**  $R_1^0$  by  $R = R_2^1$  to obtain

$$R_2^0 = R_1^0 R_2^1 \quad (2.22)$$

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

$$R_2^0 = R R_1^0 \quad (2.23)$$

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 in 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  $\theta$  about the current  $x$ -axis
2. A rotation of  $\phi$  about the current  $z$ -axis
3. A rotation of  $\alpha$  about the fixed  $z$ -axis
4. A rotation of  $\beta$  about the current  $y$ -axis
5. A rotation of  $\delta$  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 post-multiply as the case may be to obtain*

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

◊

**2.5 PARAMETERIZATIONS OF ROTATIONS**

The nine elements  $r_{ij}$  in a general rotational transformation  $R$  are not independent quantities. Indeed a rigid body possesses at most three rotational

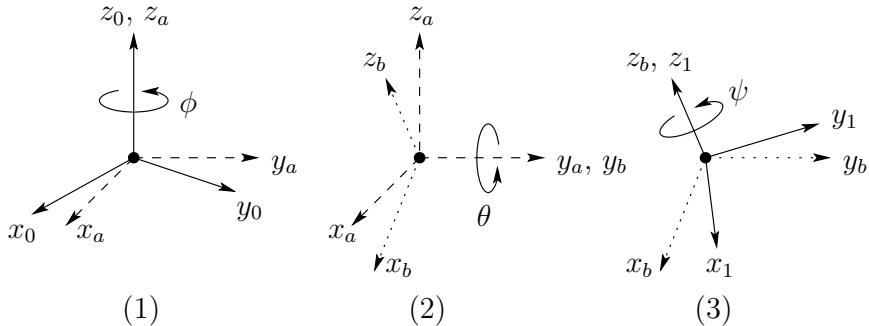


Fig. 2.11 Euler angle representation.

degrees-of-freedom and thus at most three quantities are required to specify its orientation. This can be easily seen by examining the constraints that govern the matrices in  $SO(3)$ :

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

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

Equation (2.25) follows from the fact the 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 2.11. 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  $(\phi, \theta, \psi)$ , known as Euler Angles, and obtained by three successive rotations as follows: First rotate about the  $z$ -axis by the angle  $\phi$ . Next rotate about the current  $y$ -axis by the angle  $\theta$ . Finally rotate about the current  $z$ -axis by the angle  $\psi$ . In Figure 2.11, frame  $o_ax_ay_az_a$  represents the new coordinate frame after the rotation by  $\phi$ , frame  $o_bxb_ybz_b$  represents the new coordinate frame after the rotation by  $\theta$ , and frame  $o_1x_1y_1z_1$  represents the final frame, after the rotation by  $\psi$ . Frames  $o_ax_ay_az_a$  and  $o_bxb_ybz_b$  are shown in the figure only to help you visualize the rotations.

In terms of the basic rotation matrices the resulting rotational transformation  $R_1^0$  can be generated as the product

$$\begin{aligned} R_{ZY\bar{Z}} &= R_{z,\phi} R_{y,\theta} R_{z,\psi} \\ &= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \end{aligned} \quad (2.27)$$

The matrix  $R_{ZY\bar{Z}}$  in Equation (2.27) is called the **ZYZ-Euler Angle Transformation**.

The more important and more difficult problem is the following: Given a matrix  $R \in SO(3)$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

determine a set of Euler angles  $\phi$ ,  $\theta$ , and  $\psi$  so that

$$R = R_{ZY\bar{Z}} \quad (2.28)$$

This problem will be important later when we address the inverse kinematics problem for manipulators. In order 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 (2.28) 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

$$\theta = \text{atan2}\left(r_{33}, \sqrt{1 - r_{33}^2}\right) \quad (2.29)$$

or

$$\theta = \text{atan2}\left(r_{33}, -\sqrt{1 - r_{33}^2}\right) \quad (2.30)$$

where the function `atan2` is the **two-argument arctangent function** defined in Appendix A.

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

$$\phi = \text{atan2}(r_{13}, r_{23}) \quad (2.31)$$

$$\psi = \text{atan2}(-r_{31}, r_{32}) \quad (2.32)$$

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

$$\phi = \text{atan2}(-r_{13}, -r_{23}) \quad (2.33)$$

$$\psi = \text{atan2}(r_{31}, -r_{32}) \quad (2.34)$$

Thus there are two solutions depending on the sign chosen for  $\theta$ .

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

$$\begin{aligned} \phi + \psi &= \text{atan2}(r_{11}, r_{21}) \\ &= \text{atan2}(r_{11}, -r_{12}) \end{aligned} \quad (2.36)$$

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 order. These rotations define the **roll**, **pitch**, and **yaw** angles, which we shall also denote  $\phi, \theta, \psi$ , and which are shown in Figure 2.12.

We specify the order of rotation as  $x - y - z$ , in other words, first a yaw about  $x_0$  through an angle  $\psi$ , then pitch about the  $y_0$  by an angle  $\theta$ , and finally roll about the  $z_0$  by an angle  $\phi$ <sup>5</sup>. Since the successive rotations are relative to

<sup>5</sup>It should be noted that other conventions exist for naming the roll, pitch and yaw angles.

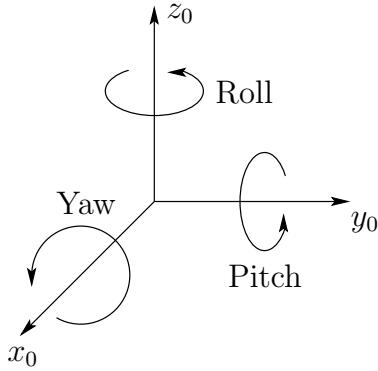


Fig. 2.12 Roll, pitch, and yaw angles.

the fixed frame, the resulting transformation matrix is given by

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

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,  $\phi, \theta, \psi$ , can be obtained for a given rotation matrix using a method that is similar to that used to derive the Euler angles above. We leave this as an exercise for the reader.

### 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  $k = (k_x, k_y, k_z)^T$ , 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  $\theta$  about this axis.

There are several ways in which the matrix  $R_{k,\theta}$  can be derived. Perhaps the simplest way is to note that the axis defined by the vector  $k$  is along the  $z$ -axis following the rotational transformation  $R_1^0 = R_{z,\alpha} R_{y,\beta}$ . Therefore, a rotation

about the axis  $k$  can be computed using a similarity transformation as

$$R_{k,\theta} = R_1^0 R_{z,\theta} R_1^{0-1} \quad (2.40)$$

$$= R_{z,\alpha} R_{y,\beta} R_{z,\theta} R_{y,-\beta} R_{z,-\alpha} \quad (2.41)$$

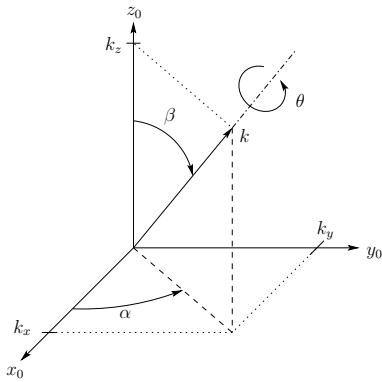


Fig. 2.13 Rotation about an arbitrary axis.

From Figure 2.13, we see that

$$\sin \alpha = \frac{k_y}{\sqrt{k_x^2 + k_y^2}} \quad (2.42)$$

$$\cos \alpha = \frac{k_x}{\sqrt{k_x^2 + k_y^2}} \quad (2.43)$$

$$\sin \beta = \sqrt{k_x^2 + k_y^2} \quad (2.44)$$

$$\cos \beta = k_z \quad (2.45)$$

Note that the final two equations follow from the fact that  $k$  is a unit vector. Substituting Equations (2.42)-(2.45) into Equation (2.41) we obtain after some lengthy calculation (Problem 2-17)

$$R_{k,\theta} = \begin{bmatrix} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{bmatrix} \quad (2.46)$$

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,

$$R = R_{k,\theta} \quad (2.47)$$

where  $k$  is a unit vector defining the axis of rotation, and  $\theta$  is the angle of rotation about  $k$ . The matrix  $R_{k,\theta}$  given in Equation (2.47) is called the **axis/angle representation** of  $R$ . Given an arbitrary rotation matrix  $R$  with components  $r_{ij}$ , the equivalent angle  $\theta$  and equivalent axis  $k$  are given by the expressions

$$\begin{aligned}\theta &= \cos^{-1} \left( \frac{\text{Tr}(R) - 1}{2} \right) \\ &= \cos^{-1} \left( \frac{r_{11} + r_{22} + r_{33} - 1}{2} \right)\end{aligned}\quad (2.48)$$

where  $\text{Tr}$  denotes the trace of  $R$ , and

$$k = \frac{1}{2 \sin \theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \quad (2.49)$$

These equations can be obtained by direct manipulation of the entries of the matrix given in Equation (2.46). The axis/angle representation is not unique since a rotation of  $-\theta$  about  $-k$  is the same as a rotation of  $\theta$  about  $k$ , that is,

$$R_{k,\theta} = R_{-k,-\theta} \quad (2.50)$$

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  $90^\circ$  about  $z_0$  followed by a rotation of  $30^\circ$  about  $y_0$  followed by a rotation of  $60^\circ$  about  $x_0$ . Then

$$\begin{aligned}R &= R_{x,60} R_{y,30} R_{z,90} \\ &= \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{4} & -\frac{3}{4} \\ \frac{\sqrt{3}}{2} & \frac{1}{4} & \frac{\sqrt{3}}{4} \end{bmatrix}\end{aligned}\quad (2.51)$$

We see that  $\text{Tr}(R) = 0$  and hence the equivalent angle is given by Equation (2.48) as

$$\theta = \cos^{-1} \left( -\frac{1}{2} \right) = 120^\circ \quad (2.52)$$

The equivalent axis is given from Equation (2.49) as

$$k = \left( \frac{1}{\sqrt{3}}, \frac{1}{2\sqrt{3}} - \frac{1}{2}, \frac{1}{2\sqrt{3}} + \frac{1}{2} \right)^T \quad (2.53)$$

◊

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  $\theta$ . However, since the equivalent axis  $k$  is given as a unit vector only

## TO APPEAR

*Fig. 2.14* Homogeneous transformations in two dimensions.

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

$$r = (r_x, r_y, r_z)^T = (\theta k_x, \theta k_y, \theta k_z)^T \quad (2.54)$$

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

**Remark 2.1** *One should be careful not to interpret the representation in Equation (2.54) to 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.*

## 2.6 RIGID MOTIONS

We have 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 2.1** *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  $SE(3) = \mathbb{R}^3 \times SO(3)$ .<sup>a</sup>*

<sup>a</sup>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  $\det R = +1$ , i.e. that  $R \in SO(3)$ .

A rigid motion is a pure translation together with a pure rotation. Referring to Figure 2.14 we see that if frame  $o_1x_1y_1z_1$  is obtained from frame  $o_0x_0y_0z_0$  by first applying a rotation specified by  $R_1^0$  followed by a translation given (with

respect to  $o_0x_0y_0z_0$ ) by  $d_1^0$ , then the coordinates  $p^0$  are given by

$$p^0 = R_1^0 p^1 + d_1^0 \quad (2.55)$$

Two points are worth noting in this figure. First, note that we cannot simply add the vectors  $p^0$  and  $p^1$  since they are defined relative to frames with different orientations, i.e. with respect to frames that are not parallel. However, we are able to add the vectors  $p^1$  and  $R_1^0 p^1$  precisely because multiplying  $p^1$  by the orientation matrix  $R_1^0$  expresses  $p^1$  in a frame that is parallel to frame  $o_0x_0y_0z_0$ . Second, it is not important in which order the rotation and translation are performed.

If we have the two rigid motions

$$p^0 = R_1^0 p^1 + d_1^0 \quad (2.56)$$

and

$$p^1 = R_2^1 p^2 + d_2^1 \quad (2.57)$$

then their composition defines a third rigid motion, which we can describe by substituting the expression for  $p^1$  from Equation (2.57) into Equation (2.56)

$$p^0 = R_1^0 R_2^1 p^2 + R_1^0 d_2^1 + d_1^0 \quad (2.58)$$

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

$$p^0 = R_2^0 p^2 + d_2^0 \quad (2.59)$$

Comparing Equations (2.58) and (2.59) we have the relationships

$$R_2^0 = R_1^0 R_2^1 \quad (2.60)$$

$$d_2^0 = d_1^0 + R_1^0 d_2^1 \quad (2.61)$$

Equation (2.60) shows that the orientation transformations can simply be multiplied together and Equation (2.61) 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 system  $o_0x_0y_0z_0$ ).

## 2.7 HOMOGENEOUS TRANSFORMATIONS

One can easily see that the calculation leading to Equation (2.58) 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 (2.60) and (2.61) with the matrix identity

$$\begin{bmatrix} R_1^0 & d_1^0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2^1 & d_1^2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1^0 R_2^1 & R_1^0 d_1^2 + d_1^0 \\ 0 & 1 \end{bmatrix} \quad (2.62)$$

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

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}; R \in SO(3), d \in \mathbb{R}^3 \quad (2.63)$$

Transformation matrices of the form given in Equation (2.63) 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  $4 \times 4$  matrices  $H$  of the form given in Equation (2.63)

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

$$H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix} \quad (2.64)$$

In order to represent the transformation given in Equation (2.55) 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,

$$P^0 = \begin{bmatrix} p^0 \\ 1 \end{bmatrix} \quad (2.65)$$

$$P^1 = \begin{bmatrix} p^1 \\ 1 \end{bmatrix} \quad (2.66)$$

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 (2.55) is equivalent to the (homogeneous) matrix equation

$$P^0 = H_1^0 P^1 \quad (2.67)$$

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

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \text{Rot}_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.68)$$

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \text{Rot}_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.69)$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \text{Rot}_{x,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.70)$$

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

$$H_1^0 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.71)$$

In the above equation  $n = (n_x, n_y, n_z)^T$  is a vector representing the direction of  $x_1$  in the  $o_0x_0y_0z_0$  system,  $s = (s_x, s_y, s_z)^T$  represents the direction of  $y_1$ , and  $a = (a_x, a_y, a_z)^T$  represents the direction of  $z_1$ . The vector  $d = (d_x, d_y, d_z)^T$  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.

### Composition Rule for Homogeneous Transformations

The same interpretation regarding composition and ordering of transformations holds for  $4 \times 4$  homogeneous transformations as for  $3 \times 3$  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  $\alpha$  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  $\theta$  about the current  $z$ -axis, is given by

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

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

◇

The homogeneous representation given in Equation (2.63) is a special case of homogeneous coordinates, which have been extensively used in the field of computer graphics. There, one is interested in scaling and/or perspective transformations in addition to translation and rotation. The most general homogeneous transformation takes the form

$$H = \left[ \begin{array}{c|c} R_{3 \times 3} & d_{3 \times 1} \\ \hline f_{1 \times 3} & s_{1 \times 1} \end{array} \right] = \left[ \begin{array}{c|c} \text{Rotation} & \text{Translation} \\ \hline \text{perspective} & \text{scale factor} \end{array} \right] \quad (2.72)$$

For our purposes we always take the last row vector of  $H$  to be  $(0, 0, 0, 1)$ , although the more general form given by (2.72) could be useful, for example, for interfacing a vision system into the overall robotic system or for graphic simulation.

## 2.8 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  $\theta$  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

$$R_2^0 = RR_1^0$$

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  $(\phi, \theta, \psi)$ , 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.

The interested reader can find deeper explanations of these concepts in a variety of sources, including [4] [18] [29] [62] [54] [75].

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. Show that the length of a free vector is not changed by rotation, i.e., that  $\|v\| = \|Rv\|$ .
3. Show that the distance between points is not changed by rotation i.e., that  $\|p_1 - p_2\| = \|Rp_1 - Rp_2\|$ .
4. If a matrix  $R$  satisfies  $R^T R = I$ , show that the column vectors of  $R$  are of unit length and mutually perpendicular.
5. If a matrix  $R$  satisfies  $R^T R = I$ , then
  - a) show that  $\det R = \pm 1$
  - b) Show that  $\det R = \pm 1$  if we restrict ourselves to right-handed coordinate systems.
6. Verify Equations (2.3)-(2.5).
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.

8. Derive Equations (2.6) and (2.7).
9. Suppose  $A$  is a  $2 \times 2$  rotation matrix. In other words  $A^T A = I$  and  $\det A = 1$ . Show that there exists a unique  $\theta$  such that  $A$  is of the form

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

10. Consider the following sequence of rotations:

- (a) Rotate by  $\phi$  about the world  $x$ -axis.
- (b) Rotate by  $\theta$  about the current  $z$ -axis.
- (c) Rotate by  $\psi$  about the world  $y$ -axis.

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

11. Consider the following sequence of rotations:

- (a) Rotate by  $\phi$  about the world  $x$ -axis.
- (b) Rotate by  $\theta$  about the world  $z$ -axis.
- (c) Rotate by  $\psi$  about the current  $x$ -axis.

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

12. Consider the following sequence of rotations:

- (a) Rotate by  $\phi$  about the world  $x$ -axis.
- (b) Rotate by  $\theta$  about the current  $z$ -axis.
- (c) Rotate by  $\psi$  about the current  $x$ -axis.
- (d) Rotate by  $\alpha$  about the world  $z$ -axis.

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

13. Consider the following sequence of rotations:

- (a) Rotate by  $\phi$  about the world  $x$ -axis.
- (b) Rotate by  $\theta$  about the world  $z$ -axis.
- (c) Rotate by  $\psi$  about the current  $x$ -axis.
- (d) Rotate by  $\alpha$  about the world  $z$ -axis.

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

14. Find the rotation matrix representing a roll of  $\frac{\pi}{4}$  followed by a yaw of  $\frac{\pi}{2}$  followed by a pitch of  $\frac{\pi}{2}$ .

15. 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.

16. 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  $R_3^2$ .

17. Verify Equation (2.46).

18. 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  $k$ .
19. Let  $k = \frac{1}{\sqrt{3}}(1, 1, 1)^T$ ,  $\theta = 90^\circ$ . Find  $R_{k,\theta}$ .
20. Show by direct calculation that  $R_{k,\theta}$  given by Equation (2.46) is equal to  $R$  given by Equation (2.51) if  $\theta$  and  $k$  are given by Equations (2.52) and (2.53), respectively.
21. Compute the rotation matrix given by the product

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

22. Suppose  $R$  represents a rotation of  $90^\circ$  about  $y_0$  followed by a rotation of  $45^\circ$  about  $z_1$ . Find the equivalent axis/angle to represent  $R$ . Sketch the initial and final frames and the equivalent axis vector  $k$ .
23. Find the rotation matrix corresponding to the set of Euler angles  $\{\frac{\pi}{2}, 0, \frac{\pi}{4}\}$ . What is the direction of the  $x_1$  axis relative to the base frame?
24. 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?
25. Unit magnitude complex numbers (i.e.,  $a + ib$  such that  $a^2 + b^2 = 1$ ) 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.
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  $a + ib$ ?
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  $\theta$  about the unit vector  $n = (n_x, n_y, n_z)^T$  can be represented by the unit quaternion  $Q = (\cos \frac{\theta}{2}, n_x \sin \frac{\theta}{2}, n_y \sin \frac{\theta}{2}, n_z \sin \frac{\theta}{2})$ . Show that such a quaternion has unit norm, i.e., that  $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ .

28. Using  $Q = (\cos \frac{\theta}{2}, n_x \sin \frac{\theta}{2}, n_y \sin \frac{\theta}{2}, n_z \sin \frac{\theta}{2})$ , and the results from Section 2.5.3, determine the rotation matrix  $R$  that corresponds to the rotation represented by the quaternion  $(q_0, q_1, q_2, q_3)$ .
29. Determine the quaternion  $Q$  that represents the same rotation as given by the rotation matrix  $R$ .
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  $= (q_1, q_2, q_3)^T$ . 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.

31. Show that  $Q_I = (1, 0, 0, 0)$  is the identity element for unit quaternion multiplication, i.e., that  $QQ_I = Q_IQ = Q$  for any unit quaternion  $Q$ .
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  $Q$ , i.e., that  $Q^*Q = QQ^* = (1, 0, 0, 0)$ .

33. Let  $v$  be a vector whose coordinates are given by  $(v_x, v_y, v_z)^T$ . 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.
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

$$Q(0, x, y, z)Q^* + T \quad (2.73)$$

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

35. 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?
36. Consider the diagram of Figure 2.15. Find the homogeneous transfor-

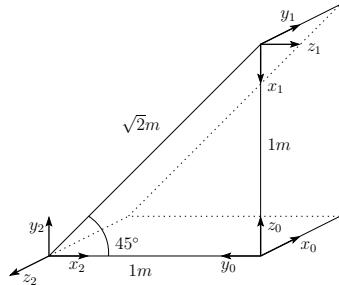


Fig. 2.15 Diagram for Problem 2-36.

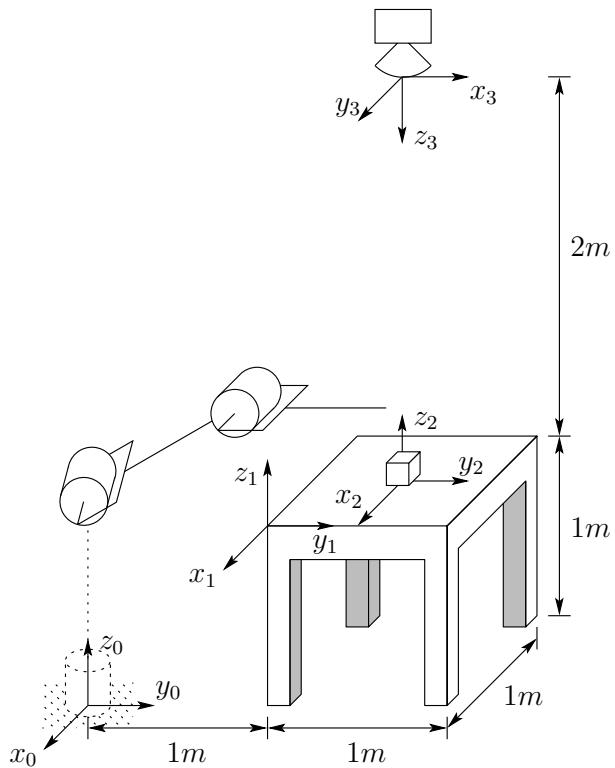


Fig. 2.16 Diagram for Problem 2-37.

mations  $H_1^0, H_2^0, H_2^1$  representing the transformations among the three frames shown. Show that  $H_2^0 = H_1^0, H_2^1$ .

37. Consider the diagram of Figure 2.16. 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 2m 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  $o_3x_3y_3z_3$ .

38. In Problem 37, suppose that, after the camera is calibrated, it is rotated  $90^\circ$  about  $z_3$ . Recompute the above coordinate transformations.
39. If the block on the table is rotated  $90^\circ$  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.
40. 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 Earth's frame with respect to the global reference frame.
41. In general, multiplication of homogeneous transformation matrices is not commutative. Consider the matrix product

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

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,  $H$ .

# 3

---

## *FORWARD AND INVERSE KINEMATICS*

**I**n this chapter we consider the forward and inverse kinematics for serial link manipulators. The problem of kinematics is to describe the motion of the manipulator without consideration of the forces and torques causing the motion. The kinematic description is therefore a geometric one. We first consider the problem of *forward kinematics*, which is to determine the position and orientation of the end-effector given the values for the joint variables of the robot. The *inverse kinematics* problem is to determine the values of 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 and allows a relative rotation about a single axis, and a prismatic joint permits 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. The objective of forward kinematic analysis is to determine the *cumulative* effect of the entire set of joint variables, that is, to determine the position and orientation of the end effector given the values of these joint variables. The objective of inverse kinematic analysis is, in contrast, to determine the values for these joint variables given the position and orientation of the end effector frame.

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) 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:

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

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 inertial frame. Figure 3.1 illustrates the idea of attaching frames rigidly to links in the case of an elbow manipulator.

Now suppose  $A_i$  is the homogeneous transformation matrix that expresses 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,

$$A_i = A_i(q_i) \quad (3.2)$$

Now 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, by convention, a **trans-**

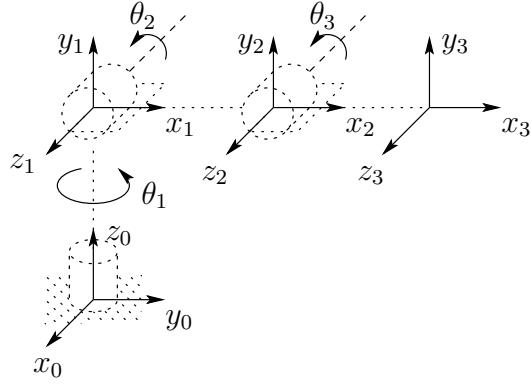


Fig. 3.1 Coordinate frames attached to elbow manipulator

**formation matrix**, and is denoted by  $T_j^i$ . From Chapter 2 we see that

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

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 frame  $n$ , is a constant independent of the configuration of the robot. 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  $3 \times 3$  rotation matrix  $R_n^0$ , and define the homogeneous transformation matrix

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

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

$$H = T_n^0 = A_1(q_1) \cdots A_n(q_n) \quad (3.5)$$

Each homogeneous transformation  $A_i$  is of the form

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

Hence

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

The matrix  $R_j^i$  expresses the orientation of  $o_jx_jy_jz_j$  relative to  $o_ix_iy_iz_i$  and is given by the rotational parts of the  $A$ -matrices as

$$R_j^i = R_{i+1}^i \cdots R_j^{j-1} \quad (3.8)$$

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

$$o_j^i = o_{j-1}^i + R_{j-1}^i o_j^{j-1} \quad (3.9)$$

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 FORWARD KINEMATICS: THE DENAVIT-HARTENBERG CONVENTION

In this section we develop the **forward** or **configuration kinematic equations** for rigid robots. The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints.

We will develop a set of conventions that provide a systematic procedure for performing this analysis. 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 robot 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

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

where the four quantities  $\theta_i$ ,  $a_i$ ,  $d_i$ ,  $\alpha_i$  are parameters associated with link  $i$  and joint  $i$ . The four parameters  $a_i$ ,  $\alpha_i$ ,  $d_i$ , and  $\theta_i$  in (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, it turns out that three of the above four quantities are constant for a given link, while the fourth parameter,  $\theta_i$  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, such as, for example, three numbers to specify the fourth column of the matrix and three Euler angles to specify the upper left  $3 \times 3$  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 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 issues

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 (3.10). Suppose we

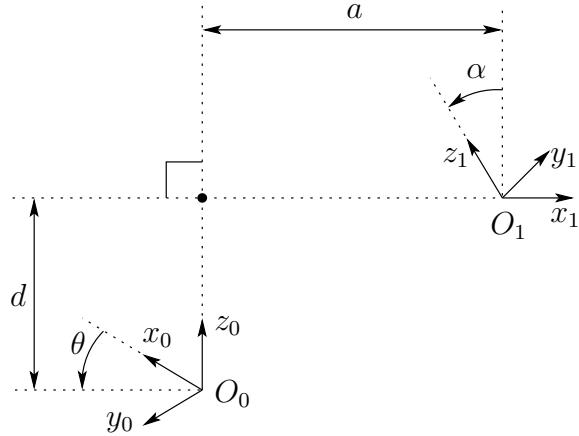


Fig. 3.2 Coordinate frames satisfying assumptions DH1 and DH2

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.

#### DH Coordinate Frame Assumptions

**(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  $a$ ,  $d$ ,  $\theta$ ,  $\alpha$  such that

$$A = \text{Rot}_{z,\theta} \text{Trans}_{z,d} \text{Trans}_{x,a} \text{Rot}_{x,\alpha} \quad (3.11)$$

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

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

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  $o_0 x_0 y_0 z_0$ , 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  $\theta$  and  $\alpha$  such that

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

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

$$\begin{aligned} r_{11}^2 + r_{21}^2 &= 1, \\ r_{32}^2 + r_{33}^2 &= 1 \end{aligned}$$

Hence there exist unique  $\theta$  and  $\alpha$  such that

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

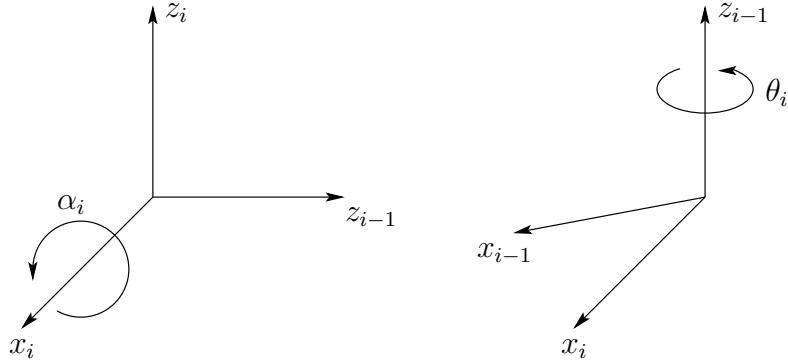
Once  $\theta$  and  $\alpha$  are found, it is routine to show that the remaining elements of  $R_1^0$  must have the form shown in (3.13), using the fact that  $R_1^0$  is a rotation matrix.

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  $o_0 x_0 y_0 z_0$ , 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 (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 represented in the form (3.10), we can in fact give a physical interpretation to each of the four quantities in (3.10). The parameter  $a$  is the distance between the axes  $z_0$  and  $z_1$ , and is measured along the axis  $x_1$ . The angle  $\alpha$  is the angle between the axes  $z_0$  and  $z_1$ , measured in a plane normal to  $x_1$ . The positive sense for  $\alpha$  is determined from  $z_0$  to  $z_1$  by the right-handed rule as shown in Figure 3.3. The parameter  $d$  is the perpendicular distance from the origin  $o_0$  to the intersection of the  $x_1$  axis with  $z_0$  measured along the  $z_0$  axis. Finally,  $\theta$  is the angle between  $x_0$  and  $x_1$  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.

Fig. 3.3 Positive sense for  $\alpha_i$  and  $\theta_i$ 

### 3.2.2 Assigning the coordinate frames

For a given robot manipulator, one can always choose the frames  $0, \dots, n$  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 link frames (assuming that the coordinate frames for link  $n$  coincide). We will begin by deriving the general procedure. We will then discuss various common special cases where it is possible to further simplify the homogeneous transformation matrix.

To start, note that the choice of  $z_i$  is arbitrary. In particular, from (3.13), we see that by choosing  $\alpha_i$  and  $\theta_i$  appropriately, we can obtain any arbitrary direction for  $z_i$ . Thus, for our first step, we assign the axes  $z_0, \dots, z_{n-1}$  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 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 joint  $i$  is fixed with respect to frame  $i$ , and 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.

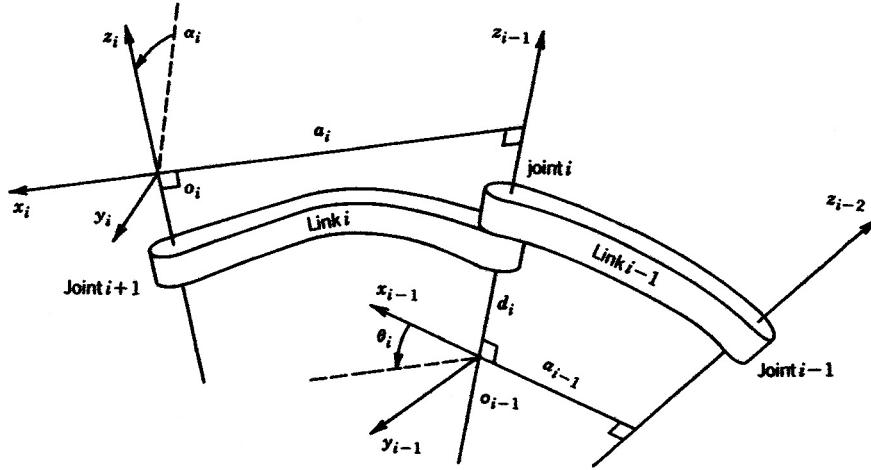


Fig. 3.4 Denavit-Hartenberg frame assignment

In order to set up frame  $i$  it is necessary 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 3.2.3. 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 line segment perpendicular to both  $z_{i-1}$  and  $z_i$  such that it connects both lines and it has minimum length. The line containing this common normal to  $z_{i-1}$  and  $z_i$  defines  $x_i$ , and the point where this line 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 (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  $o_i$  anywhere along  $z_i$ . One often chooses  $o_i$  to simplify the resulting equations. 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,  $\alpha_i$  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. The most natural choice for the origin  $o_i$  in this case is at the point of intersection of  $z_i$  and  $z_{i-1}$ . However, any convenient point along the axis  $z_i$  suffices. Note that in this case the parameter  $a_i$  equals 0.

This constructive procedure works for frames  $0, \dots, n-1$  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 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  $s$ .

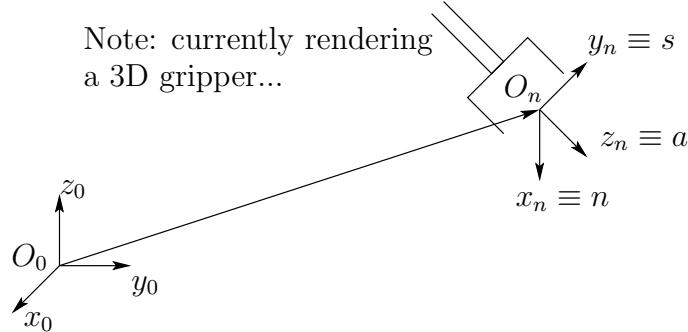


Fig. 3.5 Tool frame assignment

In most contemporary robots the final joint motion is a rotation of the end-effector by  $\theta_n$  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  $\theta_n$  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  $\alpha_i$  are always constant for all  $i$  and are characteristic of the manipulator. If joint  $i$  is prismatic, then

$\theta_i$  is also a constant, while  $d_i$  is the  $i^{th}$  joint variable. Similarly, if joint  $i$  is revolute, then  $d_i$  is constant and  $\theta_i$  is the  $i^{th}$  joint variable.

### 3.2.3 Examples

In the DH convention the only variable angle is  $\theta$ , so we simplify notation by writing  $c_i$  for  $\cos \theta_i$ , etc. We also denote  $\theta_1 + \theta_2$  by  $\theta_{12}$ , 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.

#### Example 3.1 Planar Elbow Manipulator

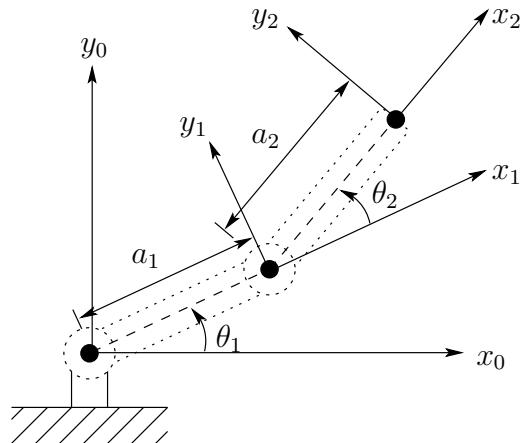


Fig. 3.6 Two-link planar manipulator. The  $z$ -axes all point out of the page, and are not shown in the figure

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. The origin is chosen at the point of intersection of the  $z_0$  axis with the page and the direction of the  $x_0$  axis is completely 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

Table 3.1 Link parameters for 2-link planar manipulator

| Link | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$   |
|------|-------|------------|-------|--------------|
| 1    | $a_1$ | 0          | 0     | $\theta_1^*$ |
| 2    | $a_2$ | 0          | 0     | $\theta_2^*$ |

\* variable

from (3.10) as

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The  $T$ -matrices are thus given by

$$T_1^0 = A_1$$

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

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,

$$x = a_1 c_1 + a_2 c_{12}$$

$$y = a_1 s_1 + a_2 s_{12}$$

are the coordinates of the end-effector in the base frame. The rotational part of  $T_2^0$  gives the orientation of the frame  $o_2 x_2 y_2 z_2$  relative to the base frame.

◊

**Example 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  $z_0$  as well as 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 2. The axis  $x_0$  is chosen normal to the page. Next, since  $z_0$  and  $z_1$  coincide, the origin  $o_1$  is chosen at joint 1 as shown. The  $x_1$  axis is normal to the page when  $\theta_1 = 0$  but, of course its

Table 3.2 Link parameters for 3-link cylindrical manipulator

| Link | $a_i$ | $\alpha_i$ | $d_i$   | $\theta_i$   |
|------|-------|------------|---------|--------------|
| 1    | 0     | 0          | $d_1$   | $\theta_1^*$ |
| 2    | 0     | -90        | $d_2^*$ | 0            |
| 3    | 0     | 0          | $d_3^*$ | 0            |

\* variable

direction will change since  $\theta_1$  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  $\theta_2$  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

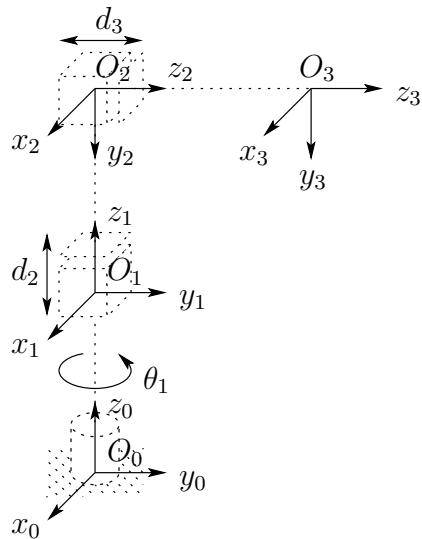


Fig. 3.7 Three-link cylindrical manipulator

$T$  matrices are

$$\begin{aligned}
 A_1 &= \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 A_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 A_3 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 T_3^0 = A_1 A_2 A_3 &= \begin{bmatrix} c_1 & 0 & -s_1 & -s_1 d_3 \\ s_1 & 0 & c_1 & c_1 d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.14)
 \end{aligned}$$

◇

### Example 3.3 Spherical Wrist

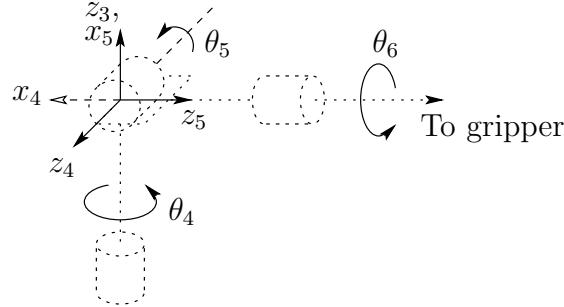


Fig. 3.8 The spherical wrist frame assignment

The spherical wrist configuration is shown in Figure 3.8, in which the joint axes  $z_3, z_4, z_5$  intersect at  $o$ . 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,  $\theta_4, \theta_5, \theta_6$  are the Euler angles  $\phi, \theta, \psi$ , respectively, 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 and

Table 3.3 DH parameters for spherical wrist

| Link | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$   |
|------|-------|------------|-------|--------------|
| 4    | 0     | -90        | 0     | $\theta_4^*$ |
| 5    | 0     | 90         | 0     | $\theta_5^*$ |
| 6    | 0     | 0          | $d_6$ | $\theta_6^*$ |

\* variable

the expression (3.10). This gives

$$\begin{aligned} A_4 &= \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_5 &= \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ A_6 &= \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Multiplying these together yields

$$\begin{aligned} T_6^3 &= A_4 A_5 A_6 \\ &= \begin{bmatrix} R_6^3 & o_6^3 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 & c_4 s_5 d_6 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 & s_4 s_5 d_6 \\ -s_5 c_6 & s_5 s_6 & c_5 & c_5 d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.15) \end{aligned}$$

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

◇

#### Example 3.4 Cylindrical Manipulator with Spherical Wrist

Suppose that we now attach a spherical wrist to the cylindrical manipulator of Example 3.2 as shown in Figure 3.9. Note that the axis of rotation of joint 4

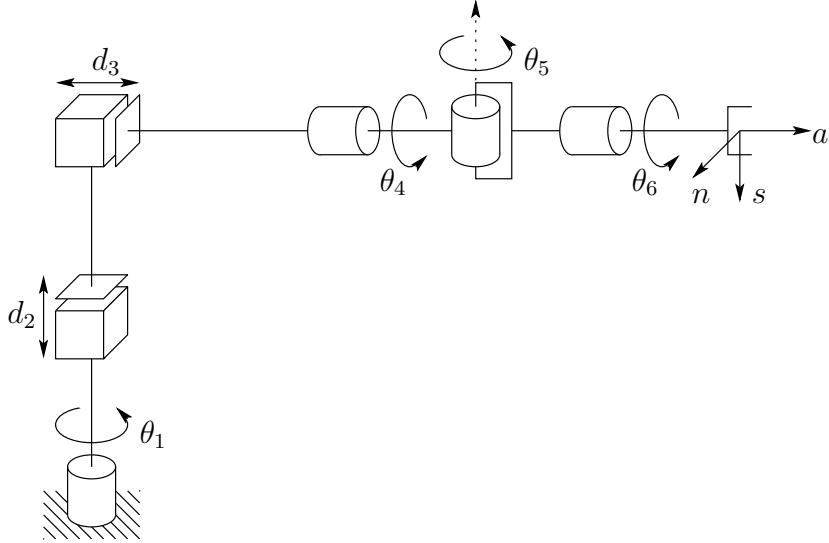


Fig. 3.9 Cylindrical robot with spherical wrist

is parallel to  $z_2$  and thus coincides with the axis  $z_3$  of Example 3.2. The implication of this is that we can immediately combine the two previous expression (3.14) and (3.15) to derive the forward kinematics as

$$T_6^0 = T_3^0 T_6^3 \quad (3.16)$$

with  $T_3^0$  given by (3.14) and  $T_6^3$  given by (3.15). Therefore the forward kinematics of this manipulator is described by

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

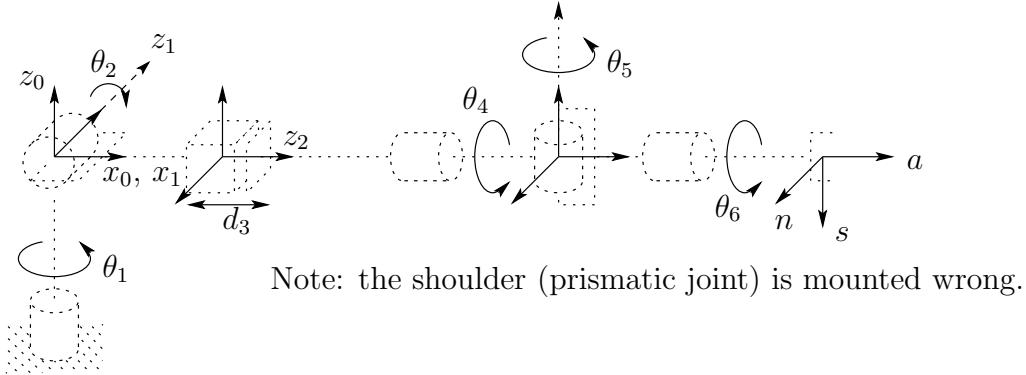


Fig. 3.10 DH coordinate frame assignment for the Stanford manipulator

in which

$$\begin{aligned}
 r_{11} &= c_1 c_4 c_5 c_6 - c_1 s_4 s_6 + s_1 s_5 c_6 \\
 r_{21} &= s_1 c_4 c_5 c_6 - s_1 s_4 s_6 - c_1 s_5 c_6 \\
 r_{31} &= -s_4 c_5 c_6 - c_4 s_6 \\
 r_{12} &= -c_1 c_4 c_5 s_6 - c_1 s_4 c_6 - s_1 s_5 c_6 \\
 r_{22} &= -s_1 c_4 c_5 s_6 - s_1 s_4 s_6 + c_1 s_5 c_6 \\
 r_{32} &= s_4 c_5 c_6 - c_4 c_6 \\
 r_{13} &= c_1 c_4 s_5 - s_1 c_5 \\
 r_{23} &= s_1 c_4 s_5 + c_1 c_5 \\
 r_{33} &= -s_4 s_5 \\
 d_x &= c_1 c_4 s_5 d_6 - s_1 c_5 d_6 - s_1 d_3 \\
 d_y &= s_1 c_4 s_5 d_6 + c_1 c_5 d_6 + c_1 d_3 \\
 d_z &= -s_4 s_5 d_6 + d_1 + d_2
 \end{aligned}$$

Notice how most of the complexity of the forward kinematics for this manipulator results from the orientation of the end-effector while the expression for the arm position from (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 the next chapter.

◇

### Example 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.

Table 3.4 DH parameters for Stanford Manipulator

| Link | $d_i$ | $a_i$ | $\alpha_i$ | $\theta_i$ |
|------|-------|-------|------------|------------|
| 1    | 0     | 0     | -90        | $\theta^*$ |
| 2    | $d_2$ | 0     | +90        | $\theta^*$ |
| 3    | $d^*$ | 0     | 0          | 0          |
| 4    | 0     | 0     | -90        | $\theta^*$ |
| 5    | 0     | 0     | +90        | $\theta^*$ |
| 6    | $d_6$ | 0     | 0          | $\theta^*$ |

\* joint variable

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

It is straightforward to compute the matrices  $A_i$  as

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

$$A_2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.19)$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.20)$$

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.21)$$

$$A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.22)$$

$$A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.23)$$

$T_6^0$  is then given as

$$T_6^0 = A_1 \cdots A_6 \quad (3.24)$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.25)$$

where

$$\begin{aligned} r_{11} &= c_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] - d_2(s_4c_5c_6 + c_4s_6) \\ r_{21} &= s_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] + c_1(s_4c_5c_6 + c_4s_6) \\ r_{31} &= -s_2(c_4c_5c_6 - s_4s_6) - c_2s_5c_6 \\ r_{12} &= c_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] - s_1(-s_4c_5s_6 + c_4c_6) \\ r_{22} &= -s_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] + c_1(-s_4c_5s_6 + c_4c_6) \\ r_{32} &= s_2(c_4c_5s_6 + s_4c_6) + c_2s_5s_6 \\ r_{13} &= c_1(c_2c_4s_5 + s_2c_5) - s_1s_4s_5 \\ r_{23} &= s_1(c_2c_4s_5 + s_2c_5) + c_1s_4s_5 \\ r_{33} &= -s_2c_4s_5 + c_2c_5 \\ d_x &= c_1s_2d_3 - s_1d_2 + d_6(c_1c_2c_4s_5 + c_1c_5s_2 - s_1s_4s_5) \\ d_y &= s_1s_2d_3 + c_1d_2 + d_6(c_1s_4s_5 + c_2c_4s_1s_5 + c_5s_1s_2) \\ d_z &= c_2d_3 + d_6(c_2c_5 - c_4s_2s_5) \end{aligned}$$

◇

### Example 3.6 SCARA Manipulator

As another example of the general procedure, consider the SCARA manipulator of Figure 3.11. This manipulator, which is an abstraction of the AdeptOne robot of Figure 1.14, 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 is completely arbitrary and only affects the zero configuration of the manipulator, that is, the position of the manipulator when  $\theta_1 = 0$ .

The joint parameters are given in Table 3.5, and the  $A$ -matrices are as fol-

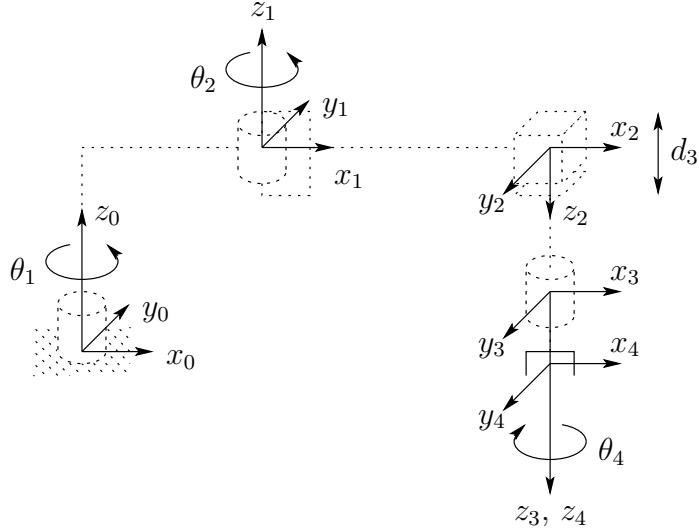


Fig. 3.11 DH coordinate frame assignment for the SCARA manipulator

Table 3.5 Joint parameters for SCARA

| Link | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$ |
|------|-------|------------|-------|------------|
| 1    | $a_1$ | 0          | 0     | $\theta^*$ |
| 2    | $a_2$ | 180        | 0     | $\theta^*$ |
| 3    | 0     | 0          | $d^*$ | 0          |
| 4    | 0     | 0          | $d_4$ | $\theta^*$ |

\* joint variable

lows.

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.26)$$

$$A_2 = \begin{bmatrix} c_2 & s_2 & 0 & a_2 c_2 \\ s_2 & -c_2 & 0 & a_2 s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.27)$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.28)$$

$$A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.29)$$

The forward kinematic equations are therefore given by

$$\begin{aligned} T_4^0 &= A_1 \cdots A_4 \\ &= \begin{bmatrix} c_{12}c_4 + s_{12}s_4 & -c_{12}s_4 + s_{12}c_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -s_{12}s_4 - c_{12}c_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.30) \end{aligned}$$

◇

### 3.3 INVERSE KINEMATICS

In the previous section we showed how to determine the end-effector position and orientation in terms of the joint variables. This section is concerned with the inverse problem of finding the joint variables in terms of the end-effector position and orientation. This is the problem of **inverse kinematics**, and it is, in general, more difficult than the forward kinematics problem.

In this chapter, 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. 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.

#### 3.3.1 The General Inverse Kinematics Problem

The general problem of inverse kinematics can be stated as follows. Given a  $4 \times 4$  homogeneous transformation

$$H = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} \in SE(3) \quad (3.31)$$

with  $R \in SO(3)$ , find (one or all) solutions of the equation

$$T_n^0(q_1, \dots, q_n) = H \quad (3.32)$$

where

$$T_n^0(q_1, \dots, q_n) = A_1(q_1) \cdots A_n(q_n) \quad (3.33)$$

Here,  $H$  represents the desired position and orientation of the end-effector, and our task is to find the values for the joint variables  $q_1, \dots, q_n$  so that  $T_n^0(q_1, \dots, q_n) = H$ .

Equation (3.32) results in twelve nonlinear equations in  $n$  unknown variables, which can be written as

$$T_{ij}(q_1, \dots, q_n) = h_{ij}, \quad i = 1, 2, 3, \quad j = 1, \dots, 4 \quad (3.34)$$

where  $T_{ij}$ ,  $h_{ij}$  refer to the twelve nontrivial entries of  $T_n^0$  and  $H$ , respectively. (Since the bottom row of both  $T_n^0$  and  $H$  are  $(0,0,0,1)$ , four of the sixteen equations represented by (3.32) are trivial.)

### Example 3.7

Recall the Stanford manipulator of Example 3.3.5. Suppose that the desired position and orientation of the final frame are given by

$$H = \begin{bmatrix} 0 & 1 & 0 & -0.154 \\ 0 & 0 & 1 & 0.763 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.35)$$

To find the corresponding joint variables  $\theta_1, \theta_2, d_3, \theta_4, \theta_5$ , and  $\theta_6$  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.

In solving the inverse kinematics problem we are most interested in finding a closed form solution of the equations rather than a numerical solution. Finding a closed form solution means finding an explicit relationship:

$$q_k = f_k(h_{11}, \dots, h_{34}), \quad k = 1, \dots, n \quad (3.36)$$

Closed form solutions are preferable for two reasons. First, in certain applications, such as tracking a welding seam whose location is provided by a vision system, the inverse kinematic equations must be solved at a rapid rate, say every 20 milliseconds, and having closed form expressions rather than an iterative search is a practical necessity. Second, the kinematic equations in general have multiple solutions. Having closed form solutions allows one to develop rules for choosing a particular solution among several.

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 (3.32) 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. For our purposes, we henceforth assume that the given homogeneous matrix  $H$  in (3.32) corresponds to a configuration within the manipulator's workspace with an attainable orientation. This guarantees that the mathematical solutions obtained correspond to achievable configurations.

### 3.3.2 Kinematic Decoupling

Although the general problem of inverse kinematics is quite difficult, it turns out that for manipulators having six joints, with the last three joints 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 (3.32) as

two sets of equations representing the rotational and positional equations

$$R_6^0(q_1, \dots, q_6) = R \quad (3.37)$$

$$o_6^0(q_1, \dots, q_6) = o \quad (3.38)$$

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  $q_1, \dots, q_6$ .

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 links about these axes will not change the position of  $o_c$ , and thus, the position of the wrist center is thus 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

$$o = o_c^0 + d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.39)$$

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

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.40)$$

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 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 (3.40) gives the relationship

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix} \quad (3.41)$$

Using Equation (3.41) 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

$$R = R_3^0 R_6^3 \quad (3.42)$$

as

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R \quad (3.43)$$

As we shall see in Section 3.3.4, 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 (3.43) 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 3.12.

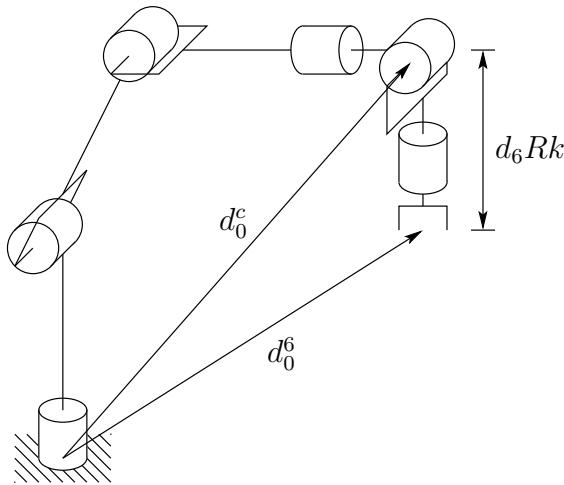


Fig. 3.12 Kinematic decoupling

### 3.3.3 Inverse Position: A Geometric Approach

For the 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 (3.40). We restrict our treatment to the geometric approach for two reasons. First, as we have said, most present manipulator designs are kinematically simple, usually consisting of one of the five basic configurations of Chapter 1 with a spherical wrist. Indeed, it is partly due to the difficulty of the general inverse kinematics problem that manipulator designs have evolved to their present state. Second, there are few techniques that can handle the general inverse kinematics problem for arbitrary configurations. Since the reader is most likely to encounter robot configurations of the type considered here, the added difficulty involved in treating the general case seems unjustified. The interested reader can find more detailed treatment of the general case in [32] [34] [61] [72].

In general the complexity of the inverse kinematics problem increases with the number of nonzero link parameters. For most manipulators, many of the  $a_i, d_i$  are zero, the  $\alpha_i$  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 problem. For example, to solve for  $\theta_1$ , we project the arm onto the  $x_0 - y_0$  plane and use trigonometry to find  $\theta_1$ . We will illustrate this method with two important examples: the articulated and spherical arms.

**3.3.3.1 Articulated Configuration** Consider the elbow manipulator shown in Figure 3.13, with the components of  $o_c^0$  denoted by  $x_c, y_c, z_c$ . We project  $o_c$  onto the  $x_0 - y_0$  plane as shown in Figure 3.14.

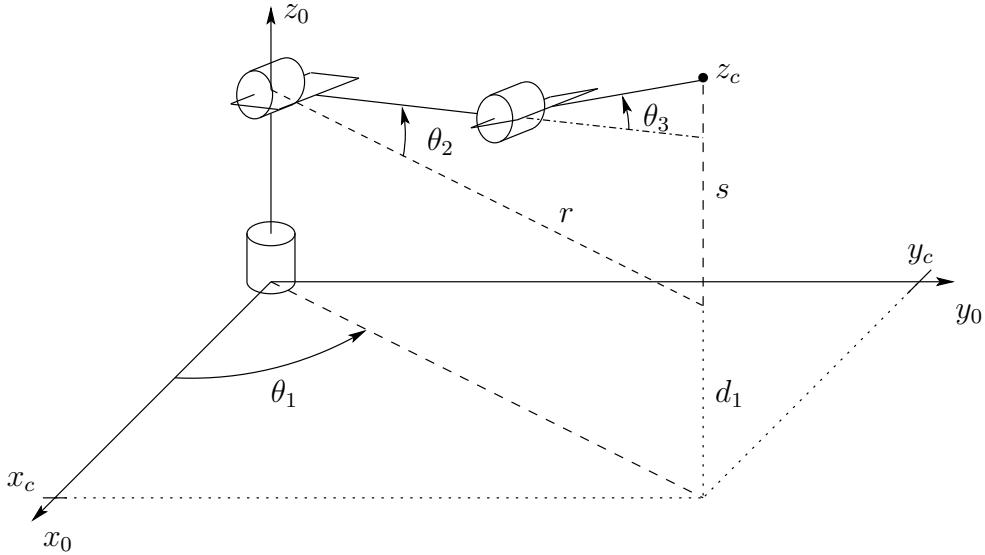


Fig. 3.13 Elbow manipulator

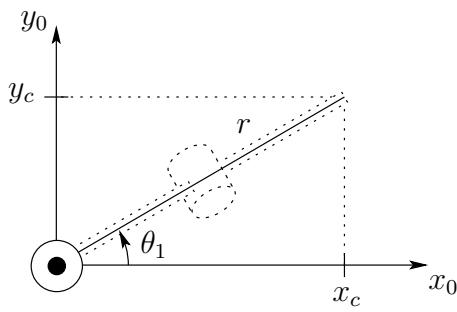


Fig. 3.14 Projection of the wrist center onto  $x_0 - y_0$  plane

We see from this projection that

$$\theta_1 = \text{atan}2(x_c, y_c) \quad (3.44)$$

in which  $\text{atan}2(x, y)$  denotes the two argument arctangent function defined in Chapter 2.

Note that a second valid solution for  $\theta_1$  is

$$\theta_1 = \pi + \text{atan}2(x_c, y_c) \quad (3.45)$$

Of course this will, in turn, lead to different solutions for  $\theta_2$  and  $\theta_3$ , as we will see below.

These solutions for  $\theta_1$ , are valid unless  $x_c = y_c = 0$ . In this case (3.44) is undefined and the manipulator is in a singular configuration, shown in Figure 3.15. In this position the wrist center  $o_c$  intersects  $z_0$ ; hence any value of  $\theta_1$  leaves  $o_c$

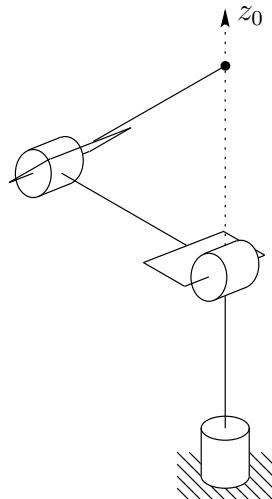


Fig. 3.15 Singular configuration

fixed. There are thus infinitely many solutions for  $\theta_1$  when  $o_c$  intersects  $z_0$ .

If there is an offset  $d \neq 0$  as shown in Figure 3.16 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  $d_3 = d$ . In this case, there will, in general, be only two solutions for  $\theta_1$ . These correspond to the so-called **left arm** and **right arm** configurations as shown in Figures 3.17 and 3.18. Figure 3.17 shows the left arm configuration. From this figure, we see geometrically that

$$\theta_1 = \phi - \alpha \quad (3.46)$$

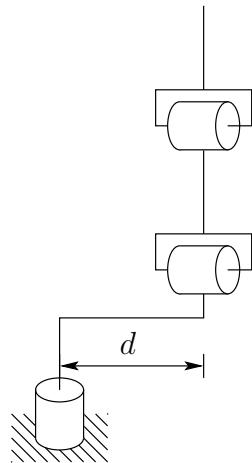


Fig. 3.16 Elbow manipulator with shoulder offset

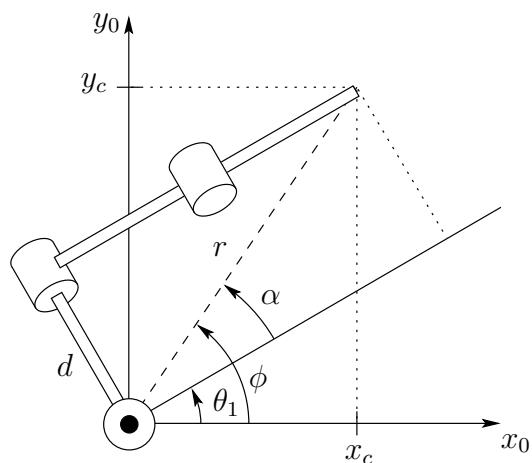


Fig. 3.17 Left arm configuration

where

$$\phi = \text{atan2}(x_c, y_c) \quad (3.47)$$

$$\alpha = \text{atan2}(\sqrt{r^2 - d^2}, d) \quad (3.48)$$

$$= \text{atan2}(\sqrt{x_c^2 + y_c^2 - d^2}, d)$$

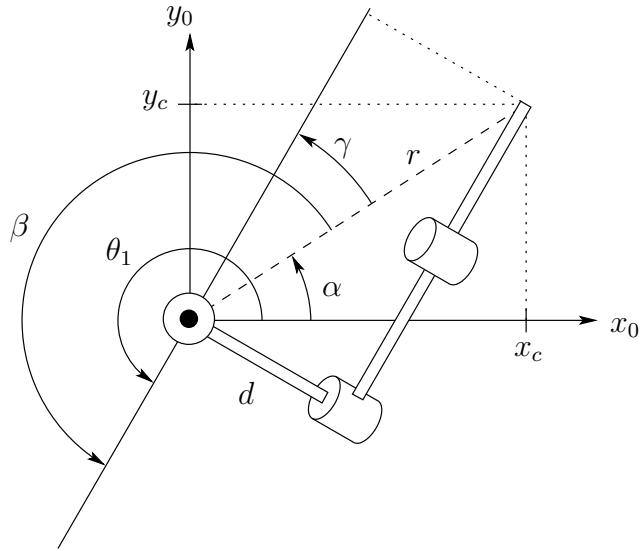


Fig. 3.18 Right arm configuration

The second solution, given by the right arm configuration shown in Figure 3.18 is given by

$$\theta_1 = \text{atan2}(x_c, y_c) + \text{atan2}(-\sqrt{r^2 - d^2}, -d) \quad (3.49)$$

To see this, note that

$$\theta_1 = \alpha + \beta \quad (3.50)$$

$$\alpha = \text{atan2}(x_c, y_c) \quad (3.51)$$

$$\beta = \gamma + \pi \quad (3.52)$$

$$\gamma = \text{atan2}(\sqrt{r^2 - d^2}, d) \quad (3.53)$$

which together imply that

$$\beta = \text{atan2}(-\sqrt{r^2 - d^2}, -d) \quad (3.54)$$

since  $\cos(\theta + \pi) = -\cos(\theta)$  and  $\sin(\theta + \pi) = -\sin(\theta)$ .

To find the angles  $\theta_2, \theta_3$  for the elbow manipulator, given  $\theta_1$ , we consider the plane formed by the second and third links as shown in Figure 3.19. Since the motion of links two and three is planar, the solution is analogous to that of the two-link manipulator of Chapter 1. As in our previous derivation (cf. (1.7) and

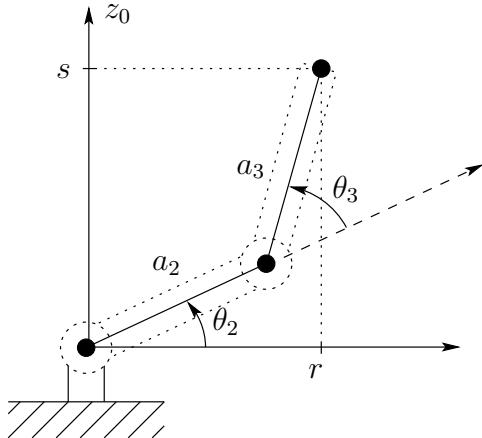


Fig. 3.19 Projecting onto the plane formed by links 2 and 3

(1.8)) we can apply the law of cosines to obtain

$$\begin{aligned}\cos \theta_3 &= \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2a_3} \\ &= \frac{x_c^2 + y_c^2 - d^2 + (z_c - d_1)^2 - a_2^2 - a_3^2}{2a_2a_3} := D\end{aligned}\quad (3.55)$$

since  $r^2 = x_c^2 + y_c^2 - d^2$  and  $s = z_c - d_1$ . Hence,  $\theta_3$  is given by

$$\theta_3 = \text{atan2}\left(D, \pm \sqrt{1 - D^2}\right) \quad (3.56)$$

The two solutions for  $\theta_3$  correspond to the elbow-up position and elbow-down position, respectively.

Similarly  $\theta_2$  is given as

$$\begin{aligned}\theta_2 &= \text{atan2}(r, s) - \text{atan2}(a_2 + a_3 c_3, a_3 s_3) \\ &= \text{atan2}\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c - d_1\right) - \text{atan2}(a_2 + a_3 c_3, a_3 s_3)\end{aligned}\quad (3.57)$$

An example of an elbow manipulator with offsets is the PUMA shown in Figure 3.20. 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.

**3.3.3.2 Spherical Configuration** We next solve the inverse position kinematics for a three degree of freedom spherical manipulator shown in Figure 3.21. As

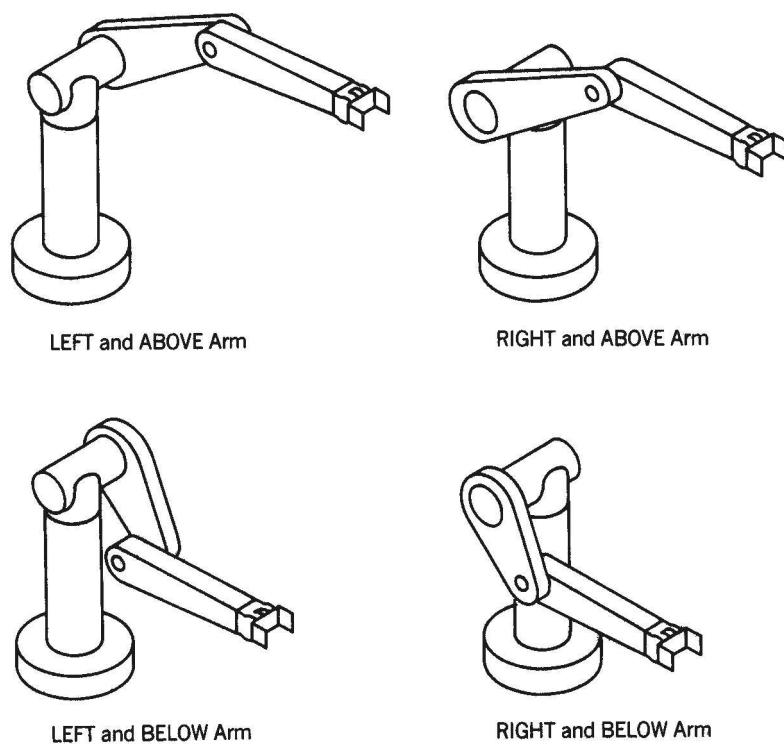


Fig. 3.20 Four solutions of the inverse position kinematics for the PUMA manipulator

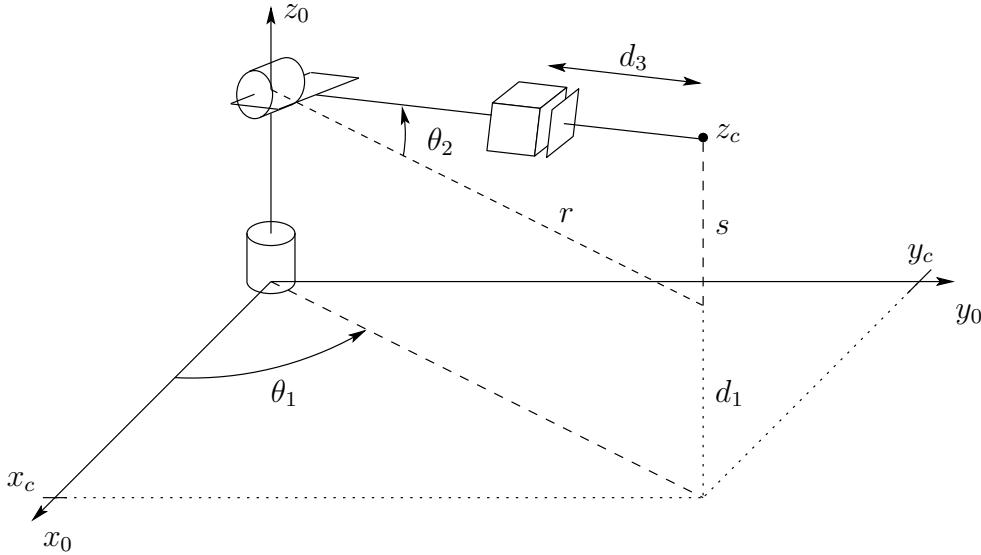


Fig. 3.21 Spherical manipulator

in the case of the elbow manipulator the first joint variable is the base rotation and a solution is given as

$$\theta_1 = \text{atan2}(x_c, y_c) \quad (3.58)$$

provided  $x_c$  and  $y_c$  are not both zero. If both  $x_c$  and  $y_c$  are zero, the configuration is singular as before and  $\theta_1$  may take on any value. As in the case of the elbow manipulator, a second solution for  $\theta_1$  is given by

$$\theta_1 = \pi + \text{atan2}(x_c, y_c). \quad (3.59)$$

The angle  $\theta_2$  is given from Figure 3.21 as

$$\theta_2 = \text{atan2}(r, s) + \frac{\pi}{2} \quad (3.60)$$

where  $r^2 = x_c^2 + y_c^2$ ,  $s = z_c - d_1$ .

The linear distance  $d_3$  is found as

$$d_3 = \sqrt{r^2 + s^2} = \sqrt{x_c^2 + y_c^2 + (z_c - d_1)^2} \quad (3.61)$$

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  $z_0$ . If there is an offset then there will be left and right arm configurations as in the case of the elbow manipulator (Problem 3-25).

**Table 3.6** Link parameters for the articulated manipulator of Figure 3.13

| Link | $a_i$ | $\alpha_i$ | $d_i$ | $\theta_i$   |
|------|-------|------------|-------|--------------|
| 1    | 0     | 90         | $d_1$ | $\theta_1^*$ |
| 2    | $a_2$ | 0          | 0     | $\theta_2^*$ |
| 3    | $a_3$ | 0          | 0     | $\theta_3^*$ |

\* variable

### 3.3.4 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 corresponding to a given position of the wrist origin. 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 (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,  $\phi, \theta, \psi$ , using Equations (2.29) – (2.34), and then use the mapping

$$\begin{aligned}\theta_4 &= \phi \\ \theta_5 &= \theta \\ \theta_6 &= \psi\end{aligned}$$

### Example 3.8 Articulated Manipulator with Spherical Wrist

The DH parameters for the frame assignment shown in Figure 3.13 are summarized in Table 3.6. Multiplying the corresponding  $A_i$  matrices gives the matrix  $R_3^0$  for the articulated or elbow manipulator as

$$R_3^0 = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 \\ s_1 c_{23} & -s_1 s_{23} & -c_1 \\ s_{23} & c_{23} & 0 \end{bmatrix} \quad (3.62)$$

The matrix  $R_6^3 = A_4 A_5 A_6$  is given as

$$R_6^3 = \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \end{bmatrix} \quad (3.63)$$

The equation to be solved for the final three variables is therefore

$$R_6^3 = (R_3^0)^T R \quad (3.64)$$

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

$$c_4 s_5 = c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33} \quad (3.65)$$

$$s_4 s_5 = -c_1 s_{23} r_{13} - s_1 s_{23} r_{23} + c_{23} r_{33} \quad (3.66)$$

$$c_5 = s_1 r_{13} - c_1 r_{23} \quad (3.67)$$

Hence, if not both of the expressions (3.65), (3.66) are zero, we obtain  $\theta_5$  from (2.29) and (2.30) as

$$\theta_5 = \text{atan2}\left(s_1 r_{13} - c_1 r_{23}, \pm \sqrt{1 - (s_1 r_{13} - c_1 r_{23})^2}\right) \quad (3.68)$$

If the positive square root is chosen in (3.68), then  $\theta_4$  and  $\theta_6$  are given by (2.31) and (2.32), respectively, as

$$\begin{aligned} \theta_4 &= \text{atan2}(c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33}, \\ &\quad -c_1 s_{23} r_{13} - s_1 s_{23} r_{23} + c_{23} r_{33}) \end{aligned} \quad (3.69)$$

$$\theta_6 = \text{atan2}(-s_1 r_{11} + c_1 r_{21}, s_1 r_{12} - c_1 r_{22}) \quad (3.70)$$

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  $\theta_4$  arbitrarily and then determine  $\theta_6$  using (2.36) or (2.38).

◇

### 3.3.5 Examples

#### Example 3.9 Elbow Manipulator - Complete Solution

To summarize the geometric approach for solving the inverse kinematics equations, we write give here one solution to the inverse kinematics of the six degree-of-freedom elbow manipulator shown in Figure 3.13 which has no joint offsets and a spherical wrist.

Given

$$o = \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix}; \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (3.71)$$

then with

$$x_c = o_x - d_6 r_{13} \quad (3.72)$$

$$y_c = o_y - d_6 r_{23} \quad (3.73)$$

$$z_c = o_z - d_6 r_{33} \quad (3.74)$$

a set of DH joint variables is given by

$$\theta_1 = \text{atan2}(x_c, y_c) \quad (3.75)$$

$$\theta_2 = \text{atan2}\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c - d_1\right) - \text{atan2}(a_2 + a_3 c_3, a_3 s_3) \quad (3.76)$$

$$\theta_3 = \text{atan2}\left(D, \pm\sqrt{1 - D^2}\right),$$

where  $D = \frac{x_c^2 + y_c^2 - d^2 + (z_c - d_1)^2 - a_2^2 - a_3^2}{2a_2 a_3}$

$$(3.77)$$

$$\begin{aligned} \theta_4 &= \text{atan2}(c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33}, \\ &\quad -c_1 s_{23} r_{13} - s_1 s_{23} r_{23} + c_{23} r_{33}) \end{aligned} \quad (3.78)$$

$$\theta_5 = \text{atan2}\left(s_1 r_{13} - c_1 r_{23}, \pm\sqrt{1 - (s_1 r_{13} - c_1 r_{23})^2}\right) \quad (3.79)$$

$$\theta_6 = \text{atan2}(-s_1 r_{11} + c_1 r_{21}, s_1 r_{12} - c_1 r_{22}) \quad (3.80)$$

The other possible solutions are left as an exercise (Problem 3-24).

◇

### Example 3.10 SCARA Manipulator

As another example, we consider the SCARA manipulator whose forward kinematics is defined by  $T_4^0$  from (3.30). The inverse kinematics solution is then given as the set of solutions of the equation

$$\begin{aligned} T_4^1 &= \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_{12}c_4 + s_{12}s_4 & s_{12}c_4 - c_{12}s_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -c_{12}c_4 - s_{12}s_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (3.81)$$

We first note that, since the SCARA has only four degrees-of-freedom, not every possible  $H$  from  $SE(3)$  allows a solution of (3.81). In fact we can easily see that there is no solution of (3.81) unless  $R$  is of the form

$$R = \begin{bmatrix} c_\alpha & s_\alpha & 0 \\ s_\alpha & -c_\alpha & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (3.82)$$

and if this is the case, the sum  $\theta_1 + \theta_2 - \theta_4$  is determined by

$$\theta_1 + \theta_2 - \theta_4 = \alpha = \text{atan2}(r_{11}, r_{12}) \quad (3.83)$$

Projecting the manipulator configuration onto the  $x_0 - y_0$  plane immediately yields the situation of Figure 3.22. We see from this that

$$\theta_2 = \text{atan2}(c_2, \pm\sqrt{1 - c_2}) \quad (3.84)$$

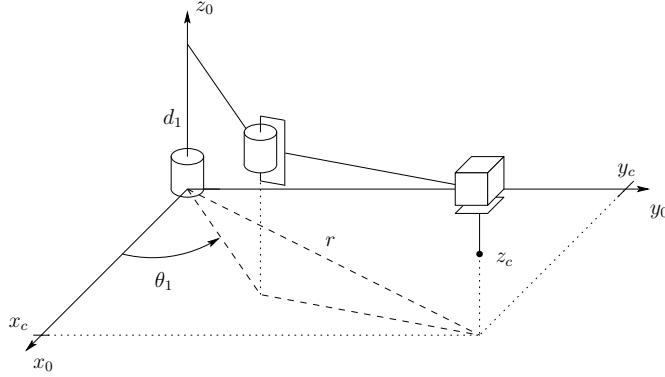


Fig. 3.22 SCARA manipulator

where

$$c_2 = \frac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2} \quad (3.85)$$

$$\theta_1 = \text{atan2}(o_x, o_y) - \text{atan2}(a_1 + a_2c_2, a_2s_2) \quad (3.86)$$

We may then determine  $\theta_4$  from (3.83) as

$$\begin{aligned} \theta_4 &= \theta_1 + \theta_2 - \alpha \\ &= \theta_1 + \theta_2 - \text{atan2}(r_{11}, r_{12}) \end{aligned} \quad (3.87)$$

Finally  $d_3$  is given as

$$d_3 = o_z + d_4 \quad (3.88)$$

◊

### 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. We began by introducing the Denavit-Hartenberg convention for assigning coordinate frames to the links of a serial manipulator. We may summarize the procedure based on the DH convention in the following algorithm for deriving the forward kinematics for any manipulator.

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

**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  $i = 1, \dots, n - 1$ , 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  $z_n = a$  along the direction  $z_{n-1}$ . 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 link parameters  $a_i, d_i, \alpha_i, \theta_i$ .

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

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

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

$\theta_i$  = the angle between  $x_{i-1}$  and  $x_i$  measured about  $z_{i-1}$ .  $\theta_i$  is variable if joint  $i$  is revolute.

**Step 8:** Form the homogeneous transformation matrices  $A_i$  by substituting the above parameters into (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.

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 problem, i.e., given a position and orientation for the end effector, solve for the corresponding set of joint variables. In this chapter, we have 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

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.89)$$

**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

$$R_6^3 = (R_3^0)^{-1}R = (R_3^0)^T R \quad (3.90)$$

In this chapter, 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  $q_i$ .

### 3.5 NOTES AND REFERENCES

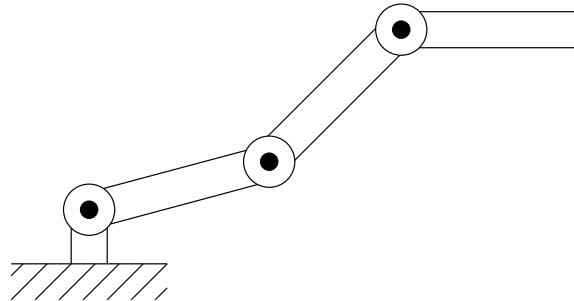
Kinematics and inverse kinematics have been the subject of research in robotics for many years. Some of the seminal work in these areas can be found in [9] [14] [20] [22] [43] [44] [60] [71] [35] [76] [2] [32] [34] [44] [45] [60] [61] [66] [72].

---

## Problems

---

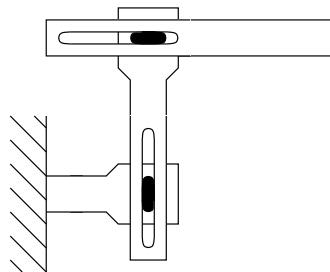
1. Verify the statement after Equation (3.14) that the rotation matrix  $R$  has the form (3.13) provided assumptions DH1 and DH2 are satisfied.
2. Consider the three-link planar manipulator shown in Figure 3.23. Derive



*Fig. 3.23* Three-link planar arm of Problem 3-2

the forward kinematic equations using the DH-convention.

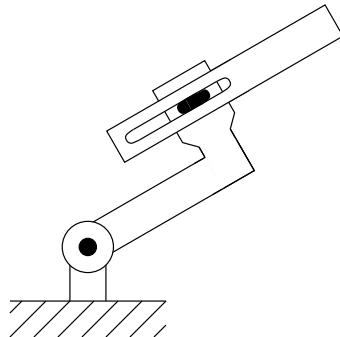
3. Consider the two-link cartesian manipulator of Figure 3.24. Derive the



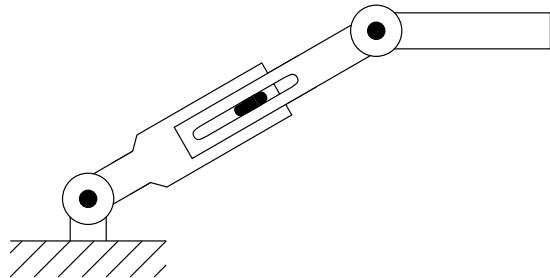
*Fig. 3.24* Two-link cartesian robot of Problem 3-3

forward kinematic equations using the DH-convention.

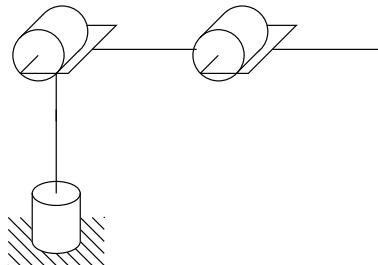
4. Consider the two-link manipulator of Figure 3.25 which has joint 1 revolute and joint 2 prismatic. Derive the forward kinematic equations using the DH-convention.
5. Consider the three-link planar manipulator of Figure 3.26. Derive the forward kinematic equations using the DH-convention.
6. Consider the three-link articulated robot of Figure 3.27. Derive the for-



*Fig. 3.25* Two-link planar arm of Problem 3-4



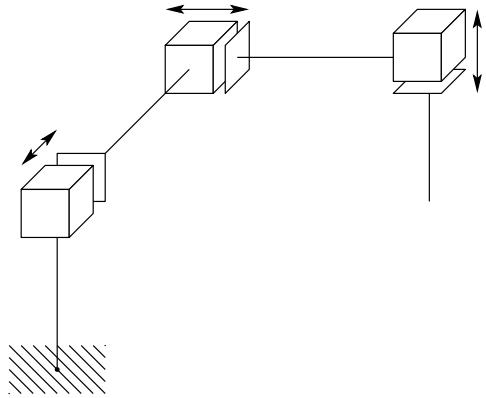
*Fig. 3.26* Three-link planar arm with prismatic joint of Problem 3-5



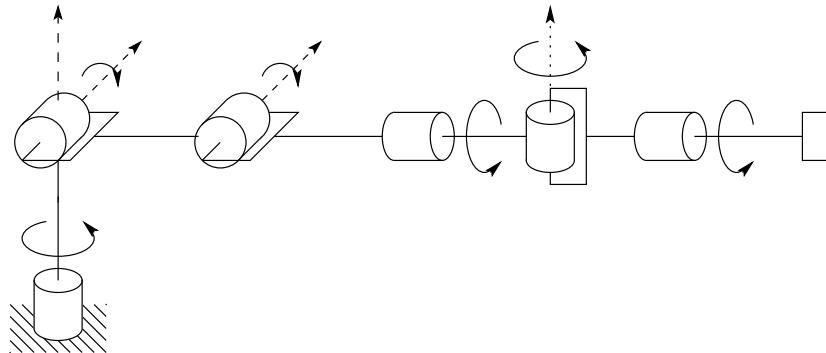
*Fig. 3.27* Three-link articulated robot

ward kinematic equations using the DH-convention.

7. Consider the three-link cartesian manipulator of Figure 3.28. Derive the forward kinematic equations using the DH-convention.
8. Attach a spherical wrist to the three-link articulated manipulator of Problem 3-6. as shown in Figure 3.29. Derive the forward kinematic equations for this manipulator.

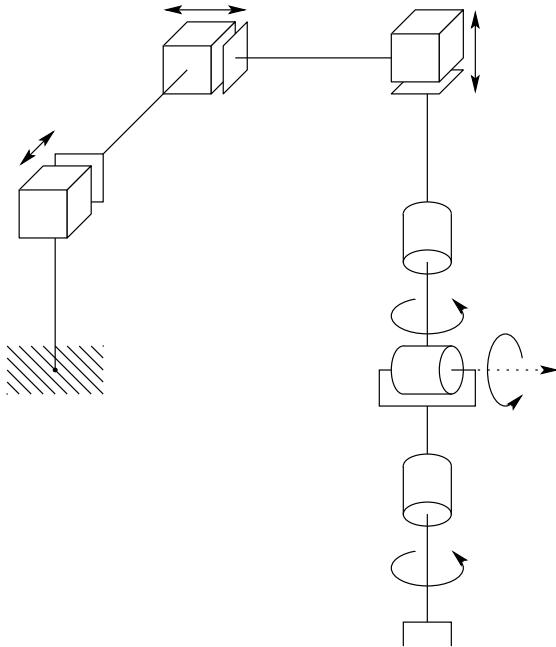


*Fig. 3.28* Three-link cartesian robot



*Fig. 3.29* Elbow manipulator with spherical wrist

9. Attach a spherical wrist to the three-link cartesian manipulator of Problem 3-7 as shown in Figure 3.30. Derive the forward kinematic equations for this manipulator.
10. Consider the PUMA 260 manipulator shown in Figure 3.31. Derive the complete set of forward kinematic equations, by establishing appropriate DH coordinate frames, constructing a table of link parameters, forming the A-matrices, etc.
11. Repeat Problem 3-9 for the five degree-of-freedom Rhino XR-3 robot shown in Figure 3.32. (Note: you should replace the Rhino wrist with the spherical wrist.)
12. Suppose that a Rhino XR-3 is bolted to a table upon which a coordinate frame  $o_s x_s y_s z_s$  is established as shown in Figure 3.33. (The frame  $o_s x_s y_s z_s$  is often referred to as the **station frame**.) Given the base frame



*Fig. 3.30* Cartesian manipulator with spherical wrist

that you established in Problem 3-11, find the homogeneous transformation  $T_0^s$  relating the base frame to the station frame. Find the homogeneous transformation  $T_5^s$  relating the end-effector frame to the station frame. What is the position and orientation of the end-effector in the station frame when  $\theta_1 = \theta_2 = \dots = \theta_5 = 0$ ?

13. Consider the GMF S-400 robot shown in Figure 3.34 Draw the symbolic representation for this manipulator. Establish DH-coordinate frames and write the forward kinematic equations.
14. 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 3.35? If the orientation of the end-effector is also specified, how many solutions are there? Use the geometric approach to find them.
15. Repeat Problem 3-14 for the three-link planar arm with prismatic joint of Figure 3.36.
16. Solve the inverse position kinematics for the cylindrical manipulator of Figure 3.37.
17. Solve the inverse position kinematics for the cartesian manipulator of Figure 3.38.

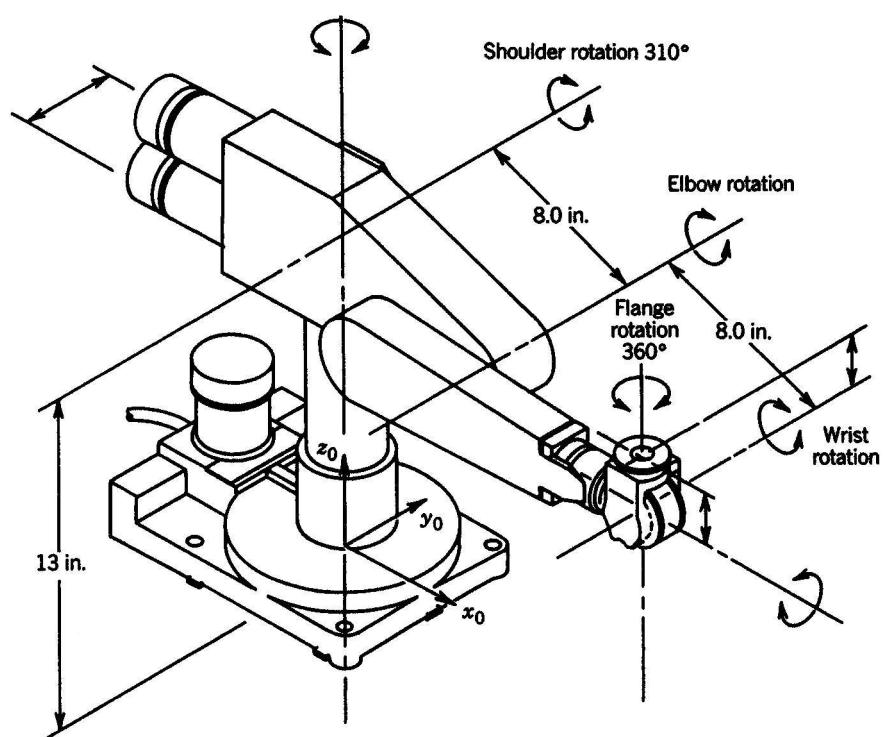


Fig. 3.31 PUMA 260 manipulator

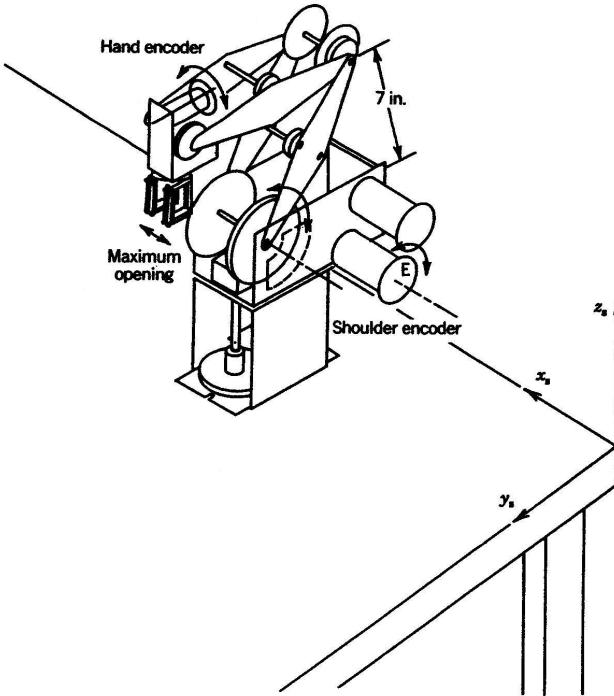


Fig. 3.32 Rhino XR-3 robot

18. Add a spherical wrist to the three-link cylindrical arm of Problem 3-16 and write the complete inverse kinematics solution.
19. Repeat Problem 3-16 for the cartesian manipulator of Problem 3-17.
20. Write a computer program to compute the inverse kinematic equations for the elbow manipulator using Equations (3.75)-(3.80). Include procedures for identifying singular configurations and choosing a particular solution when the configuration is singular. Test your routine for various special cases, including singular configurations.
21. The Stanford manipulator of Example 3.3.5 has a spherical wrist. Therefore, given a desired position  $O$  and orientation  $R$  of the end-effector,
  - a) Compute the desired coordinates of the wrist center  $O_c^0$ .
  - b) 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?

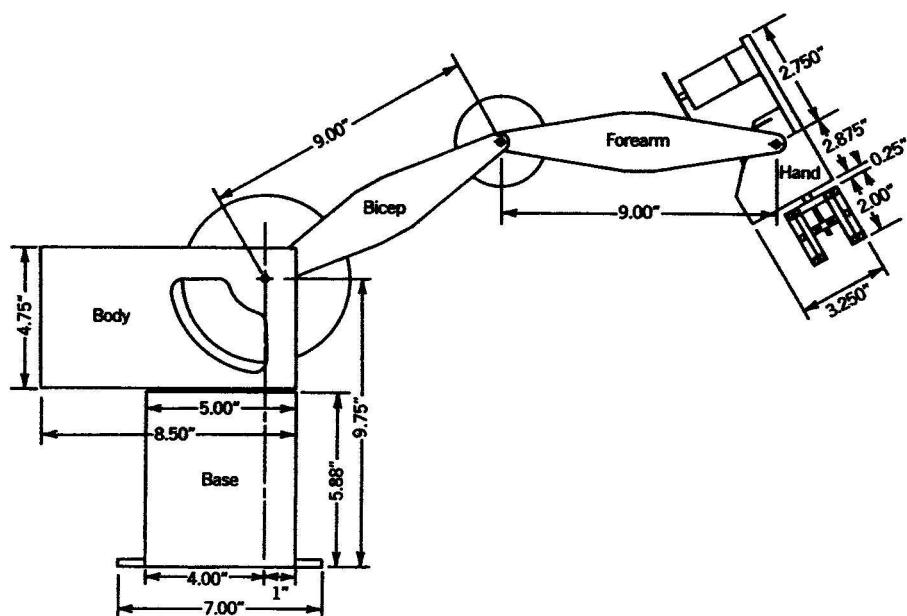


Fig. 3.33 Rhino robot attached to a table. From: *A Robot Engineering Textbook*, by Mohsen Shahinpoor. Copyright 1987, Harper & Row Publishers, Inc

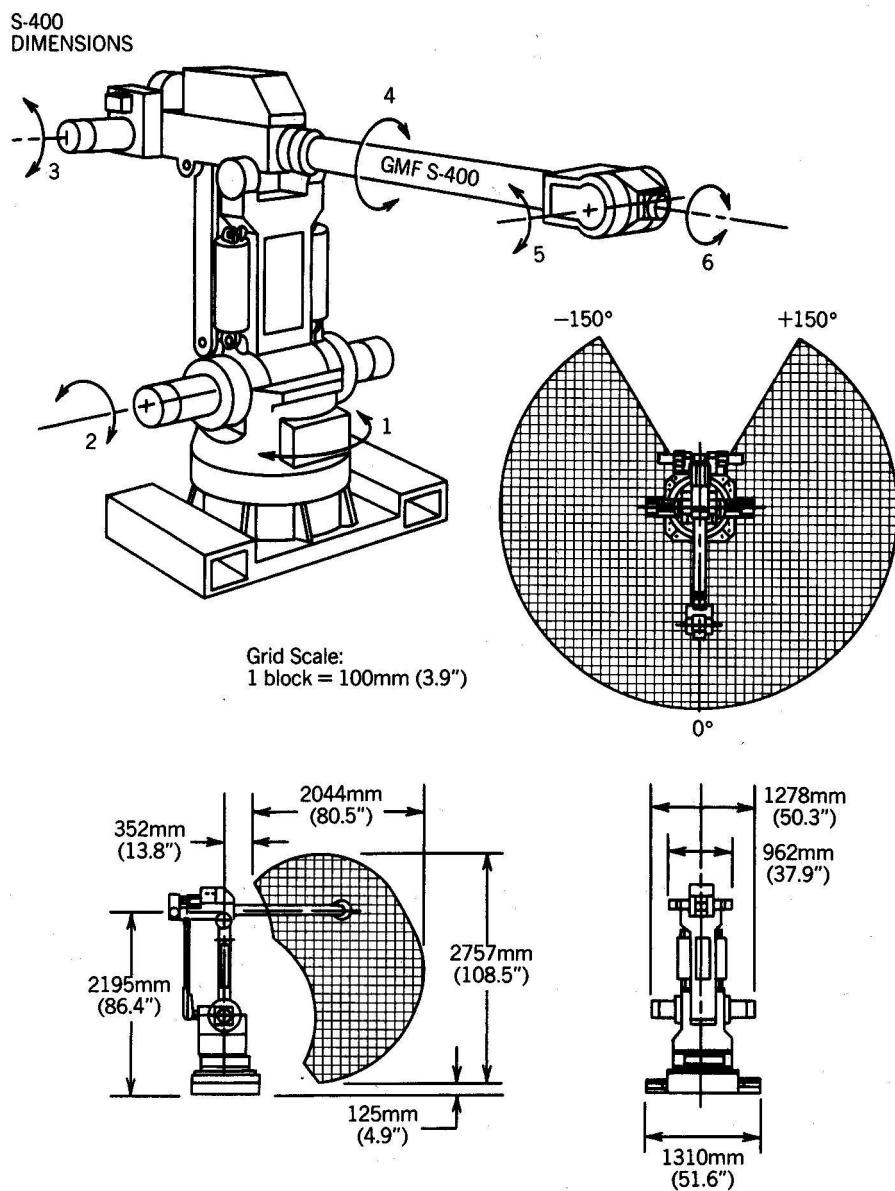
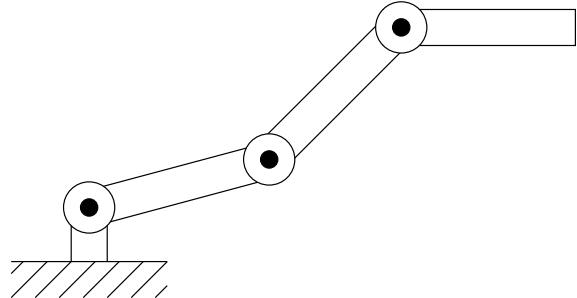
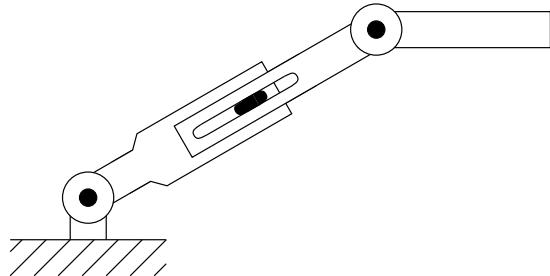


Fig. 3.34 GMF S-400 robot. (Courtesy GMF Robotics.)



*Fig. 3.35* Three-link planar robot with revolute joints.



*Fig. 3.36* Three-link planar robot with prismatic joint

- c) 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 (3.63).
- 22. Repeat Problem 3-21 for the PUMA 260 manipulator of Problem 3-9, which also has a spherical wrist. How many total solutions did you find?
- 23. Solve the inverse position kinematics for the Rhino robot.
- 24. ). Find all other solutions to the inverse kinematics of the elbow manipulator of Example 3.9.
- 25. . Modify the solutions  $\theta_1$  and  $\theta_2$  for the spherical manipulator given by Equations (3.58) and (3.60) in the case of a shoulder offset.

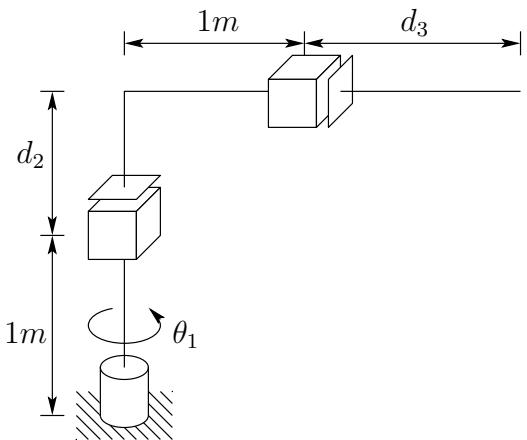


Fig. 3.37 Cylindrical configuration

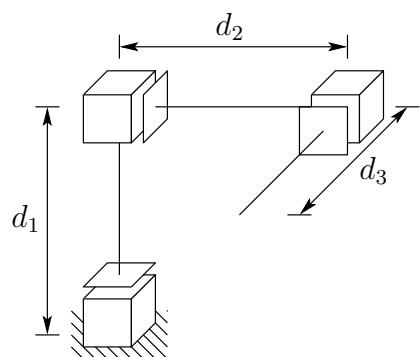


Fig. 3.38 Cartesian configuration

# 4

---

## VELOCITY KINEMATICS – THE MANIPULATOR JACOBIAN

In the previous chapter we derived the forward and inverse position 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 between the space of cartesian positions and orientations and the space of joint positions. The velocity relationships are then determined by the **Jacobian** of this function. The Jacobian is a matrix that can be thought of as the vector version 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 6-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 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 discuss the notion of **singular configurations**. These are configurations in which the manipulator loses one or more degrees-of-freedom. 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 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  $\theta$ , 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

$$\omega = \dot{\theta}k \quad (4.1)$$

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

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 equation

$$v = \omega \times r \quad (4.2)$$

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  $\theta$  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  $\omega$  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

In Section 4.3 we will derive properties of rotation matrices that can be used to compute relative velocity transformations between coordinate frames. Such transformations involve derivatives of rotation matrices. By introducing the notion of a skew symmetric matrix it is possible to simplify many of the computations involved.

**Definition 4.1** An  $n \times n$  matrix  $S$  is said to be **skew symmetric** if and only if

$$S^T + S = 0 \quad (4.3)$$

We denote the set of all  $3 \times 3$  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

$$s_{ij} + s_{ji} = 0 \quad i, j = 1, 2, 3 \quad (4.4)$$

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

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

If  $a = (a_x, a_y, a_z)^T$  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**

We 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

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

◊

**4.2.1 Properties of Skew Symmetric Matrices**

Skew symmetric matrices possess several properties that will prove useful for subsequent derivations.<sup>1</sup> Among these properties are

1. The operator  $S$  is linear, i.e.,

$$S(\alpha a + \beta b) = \alpha S(a) + \beta S(b) \quad (4.6)$$

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

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

$$S(a)p = a \times p \quad (4.7)$$

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

3. If  $R \in SO(3)$  and  $a, b$  are vectors in  $\mathbb{R}^3$  it can also be shown by direct calculation that

$$R(a \times b) = Ra \times Rb \quad (4.8)$$

Equation (4.8) is **not** true in general unless  $R$  is orthogonal. Equation (4.8) 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

<sup>1</sup>These properties are consequences of the fact that  $so(3)$  is a *Lie Algebra*, a vector space with a suitably defined product operation [8].

$Ra$  and  $Rb$ , the result is the same as that obtained by first forming the cross product  $a \times b$  and then rotating to obtain  $R(a \times b)$ .

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

$$RS(a)R^T = S(Ra) \quad (4.9)$$

This property follows easily from Equations (4.7) and (4.8) 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.

As we will see, Equation (4.9) is one of the most useful expressions that we will derive. The left hand side of Equation (4.9) 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  $\theta$ . Hence  $R = R(\theta) \in SO(3)$  for every  $\theta$ . Since  $R$  is orthogonal for all  $\theta$  it follows that

$$R(\theta)R(\theta)^T = I \quad (4.10)$$

Differentiating both sides of Equation (4.10) with respect to  $\theta$  using the product rule gives

$$\frac{dR}{d\theta}R(\theta)^T + R(\theta)\frac{dR^T}{d\theta} = 0 \quad (4.11)$$

Let us define the matrix  $S$  as

$$S := \frac{dR}{d\theta}R(\theta)^T \quad (4.12)$$

Then the transpose of  $S$  is

$$S^T = \left( \frac{dR}{d\theta}R(\theta)^T \right)^T = R(\theta)\frac{dR^T}{d\theta} \quad (4.13)$$

Equation (4.11) says therefore that

$$S + S^T = 0 \quad (4.14)$$

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

$$\frac{dR}{d\theta} = SR(\theta) \quad (4.15)$$

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 (2.6), then direct computation shows that

$$\begin{aligned} S = \frac{dR}{d\theta} R^T &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin \theta & -\cos \theta \\ 0 & \cos \theta & -\sin \theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = S(i) \end{aligned}$$

Thus we have shown that

$$\frac{dR_{x,\theta}}{d\theta} = S(i)R_{x,\theta}$$

Similar computations show that

$$\frac{dR_{y,\theta}}{d\theta} = S(j)R_{y,\theta} \quad \text{and} \quad \frac{dR_{z,\theta}}{d\theta} = S(k)R_{z,\theta} \quad (4.16)$$

◇

### Example 4.3

Let  $R_{k,\theta}$  be a rotation about the axis defined by  $k$  as in Equation (2.46). Note that in this example  $k$  is not the unit coordinate vector  $(0, 0, 1)^T$ . It is easy to check that  $S(k)^3 = -S(k)$ . Using this fact together with Problem 4-25 it follows that

$$\frac{dR_{k,\theta}}{d\theta} = S(k)R_{k,\theta} \quad (4.17)$$

◇

## 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  $R(t)$  is given by

$$\dot{R}(t) = S(t)R(t) \quad (4.18)$$

where the matrix  $S(t)$  is skew symmetric. Now, since  $S(t)$  is skew symmetric, it can be represented as  $S(\omega(t))$  for a unique vector  $\omega(t)$ . This vector  $\omega(t)$  is the **angular velocity** of the rotating frame with respect to the fixed frame at time  $t$ . Thus, the time derivative  $\dot{R}(t)$  is given by

$$\dot{R}(t) = S(\omega(t))R(t) \quad (4.19)$$

in which  $\omega(t)$  is the angular velocity.

Equation (4.19) 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 (4.19). 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  $\omega$  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, e.g., using  $\omega_2$  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

$$\dot{R} = \frac{dR}{dt} = \frac{dR}{d\theta} \frac{d\theta}{dt} = \dot{\theta} S(i)R(t) = S(\omega(t))R(t) \quad (4.20)$$

in which  $\omega = i\dot{\theta}$  is the **angular velocity**. Note, here  $i = (1, 0, 0)^T$ .

◇

#### 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). As in Chapter 2,

$$R_2^0(t) = R_1^0(t)R_2^1(t) \quad (4.21)$$

Taking derivatives of both sides of Equation (4.21) with respect to time yields

$$\dot{R}_2^0 = \dot{R}_1^0R_2^1 + R_1^0\dot{R}_2^1 \quad (4.22)$$

Using Equation (4.19), the term  $\dot{R}_2^0$  on the left-hand side of Equation (4.22) can be written

$$\dot{R}_2^0 = S(\omega_{0,2}^0)R_2^0 \quad (4.23)$$

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 (4.22) is simply

$$\dot{R}_1^0R_2^1 = S(\omega_{0,1}^0)R_1^0R_2^1 = S(\omega_{0,1}^0)R_2^0 \quad (4.24)$$

Note that 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 (4.22). Using Equation (4.9) we have

$$R_1^0\dot{R}_2^1 = R_1^0S(\omega_{1,2}^1)R_2^1 \quad (4.25)$$

$$\begin{aligned} &= R_1^0S(\omega_{1,2}^1){R_1^0}^T R_1^0R_2^1 = S(R_1^0\omega_{1,2}^1)R_1^0R_2^1 \\ &= S(R_1^0\omega_{1,2}^1)R_2^0. \end{aligned} \quad (4.26)$$

Note that 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  $o_0x_0y_0z_0$ , i.e.,  $R_1^0\omega_{1,2}^1$  gives the coordinates of the free vector  $\omega_{1,2}$  with respect to frame 0.

Now, combining the above expressions we have shown that

$$S(\omega_2^0)R_2^0 = \{S(\omega_{0,1}^0) + S(R_1^0\omega_{1,2}^1)\}R_2^0 \quad (4.27)$$

Since  $S(a) + S(b) = S(a + b)$ , we see that

$$\omega_2^0 = \omega_{0,1}^0 + R_1^0\omega_{1,2}^1 \quad (4.28)$$

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

$$R_n^0 = R_1^0 R_2^1 \cdots R_n^{n-1} \quad (4.29)$$

Although it is a slight abuse of notation, let us represent by  $\omega_i^{i-1}$  the angular velocity due to the rotation given by  $R_i^{i-1}$ , expressed relative to frame  $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$ . Extending the above reasoning we obtain

$$\dot{R}_n^0 = S(\omega_{0,n}^0) R_n^0 \quad (4.30)$$

in which

$$\omega_{0,n}^0 = \omega_{0,1}^0 + R_1^0 \omega_{1,2}^1 + R_2^0 \omega_{2,3}^2 + R_3^0 \omega_{3,4}^3 + \cdots + R_{n-1}^0 \omega_{n-1,n}^{n-1} \quad (4.31)$$

$$= \omega_{0,1}^0 + \omega_{1,2}^0 + \omega_{2,3}^0 + \omega_{3,4}^0 + \cdots + \omega_{n-1,n}^0 \quad (4.32)$$

## 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

$$p^0 = R_1^0(t)p^1. \quad (4.33)$$

The velocity  $\dot{p}^0$  is then given by the product rule for differentiation as

$$\dot{p}^0 = \dot{R}_1^0(t)p^1 + R_1^0(t)\dot{p}^1 \quad (4.34)$$

$$= S(\omega^0)R_1^0(t)p^1 \quad (4.35)$$

$$= S(\omega^0)p^0 = \omega^0 \times p^0 \quad (4.36)$$

which is the familiar expression for the velocity in terms of the vector cross product. Note that Equation (4.35) 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

$$H_1^0(t) = \begin{bmatrix} R_1^0(t) & o_1^0(t) \\ 0 & 1 \end{bmatrix} \quad (4.37)$$

For simplicity we omit the argument  $t$  and the subscripts and superscripts on  $R_1^0$  and  $o_1^0$ , and write

$$p^0 = Rp^1 + o \quad (4.38)$$

Differentiating the above expression using the product rule gives

$$\begin{aligned}\dot{p}^0 &= \dot{R}p^1 + \dot{o} \\ &= S(\omega)Rp^1 + \dot{o} \\ &= \omega \times r + v\end{aligned}\tag{4.39}$$

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  $q_1, \dots, q_n$ . Let

$$T_n^0(q) = \begin{bmatrix} R_n^0(q) & o_n^0(q) \\ 0 & 1 \end{bmatrix}\tag{4.40}$$

denote the transformation from the end-effector frame to the base frame, where  $q = (q_1, \dots, q_n)^T$  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

$$S(\omega_n^0) = \dot{R}_n^0(R_n^0)^T\tag{4.41}$$

define the angular velocity vector  $\omega_n^0$  of the end-effector, and let

$$v_n^0 = \dot{o}_n^0\tag{4.42}$$

denote the linear velocity of the end effector. We seek expressions of the form

$$v_n^0 = J_v \dot{q}\tag{4.43}$$

$$\omega_n^0 = J_\omega \dot{q}\tag{4.44}$$

where  $J_v$  and  $J_\omega$  are  $3 \times n$  matrices. We may write Equations (4.43) and (4.44) together as

$$\xi = J \dot{q}\tag{4.45}$$

in which  $\xi$  and  $J$  are given by

$$\xi = \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} \quad \text{and} \quad J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}\tag{4.46}$$

The vector  $\xi$  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 (4.31) 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  $\theta_i$  and the axis of rotation is  $z_{i-1}$ . Following the convention that we introduced above, let  $\omega_i^{i-1}$  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

$$\omega_i^{i-1} = \dot{q}_i z_{i-1}^{i-1} = \dot{q}_i k \quad (4.47)$$

in which, as above,  $k$  is the unit coordinate vector  $(0, 0, 1)^T$ .

If the  $i$ -th joint is prismatic, then the motion of frame  $i$  relative to frame  $i-1$  is a translation and

$$\omega_i^{i-1} = 0 \quad (4.48)$$

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,  $\omega_n^0$ , in the base frame is determined by Equation (4.31) as

$$\begin{aligned} \omega_n^0 &= \rho_1 \dot{q}_1 k + \rho_2 \dot{q}_2 R_1^0 k + \cdots + \rho_n \dot{q}_n R_{n-1}^0 k \\ &= \sum_{i=1}^n \rho_i \dot{q}_i z_{i-1}^0 \end{aligned} \quad (4.49)$$

in which  $\rho_i$  is equal to 1 if joint  $i$  is revolute and 0 if joint  $i$  is prismatic, since

$$z_{i-1}^0 = R_{i-1}^0 k \quad (4.50)$$

Of course  $z_0^0 = k = (0, 0, 1)^T$ .

The lower half of the Jacobian  $J_\omega$ , in Equation (4.46) is thus given as

$$J_\omega = [\rho_1 z_0 \cdots \rho_n z_{n-1}] . \quad (4.51)$$

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 occasionally will follow this convention when there is no ambiguity concerning the reference frame.

### 4.6.2 Linear Velocity

The linear velocity of the end-effector is just  $\dot{o}_n^0$ . By the chain rule for differentiation

$$\dot{o}_n^0 = \sum_{i=1}^n \frac{\partial o_n^0}{\partial q_i} \dot{q}_i \quad (4.52)$$

Thus we see that the  $i$ -th column of  $J_v$ , which we denote as  $J_{v_i}$  is given by

$$J_{v_i} = \frac{\partial o_n^0}{\partial q_i} \quad (4.53)$$

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 fixed but the  $i$ -th and actuating the  $i$ -th at unit velocity. We now consider the two cases (prismatic and revolute joints) separately.

#### (i) Case 1: Prismatic Joints

If joint  $i$  is prismatic, then it imparts a pure translation to the end-effector. From our study of the DH convention in Chapter 3, we can write the  $T_n^0$  as the product of three transformations as follows

$$\begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix} = T_n^0 \quad (4.54)$$

$$= T_{i-1}^0 T_i^{i-1} T_n^i \quad (4.55)$$

$$= \begin{bmatrix} R_{i-1}^0 & o_{i-1}^0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_n^i & o_n^i \\ 0 & 1 \end{bmatrix} \quad (4.56)$$

$$= \begin{bmatrix} R_n^0 & R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0 \\ 0 & 1 \end{bmatrix}, \quad (4.57)$$

which gives

$$o_n^0 = R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0 \quad (4.58)$$

If only joint  $i$  is allowed to move, then both of  $o_n^i$  and  $o_{i-1}^0$  are constant. Furthermore, if joint  $i$  is prismatic, then the rotation matrix  $R_{i-1}^0$  is also constant (again, assuming that only joint  $i$  is allowed to move). Finally, recall from Chapter 3 that, by the DH convention,  $o_i^{i-1} = (a_i c_i, a_i s_i, d_i)^T$ . Thus,

differentiation of  $o_n^0$  gives

$$\frac{\partial o_n^0}{\partial q_i} = \frac{\partial}{\partial d_i} R_{i-1}^0 o_i^{i-1} \quad (4.59)$$

$$= R_{i-1}^0 \frac{\partial}{\partial d_i} \begin{bmatrix} a_i c_i \\ a_i s_i \\ d_i \end{bmatrix} \quad (4.60)$$

$$= \dot{d}_i R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.61)$$

$$= \dot{d}_i z_{i-1}^0, \quad (4.62)$$

in which  $d_i$  is the joint variable for prismatic joint  $i$ . Thus, (again, dropping the zero superscript on the z-axis) for the case of prismatic joints we have

$$J_{v_i} = z_{i-1} \quad (4.63)$$

### (ii) Case 2: Revolute Joints

If joint  $i$  is revolute, then we have  $q_i = \theta_i$ . Starting with Equation (4.58), and letting  $q_i = \theta_i$ , since  $R_i^0$  is not constant with respect to  $\theta_i$ , we obtain

$$\frac{\partial}{\partial \theta_i} o_n^0 = \frac{\partial}{\partial \theta_i} [R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1}] \quad (4.64)$$

$$= \frac{\partial}{\partial \theta_i} R_i^0 o_n^i + R_{i-1}^0 \frac{\partial}{\partial \theta_i} o_i^{i-1} \quad (4.65)$$

$$= \dot{\theta}_i S(z_{i-1}^0) R_i^0 o_n^i + \dot{\theta}_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1} \quad (4.66)$$

$$= \dot{\theta}_i S(z_{i-1}^0) [R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1}] \quad (4.67)$$

$$= \dot{\theta}_i S(z_{i-1}^0) (o_n^0 - o_{i-1}^0) \quad (4.68)$$

$$= \dot{\theta}_i z_{i-1}^0 \times (o_n^0 - o_{i-1}^0) \quad (4.69)$$

The second term in Equation (4.66) is derived as follows:

$$R_{i-1}^0 \frac{\partial}{\partial \theta_i} \begin{bmatrix} a_i c_i \\ a_i s_i \\ d_i \end{bmatrix} = R_{i-1}^0 \begin{bmatrix} -a_i s_i \\ a_i c_i \\ 0 \end{bmatrix} \dot{\theta}_i \quad (4.70)$$

$$= R_{i-1}^0 S(k\dot{\theta}_i) o_i^{i-1} \quad (4.71)$$

$$= R_{i-1}^0 S(k\dot{\theta}_i) (R_{i-1}^0)^T R_{i-1}^0 o_i^{i-1} \quad (4.72)$$

$$= S(R_{i-1}^0 k\dot{\theta}_i) R_{i-1}^0 o_i^{i-1} \quad (4.73)$$

$$= \dot{\theta}_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1} \quad (4.74)$$

Equation (4.71) follows by straightforward computation. Thus for a revolute joint

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1}) \quad (4.75)$$

in which we have, following our convention, omitted the zero superscripts. Figure 4.1 illustrates a second interpretation of Equation (4.75). As can be seen in the figure,  $o_n - o_{i-1} = r$  and  $z_{i-1} = \omega$  in the familiar expression  $v = \omega \times r$ .

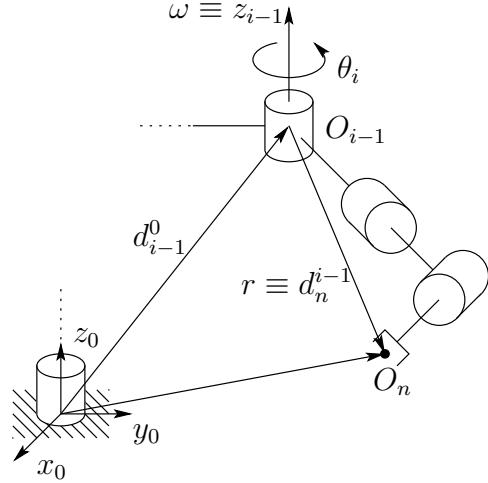


Fig. 4.1 Motion of the end-effector due to link  $i$ .

#### 4.6.3 Combining the Angular and Linear Jacobians

As we have seen in the preceding section, the upper half of the Jacobian  $J_v$  is given as

$$J_v = [J_{v_1} \cdots J_{v_n}] \quad (4.76)$$

where the  $i$ -th column  $J_{v_i}$  is

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases} \quad (4.77)$$

The lower half of the Jacobian is given as

$$J_\omega = [J_{\omega_1} \cdots J_{\omega_n}] \quad (4.78)$$

where the  $i$ -th column  $J_{\omega_i}$  is

$$J_{\omega_i} = \begin{cases} z_{i-1} & \text{for revolute joint } i \\ 0 & \text{for prismatic joint } i \end{cases} \quad (4.79)$$

Putting the upper and lower halves of the Jacobian together, we see the Jacobian for an  $n$ -link manipulator is of the form

$$J = [J_1 J_2 \cdots J_n] \quad (4.80)$$

where the  $i$ -th column  $J_i$  is given by

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} \quad (4.81)$$

if joint  $i$  is revolute and

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \quad (4.82)$$

if joint  $i$  is prismatic.

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  $o_1, \dots, o_n$ . 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.

## 4.7 EXAMPLES

We now provide a few examples to illustrate the derivation of the manipulator Jacobian.

### Example 4.5 Two-Link Planar Manipulator

Consider the two-link planar manipulator of Example 3.1. Since both joints are revolute the Jacobian matrix, which in this case is  $6 \times 2$ , is of the form

$$J(q) = \begin{bmatrix} z_0 \times (o_2 - o_0) & z_1 \times (o_2 - o_1) \\ z_0 & z_1 \end{bmatrix} \quad (4.83)$$

The various quantities above are easily seen to be

$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix} \quad o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix} \quad (4.84)$$

$$z_0 = z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.85)$$

Performing the required calculations then yields

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \quad (4.86)$$

It is easy to see how the above Jacobian compares with Equation (1.1) derived in Chapter 1. The first two rows of Equation (4.85) are exactly the  $2 \times 2$  Jacobian of Chapter 1 and give the linear velocity of the origin  $o_2$  relative to the base. The third row in Equation (4.86) 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

Consider the three-link planar manipulator of Figure 4.2. Suppose we wish

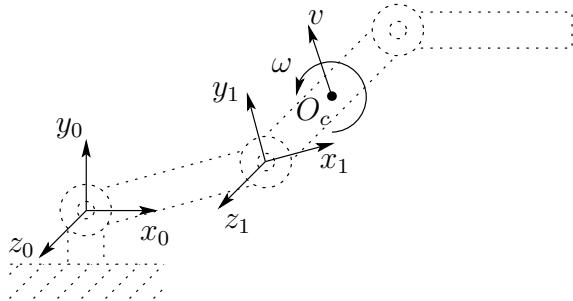


Fig. 4.2 Finding the velocity of link 2 of a 3-link planar robot.

to compute the linear velocity  $v$  and the angular velocity  $\omega$  of the center of link 2 as shown. In this case we have that

$$J(q) = \begin{bmatrix} z_0 \times (o_c - o_0) & z_1 \times (o_c - o_1) & 0 \\ z_0 & z_1 & 0 \end{bmatrix} \quad (4.87)$$

which is merely the usual the 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 is unaffected by motion of the third link<sup>2</sup>. Note that in this case the vector  $o_c$  must be computed as it is not given directly by the  $T$  matrices (Problem 4-13).

<sup>2</sup>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.

◊

**Example 4.7 Stanford Manipulator**

Consider the Stanford manipulator of Example 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

$$\begin{aligned} J_i &= \begin{bmatrix} z_{i-1} \times (o_6 - o_{i-1}) \\ z_{i-1} \end{bmatrix} \quad i = 1, 2 \\ J_3 &= \begin{bmatrix} z_2 \\ 0 \end{bmatrix} \\ J_i &= \begin{bmatrix} z_{i-1} \times (o_6 - o) \\ z_{i-1} \end{bmatrix} \quad i = 4, 5, 6 \end{aligned}$$

Now, using the  $A$ -matrices given by Equations (3.18)-(3.23) 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  $o_0 = (0, 0, 0)^T = o_1$ . The vector  $z_j$  is given as

$$z_j = R_j^0 k \quad (4.88)$$

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 manipulator:

$$o_6 = \begin{bmatrix} c_1 s_2 d_3 - s_1 d_2 + d_6(c_1 c_2 c_4 s_5 + c_1 c_5 s_2 - s_1 s_4 s_5) \\ s_1 s_2 d_3 - c_1 d_2 + d_6(c_1 s_4 s_5 + c_2 c_4 s_1 s_5 + c_5 s_1 s_2) \\ c_2 d_3 + d_6(c_2 c_5 - c_4 s_2 s_5) \end{bmatrix} \quad (4.89)$$

$$o_3 = \begin{bmatrix} c_1 s_2 d_3 - s_1 d_2 \\ s_1 s_2 d_3 + c_1 d_2 \\ c_2 d_3 \end{bmatrix} \quad (4.90)$$

The  $z_i$  are given as

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad z_1 = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \quad (4.91)$$

$$z_2 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \quad z_3 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \quad (4.92)$$

$$z_4 = \begin{bmatrix} -c_1 c_2 s_4 - s_1 c_4 \\ -s_1 c_2 s_4 + c_1 c_4 \\ s_2 s_4 \end{bmatrix} \quad (4.93)$$

$$z_5 = \begin{bmatrix} c_1 c_2 c_4 s_5 - s_1 s_4 s_5 + c_1 s_2 c_5 \\ s_1 c_2 c_4 s_5 + c_1 s_4 s_5 + s_1 s_2 c_5 \\ -s_2 c_4 s_5 + c_2 c_5 \end{bmatrix}. \quad (4.94)$$

The Jacobian of the Stanford Manipulator is now given by combining these expressions according to the given formulae (Problem 4-19).

◇

#### Example 4.8 SCARA Manipulator

We will now derive the Jacobian of the SCARA manipulator of Example 3.6. This Jacobian is a  $6 \times 4$  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 (3.26)-(3.29).

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

$$J = \begin{bmatrix} z_0 \times (o_4 - o_0) & z_1 \times (o_4 - o_1) & z_2 & 0 \\ z_0 & z_1 & 0 & z_3 \end{bmatrix} \quad (4.95)$$

Performing the indicated calculations, one obtains

$$o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix} \quad o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix} \quad (4.96)$$

$$o_4 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_2 + a_2 s_{12} \\ d_3 - d_4 \end{bmatrix} \quad (4.97)$$

Similarly  $z_0 = z_1 = k$ , and  $z_2 = z_3 = -k$ . Therefore the Jacobian of the SCARA Manipulator is

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \quad (4.98)$$

◊

## 4.8 THE ANALYTICAL JACOBIAN

The Jacobian matrix derived above is sometimes called the *Geometric Jacobian* to distinguish it from the *Analytical Jacobian*, denoted  $J_a(q)$ , considered in this section, which is based on a minimal representation for the orientation of the end-effector frame. Let

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix} \quad (4.99)$$

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  $\alpha$  denotes a minimal representation for the orientation of the end-effector frame relative to the base frame. For example, let  $\alpha = [\phi, \theta, \psi]^T$  be a vector of Euler angles as defined in Chapter 2. Then we look for an expression of the form

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q} \quad (4.100)$$

to define the analytical Jacobian.

It can be shown (Problem 4-9) that, if  $R = R_{z,\psi}R_{y,\theta}R_{z,\phi}$  is the Euler angle transformation then

$$\dot{R} = S(\omega)R \quad (4.101)$$

in which  $\omega$ , defining the angular velocity is given by

$$\omega = \begin{bmatrix} c_\psi s_\theta \dot{\phi} - s_\psi \dot{\theta} \\ s_\psi s_\theta \dot{\psi} + c_\psi \dot{\theta} \\ \dot{\psi} + c_\theta \dot{\psi} \end{bmatrix} \quad (4.102)$$

$$= \begin{bmatrix} c_\psi s_\theta & -s_\psi & 0 \\ s_\psi s_\theta & c_\psi & 0 \\ c_\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = B(\alpha)\dot{\alpha} \quad (4.103)$$

The components of  $\omega$  are called the *nutation*, *spin*, and *precession*, respectively. Combining the above relationship with the previous definition of the Jacobian, i.e.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ \omega \end{bmatrix} = J(q)\dot{q} \quad (4.104)$$

yields

$$J(q)\dot{q} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ B(\alpha)\dot{\alpha} \end{bmatrix} \quad (4.105)$$

$$= \begin{bmatrix} I & 0 \\ 0 & B(\alpha) \end{bmatrix} \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} \quad (4.106)$$

$$= \begin{bmatrix} I & 0 \\ 0 & B(\alpha) \end{bmatrix} J_a(q)\dot{q} \quad (4.107)$$

Thus the analytical Jacobian,  $J_a(q)$ , may be computed from the geometric Jacobian as

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B(\alpha)^{-1} \end{bmatrix} J(q) \quad (4.108)$$

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 4-20) 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

$$\xi = J(q)\dot{q} \quad (4.109)$$

between the vector  $\dot{q}$  of joint velocities and the vector  $\xi = (v, \omega)^T$  of end-effector velocities. This implies that the all possible end-effector velocities are linear combinations of the columns of the Jacobian matrix,

$$\xi = J_1\dot{q}_1 + J_2\dot{q}_2 + \dots + J_n\dot{q}_n$$

For example, for the two-link planar arm, the Jacobian matrix given in Equation (4.86) 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  $\text{rank } J = 6$ , the end-effector can execute any arbitrary velocity. For a matrix  $J \in \mathbb{R}^{6 \times n}$ , it is always the case that  $\text{rank } J \leq \min(6, n)$ . For example, for the two-link planar arm, we always have  $\text{rank } J \leq 2$ , while for an anthropomorphic arm with spherical wrist we always have  $\text{rank } J \leq 6$ .

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 the rank  $J(q)$  is less than its maximum value are called **singularities** or **singular configurations**. Identifying manipulator singularities is important for several reasons.

1. Singularities represent configurations from which certain directions of motion may be unattainable.
2. At singularities, bounded end-effector velocities may correspond to unbounded joint velocities.

3. At singularities, bounded end-effector forces and torques may correspond to unbounded joint torques. (We will see this in Chapter 9).
4. Singularities usually (but not always) correspond to points on the boundary of the manipulator workspace, that is, to points of maximum reach of the manipulator.
5. Singularities correspond to points in the manipulator workspace that may be unreachable under small perturbations of the link parameters, such as length, offset, etc.
6. Near singularities there will not exist a unique solution to the inverse kinematics problem. In such cases there may be no solution or there may be infinitely many solutions.

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.

For the sake of argument, suppose 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  $6 \times 6$  matrix and a configuration  $q$  is singular if and only if

$$\det J(q) = 0 \quad (4.110)$$

If we now partition the Jacobian  $J$  into  $3 \times 3$  blocks as

$$J = [J_P \mid J_O] = \left[ \begin{array}{c|c} J_{11} & J_{12} \\ \hline J_{21} & J_{22} \end{array} \right] \quad (4.111)$$

then, since the final three joints are always revolute

$$J_O = \begin{bmatrix} z_3 \times (o_6 - o_3) & z_4 \times (o_6 - o_4) & z_5 \times (o_6 - o_5) \\ z_3 & z_4 & z_5 \end{bmatrix} \quad (4.112)$$

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

$$J_O = \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix} \quad (4.113)$$

In this case the Jacobian matrix has the block triangular form

$$J = \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix} \quad (4.114)$$

with determinant

$$\det J = \det J_{11} \det J_{22} \quad (4.115)$$

where  $J_{11}$  and  $J_{22}$  are each  $3 \times 3$  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

$$J_{22} = [z_3 \ z_4 \ z_5] \quad (4.116)$$

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 (4.116) 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 4.3 we see that this happens when the joint axes  $z_3$  and  $z_5$  are collinear. 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. This is the only singularity of the spherical wrist, and is 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.

#### 4.9.3 Arm Singularities

To investigate arm singularities we need only to compute  $J_{11}$ , which is done using Equation (4.77) but with the wrist center  $o$  in place of  $o_n$ .

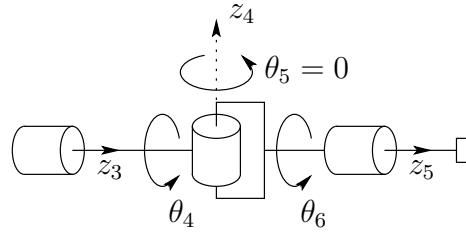


Fig. 4.3 Spherical wrist singularity.

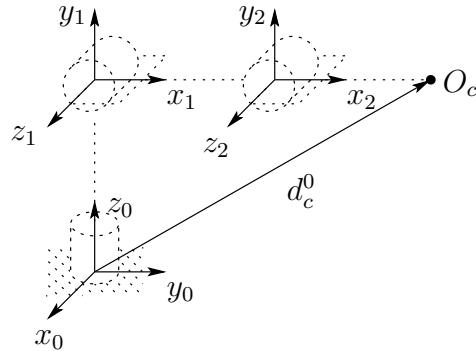


Fig. 4.4 Elbow manipulator.

**Example 4.9 Elbow Manipulator Singularities**

Consider the three-link articulated manipulator with coordinate frames attached as shown in Figure 4.4. It is left as an exercise (Problem 4-14) to show that

$$J_{11} = \begin{bmatrix} -a_2 s_1 c_2 - a_3 s_1 c_{23} & -a_2 s_2 c_1 - a_3 s_{23} c_1 & -a_3 c_1 s_{23} \\ a_2 c_1 c_2 + a_3 c_1 c_{23} & -a_2 s_1 s_2 - a_3 s_1 s_{23} & -a_3 s_1 s_{23} \\ 0 & a_2 c_2 + a_3 c_{23} & a_3 c_{23} \end{bmatrix} \quad (4.117)$$

and that the determinant of  $J_{11}$  is

$$\det J_{11} = a_2 a_3 s_3 (a_2 c_2 + a_3 c_{23}). \quad (4.118)$$

We see from Equation (4.118) that the elbow manipulator is in a singular configuration whenever

$$s_3 = 0, \quad \text{that is, } \theta_3 = 0 \text{ or } \pi \quad (4.119)$$

and whenever

$$a_2 c_2 + a_3 c_{23} = 0 \quad (4.120)$$

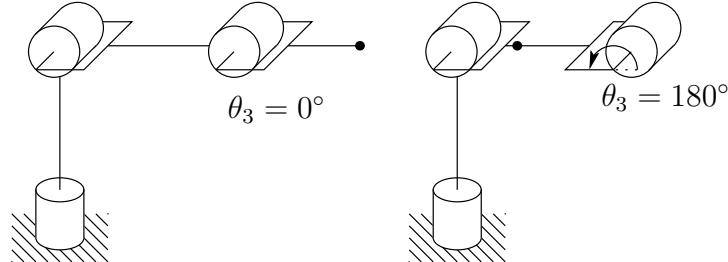


Fig. 4.5 Elbow singularities of the elbow manipulator.

The situation of Equation (4.119) is shown in Figure 4.5 and arises when the elbow is fully extended or fully retracted as shown. The second situation of Equation (4.120) is shown in Figure 4.6. This configuration occurs when

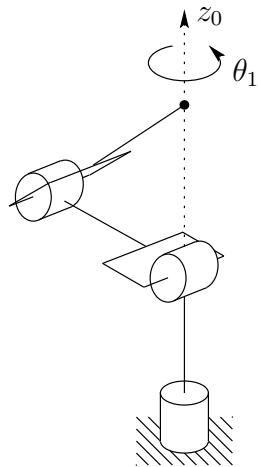


Fig. 4.6 Singularity of the elbow manipulator with no offsets.

the wrist center intersects the axis of the base rotation,  $z_0$ . As we saw in Chapter 3, 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 4.7, 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.

◇

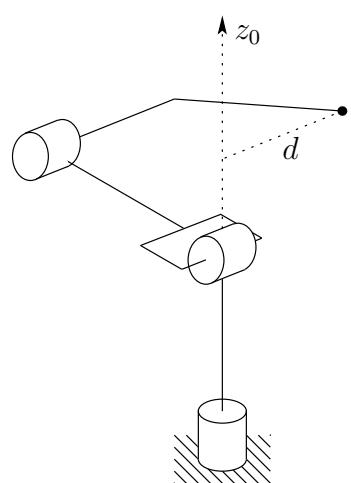


Fig. 4.7 Elbow manipulator with shoulder offset.

**Example 4.10 Spherical Manipulator**

Consider the spherical arm of Figure 4.8. This manipulator is in a singular

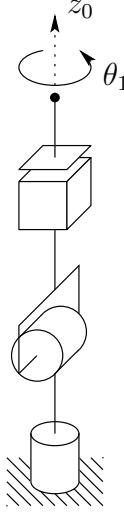


Fig. 4.8 Singularity of spherical manipulator with no offsets.

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 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 4.9 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

$$J_{11} = \begin{bmatrix} \alpha_1 & \alpha_3 & 0 \\ \alpha_2 & \alpha_4 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (4.121)$$

where

$$\alpha_1 = -a_1 s_1 - a_2 s_{12} \quad (4.122)$$

$$\alpha_2 = a_1 c_1 + a_2 c_{12}$$

$$\alpha_3 = -a_1 s_{12}$$

$$\alpha_4 = a_1 c_{12} \quad (4.123)$$

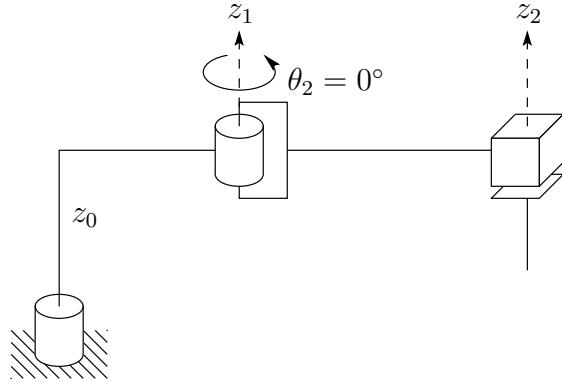


Fig. 4.9 SCARA manipulator singularity.

we see that the rank of  $J_{11}$  will be less than three precisely whenever  $\alpha_1\alpha_4 - \alpha_2\alpha_3 = 0$ . It is easy to compute this quantity and show that it is equivalent to (Problem 4-16)

$$s_2 = 0, \quad \text{which implies} \quad \theta_2 = 0, \pi. \quad (4.124)$$

◊

## 4.10 INVERSE VELOCITY AND ACCELERATION

The Jacobian relationship

$$\xi = J\dot{q} \quad (4.125)$$

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 inverse position. When the Jacobian is square (i.e.,  $J \in \mathbb{R}^{n \times n}$ ) and nonsingular, this problem can be solved by simply inverting the Jacobian matrix to give

$$\dot{q} = J^{-1}\xi \quad (4.126)$$

For manipulators that do not have exactly six links, the Jacobian can not be inverted. In this case there will be a solution to Equation (4.125) if and only if  $\xi$  lies in the range space of the Jacobian. This can be determined by the following simple rank test. A vector  $\xi$  belongs to the range of  $J$  if and only if

$$\text{rank } J(q) = \text{rank } [J(q) \mid \xi] \quad (4.127)$$

In other words, Equation (4.125) 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  $J$ . To construct this psuedoinverse, we use the following result from linear algebra.

**Proposition:** For  $J \in \mathbb{R}^{m \times n}$ , if  $m < n$  and rank  $J = m$ , then  $(JJ^T)^{-1}$  exists.

In this case  $(JJ^T) \in \mathbb{R}^{m \times m}$ , and has rank  $m$ . Using this result, we can regroup terms to obtain

$$\begin{aligned}(JJ^T)(JJ^T)^{-1} &= I \\ J[J^T(JJ^T)^{-1}] &= I \\ JJ^+ &= I\end{aligned}$$

Here,  $J^+ = J^T(JJ^T)^{-1}$  is called a right pseudoinverse of  $J$ , since  $JJ^+ = I$ . Note that,  $J^+J \in \mathbb{R}^{n \times n}$ , and that in general,  $J^+J \neq I$  (recall that matrix multiplication is not commutative).

It is now easy to demonstrate that a solution to Equation (4.125) is given by

$$\dot{q} = J^+\xi + (I - J^+J)b \quad (4.128)$$

in which  $b \in \mathbb{R}^n$  is an arbitrary vector. To see this, siimly multiply both sides of Equation (4.128) by  $J$ :

$$\begin{aligned}J\dot{q} &= J[J^+\xi + (I - J^+J)b] \\ &= JJ^+\xi + J(I - J^+J)b \\ &= JJ^+\xi + (J - JJ^+J)b \\ &= \xi + (J - J)b \\ &= \xi\end{aligned}$$

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  $J$ , i.e., 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 (4.125), 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  $b = 0$ . To see this, apply the triangle inequality to obtain

$$\begin{aligned}\|\dot{q}\| &= \|J^+\xi + (I - J^+J)b\| \\ &\leq \|J^+\xi\| + \|(I - J^+J)b\|\end{aligned}$$

It is a simple matter construct the right pseudoinverse of  $J$  using its singular value decomposition (see Appendix B),

$$J^+ = V\Sigma^+U^T$$

in which

$$\Sigma^+ = \left[ \begin{array}{c|c} \sigma_1^{-1} & \\ \sigma_2^{-1} & \\ \vdots & \vdots \\ \sigma_m^{-1} & \end{array} \right]^T$$

We can apply a similar approach when the analytical Jacobian is used in place of the manipulator Jacobian. Recall from Equation (4.100) that the joint velocities and the end-effector velocities are related by the analytical Jacobian as

$$\dot{X} = J_a(q)\dot{q} \quad (4.129)$$

Thus the inverse velocity problem becomes one of solving the system of linear equations (4.129), which can be accomplished as above for the manipulator Jacobian.

Differentiating Equation (4.129) yields the acceleration equations

$$\ddot{X} = J_a(q)\ddot{q} + \left( \frac{d}{dt} J_a(q) \right) \dot{q} \quad (4.130)$$

Thus, given a vector  $\ddot{X}$  of end-effector accelerations, the instantaneous joint acceleration vector  $\ddot{q}$  is given as a solution of

$$b = J_a(q)\ddot{q} \quad (4.131)$$

where

$$b = \ddot{X} - \frac{d}{dt} J_a(q)\dot{q} \quad (4.132)$$

For 6-DOF manipulators the inverse velocity and acceleration equations can therefore be written as

$$\dot{q} = J_a(q)^{-1} \dot{X} \quad (4.133)$$

and

$$\ddot{q} = J_a(q)^{-1} b \quad (4.134)$$

provided  $\det J_a(q) \neq 0$ .

## 4.11 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,  $\xi$ . 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

$$\|\dot{q}\| = (\dot{q}_1^2 + \dot{q}_2^2 + \dots + \dot{q}_m^2)^{1/2} \leq 1 \quad (4.135)$$

If we use the minimum norm solution  $\dot{q} = J^+ \xi$ , we obtain

$$\begin{aligned} \|\dot{q}\| &= \dot{q}^T \dot{q} \\ &= (J^+ \xi)^T J^+ \xi \\ &= \xi^T (J J^T)^{-1} \xi \leq 1 \end{aligned} \quad (4.136)$$

The derivation of this is left as an exercise (Problem 4-26). This final inequality gives us a quantitative characterization of the scaling that is effected by the Jacobian. In particular, if the manipulator Jacobian is full rank, i.e., rank  $J = m$ , then Equation (4.136) defines an  $m$ -dimensional ellipsoid that is known as the *manipulability ellipsoid*. If the input (i.e., joint velocity) vector has unit norm, then the output (i.e., end-effector velocity) will lie within the ellipsoid given by Equation (4.136). We can more easily see that Equation (4.136) defines an ellipsoid by replacing the Jacobian by its SVD  $J = U \Sigma V^T$  (see Appendix B) to obtain

$$\xi^T (J J^T)^{-1} \xi^T = (U^T \xi)^T \Sigma_m^{-2} (U^T \xi) \quad (4.137)$$

in which

$$\Sigma_m^{-2} = \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_m^{-2} \end{bmatrix}$$

The derivation of Equation (4.137) is left as an exercise (Problem 4-27). If we make the substitution  $w = U^T \xi$ , then Equation (4.137) can be written as

$$w^T \Sigma_m^{-2} w = \sum \sigma_i^{-2} w_i^2 \leq 1 \quad (4.138)$$

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  $U$ . 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, as defined by Yoshikawa [78], is given by

$$\mu = \sigma_1 \sigma_2 \cdots \sigma_m \quad (4.139)$$

Note that the constant  $K$  is not included in the definition of manipulability, since it is fixed once the task has been defined (i.e., once the dimension of the task space has been fixed).

Now, consider the special case that the robot is not redundant, i.e.,  $J \in \mathbb{R}^{m \times m}$ . 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

$$\begin{aligned}\det JJ^T &= \det J \det J^T \\ &= \det J \det J \\ &= (\lambda_1 \lambda_2 \cdots \lambda_m)(\lambda_1 \lambda_2 \cdots \lambda_m) \\ &= \lambda_1^2 \lambda_2^2 \cdots \lambda_m^2\end{aligned}\tag{4.140}$$

in which  $\lambda_1 \geq \lambda_2 \cdots \leq \lambda_m$  are the eigenvalues of  $J$ . This leads to

$$\mu = \sqrt{\det JJ^T} = |\lambda_1 \lambda_2 \cdots \lambda_m| = |\det J|\tag{4.141}$$

The manipulability,  $\mu$ , has the following properties.

- In general,  $\mu = 0$  holds if and only if  $\text{rank}(J) < m$ , (i.e., when  $J$  is not full rank).
- 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

$$(\sigma_1)^{-1} \leq \frac{||\Delta\dot{q}||}{||\Delta\xi||} \leq (\sigma_m)^{-1}\tag{4.142}$$

#### Example 4.12 Two-link Planar Arm.

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 dexterity. We can use manipulability as a measure of dexterity. Consider the two-link planar arm and the task of positioning in the plane. For the two link arm, the Jacobian is given by

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}\tag{4.143}$$

and the manipulability is given by

$$\mu = |\det J| = a_1 a_2 |s_2|$$

Thus, 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, the 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.

◊

## 4.12 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

$$\dot{R}(t) = S(\omega(t))R(t) \quad (4.144)$$

and the vector  $\omega(t)$  is the instantaneous angular velocity of the frame. The operator  $S$  gives a skew symmetrix matrix

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (4.145)$$

The manipulator Jacobian relates the vector of joint velocities to the body velocity  $\xi = (v, \omega)^T$  of the end effector,

$$\xi = J\dot{q} \quad (4.146)$$

This relationship can be written as two equations, one for linear velocity and one for angular velocity,

$$v = J_v \dot{q} \quad (4.147)$$

$$\omega = J_\omega \dot{q} \quad (4.148)$$

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

$$J_i = \begin{cases} \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{if joint } i \text{ is revolute} \\ \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{if joint } i \text{ is prismatic} \end{cases} \quad (4.149)$$

For a given parameterization of orientation, e.g. Euler angles, the analytical Jacobian relates joint velocities to the time derivative of the pose parameters

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix} \quad \dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}$$

in which  $d(q)$  is the usual vector from the origin of the base frame to the origin of the end-effector frame and  $\alpha$  denotes a parameterization of 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

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B(\alpha)^{-1} \end{bmatrix} J(q) \quad (4.150)$$

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 (i.e., a configuration  $q$  such that  $\text{rank } J \leq \max_q \text{rank } J(q)$ ) 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  $3 \times 3$  block of the manipulator Jacobian.

For nonsingular configurations, the Jacobian relationship can be used to find the joint velocities  $\dot{q}$  necessary to achieve a desired end-effector velocity  $\xi$ . 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  $\sigma_i$  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 Verify Equation (4.8) by direct calculation.
- 4-4 Verify Equation (4.16) by direct calculation.
- 4-5 Show that  $S(k)^3 = -S(k)$ . Use this and Problem 25 to verify Equation (4.17).
- 4-6 Given any square matrix  $A$ , the exponential of  $A$  is a matrix defined as

$$e^A = I + A + \frac{1}{2}A^2 + \frac{1}{3!}A^3 + \dots$$

Given  $S \in so(3)$  show that  $e^S \in SO(3)$ .

[Hint: Verify the facts that  $e^A e^B = e^{A+B}$  provided that  $A$  and  $B$  commute, that is,  $AB = BA$ , and also that  $\det(e^A) = e^{Tr(A)}$ .]

- 4-7 Show that  $R_{k,\theta} = e^{S(k)\theta}$ .

[Hint: Use the series expansion for the matrix exponential together with Problems 25 and 5. Alternatively use the fact that  $R_{k,\theta}$  satisfies the differential equation

$$\frac{dR}{d\theta} = S(k)R.$$

- 4-8 Use Problem 7 to show the converse of Problem 6, that is, if  $R \in SO(3)$  then there exists  $S \in so(3)$  such that  $R = e^S$ .

- 4-9 Given the Euler angle transformation

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

show that  $\frac{d}{dt}R = S(\omega)R$  where

$$\omega = \{c_\psi s_\theta \dot{\phi} - s_\psi \dot{\theta}\}i + \{s_\psi s_\theta \dot{\phi} + c_\psi \dot{\theta}\}j + \{[s_i + c_\theta \dot{\phi}]k\}.$$

The components of  $i, j, k$ , respectively, are called the **nutation**, **spin**, and **precession**.

- 4-10 Repeat Problem 9 for the Roll-Pitch-Yaw transformation. In other words, find an explicit expression for  $\omega$  such that  $\frac{d}{dt}R = S(\omega)R$ , where  $R$  is given by Equation (2.39).

- 4-11 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  $v_1(t) = (3, 1, 0)^T$  relative to frame  $o_1x_1y_1z_1$ . What is the velocity of the particle in frame  $o_0x_0y_0z_0$ ?

- 4-12 Three frames  $o_0x_0y_0z_0$  and  $o_1x_1y_1z_1$ , and  $o_2x_2y_2z_2$  are given below. If the angular velocities  $\omega_1^0$  and  $\omega_2^1$  are given as

$$\omega_1^0 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}; \quad \omega_2^1 = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

what is the angular velocity  $\omega_2^0$  at the instant when

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

- 4-13 ). For the three-link planar manipulator of Example 4.6, compute the vector  $O_c$  and derive the manipulator Jacobian matrix.

- 4-14 Compute the Jacobian  $J_{11}$  for the 3-link elbow manipulator of Example 4.9 and show that it agrees with Equation (4.117). Show that the determinant of this matrix agrees with Equation (4.118).

- 4-15 Compute the Jacobian  $J_{11}$  for the three-link spherical manipulator of Example 4.10.

- 4-16 Show from Equation (4.122) that the singularities of the SCARA manipulator are given by Equation (4.124).

- 4-17 Find the  $6 \times 3$  Jacobian for the three links of the cylindrical manipulator of Figure 3.7. Show that there are no singular configurations for this arm. Thus the only singularities for the cylindrical manipulator must come from the wrist.

- 4-18 Repeat Problem 17 for the cartesian manipulator of Figure 3.28.

- 4-19 Complete the derivation of the Jacobian for the Stanford manipulator from Example 4.7.

- 4-20 Show that  $B(\alpha)$  given by Equation (4.103) is invertible provided  $s_\theta \neq 0$ .

- 4-21 Verify Equation (4.7) by direct calculation.

- 4-22 Prove the assertion given in Equation (4.8) that  $R(a \times b) = Ra \times Rb$ , for  $R \in SO(3)$ .
- 4-23 Suppose that  $a = (1, -1, 2)^T$  and that  $R = R_{x,90}$ . Show by direct calculation that

$$RS(a)R^T = S(Ra).$$

- 4-24 Given  $R_1^0 = R_{x,\theta}R_{y,\phi}$ , compute  $\frac{\partial R_1^0}{\partial \phi}$ . Evaluate  $\frac{\partial R_1^0}{\partial \phi}$  at  $\theta = \frac{\pi}{2}$ ,  $\phi = \frac{\phi}{2}$ .

- 4-25 Use Equation (2.46) to show that

$$R_{k,\theta} = I + S(k) \sin(\theta) + S^2(k) \operatorname{vers}(\theta).$$

- 4-26 Verify Equation (4.136).

- 4-27 Verify Equation (4.137).

# 5

---

## *PATH AND TRAJECTORY PLANNING*

**I**n previous chapters, we have 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. In this chapter, we address the problem of planning collision free paths for the robot. 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 for the robot that connects them.

The description of this problem is deceptively simple, yet the path planning problem is among the most difficult problems in computer science. The computational complexity of the best known complete<sup>1</sup> path planning algorithm grows exponentially with the number of internal degrees of freedom of the robot. For this reason, for robot systems with more than a few degrees of freedom, complete algorithms are not used in practice.

In this chapter we treat the path planning problem as a search problem. The algorithms we describe are not guaranteed to find a solution to all problems, but they are quite effective in a wide range of practical applications. Furthermore, these algorithms are fairly easy to implement, and require only moderate computation time for most problems.

<sup>1</sup>An algorithm is said to be complete if it finds a solution whenever one exists.

Path planning provides a geometric description of robot motion, but it does not specify any dynamic aspects of the motion. For example, what should be the joint velocities and accelerations while traversing the path? These questions are addressed by a trajectory planner. The trajectory planner computes a function  $q^d(t)$  that completely specifies the motion of the robot as it traverses the path.

We begin in Section 5.1 by introducing the notion of *configuration space*, and describing how obstacles in the workspace can be mapped into the configuration space. We then introduce path planning methods that use artificial potential fields in Sections 5.2 and 5.3. The corresponding algorithms use gradient descent search to find a collision-free path to the goal, and, as with all gradient descent methods, these algorithms can become trapped in local minima in the potential field. Therefore, in Section 5.4 we describe how random motions can be used to escape local minima. In Section 5.5 we describe another randomized method known as the Probabilistic Roadmap (PRM) method. Finally, since each of these methods generates a sequence of configurations, we describe in Section 5.6 how polynomial splines can be used to generate smooth trajectories from a sequence of configurations.

## 5.1 THE CONFIGURATION SPACE

In Chapter 3, we learned 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  $A_i$  matrices can be used to infer the position and orientation of the reference frame for any link of the robot. Since each link of the robot is assumed to be a rigid body, the  $A_i$  matrices can therefore be used to infer the position of any point on the robot, given the values of the joint variables. 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*. For our purposes, the vector of joint variables,  $q$ , provides a convenient representation of a configuration. We will denote the configuration space by  $\mathcal{Q}$ .

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 could also say  $\mathcal{Q} = SO(2)$ . In fact, the choice of  $S^1$  or  $SO(2)$  is not particularly important, since these two are equivalent representations. In either case, we can parameterize  $\mathcal{Q}$  by a single parameter, the joint angle  $\theta_1$ . 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  $q = (d_1, d_2, d_3) = (x, y, z)$ .

Although we have chosen to represent a configuration by a vector of joint variables, the notion of a configuration is more general than this. For example, as we saw in Chapter 2, for any rigid two-dimensional 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 can represent a configuration by the triple  $q = (x, y, \theta)$ , and the configuration space can be represented by  $\mathcal{Q} = \mathbb{R}^2 \times SO(2)$ . Again, this is merely one possible representation of the configuration space, but it is a convenient one given the representations of position and orientation that we have learned in preceding chapters.

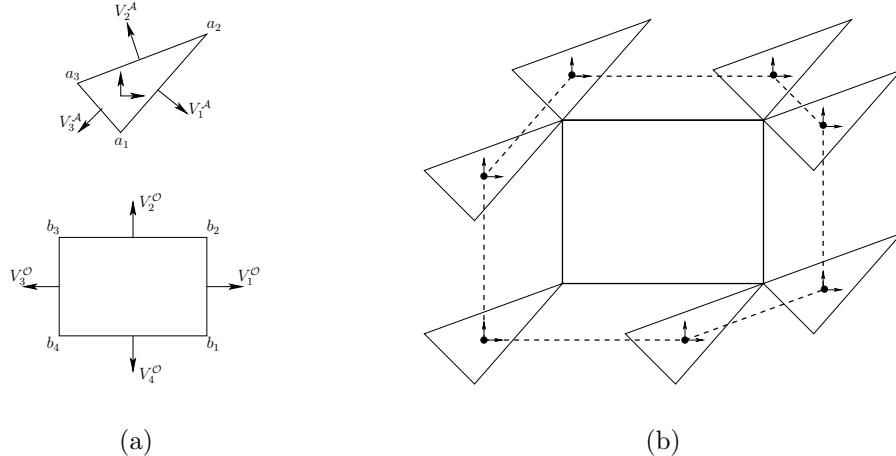
A collision occurs when the robot contacts an obstacle in the workspace. To describe collisions, we introduce some additional notation. We will denote the robot by  $\mathcal{A}$ , and by  $\mathcal{A}(q)$  the subset of the workspace that is occupied by the robot at configuration  $q$ . We denote by  $\mathcal{O}_i$  the obstacles in the workspace, and by  $\mathcal{W}$  the workspace (i.e., the Cartesian space in which the robot moves). 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 in the workspace. 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

$$\mathcal{Q}\mathcal{O} = \{q \in \mathcal{Q} \mid \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$$

Here, we define  $\mathcal{O} = \cup \mathcal{O}_i$ . The set of collision-free configurations, referred to as the free configuration space, is then simply

$$\mathcal{Q}_{\text{free}} = \mathcal{Q} \setminus \mathcal{Q}\mathcal{O}$$

**Example 5.1** *A Rigid Body that Translates in the Plane.*



*Fig. 5.1* (a) a rigid body,  $\mathcal{A}$ , in a workspace containing a single rectangular obstacle,  $\mathcal{O}$ , (b) illustration of the algorithm to construct  $\mathcal{Q}\mathcal{O}$ , with the boundary of  $\mathcal{Q}\mathcal{O}$  shown as the dashed line

Consider a simple gantry robot with two prismatic joints and forward kinematics given by  $x = d_1, y = d_2$ . 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 end-effector and the obstacle are convex polygons, it is a simple matter to compute the configuration space obstacle region,  $\mathcal{QO}$  (we assume here that the arm itself is positioned above the workspace, so that the only possible collisions are between the end-effector and obstacles the obstacle).

Let  $V_i^A$  denote the vector that is normal to the  $i^{th}$  edge of the robot and  $V_j^O$  denote the vector that is normal to the  $j^{th}$  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 robot and  $b_j$  to be the vector from the origin of the world coordinate frame to the  $j^{th}$  vertex of the obstacle. An example is shown in Figure 5.1(a). The vertices of  $\mathcal{QO}$  can be determined as follows.

- For each pair  $V_j^O$  and  $V_{j-1}^O$ , if  $V_i^A$  points between  $-V_j^O$  and  $-V_{j-1}^O$  then add to  $\mathcal{QO}$  the vertices  $b_j - a_i$  and  $b_j - a_{i+1}$ .
- For each pair  $V_i^A$  and  $V_{i-1}^A$ , if  $V_j^O$  points between  $-V_i^A$  and  $-V_{i-1}^A$  then add to  $\mathcal{QO}$  the vertices  $b_j - a_i$  and  $b_{j+1} - a_i$ .

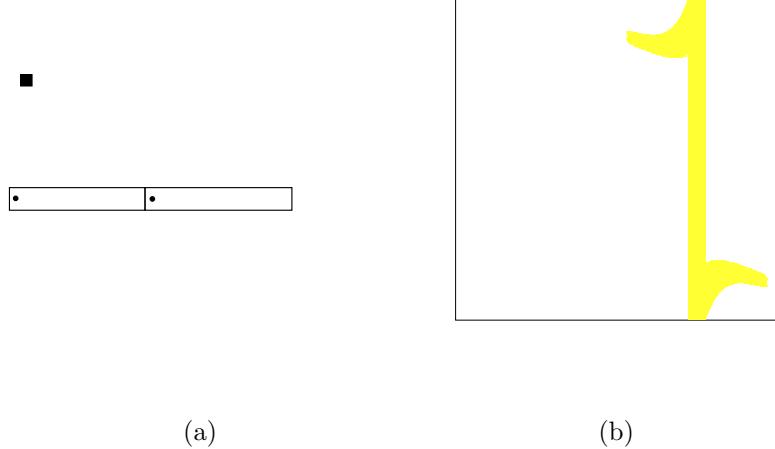
This is illustrated in Figure 5.1(b). Note that this algorithm essentially places the robot at all positions where vertex-vertex contact between robot and obstacle are possible. The origin of the robot's local coordinate frame at each such configuration defines a vertex of  $\mathcal{QO}$ . The polygon 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 C-space obstacle region,  $\mathcal{QO}_i$  for each piece, and finally, computing the union of the  $\mathcal{QO}_i$ .

◇

### Example 5.2 A Two Link Planar Arm.

For robots with revolute joints, computation of  $\mathcal{QO}$  is more difficult. Consider a two-link planar arm in a workspace containing a single obstacle as shown in Figure 5.2(a). The configuration space obstacle region is illustrated in 5.2(b). The horizontal axis in 5.2(b) corresponds to  $\theta_1$ , and the vertical axis to  $\theta_2$ . For values of  $\theta_1$  very near  $\pi/2$ , the first link of the arm collides with the obstacle. Further, when the first link is near the obstacle ( $\theta_1$  near  $\pi/2$ ), for some values of  $\theta_2$  the second link of the arm collides with the obstacle. The region  $\mathcal{QO}$  shown in 5.2(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. Thus, this is only an approximate representation of  $\mathcal{QO}$ .



*Fig. 5.2* (a) A two-link planar arm and a single polygonal obstacle. (b) The corresponding configuration space obstacle region

◊

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 (e.g., 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  $\mathcal{Q}_{\text{free}}$ .

The path planning problem is to find a path from an initial configuration  $q_{\text{init}}$  to a final configuration  $q_{\text{final}}$ , such that the robot does not collide with any obstacle as it traverses the path. More formally, A collision-free path from  $q_{\text{init}}$  to  $q_{\text{final}}$  is a continuous map,  $\tau : [0, 1] \rightarrow \mathcal{Q}_{\text{free}}$ , with  $\tau(0) = q_{\text{init}}$  and  $\tau(1) = q_{\text{final}}$ . We will develop path planning methods that compute a sequence of discrete configurations (set points) in the configuration space. In Section 5.6 we will show how smooth trajectories can be generated from such a sequence.

## 5.2 PATH PLANNING USING CONFIGURATION SPACE POTENTIAL FIELDS

As mentioned above, it is not feasible to build an explicit representation 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. Such a search algorithm requires a strategy for exploring  $\mathcal{Q}_{\text{free}}$ , and one of the most popular is to use an *artificial potential field* to guide the search. In this section, we will introduce artificial potential field methods. Here we describe how the potential field can be constructed directly on the configuration space of the robot. However, as will become clear, computing the gradient of such a field is not feasible in general, so in Section 5.3 we will develop an alternative, in which the potential field is first constructed on the workspace, and then its effects are mapped to the configuration space.

The basic idea behind the potential field approaches is as follows. The robot is treated 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,  $q_{\text{final}}$ , while being repelled from the boundaries of  $\mathcal{Q}\mathcal{O}$ . If  $U$  is constructed appropriately, there will be a single global minimum of  $U$  at  $q_{\text{final}}$ , and no local minima. Unfortunately, as we will discuss below, it is often difficult to construct such a field.

In general, the field  $U$  is an additive field consisting of one component that attracts the robot to  $q_{\text{final}}$  and a second component that repels the robot from the boundary of  $\mathcal{Q}\mathcal{O}$ ,

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \quad (5.1)$$

Given this formulation, path planning can be treated as an optimization problem, i.e., find the global minimum in  $U$ , starting from initial configuration  $q_{\text{init}}$ . One of the easiest algorithms to solve this problem is gradient descent. In this case, the negative gradient of  $U$  can be considered as a force acting on the robot (in configuration space),

$$F(q) = -\nabla U(q) = -\nabla U_{\text{att}}(q) - \nabla U_{\text{rep}}(q) \quad (5.2)$$

In the remainder of this section, we will describe typical choices for the attractive and repulsive potential fields, and a gradient descent algorithm that can be used to plan paths in this field.

### 5.2.1 The Attractive Field

There are several criteria that the potential field  $U_{\text{att}}$  should satisfy. First,  $U_{\text{att}}$  should be monotonically increasing with distance from  $q_{\text{final}}$ . The simplest choice for such a field is a field that grows linearly with the distance from  $q_{\text{final}}$ , a so-called conic well potential. However, the gradient of such a field has unit magnitude everywhere but the origin, where it is zero. This can lead to stability problems, since there is a discontinuity in the attractive force at the origin. We

prefer a field that is continuously differentiable, such that the attractive force decreases as the robot approaches  $q_{\text{final}}$ . The simplest such field is a field that grows quadratically with the distance to  $q_{\text{final}}$ . Let  $\rho_f(q)$  be the Euclidean distance between  $q$  and  $q_{\text{final}}$ , i.e.,  $\rho_f(q) = \|q - q_{\text{final}}\|$ . Then we can define the quadratic field by

$$U_{\text{att}}(q) = \frac{1}{2}\zeta\rho_f^2(q) \quad (5.3)$$

in which  $\zeta$  is a parameter used to scale the effects of the attractive potential. This field is sometimes referred to as a parabolic well. For  $q = (q^1, \dots, q^n)^T$ , the gradient of  $U_{\text{att}}$  is given by

$$\begin{aligned} \nabla U_{\text{att}}(q) &= \nabla \frac{1}{2}\zeta\rho_f^2(q) \\ &= \nabla \frac{1}{2}\zeta\|q - q_{\text{final}}\|^2 \\ &= \frac{1}{2}\zeta\nabla \sum_i (q^i - q_{\text{final}}^i)^2 \\ &= \zeta(q^1 - q_{\text{final}}^1, \dots, q^n - q_{\text{final}}^n)^T \\ &= \zeta(q - q_{\text{final}}) \end{aligned} \quad (5.4)$$

Here, (5.4) follows since

$$\frac{\partial}{\partial q^j} \sum_i (q^i - q_{\text{final}}^i)^2 = 2(q^j - q_{\text{final}}^j)$$

So, for the parabolic well, the attractive force,  $F_{\text{att}}(q) = -\nabla U_{\text{att}}(q)$  is a vector directed toward  $q_{\text{final}}$  with magnitude linearly related to the distance from  $q$  to  $q_{\text{final}}$ .

Note that while  $F_{\text{att}}(q)$  converges linearly to zero as  $q$  approaches  $q_{\text{final}}$  (which is a desirable property), it grows without bound as  $q$  moves away from  $q_{\text{final}}$ . If  $q_{\text{init}}$  is very far from  $q_{\text{final}}$ , this may produce an attractive force that is too large. For this reason, we may choose to combine the quadratic and conic potentials so that the conic potential attracts the robot when it is very distant from  $q_{\text{final}}$  and the quadratic potential attracts the robot when it is near  $q_{\text{final}}$ . Of course it is necessary that the gradient be defined at the boundary between the conic and quadratic portions. Such a field can be defined by

$$U_{\text{att}}(q) = \begin{cases} \frac{1}{2}\zeta\rho_f^2(q) & : \rho_f(q) \leq d \\ d\zeta\rho_f(q) - \frac{1}{2}\zeta d^2 & : \rho_f(q) > d \end{cases} \quad (5.5)$$

and in this case we have

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(q) = \begin{cases} -\zeta(q - q_{\text{final}}) & : \rho_f(q) \leq d \\ -\frac{d\zeta(q - q_{\text{final}})}{\rho_f(q)} & : \rho_f(q) > d \end{cases} \quad (5.6)$$

The gradient is well defined at the boundary of the two fields since at the boundary  $d = \rho_f(q)$ , and the gradient of the quadratic potential is equal to the gradient of the conic potential,  $F_{\text{att}}(q) = -\zeta(q - q_{\text{final}})$ .

### 5.2.2 The Repulsive field

There are several criteria that the 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 that goes to infinity at obstacle boundaries, and drops to zero at a certain distance from the obstacle. If we define  $\rho_0$  to be the distance of influence of an obstacle (i.e., an obstacle will not repel the robot if the distance from the robot to the obstacle is greater than  $\rho_0$ ), one potential that meets these criteria is given by

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & : \rho(q) \leq \rho_0 \\ 0 & : \rho(q) > \rho_0 \end{cases} \quad (5.7)$$

in which  $\rho(q)$  is the shortest distance from  $q$  to a configuration space obstacle boundary, and  $\eta$  is a scalar gain coefficient that determines the influence of the repulsive field. If  $\mathcal{QO}$  consists of a single convex region, the corresponding repulsive force is given by the negative gradient of the repulsive field,

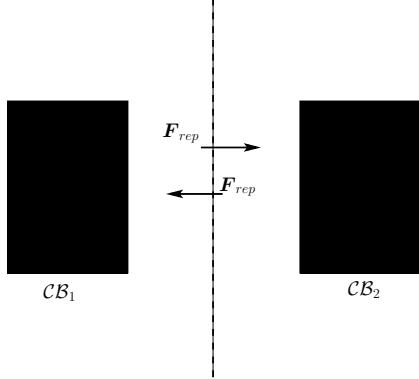
$$F_{\text{rep}}(q) = \begin{cases} \eta \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(q)} \nabla \rho(q) & : \rho(q) \leq \rho_0 \\ 0 & : \rho(q) > \rho_0 \end{cases} \quad (5.8)$$

When  $\mathcal{QO}$  is convex, the gradient of the distance to the nearest obstacle is given by

$$\nabla \rho(q) = \frac{q - b}{||q - b||} \quad (5.9)$$

in which  $b$  is the point in the boundary of  $\mathcal{QO}$  that is nearest to  $q$ . The derivation of (5.8) and (5.9) are left as exercises ??.

If  $\mathcal{QO}$  is not convex, then  $\rho$  won't necessarily be differentiable everywhere, which implies discontinuity in the force vector. Figure 5.3 illustrates such a case.



*Fig. 5.3* Situation in which the gradient of the repulsive potential of (5.7) is not continuous

Here  $\mathcal{Q}\mathcal{O}$  contains 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 the configuration of the robot 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.

### 5.2.3 Gradient Descent Planning

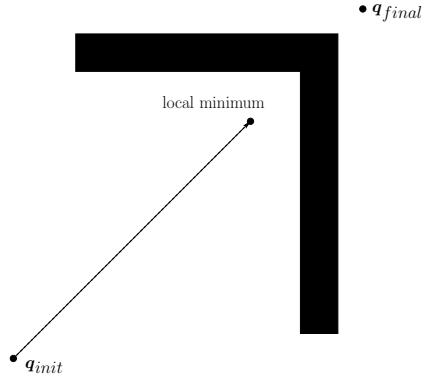
Gradient descent is a well known approach for solving optimization problems. The idea is simple. Starting at the initial configuration, take a small step in the direction of the negative gradient (i.e., in the direction that decreases the potential as quickly as possible). This gives a new configuration, and the process is repeated until the final configuration is reached. More formally, we can define a gradient descent algorithm as follows.

1.  $q^0 \leftarrow q_{\text{init}}, i \leftarrow 0$
2. **IF**  $q^i \neq q_{\text{final}}$ 

$$q^{i+1} \leftarrow q^i + \alpha^i \frac{F(q^i)}{\|F(q^i)\|}$$

$$i \leftarrow i + 1$$
**ELSE** return  $< q^0, q^1 \dots q^i >$
3. **GO TO 2**

In this algorithm, the notation  $q^i$  is used to denote the value of  $q$  at the  $i^{\text{th}}$  iteration (not the  $i^{\text{th}}$  component of the vector  $q$ ) and the final path consists of the sequence of iterates  $< q^0, q^1 \dots q^i >$ . The value of the scalar  $\alpha^i$  determines the step size at the  $i^{\text{th}}$  iteration; it is multiplied by the unit vector in the direction of the resultant force. It is important that  $\alpha^i$  be small enough that



*Fig. 5.4* This figure illustrates the progress of a gradient descent algorithm from  $q_{\text{init}}$  to a local minimum in the field  $U$

the robot is not allowed to “jump into” obstacles, while being large enough that the algorithm doesn’t require excessive computation time. In motion planning problems, the choice for  $\alpha^i$  is often made on an ad hoc or empirical basis, perhaps based on the distance to the nearest obstacle or to the goal. A number of systematic methods for choosing  $\alpha^i$  can be found in the optimization literature [7]. The constants  $\zeta$  and  $\eta$  used to define  $U_{\text{att}}$  and  $U_{\text{rep}}$  essentially arbitrate between attractive and repulsive forces. Finally, it is unlikely that we will ever exactly satisfy the condition  $q^i = q_{\text{final}}$ . For this reason, this condition is often replaced with the more forgiving condition  $\|q^i - q_{\text{final}}\| < \epsilon$ , in which  $\epsilon$  is chosen to be sufficiently small, based on the task requirements.

The problem that plagues all gradient descent algorithms is the possible existence of local minima in the potential field. For appropriate choice of  $\alpha^i$ , it can be shown that the gradient descent algorithm is guaranteed 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  $q_{\text{final}}$ . An example of this situation is shown in Figure 5.4. We will discuss ways to deal with this below in Section 5.4.

One of the main difficulties with this planning approach lies in the evaluation of  $\rho$  and  $\nabla\rho$ . In the general case, in which both rotational and translational degrees of freedom are allowed, this becomes even more difficult. We address this general case in the next section.

### 5.3 PLANNING USING WORKSPACE POTENTIAL FIELDS

As described above, in the general case, it is extremely difficult to compute an explicit representation of  $Q\mathcal{O}$ , and thus it can be extremely difficult to compute  $\rho$  and  $\nabla\rho$ . In fact, in general for a curved surface there does not exist a closed

form expression for the distance from a point to the surface. Thus, even if we had access to an explicit representation of  $\mathcal{Q}\mathcal{O}$ , it would still be difficult to compute  $\rho$  and  $\nabla\rho$  in (5.8). In order to deal with these difficulties, in this section we will modify the potential field approach of Section 5.2 so that the potential function is defined on the workspace,  $\mathcal{W}$ , instead of the configuration space,  $\mathcal{Q}$ . Since  $\mathcal{W}$  is a subset of a low dimensional space (either  $\mathbb{R}^2$  or  $\mathbb{R}^3$ ), it will be much easier to implement and evaluate potential functions over  $\mathcal{W}$  than over  $\mathcal{Q}$ .

We begin by describing a method to define an appropriate potential field on the workspace. This field should have the properties that the potential is at a minimum when the robot is in its goal configuration, and the potential should grow without bound as the robot approaches an obstacle. As above, we will define a global potential field that is the sum of attractive and repulsive fields. Once we have constructed the workspace potential, we will develop the tools to map its gradient to changes in the joint variable values (i.e., we will map workspace forces to changes in configuration). Finally, we will present a gradient descent algorithm similar to the one presented above, but which can be applied to robots with more complicated kinematics.

### 5.3.1 Defining Workspace Potential Fields

As before, our goal in defining potential functions is to construct a field that repels the robot from obstacles, with a global minimum that corresponds to  $q_{\text{final}}$ . In the configuration space, this task was conceptually simple because the robot was represented by a single point, which we treated as a point mass under the influence of a potential field. In the workspace, things are not so simple; the robot is an articulated arm with finite volume. Evaluating the effect of a potential field over the arm would involve computing an integral over the volume of the arm, and this can be quite complex (both mathematically and computationally). An alternative approach is to select a subset of points on the robot, called control points, and to define a workspace potential for each of these points. The global potential is obtained by summing the effects of the individual potential functions. Evaluating the effect a particular potential field on a single point is no different than the evaluations required in Section 5.2, but the required distance and gradient calculations are much simpler.

Let  $\mathcal{A}_{\text{att}} = \{a_1, a_2 \dots a_n\}$  be a set of control points used to define the attractive potential fields. For an  $n$ -link arm, we could use the centers of mass for the  $n$  links, or the origins for the DH frames (excluding the fixed frame 0). We denote by  $a_i(q)$  the position of the  $i^{\text{th}}$  control point when the robot is at

configuration  $q$ . For each  $a_i \in \mathcal{A}_{\text{att}}$  we can define an attractive potential by

$$U_{\text{att},i}(q) = \begin{cases} \frac{1}{2}\zeta_i||a_i(q) - a_i(q_{\text{final}})||^2 & : ||a_i(q) - a_i(q_{\text{final}})|| \leq d \\ d\zeta_i||a_i(q) - a_i(q_{\text{final}})|| - \frac{1}{2}\zeta_id^2 & : ||a_i(q) - a_i(q_{\text{final}})|| > d \end{cases} \quad (5.10)$$

For the single point  $a_i$ , this function is analogous the attractive potential defined in Section 5.2; it combines the conic and quadratic potentials, and reaches its global minimum when the control point reaches its goal position in the workspace. If  $\mathcal{A}_{\text{att}}$  contains a sufficient number of independent control points (the origins of the DH frames, e.g.), then when all control points reach their global minimum potential value, the configuration of the arm will be  $q_{\text{final}}$ .

With this potential function, the workspace force for attractive control point  $a_i$  is defined by

$$\mathcal{F}_{\text{att},i}(q) = -\nabla U_{\text{att},i}(q) \quad (5.11)$$

$$= \begin{cases} -\zeta_i(a_i(q) - a_i(q_{\text{final}})) & : ||a_i(q) - a_i(q_{\text{final}})|| \leq d \\ -\frac{d\zeta_i(a_i(q) - a_i(q_{\text{final}}))}{||a_i(q) - a_i(q_{\text{final}})||} & : ||a_i(q) - a_i(q_{\text{final}})|| > d \end{cases} \quad (5.12)$$

For the workspace repulsive potential fields, we will select a set of fixed control points on the robot  $\mathcal{A}_{\text{rep}} = \{a_1, \dots, a_m\}$ , and define the repulsive potential for  $a_j$  as

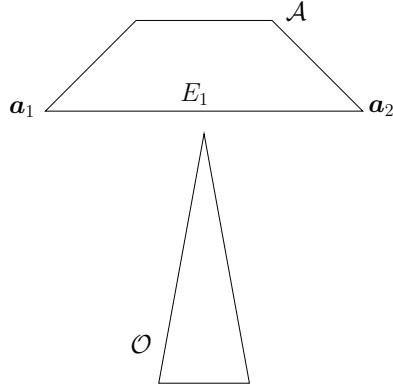
$$U_{\text{rep},j}(q) = \begin{cases} \frac{1}{2}\eta_j \left( \frac{1}{\rho(a_j(q))} - \frac{1}{\rho_0} \right)^2 & : \rho(a_j(q)) \leq \rho_0 \\ 0 & : \rho(a_j(q)) > \rho_0 \end{cases} \quad (5.13)$$

in which  $\rho(a_j(q))$  is the shortest distance between the control point  $a_j$  and any *workspace* obstacle, and  $\rho_0$  is the workspace distance of influence in the workspace for obstacles. The negative gradient of each  $U_{\text{rep},j}$  corresponds to a workspace repulsive force,

$$\mathcal{F}_{\text{rep},j}(q) = \begin{cases} \eta_j \left( \frac{1}{\rho(a_j(q))} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(a_j(q))} \nabla \rho(a_j(q)) & : \rho(a_j(q)) \leq \rho_0 \\ 0 & : \rho(a_j(q)) > \rho_0 \end{cases} \quad (5.14)$$

in which the notation  $\nabla \rho(a_j(q))$  indicates the gradient  $\nabla \rho(x)$  evaluated at  $x = a_j(q)$ . If  $b$  is the point on the workspace obstacle boundary that is closest to the repulsive control point  $a_j$ , then  $\rho(a_j(q)) = ||a_j(q) - b||$ , and its gradient is

$$\nabla \rho(x) \Big|_{x=a_j(q)} = \frac{a_j(q) - b}{||a_j(q) - b||} \quad (5.15)$$



*Fig. 5.5* The repulsive forces exerted on the robot vertices  $a_1$  and  $a_2$  may not be sufficient to prevent a collision between edge  $E_1$  and the obstacle

i.e., the unit vector directed from  $b$  toward  $a_j(q)$ .

It is important to note that this selection of repulsive control points does not guarantee that the robot cannot collide with an obstacle. Figure 5.5 shows an example where this is the case. In this figure, the repulsive control points  $a_1$  and  $a_2$  are very far from the obstacle  $\mathcal{O}$ , and therefore the repulsive influence may not be great enough to prevent the robot edge  $E_1$  from colliding with the obstacle. To cope with this problem, we can use a set of floating repulsive control points,  $a_{\text{float},i}$ , typically one per link of the robot arm. 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  $a_{\text{float},i}$  depends on the configuration  $q$ . For the example shown in Figure 5.5,  $a_{\text{float}}$  would be located at the center of edge  $E_1$ , thus repelling the robot from the obstacle. The repulsive force acting on  $a_{\text{float}}$  is defined in the same way as for the other control points, using (5.14).

### 5.3.2 Mapping workspace forces to joint forces and torques

Above we have constructed potential fields in the robot's workspace, and these fields induce artificial forces on the individual control points on the robot. In this section, we describe how these forces can be used to drive a gradient descent algorithm on the configuration space.

Suppose a force,  $\mathcal{F}$  were applied to a point on the robot arm. Such a force would induce forces and torques on the robot's joints. If the joints did not resist these forces, a motion would occur. This is the key idea behind mapping artificial forces in the workspace to motions of the robot arm. Therefore, we now derive the relationship between forces applied to the robot arm, and the resulting forces and torques that are induced on the robot joints.

Let  $F$  denote the vector of joint torques (for revolute joints) and forces (for prismatic joints) induced by the workspace force. As we will describe in Chapter 6, the principle of *virtual work* can be used to derive the relationship between  $\mathcal{F}$  and  $F$ . Let  $(\delta x, \delta y, \delta z)^T$  be a virtual displacement in the workspace and let  $\delta q$  be a virtual displacement of the robot's joints. Then, recalling that work is the inner product of force and displacement, by applying the principle of virtual work we obtain

$$\mathcal{F} \cdot (\delta x, \delta y, \delta z)^T = F \cdot \delta q \quad (5.16)$$

which can be written as

$$\mathcal{F}^T (\delta x, \delta y, \delta z)^T = F^T \delta q \quad (5.17)$$

Now, recall from Chapter 5.6 that

$$\begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = J \delta q$$

in which  $J$  is the Jacobian of the forward kinematic map for linear velocity (i.e., the top three rows of the manipulator Jacobian). Substituting this into (5.16) we obtain

$$\mathcal{F}^T J \delta q = F^T \delta q \quad (5.18)$$

and since this must hold for all virtual displacements  $\delta q$ , we obtain

$$\mathcal{F}^T J = F^T \quad (5.19)$$

which implies that

$$J^T \mathcal{F} = F \quad (5.20)$$

Thus we see that one can easily map forces in the workspace to joint forces and torques using (5.20).

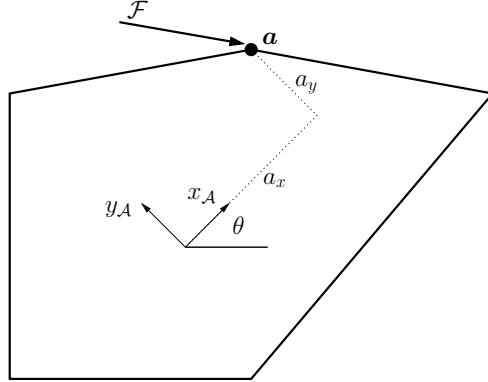
### Example 5.3 A Force Acting on a Vertex of a Polygonal Robot.

Consider the polygonal robot shown in Figure 5.6. The vertex  $a$  has coordinates  $(a_x, a_y)^T$  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  $a$  (i.e., the mapping from  $q = (x, y, \theta)$  to the global coordinates of the vertex  $a$ ) is given by

$$a(x, y, \theta) = \begin{bmatrix} x + a_x \cos \theta - a_y \sin \theta \\ y + a_x \sin \theta + a_y \cos \theta \end{bmatrix} \quad (5.21)$$

The corresponding Jacobian matrix is given by

$$J_a(x, y, \theta) = \begin{bmatrix} 1 & 0 & -a_x \sin \theta - a_y \cos \theta \\ 0 & 1 & a_x \cos \theta - a_y \sin \theta \end{bmatrix} \quad (5.22)$$



*Fig. 5.6* The robot  $\mathcal{A}$ , with coordinate frame oriented at angle  $\theta$  from the world frame, and vertex  $a$  with local coordinates  $(a_x, a_y)$

Therefore, the configuration space force is given by

$$\begin{bmatrix} F_x \\ F_y \\ F_\theta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -a_x \sin \theta - a_y \cos \theta & a_x \cos \theta - a_y \sin \theta \end{bmatrix} \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \end{bmatrix}$$

$$= \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ -\mathcal{F}_x(a_x \sin \theta - a_y \cos \theta) + \mathcal{F}_y(a_x \cos \theta - a_y \sin \theta) \end{bmatrix} \quad (5.23)$$

and  $F_\theta$  corresponds to the torque exerted about the origin of the robot frame.

In this simple case, one can use basic physics to arrive at the same result. In particular, recall that a force,  $\mathcal{F}$ , exerted at point,  $a$ , produces a torque,  $\tau$ , about the point  $O_{\mathcal{A}}$ , and this torque is given by the relationship  $\tau = r \times \mathcal{F}$ , in which  $r$  is the vector from  $O_{\mathcal{A}}$  to  $a$ . Of course we must express all vectors relative to a common frame, and in three dimensions (since torque will be defined as a vector perpendicular to the plane in which the force acts). If we choose the world frame as our frame of reference, then we have

$$r = \begin{bmatrix} a_x \cos \theta - a_y \sin \theta \\ a_x \sin \theta + a_y \cos \theta \\ 0 \end{bmatrix}$$

and the cross product gives

$$\begin{aligned} \tau &= r \times \mathcal{F} \\ &= \begin{bmatrix} a_x \cos \theta - a_y \sin \theta \\ a_x \sin \theta + a_y \cos \theta \\ 0 \end{bmatrix} \times \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ -\mathcal{F}_x(a_x \sin \theta - a_y \cos \theta) + \mathcal{F}_y(a_x \cos \theta - a_y \sin \theta) \end{bmatrix} \quad (5.24) \end{aligned}$$

Thus we see that the more general expression  $J^T \mathcal{F} = F$  gives the same value for torque as the expression  $\tau = r \times \mathcal{F}$  from mechanics.  $\diamond$

**Example 5.4** Two-link Planar Arm

Consider a two-link planar arm with the usual DH frame assignment. If we assign the control points as the origins of the DH frames (excluding the base frame), the forward kinematic equations for the arm give

$$\begin{bmatrix} a_1(\theta_1, \theta_2) & a_2(\theta_1, \theta_2) \end{bmatrix} = \begin{bmatrix} l_1 \cos \theta_1 & l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin \theta_1 & l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix}$$

in which  $l_i$  are the link lengths (we use  $l_i$  rather than  $a_i$  to avoid confusion of link lengths and control points). For the problem of motion planning, we require only the Jacobian that maps joint velocities to linear velocities,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \quad (5.25)$$

For the two-link arm, The Jacobian matrix for  $a_2$  is merely the Jacobian that we derived in Chapter 4:

$$J_{a_2}(\theta_1, \theta_2) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix} \quad (5.26)$$

The Jacobian matrix for  $a_1$  is similar, but takes into account that motion of joint two does not affect the velocity of  $a_1$ ,

$$J_{a_1}(\theta_1, \theta_2) = \begin{bmatrix} -a_1 s_1 & 0 \\ a_1 c_1 & 0 \end{bmatrix} \quad (5.27)$$

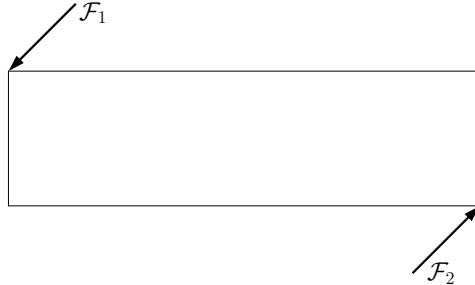
$\diamond$

The total configuration space force acting on the robot is the sum of the configuration space forces that result from all attractive and repulsive control points

$$\begin{aligned} F(q) &= \sum_i F_{\text{att},i}(q) + \sum_i F_{\text{rep},i}(q) \\ &= \sum_i J_i^T(q) \mathcal{F}_{\text{att},i}(q) + \sum_i J_i^T(q) \mathcal{F}_{\text{rep},i}(q) \end{aligned} \quad (5.28)$$

in which  $J_i(q)$  is the Jacobian matrix for control point  $a_i$ . It is essential that the addition of forces be done in the configuration space and *not* in the workspace. For example, Figure 5.7 shows a case where two workspace forces,  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , act on opposite corners of a rectangle. It is easy to see that  $\mathcal{F}_1 + \mathcal{F}_2 = 0$ , but that the combination of these forces produces a pure torque about the center of the square.

**Example 5.5** Two-link planar arm revisited. Consider again the two-link



*Fig. 5.7* This example illustrates why forces must be mapped to the configuration space before they are added. 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 moment induced by these forces

planar arm. Suppose that the workspace repulsive forces are given by  $\mathcal{F}_{\text{rep},i}(\theta_1, \theta_2) = [\mathcal{F}_{x,i}, \mathcal{F}_{y,i}]^T$ . For the two-link planar arm, the repulsive forces in the configuration space are then given by

$$\begin{aligned} \mathcal{F}_{\text{rep}}(q) = & \begin{bmatrix} -a_1 s_1 & a_1 c_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathcal{F}_{x,1} \\ \mathcal{F}_{y,1} \end{bmatrix} \\ & + \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} \\ -a_2 s_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} \mathcal{F}_{x,2} \\ \mathcal{F}_{y,2} \end{bmatrix} \quad (5.29) \end{aligned}$$

◊

### 5.3.3 Motion Planning Algorithm

Having defined a configuration space force, we can use the same gradient descent method for this case as in Section 5.3. As before, there are a number of design choices that must be made.

$\zeta_i$  controls the relative influence of the attractive potential for control point  $a_i$ .

It is not necessary that all of the  $\zeta_i$  be set to the same value. Typically, we weight one of the control points more heavily than the others, producing a “follow the leader” type of motion, in which the leader control point is quickly attracted to its final position, and the robot then reorients itself so that the other attractive control points reach their final positions.

$\eta_j$  controls the relative influence of the repulsive potential for control point  $a_j$ .

As with the  $\zeta_i$  it is not necessary that all of the  $\eta_j$  be set to the same value. In particular, we typically set the value of  $\eta_j$  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).

$\rho_0$  As with the  $\eta_j$ , we can define a distinct  $\rho_0$  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  $\rho_0$ 's to the obstacles to avoid the possibility of overlapping regions of influence for distinct obstacles.

#### 5.4 USING RANDOM MOTIONS TO ESCAPE LOCAL MINIMA

As noted above, one problem that plagues artificial potential field methods for path planning is the existence of local minima in the potential field. In the case of articulated manipulators, the resultant field  $U$  is the sum of many attractive and repulsive fields defined over  $\Re^3$ . 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, the robot path planning community has developed what are known as *randomized methods* to deal with this and other problems. The first of these methods was developed specifically to cope with the problem of local minima in potential fields.

The first planner to use randomization to escape local minima was called RPP (for Randomized Potential Planner). The basic approach is straightforward: use gradient descent until the planner finds itself stuck in a local minimum, then use a random walk to escape the local minimum. The algorithm is a slight modification of the gradient descent algorithm of Section 5.3.

1.  $q^0 \leftarrow q_{\text{init}}, i \leftarrow 0$
2. **IF**  $q^i \neq q_{\text{final}}$ 

$$q^{i+1} \leftarrow q^i + \alpha^i \frac{F(q^i)}{\|F(q^i)\|}$$
 $i \leftarrow i + 1$ 
**ELSE** return  $< q^0, q^1 \dots q^i >$
3. **IF** stuck in a local minimum
  - execute a random walk, ending at  $q'$
  - $q^{i+1} \leftarrow q'$
4. **GO TO 2**

The two new problems that must be solved are 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 (e.g., if for some small positive  $\epsilon$  we have  $\|q^i - q^{i+1}\| < \epsilon$ ,  $\|q^i - q^{i+2}\| < \epsilon$ , and  $\|q^i - q^{i+3}\| < \epsilon$  then assume  $q^i$  is near a local minimum).

Defining the random walk requires a bit more care. The original approach used in RPP is as follows. The random walk consists of  $t$  random steps. A random step from  $q = (q_1, \dots, q_n)$  is obtained by randomly adding a small fixed constant to each  $q_i$ ,

$$q_{\text{random-step}} = (q_1 \pm v_1, \dots, q_n \pm v_n)$$

with  $v_i$  a fixed small constant and the probability of adding  $+v_i$  or  $-v_i$  equal to  $1/2$  (i.e., a uniform distribution). Without loss of generality, assume that  $q = \mathbf{0}$ . We can use probability theory to characterize the behavior of the random walk consisting of  $t$  random steps. In particular, the probability density function for  $q' = (q_1, \dots, q_n)$  is given by

$$p_i(q_i, t) = \frac{1}{v_i\sqrt{2\pi t}} \exp\left(-\frac{q_i^2}{2v_i^2 t}\right) \quad (5.30)$$

which is a zero mean Gaussian density function with variance  $v_i^2 t$ . This is a result of the fact that the sum of a set of uniformly distributed random variables is a Gaussian random variable.<sup>2</sup> The variance  $v_i^2 t$  essentially determines the range of the random walk. If certain characteristics of local minima (e.g., 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 (the latter approach was used in the original RPP).

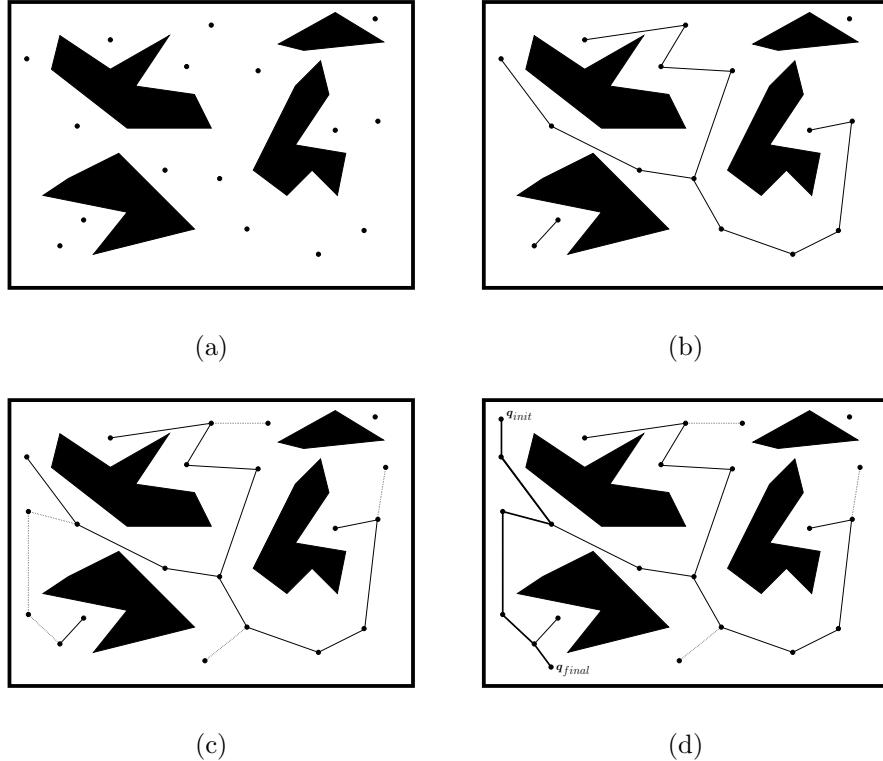
## 5.5 PROBABILISTIC ROADMAP METHODS

The potential field approaches described above incrementally explore  $\mathcal{Q}_{\text{free}}$ , searching for a path from  $q_{\text{init}}$  to  $q_{\text{final}}$ . At termination, these planners return a single path. Thus, if multiple path planning problems must be solved, such a planner must be applied once for each problem. An alternative approach is to construct a representation of  $\mathcal{Q}_{\text{free}}$  that can be used to quickly generate paths when new path planning problems arise. This is useful, for example, when a robot operates for a prolonged period in a single workspace.

In this section, we will describe probabilistic roadmaps (PRMs), which are one-dimensional roadmaps in  $\mathcal{Q}_{\text{free}}$  that can be used to quickly generate paths. Once a PRM has been constructed, the path planning problem is reduced to finding paths to connect  $q_{\text{init}}$  and  $q_{\text{final}}$  to the roadmap (a problem that is typically much easier than finding a path from  $q_{\text{init}}$  to  $q_{\text{final}}$ ).

A PRM is a network of simple curve segments, or arcs, that meet at nodes. Each node corresponds to a configuration. Each arc between two nodes corresponds to a collision free path between two configurations. Constructing a PRM is a conceptually straightforward process. First, a set of random configurations is generated to serve as the nodes in the network. Then, a simple, local path planner is used to generate paths that connect pairs of configurations.

<sup>2</sup>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.



*Fig. 5.8* (a) A two-dimensional configuration space populated with several random samples (b) One possible PRM for the given configuration space and random samples (c) PRM after enhancement (d) path from  $q_{init}$  to  $q_{final}$  found by connecting  $q_{init}$  and  $q_{final}$  to the roadmap and then searching the roadmap for a path from  $q_{init}$  to  $q_{final}$

Finally, if the initial network consists of multiple connected components<sup>3</sup>, it is augmented by an enhancement phase, in which new nodes and arcs are added in an attempt to connect disjoint components of the network. To solve a path planning problem, the simple, local planner is used to connect  $q_{init}$  and  $q_{final}$  to the roadmap, and the resulting network is searched for a path from  $q_{init}$  to  $q_{final}$ . These four steps are illustrated in Figure 5.8. We now discuss these steps in more detail.

<sup>3</sup> A connected component is a maximal subnetwork of the network such that a path exists in the subnetwork between any two nodes.

|   |  |
|---|--|
| <b>2-norm in C-space:</b>                     | $\ q' - q\  = \left[ \sum_{i=1}^n (q'_i - q_i)^2 \right]^{\frac{1}{2}}$    |
| <b><math>\infty</math>-norm in C-space:</b>   | $\max_n  q'_i - q_i $  |
| <b>2-norm in workspace:</b>                   | $\left[ \sum_{p \in \mathcal{A}} \ p(q') - p(q)\ ^2 \right]^{\frac{1}{2}}$ |
| <b><math>\infty</math>-norm in workspace:</b> | $\max_{p \in \mathcal{A}} \ p(q') - p(q)\ $                                |

Table 5.1 Four commonly used distance functions

### 5.5.1 Sampling the configuration space

The simplest way to generate sample configurations is to sample the configuration space uniformly at random. Sample configurations that lie in  $\mathcal{Q}\mathcal{O}$  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.

### 5.5.2 Connecting Pairs of Configurations

Given a set of nodes that correspond to configurations, the next step in building the PRM is to determine which pairs of nodes should be connected by a simple path. The typical approach is to attempt to connect each node 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 5.1 lists four distance functions that have been popular in the PRM literature. For the equations in this table, the robot has  $n$  joints,  $q$  and  $q'$  are the two configurations corresponding to different nodes in the roadmap,  $q_i$  refers to the configuration of the  $i$ th joint, and  $p(q)$  refers to the workspace reference point  $p$  of a set of reference points of the robot,  $\mathcal{A}$ , at configuration  $q$ . Of these, the simplest, and perhaps most commonly used, is the 2-norm in configuration space.

Once pairs of neighboring nodes have been identified, a simple local planner is used to connect these nodes. Often, a straight line in configuration space is used as the candidate plan, and thus, planning the path between two nodes 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 (e.g., RPP discussed above) can be used to attempt to connect the nodes.

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 terribly 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 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.

### 5.5.3 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 to merely attempt to directly connect nodes in two disjoint components, perhaps by using a more sophisticated planner such as RPP. A common approach is to identify the largest connected component, and to attempt to connect the smaller components to it. The node 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 node randomly as candidate for connection, and to bias the random choice based on the number of neighbors of the node; a node with fewer neighbors in the network is more likely to be near a narrow passage, and should be a more likely candidate for connection.

A second approach to enhancement is to add samples more random nodes to the PRM, in the hope of finding nodes that lie in or near the narrow passages. One approach is to identify nodes that have few neighbors, and to generate sample configurations in regions around these nodes. The local planner is then used to attempt to connect these new configurations to the network.

### 5.5.4 Path Smoothing

After the PRM has been generated, path planning amounts to connecting  $q_{\text{init}}$  and  $q_{\text{final}}$  to the network 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.

## 5.6 TRAJECTORY PLANNING

A path from  $q_{init}$  to  $q_{final}$  is defined as a continuous map,  $\tau : [0, 1] \rightarrow \mathcal{Q}$ , with  $\tau(0) = q_{init}$  and  $\tau(1) = q_{final}$ . A **trajectory** is a function of time  $q(t)$  such that  $q(t_0) = q_{init}$  and  $q(t_f) = q_{final}$ . 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  $\tau$  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  $\tau$  gives a complete specification of the robot's trajectory, including the time derivatives (since one need only differentiate  $\tau$  to obtain these).

As seen above, a path planning algorithm will not typically give the map  $\tau$ ; it will give only a sequence of points (called **via points**) along the path. Further, there are other ways that the 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, e.g. in environments such as the one shown in Figure 5.9. 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) and the robot can be controlled entirely in joint space. Finally, in cases for which no obstacles are present, the manipulator is essentially unconstrained. It is often the case that a manipulator motion can be decomposed into a segments consisting of free and **guarded motions**, shown in Figure 5.10. During the free motion, the manipulator can move very fast, since no obstacles are near by, but at the start and end of the motion, care must be taken to avoid obstacles.

We first consider **point to point** motion. In this case the task is to plan a trajectory from  $q(t_0)$  to  $q(t_f)$ , i.e., the path is specified by its initial and final configurations. In some cases, there may be constraints on the trajectory (e.g., 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  $n$ , with  $n$  dependant 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.

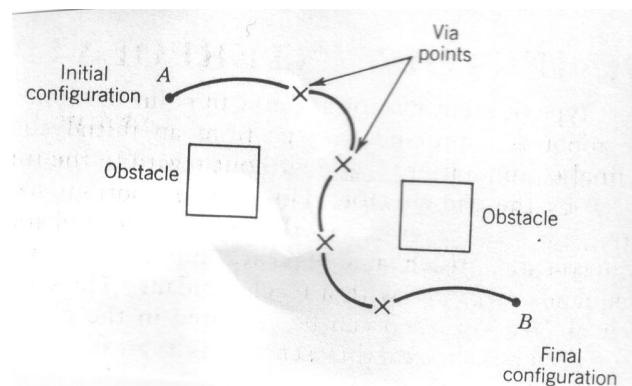


Fig. 5.9 Via points to plan motion around obstacles

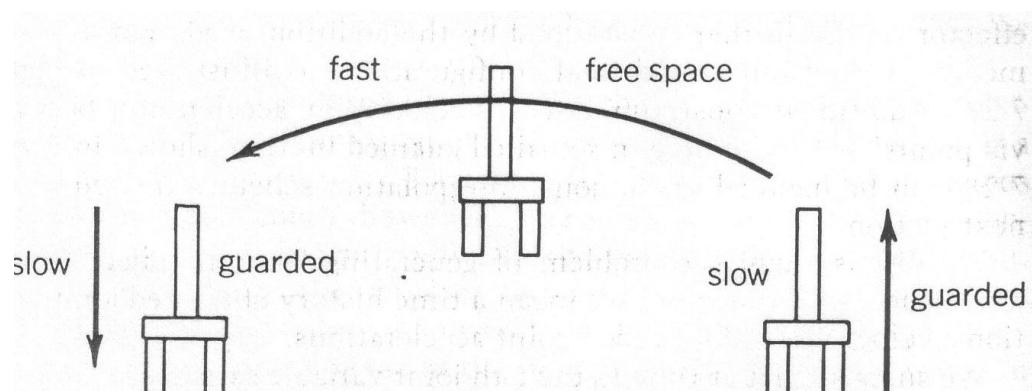


Fig. 5.10 Guarded and free motions

### 5.6.1 Trajectories for Point to Point Motion

As described above, the problem here is to find a trajectory that connects an initial to a final configuration while satisfying other specified constraints at the endpoints (e.g., 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

$$q(t_0) = q_0 \quad (5.31)$$

$$\dot{q}(t_0) = v_0 \quad (5.32)$$

and we wish to attain the values at  $t_f$

$$q(t_f) = q_f \quad (5.33)$$

$$\dot{q}(t_f) = v_f \quad (5.34)$$

Figure 5.11 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 the additional equations

$$\ddot{q}(t_0) = \alpha_0 \quad (5.35)$$

$$\ddot{q}(t_f) = \alpha_f \quad (5.36)$$

**5.6.1.1 Cubic Polynomial Trajectories** Suppose that we wish to generate a trajectory between two configurations, and that we wish to specify the start and end velocities for the trajectory. One way to generate a smooth curve such as that shown in Figure 5.11 is by a polynomial function of  $t$ . If we have four constraints to satisfy, such as (5.31)-(5.33), 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

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (5.37)$$

Then the desired velocity is given as

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (5.38)$$

Combining equations (5.37) and (5.38) with the four constraints yields four equations in four unknowns

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \quad (5.39)$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 \quad (5.40)$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \quad (5.41)$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 \quad (5.42)$$

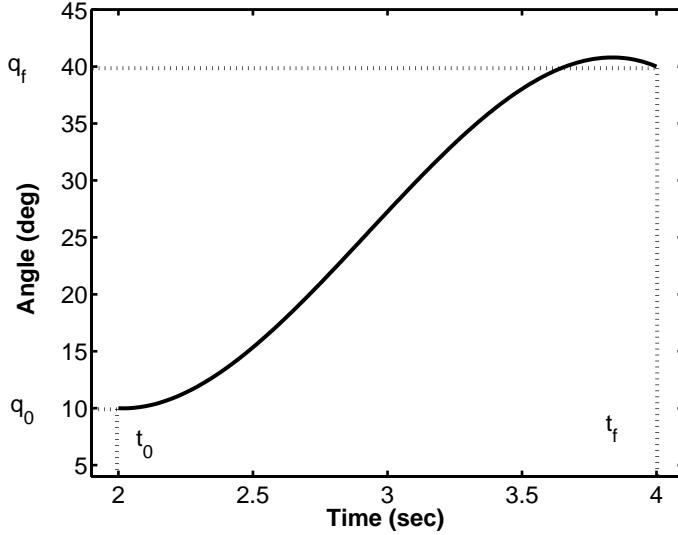


Fig. 5.11 Typical Joint Space Trajectory

These four equations can be combined into a single matrix equation

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \quad (5.43)$$

It can be shown (Problem 5-1) that the determinant of the coefficient matrix in Equation (5.43) is equal to  $(t_f - t_0)^4$  and, hence, Equation (5.43) always has a unique solution provided a nonzero time interval is allowed for the execution of the trajectory.

### Example 5.6

Writing Equation (5.43) as

$$Ma = b \quad (5.44)$$

where  $M$  is the coefficient matrix,  $a = [a_0, a_1, a_2, a_3]^T$  is the vector of coefficients of the cubic polynomial, and  $b = [q_0, v_0, q_f, v_f]^T$  is the vector of initial data (initial and final positions and velocities), the Matlab script shown in Figure 5.12 computes the general solution as

$$a = M^{-1}b \quad (5.45)$$

◇

### Example 5.7

---

```
%%
%% cubic.m
%%
%% M-file to compute a cubic polynomial reference trajectory
%%
%% q0 = initial position
%% v0 = initial velocity
%% q1 = final position
%% v1 = final velocity
%% t0 = initial time
%% tf = final time
%%
clear
d = input(' initial data = [q0,v0,q1,v1,t0,tf] = ')
q0 = d(1); v0 = d(2); q1 = d(3); v1 = d(4);
t0 = d(5); tf = d(6);
t = linspace(t0,tf,100*(tf-t0));
c = ones(size(t));
M = [ 1 t0 t0^2 t0^3;
      0 1 2*t0 3*t0^2;
      1 tf tf^2 tf^3;
      0 1 2*tf 3*tf^2];
%%
b = [q0; v0; q1; v1];
a = inv(M)*b;
%%
% qd = reference position trajectory
% vd = reference velocity trajectory
% ad = reference acceleration trajectory
%
qd = a(1).*c + a(2).*t +a(3).*t.^2 + a(4).*t.^3;
vd = a(2).*c +2*a(3).*t +3*a(4).*t.^2;
ad = 2*a(3).*c + 6*a(4).*t;
```

---

Fig. 5.12 Matlab code for Example 5.6

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, with

$$v_0 = 0 \quad v_f = 0 \quad (5.46)$$

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 the Equation (5.43) we obtain

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ 0 \\ q_f \\ 0 \end{bmatrix} \quad (5.47)$$

This is then equivalent to the four equations

$$a_0 = q_0 \quad (5.48)$$

$$a_1 = 0 \quad (5.49)$$

$$a_2 + a_3 = q_f - q_0 \quad (5.50)$$

$$2a_2 + 3a_3 = 0 \quad (5.51)$$

These latter two can be solved to yield

$$a_2 = 3(q_f - q_0) \quad (5.52)$$

$$a_3 = -2(q_f - q_0) \quad (5.53)$$

The required cubic polynomial function is therefore

$$q_i(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3 \quad (5.54)$$

Figure 5.13(a) shows this trajectory with  $q_0 = 10^\circ$ ,  $q_f = -20^\circ$ . The corresponding velocity and acceleration curves are given in Figures 5.13(b) and 5.13(c).

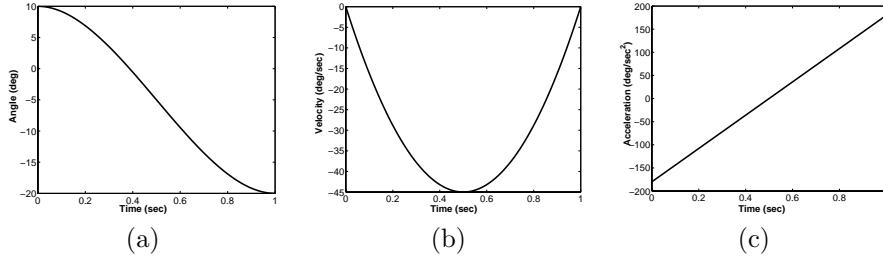


Fig. 5.13 (a) Cubic polynomial trajectory (b) Velocity profile for cubic polynomial trajectory (c) Acceleration profile for cubic polynomial trajectory

◇

**5.6.1.2 Quintic Polynomial Trajectories** As can be seen in Figure 5.13, 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

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (5.55)$$

Using (5.31) - (5.36) 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

$$\left[ \begin{array}{cccccc} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{array} \right] \left[ \begin{array}{c} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{array} \right] = \left[ \begin{array}{c} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{array} \right] \quad (5.56)$$

Figure 5.14 shows the Matlab script that gives the general solution to this equation.

### Example 5.8

Figure 5.15 shows a quintic polynomial trajectory with  $q(0) = 0$ ,  $q(2) = 40$  with zero initial and final velocities and accelerations.

◇

**5.6.1.3 Linear Segments with Parabolic Blends (LSPB)** Another way to generate suitable joint space trajectories is by so-called **Linear Segments with Parabolic Blends** or **(LSPB)** for short. This type of trajectory 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

---

```

%%
%% quintic.m
&&
%% M-file to compute a quintic polynomial reference trajectory
%%
%% q0    = initial position
%% v0    = initial velocity
%% ac0   = initial acceleration
%% q1    = final position
%% v1    = final velocity
%% ac1   = final acceleration
%% t0    = initial time
%% tf    = final time
%%
clear
d = input(' initial data = [q0,v0,ac0,q1,v1,ac1,t0,tf] = ')
q0 = d(1); v0 = d(2); ac0 = d(3);
q1 = d(4); v1 = d(5); ac1 = d(6);
t0 = d(7); tf = d(8);
t = linspace(t0,tf,100*(tf-t0));
c = ones(size(t));
M = [ 1 t0 t0^2 t0^3 t0^4 t0^5;
      0 1 2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0 2 6*t0 12*t0^2 20*t0^3;
      1 tf tf^2 tf^3 tf^4 tf^5;
      0 1 2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0 2 6*tf 12*tf^2 20*tf^3];
%%
b=[q0; v0; ac0; q1; v1; ac1];
a = inv(M)*b;
%%
%% qd = position trajectory
%% vd = velocity trajectory
%% ad = acceleration trajectory
%%
qd = a(1).*c + a(2).*t + a(3).*t.^2 + a(4).*t.^3 + a(5).*t.^4 + a(6).*t.^5;
vd = a(2).*c + 2*a(3).*t + 3*a(4).*t.^2 + 4*a(5).*t.^3 + 5*a(6).*t.^4;
ad = 2*a(3).*c + 6*a(4).*t + 12*a(5).*t.^2 + 20*a(6).*t.^3;

```

---

*Fig. 5.14 Matlab code to generate coefficients for quintic trajectory segment*

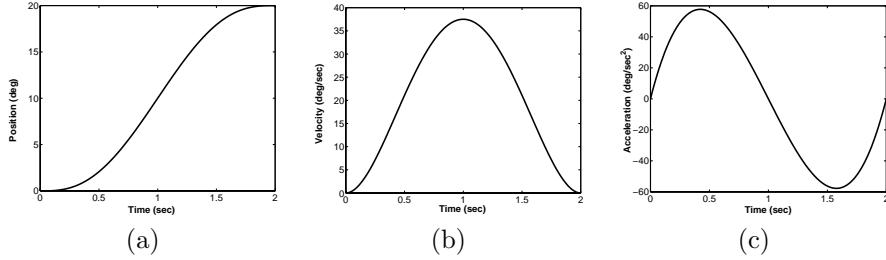


Fig. 5.15 (a) Quintic Polynomial Trajectory. (b) Velocity Profile for Quintic Polynomial Trajectory. (c) Acceleration Profile for Quintic Polynomial Trajectory

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 time  $t_f - t_b$  the trajectory switches once again, this time to a quadratic polynomial so that the velocity is linear.

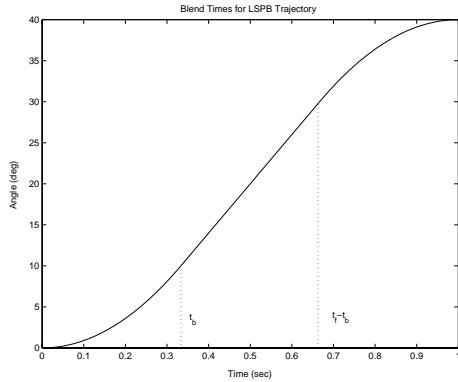


Fig. 5.16 Blend times for LSPB trajectory

We choose the blend time  $t_b$  so that the position curve is symmetric as shown in Figure 5.16. 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

$$q(t) = a_0 + a_1 t + a_2 t^2 \quad (5.57)$$

so that the velocity is

$$\dot{q}(t) = a_1 + 2a_2 t \quad (5.58)$$

The constraints  $q_0 = 0$  and  $\dot{q}(0) = 0$  imply that

$$a_0 = q_0 \quad (5.59)$$

$$a_1 = 0 \quad (5.60)$$

At time  $t_b$  we want the velocity to equal a given constant, say  $V$ . Thus, we have

$$\dot{q}(t_b) = 2a_2 t_b = V \quad (5.61)$$

which implies that

$$a_2 = \frac{V}{2t_b} \quad (5.62)$$

Therefore the required trajectory between 0 and  $t_b$  is given as

$$\begin{aligned} q(t) &= q_0 + \frac{V}{2t_b} t^2 \\ &= q_0 + \frac{\alpha}{2} t^2 \end{aligned} \quad (5.63)$$

$$\dot{q}(t) = \frac{V}{t_b} t = \alpha t \quad (5.64)$$

$$\ddot{q} = \frac{V}{t_b} = \alpha \quad (5.65)$$

where  $\alpha$  denotes the acceleration.

Now, between time  $t_f$  and  $t_f - t_b$ , the trajectory is a linear segment (corresponding to a constant velocity  $V$ )

$$q(t) = a_0 + a_1 t = a_0 + Vt \quad (5.66)$$

Since, by symmetry,

$$q\left(\frac{t_f}{2}\right) = \frac{q_0 + q_f}{2} \quad (5.67)$$

we have

$$\frac{q_0 + q_f}{2} = a_0 + V \frac{t_f}{2} \quad (5.68)$$

which implies that

$$a_0 = \frac{q_0 + q_f - Vt_f}{2} \quad (5.69)$$

Since the two segments must “blend” at time  $t_b$  we require

$$q_0 + \frac{V}{2} t_b = \frac{q_0 + q_f - Vt_f}{2} + Vt_b \quad (5.70)$$

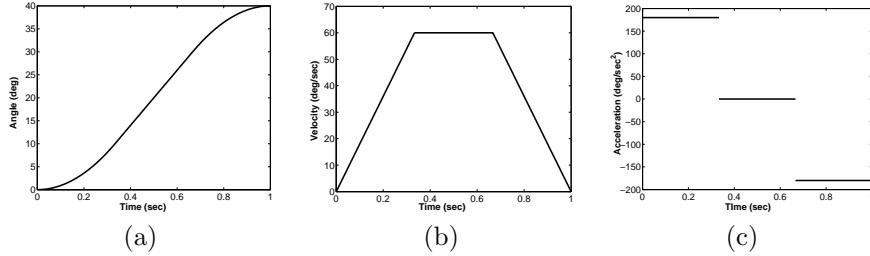


Fig. 5.17 (a) LSPB trajectory (b) Velocity profile for LSPB trajectory (c) Acceleration for LSPB trajectory

which gives upon solving for the blend time  $t_b$

$$t_b = \frac{q_0 - q_f + Vt_f}{V} \quad (5.71)$$

Note that we have the constraint  $0 < t_b \leq \frac{t_f}{2}$ . This leads to the inequality

$$\frac{q_f - q_0}{V} < t_f \leq \frac{2(q_f - q_0)}{V} \quad (5.72)$$

To put it another way we have the inequality

$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f} \quad (5.73)$$

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 5-6). The complete LSPB trajectory is given by

$$q(t) = \begin{cases} q_0 + \frac{a}{2}t^2 & 0 \leq t \leq t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \leq t_f - t_b \\ q_f - \frac{at_f^2}{2} + at_f t - \frac{a}{2}t^2 & t_f - t_b < t \leq t_f \end{cases} \quad (5.74)$$

Figure 5.17(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 5.17(b) and 5.17(c), respectively.

**5.6.1.4 Minimum Time Trajectories** An important variation of this 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  $\alpha$ , 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

$$V_s = \alpha t_s \quad (5.75)$$

and also

$$t_s = \frac{q_0 - q_f + V_s t_f}{V_s} \quad (5.76)$$

The symmetry condition  $t_s = \frac{t_f}{2}$  implies that

$$V_s = \frac{q_f - q_0}{t_s} \quad (5.77)$$

Combining these two we have the conditions

$$\frac{q_f - q_0}{t_s} = \alpha t_s \quad (5.78)$$

which implies that

$$t_s = \sqrt{\frac{q_f - q_0}{\alpha}} \quad (5.79)$$

### 5.6.2 Trajectories for Paths Specified by Via Points

Now that we have examined the problem of planning a trajectory between two configuration, we generalize our approach to the case of planning a trajectory that passes through a sequence of configurations, called **via points**. Consider the simple of example of a path specified by three points,  $q_0, q_1, q_2$ , such that the via points are reached at times  $t_0, t_1$  and  $t_2$ . 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

$$q(t) = a_6 t^6 + a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t^1 + a_0 \quad (5.80)$$

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 sometimes referred to as interpolating polynomials or blending polynomials. With this approach, we must take care that continuity constraints (e.g., in velocity and acceleration) are satisfied at the via points, where we switch from one polynomial to another.

Given initial and final times,  $t_0$  and  $t_f$ , respectively, with

$$\begin{aligned}
q^d(t_0) &= q_0 & q^d(t_f) &= q_1 \\
\dot{q}^d(t_0) &= q'_0 & \dot{q}^d(t_f) &= q'_1
\end{aligned} \quad (5.81)$$

the required cubic polynomial  $q^d(t)$  can be computed from

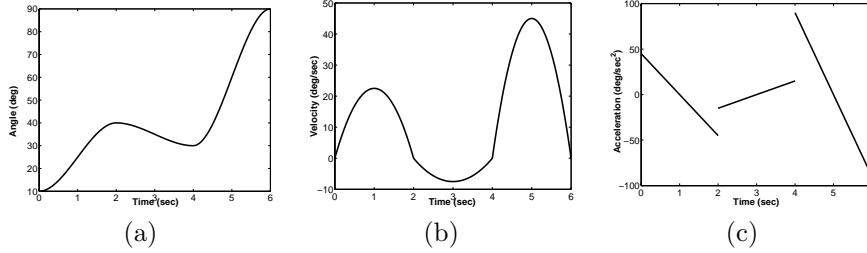
$$q^d(t_0) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \quad (5.82)$$

where

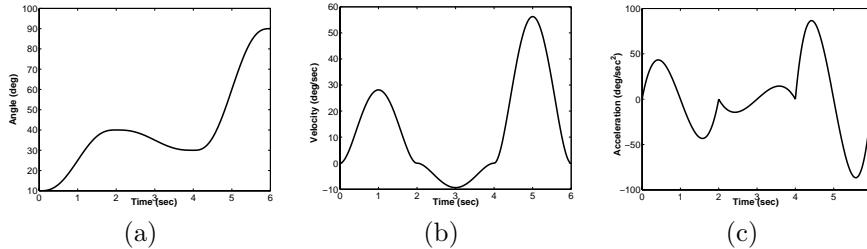
$$a_2 = \frac{3(q_1 - q_0) - (2q'_0 + q'_1)(t_f - t_0)}{(t_f - t_0)^2} \quad a_3 = \frac{2(q_0 - q_1) + (q'_0 + q'_1)(t_f - t_0)}{(t_f - t_0)^3}$$

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  $i+1$ -st move.

**Example 5.9** Figure 5.18 shows a 6-second move, computed in three parts using (5.82), where the trajectory begins at  $10^\circ$  and is required to reach  $40^\circ$  at



*Fig. 5.18* (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



*Fig. 5.19* (a) Trajectory with Multiple Quintic Segments (b) Velocity Profile for Multiple Quintic Segments (c) Acceleration Profile for Multiple Quintic Segments

2-seconds, 30° at 4seconds, and 90° at 6-seconds, with zero velocity at 0,2,4, and 6 seconds.

◇

**Example 5.10** Figure 5.19 shows the same six second trajectory as in Example 5.9 with the added constraints that the accelerations should be zero at the blend times.

◇

## 5.7 HISTORICAL PERSPECTIVE

The earliest work on robot planning was done in the late sixties and early seventies in a few University-based Artificial Intelligence (AI) labs [25, 28, 57]. 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 plausible manner. The configuration space and its application to path planning were introduced in [47]. This was the first rigorous, formal treatment of the geometric path plan-

ning 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 (e.g., [65]), and approximate methods (e.g., [11, 36, 47]). 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 C-space grows exponentially in the dimension of the C-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 [12]. 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 [39, 40].

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 [42]. 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. Soon after, the research field of *Algorithmic Robotics* was born at a small workshop in San Francisco [31].

In the early nineties, randomization was introduced in the robot planning community [5], 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 [38]. 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 [37, 58, 38].

Finally, much work has been done in the area of collision detection in recent years. [46, 52, 73, 74]. 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.

---

## Problems

---

### MOTION PLANNING PROBLEMS TO BE WRITTEN

- 5-1 Show by direct calculation that the determinant of the coefficient matrix in Equation (5.43) is  $(t_f - t_0)^4$ .
- 5-2 Use Gaussian elimination to reduce the system (5.43) to upper triangular form and verify that the solution is indeed given by Equation (5.82).
- 5-3 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.
- 5-4 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.
- 5-5 Compute a LSPB trajectory to satisfy the same requirements as in Problem 5-4. Sketch the resulting position, velocity, and acceleration profiles.
- 5-6 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 hence verify Equations (5.74).
- 5-7 Write a Matlab m-file, lspb.m, to generate an LSPB trajectory, given appropriate initial data.
- 5-8 Rewrite the Matlab m-files, cubic.m, quintic.m, and lspb.m to turn them into Matlab functions. Document them appropriately.

# 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.

The Euler-Lagrange equations have several very important properties 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 so-called skew symmetry and passivity properties. We discuss these properties in Section 6.5.

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.

## 6.1 THE EULER-LAGRANGE EQUATIONS

In this section we derive 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 displacements; but it is also possible to derive the same equations based on Hamilton's principle of least action [?].

### 6.1.1 One Dimensional System

To motivate the subsequent derivation, we show first how the Euler-Lagrange equations can be derived from Newton's Second Law for a single degree of freedom system consisting of a particle of constant mass  $m$ , constrained to move in the  $y$ -direction, and subject to a force  $f$  and the gravitational force  $mg$ , as shown in Figure 6.1. By Newton's Second law, the equation of motion

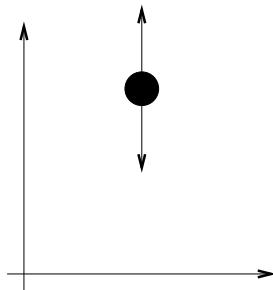


Fig. 6.1 One Degree of Freedom System

of the particle is

$$m\ddot{y} = f - mg \quad (6.1)$$

Notice that the left hand side of Equation (6.1) can be written as

$$m\ddot{y} = \frac{d}{dt}(m\dot{y}) = \frac{d}{dt} \frac{\partial}{\partial \dot{y}} \left( \frac{1}{2} m \dot{y}^2 \right) = \frac{d}{dt} \frac{\partial \mathcal{K}}{\partial \dot{y}} \quad (6.2)$$

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

$$mg = \frac{\partial}{\partial y}(mgy) = \frac{\partial \mathcal{P}}{\partial y} \quad (6.3)$$

where  $\mathcal{P} = mgy$  is the **potential energy due to gravity**. If we define

$$\mathcal{L} = \mathcal{K} - \mathcal{P} = \frac{1}{2}m\dot{y}^2 - mgy \quad (6.4)$$

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

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{y}} - \frac{\partial \mathcal{L}}{\partial y} = f \quad (6.5)$$

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 Euler-Lagrange equations provide a formulation of the dynamic equations of motion equivalent to those derived using Newton's Second Law. However, as we shall see, the Lagrangian approach is advantageous for more 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

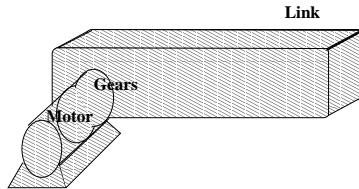


Fig. 6.2 Single-Link Robot.

link coupled through a gear train to a DC-motor. Let  $\theta_\ell$  and  $\theta_m$  denote the angles of the link and motor shaft, respectively. Then  $\theta_m = r\theta_\ell$  where  $r : 1$  is the gear ratio. The algebraic relation between the link and motor shaft angles means that the system has only one degree-of-freedom and we can therefore write the equations of motion using either  $\theta_m$  or  $\theta_\ell$ . In terms of  $\theta_\ell$ , the kinetic energy of the system is given by

$$\begin{aligned} K &= \frac{1}{2}J_m\dot{\theta}_m^2 + \frac{1}{2}J_\ell\dot{\theta}_\ell^2 \\ &= \frac{1}{2}(r^2J_m + J_\ell)\dot{\theta}_\ell^2 \end{aligned} \quad (6.6)$$

where  $J_m, J_\ell$  are the rotational inertias of the motor and link, respectively. The potential energy is given as

$$P = Mg\ell(1 - \cos \theta_\ell) \quad (6.7)$$

where  $M$  is the total mass of the link and  $\ell$  is the distance from the joint axis to the link center of mass. Defining  $J = r^2 J_m + J_\ell$ , the Lagrangian  $\mathcal{L}$  is given by

$$\mathcal{L} = \frac{1}{2}J\dot{\theta}_\ell^2 - Mg\ell(1 - \cos \theta_\ell) \quad (6.8)$$

Substituting this expression into the Euler-Lagrange equations yields the equation of motion

$$J\ddot{\theta}_\ell + Mg\ell \sin \theta_\ell = \tau_\ell \quad (6.9)$$

The generalized force  $\tau_\ell$  represents those external forces and torques that are not derivable from a potential function. For this example,  $\tau_\ell$  consists of the 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

$$\tau = u - B\dot{\theta}_\ell$$

where  $B = rB_m + B_\ell$ . Therefore the complete expression for the dynamics of this system is

$$J\ddot{\theta}_\ell + B\dot{\theta}_\ell + Mg\ell \sin \theta_\ell = u \quad (6.10)$$

In general, for any system of the type considered, an application of the Euler-Lagrange equations leads to a system of  $n$  coupled, second order nonlinear ordinary differential equations of the form

### Euler-Lagrange Equations

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \quad i = 1, \dots, n \quad (6.11)$$

The order,  $n$ , of the system is determined by the number of so-called **generalized coordinates** that are required to describe the evolution of the system. We shall see that the  $n$  Denavit-Hartenberg joint variables serve as a set of generalized coordinates for an  $n$ -link rigid robot.

#### 6.1.2 The General Case

Now, consider a system of  $k$  particles, with corresponding position vectors  $r_1, \dots, r_k$ . 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

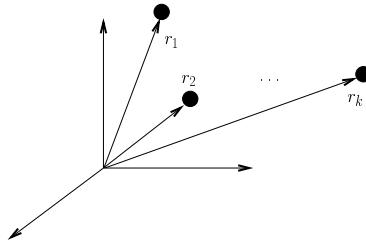


Fig. 6.3 System of  $k$  particles

rate of change of the momentum of each mass 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, which are joined by a massless rigid wire of length  $\ell$ . Then the two coordinates  $r_1$  and  $r_2$  must satisfy the constraint

$$\|r_1 - r_2\| = \ell, \quad \text{or} \quad (r_1 - r_2)^T(r_1 - r_2) = \ell^2 \quad (6.12)$$

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 can follow one of two options. We can 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 in general quite an involved task to compute the constraint forces. The contents of this section are aimed at achieving this latter objective.

First it is necessary to introduce some terminology. A constraint on the  $k$  coordinates  $r_1, \dots, r_k$  is called **holonomic** if it is an equality constraint of the form

$$g_i(r_1, \dots, r_k) = 0, \quad i = 1, \dots, \ell \quad (6.13)$$

and **nonholonomic** otherwise. The constraint (6.12) imposed by connecting two particles by a massless rigid wire is a holonomic constraint. As an example of a nonholonomic constraint, consider a particle moving inside a sphere of radius  $\rho$  centered at the origin of the coordinate system. In this case the coordinate vector  $r$  of the particle must satisfy the constraint

$$\|r\| \leq \rho \quad (6.14)$$

Note that the motion of the particle is unconstrained so long as the particle remains away from the wall of the sphere; but when the particle comes into contact with the wall, it experiences a constraining force.

If a system is subjected to  $\ell$  holonomic constraints, then one can think in terms of the constrained system having  $\ell$  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**  $q_1, \dots, q_n$ . In other words, we assume that the coordinates of the various particles, subjected to the set of constraints (6.13), can be expressed in the form

$$r_i = r_i(q_1, \dots, q_n), \quad i = 1, \dots, k \quad (6.15)$$

where  $q_1, \dots, q_n$  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, only 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, and three Euler angles to specify the orientation of the body. To keep the discussion simple, however, we assume in what follows that the number of particles is finite. Typically, generalized coordinates are positions, angles, etc. In fact, in Chapter 3 we chose to denote the joint variables by the symbols  $q_1, \dots, q_n$  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,  $\delta r_1, \dots, \delta r_k$ , of infinitesimal displacements that are consistent with the constraints. For example, consider once again the constraint (6.12) 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, we must have

$$(r_1 + \delta r_1 - r_2 - \delta r_2)^T(r_1 + \delta r_1 - r_2 - \delta r_2) = \ell^2 \quad (6.16)$$

Now let us expand the above product and take advantage of the fact that the original coordinates  $r_1, r_2$  satisfy the constraint (6.12); let us also neglect quadratic terms in  $\delta r_1, \delta r_2$ . This shows that

$$(r_1 - r_2)^T(\delta r_1 - \delta r_2) = 0 \quad (6.17)$$

Thus any perturbations in the positions of the two particles must satisfy the above equation in order that the perturbed positions continue to satisfy the constraint (6.12). Any pair of infinitesimal vectors  $\delta r_1, \delta r_2$  that satisfy (6.17) would constitute a set of virtual displacements for this problem.

Now the reason for using generalized coordinates is to avoid dealing with complicated relationships such as (6.17) above. If (6.15) holds, then one can

see that the set of all virtual displacements is precisely

$$\delta r_i = \sum_{j=1}^n \frac{\partial r_i}{\partial q_j} \delta q_j, \quad i = 1, \dots, k \quad (6.18)$$

where the virtual displacements  $\delta q_1, \dots, \delta q_n$  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,

$$\sum_{i=1}^k F_i^T \delta r_i = 0 \quad (6.19)$$

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,

$$\sum_{i=1}^k (f_i^{(a)})^T \delta r_i = 0 \quad (6.20)$$

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 (6.20) into (6.19) results in

$$\sum_{i=1}^k f_i^T \delta r_i = 0 \quad (6.21)$$

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 follows:

#### Principle of Virtual Work

The work done by external forces corresponding to any set of virtual displacements is zero.

Note that the principle is not universally applicable, but requires that (6.20) hold, that is, that the constraint forces do no work. Thus, if the principle of virtual work applies, then 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 (6.12), the principle applies. To see this, consider once again a single constraint of the form (6.12). 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 particle 1 by the wire must be of the form

$$f_1^{(a)} = c(r_1 - r_2) \quad (6.22)$$

for some constant  $c$  (which could change as the particles move about). By the law of action and reaction, the force exerted on particle 2 by the wire must be just the negative of the above, that is,

$$f_2^{(a)} = -c(r_1 - r_2) \quad (6.23)$$

Now the work done by the constraint forces corresponding to a set of virtual displacements is

$$(f_1^{(a)})^T \delta r_1 + (f_2^{(a)})^T \delta r_2 = c(r_1 - r_2)^T (\delta r_1 - \delta r_2) \quad (6.24)$$

But (6.17) shows that for any set of virtual displacements, the above inner product must be zero. Thus the principle of virtual work applies in a system constrained by (6.12). The same reasoning can be applied if the system consists of several particles, which are pairwise connected by rigid massless wires of fixed lengths, in which case the system is subjected to several constraints of the form (6.12). 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 (6.12). 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, typically 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.

In (6.21), the virtual displacements  $\delta r_i$  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 particle  $i$  for each  $i$ , where  $p_i$  is the momentum of particle  $i$ , then each particle will be in equilibrium. Thus, if one modifies (6.19) 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 equations

$$\sum_{i=1}^k f_i^T \delta r_i - \sum_{i=1}^k \dot{p}_i^T \delta r_i = 0 \quad (6.25)$$

The above equation does not mean that each coefficient of  $\delta r_i$  is zero. For this purpose, express each  $\delta r_i$  in terms of the corresponding virtual displacements of generalized coordinates, as is done in (6.18). Then the virtual work done by the forces  $f_i$  is given by

$$\sum_{i=1}^k f_i^T \delta r_i = \sum_{i=1}^k \sum_{j=1}^n f_i^T \frac{\partial r_i}{\partial q_j} \delta q_j = \sum_{j=1}^n \psi_j \delta q_j \quad (6.26)$$

where

$$\psi_j = \sum_{i=1}^k f_i^T \frac{\partial r_i}{\partial q_j} \quad (6.27)$$

is called the  $j$ -th **generalized force**. Note that  $\psi_j$  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 (6.25). Since  $p_i = m_i \dot{r}_i$ , it follows that

$$\sum_{i=1}^k \dot{p}_i^T \delta r_i = \sum_{i=1}^k m_i \ddot{r}_i^T \delta r_i = \sum_{i=1}^k \sum_{j=1}^n m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \delta q_j \quad (6.28)$$

Next, using the product rule of differentiation, we see that

$$\sum_{i=1}^k m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^k \left\{ \frac{d}{dt} \left[ m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] - m_i \ddot{r}_i^T \frac{d}{dt} \left[ \frac{\partial r_i}{\partial q_j} \right] \right\} \quad (6.29)$$

Now differentiate (6.15) using the chain rule; this gives

$$v_i = \dot{r}_i = \sum_{j=1}^n \frac{\partial r_i}{\partial q_j} \dot{q}_j \quad (6.30)$$

Observe from the above equation that

$$\frac{\partial v_i}{\partial \dot{q}_i} = \frac{\partial r_i}{\partial q_j} \quad (6.31)$$

Next,

$$\frac{d}{dt} \left[ \frac{\partial r_i}{\partial q_j} \right] = \sum_{\ell=1}^n \frac{\partial^2 r_i}{\partial q_j \partial q_\ell} \dot{q}_\ell = \frac{\partial v_i}{\partial q_j} \quad (6.32)$$

where the last equality follows from (6.30). Substituting from (6.31) and (6.32) into (6.29) and noting that  $\dot{r}_i = v_i$  gives

$$\sum_{i=1}^k m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^k \left\{ \frac{d}{dt} \left[ m_i v_i^T \frac{\partial v_i}{\partial q_j} \right] - m_i v_i^T \frac{\partial v_i}{\partial q_j} \right\} \quad (6.33)$$

If we define the *kinetic energy*  $K$  to be the quantity

$$K = \sum_{i=1}^k \frac{1}{2} m_i v_i^T v_i \quad (6.34)$$

then the sum above can be compactly expressed as

$$\sum_{i=1}^k m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \frac{d}{dt} \frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} \quad (6.35)$$

Now, substituting from (6.35) into (6.28) shows that the second summation in (6.25) is

$$\sum_{i=1}^k \dot{p}_i^T \delta r_i = \sum_{j=1}^n \left\{ \frac{d}{dt} \frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} \right\} \delta q_j \quad (6.36)$$

Finally, combining (6.36) and (6.26) gives

$$\sum_{j=1}^n \left\{ \frac{d}{dt} \frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} - \psi_j \right\} \delta q_j = 0 \quad (6.37)$$

Now, since the virtual displacements  $\delta q_j$  are independent, we can conclude that each coefficient in (6.37) is zero, that is, that

$$\frac{d}{dt} \frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} = \psi_j, \quad j = 1, \dots, n \quad (6.38)$$

If the generalized force  $\psi_j$  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  $\tau_j$  and a potential energy function  $P(q)$  such that

$$\psi_j = -\frac{\partial P}{\partial q_j} + \tau_j \quad (6.39)$$

Then (6.38) can be written in the form

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j \quad (6.40)$$

where  $\mathcal{L} = K - P$  is the Lagrangian and we have recovered the **Euler-Lagrange equations of motion** as in Equation (6.11).

## 6.2 GENERAL EXPRESSIONS FOR 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 energy 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 therefore 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 rigid robot 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 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 6.4 we attach a coordinate frame at the center of mass (called the *body attached frame*) as shown. The kinetic

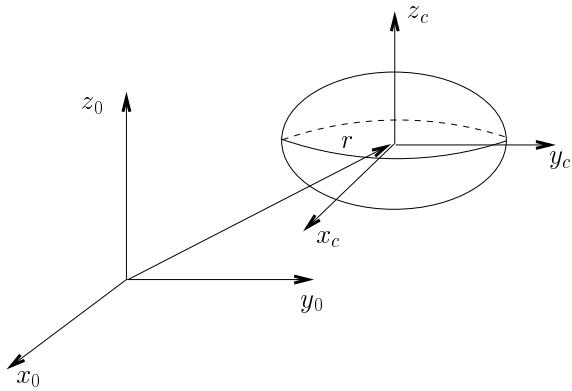


Fig. 6.4 A General Rigid Body

energy of the rigid body is then given as

$$\mathcal{K} = \frac{1}{2}mv^T v + \frac{1}{2}\omega^T \mathcal{I}\omega \quad (6.41)$$

where  $m$  is the total mass of the object,  $v$  and  $\omega$  are the linear and angular velocity vectors, respectively, and  $\mathcal{I}$  is a symmetric  $3 \times 3$  matrix called the **Inertia Tensor**.

### 6.2.1 The Inertia Tensor

It is understood that the linear and angular velocity vectors,  $v$  and  $\omega$ , respectively, in the above expression for the kinetic energy are expressed in the inertial frame. In this case we know that  $\omega$  is found from the skew symmetric matrix

$$S(\omega) = \dot{R}R^T \quad (6.42)$$

where  $R$  is the orientation transformation between 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

$$\mathcal{I} = RIR^T \quad (6.43)$$

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

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \quad (6.44)$$

where

$$\begin{aligned} I_{xx} &= \int \int \int (y^2 + z^2) \rho(x, y, z) dx dy dz \\ I_{yy} &= \int \int \int (x^2 + z^2) \rho(x, y, z) dx dy dz \\ I_{zz} &= \int \int \int (x^2 + y^2) \rho(x, y, z) dx dy dz \\ I_{xy} = I_{yx} &= - \int \int \int xy \rho(x, y, z) dx dy dz \\ I_{xz} = I_{zx} &= - \int \int \int xz \rho(x, y, z) dx dy dz \\ I_{yz} = I_{zy} &= - \int \int \int yz \rho(x, y, z) dx dy dz \end{aligned}$$

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  $x, y, z$  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: 6.2 Uniform Rectangular Solid

Consider the rectangular solid of length,  $a$ , width,  $b$ , and height,  $c$ , shown in Figure 6.5 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

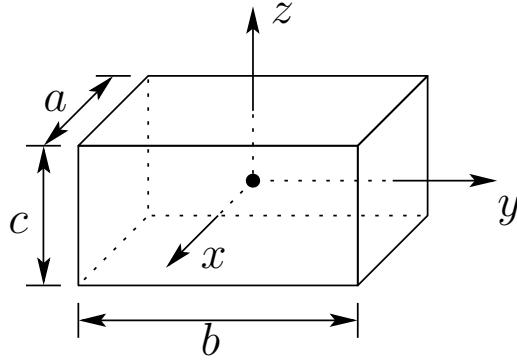


Fig. 6.5 Uniform Rectangular Solid

to compute

$$I_{xx} = \int_{-c/2}^{c/2} \int_{-b/2}^{b/2} \int_{-a/2}^{a/2} (y^2 + z^2) \rho(x, y, z) dx dy dz = \rho \frac{abc}{12} (b^2 + c^2)$$

Likewise

$$I_{yy} = \rho \frac{abc}{12} (a^2 + c^2) \quad ; \quad I_{zz} = \rho \frac{abc}{12} (a^2 + b^2)$$

and the cross products of inertia are zero.

### 6.2.2 Kinetic Energy for an $n$ -Link Robot

Now consider a manipulator consisting of  $n$  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 derivative of the joint variables. Since in our case the joint variables are indeed the generalized coordinates, it follows that, for appropriate Jacobian matrices  $J_{v_i}$  and  $J_{\omega_i}$ , we have that

$$v_i = J_{v_i}(q)\dot{q}, \quad \omega_i = J_{\omega_i}(q)\dot{q} \quad (6.45)$$

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 (6.41) it follows that the overall kinetic energy of the manipulator equals

$$K = \frac{1}{2} \dot{q}^T \sum_{i=1}^n [m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q)] \dot{q} \quad (6.46)$$

In other words, the kinetic energy of the manipulator is of the form

$$K = \frac{1}{2} \dot{q}^T D(q) \dot{q} \quad (6.47)$$

where  $D(q)$  is a symmetric positive definite matrix that is in general configuration dependent. The matrix  $D$  is called the **inertia matrix**, and in Section 6.4 we will compute this matrix for several commonly occurring manipulator configurations.

### 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

$$P_i = g^T r_{ci} m_i \quad (6.48)$$

where  $g$  is 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

$$P = \sum_{i=1}^n P_i = \sum_{i=1}^n g^T r_{ci} m_i \quad (6.49)$$

In the case that the robot contains elasticity, for example, flexible joints, 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, i.e. the potential energy depends on the configuration of the robot but not on its velocity.

## 6.3 EQUATIONS OF MOTION

In this section, we specialize the Euler-Lagrange equations derived in Section 6.1 to the special case when two conditions hold: first, the kinetic energy is a quadratic function of the vector  $\dot{q}$  of the form

$$K = \frac{1}{2} \sum_{i,j}^n d_{ij}(q) \dot{q}_i \dot{q}_j := \frac{1}{2} \dot{q}^T D(q) \dot{q} \quad (6.50)$$

where the  $n \times n$  “inertia matrix”  $D(q)$  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 this condition.

The Euler-Lagrange equations for such a system can be derived as follows. Since

$$L = K - P = \frac{1}{2} \sum_{i,j} d_{ij}(q) \dot{q}_i \dot{q}_j - P(q) \quad (6.51)$$

we have that

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj} \dot{q}_j \quad (6.52)$$

and

$$\begin{aligned} \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} &= \sum_i d_{kj} \ddot{q}_j + \sum_j \frac{d}{dt} d_{kj} \dot{q}_j \\ &= \sum_j d_{kj} \ddot{q}_j + \sum_{i,j} \frac{\partial d_{kj}}{\partial q_i} \dot{q}_i \dot{q}_j \end{aligned} \quad (6.53)$$

Also

$$\frac{\partial L}{\partial q_k} = \frac{1}{2} \sum_{i,j} \frac{\partial d_{ij}}{\partial q_k} \dot{q}_i \dot{q}_j - \frac{\partial P}{\partial q_k} \quad (6.54)$$

Thus the Euler-Lagrange equations can be written

$$\sum_j d_{kj} \ddot{q}_j + \sum_{i,j} \left\{ \frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2} \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i \dot{q}_j - \frac{\partial P}{\partial q_k} = \tau_k \quad (6.55)$$

By interchanging the order of summation and taking advantage of symmetry, we can show that

$$\sum_{i,j} \left\{ \frac{\partial d_{kj}}{\partial q_i} \right\} \dot{q}_i \dot{q}_j = \frac{1}{2} \sum_{i,j} \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} \right\} \dot{q}_i \dot{q}_j \quad (6.56)$$

Hence

$$\sum_{i,j} \left\{ \frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2} \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i \dot{q}_j = \sum_{i,j} \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i \dot{q}_j \quad (6.57)$$

### Christoffel Symbols of the First Kind

$$c_{ijk} := \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \quad (6.58)$$

The terms  $c_{ijk}$  in Equation (6.58) 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

$$\phi_k = \frac{\partial P}{\partial q_k} \quad (6.59)$$

then we can write the Euler-Lagrange equations as

$$\sum_i d_{kj}(q) \ddot{q}_j + \sum_{i,j} c_{ijk}(q) \dot{q}_i \dot{q}_j + \phi_k(q) = \tau_k, \quad k = 1, \dots, n \quad (6.60)$$

In the above equation, there are three types of terms. The first involve the second derivative of the generalized coordinates. The second are quadratic terms in the first derivatives of  $q$ , where the coefficients may depend on  $q$ . These are further classified into two types. Terms involving a product of the type  $\dot{q}_i^2$  are called **centrifugal**, while those involving a product of the type  $\dot{q}_i \dot{q}_j$  where  $i \neq j$  are called **Coriolis** terms. The third type of terms are those involving only  $q$  but not its derivatives. Clearly the latter arise from differentiating the potential energy. It is common to write (6.60) in matrix form as

### Matrix Form of Euler-Lagrange Equations

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (6.61)$$

where the  $k, j$ -th element of the matrix  $C(q, \dot{q})$  is defined as

$$\begin{aligned} c_{kj} &= \sum_{i=1}^n c_{ijk}(q)\dot{q}_i \\ &= \sum_{i=1}^n \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i \end{aligned} \quad (6.62)$$

Let us examine an important special case, where the *inertia matrix is diagonal and independent of  $q$* . In this case it follows from (6.58) that all of the Christoffel symbols are zero, since each  $d_{ij}$  is a constant. Moreover, the quantity  $d_{kj}$  is nonzero if and only if  $k = j$ , so that the Equations 6.60) decouple nicely into the form

$$d_{kk}\ddot{q} - \phi_k(q) = \tau_k, \quad k = 1, \dots, n \quad (6.63)$$

In summary, the development in this section is very general and applies to any mechanical system whose kinetic energy is of the form (6.50) and whose potential energy is independent of  $\dot{q}$ . In the next section we apply this discussion to study specific robot configurations.

## 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 6.6, consisting of two links and two

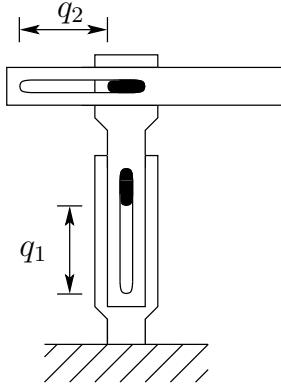


Fig. 6.6 Two-link cartesian robot.

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. Then 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 (6.50) and that the potential energy is only a function of  $q_1$  and  $q_2$ . Hence we can use the formulae 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

$$v_{c1} = J_{v_{c1}} \dot{q} \quad (6.64)$$

where

$$J_{v_{c1}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \quad (6.65)$$

Similarly,

$$v_{c2} = J_{v_{c2}} \dot{q} \quad (6.66)$$

where

$$J_{v_{c2}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (6.67)$$

Hence the kinetic energy is given by

$$K = \frac{1}{2} \dot{q}^T \{ m_1 J_{v_c}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}} \} \dot{q} \quad (6.68)$$

Comparing with (6.50), we see that the inertia matrix  $D$  is given simply by

$$D = \begin{bmatrix} m_1 + m_2 & 0 \\ 0 & m_2 \end{bmatrix} \quad (6.69)$$

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

$$P = g(m_1 + m_2) q_1 \quad (6.70)$$

Now we are ready to write down the equations of motion. Since the inertia matrix is constant, all Christoffel symbols are zero. Further, the vectors  $\phi_k$  are given by

$$\phi_1 = \frac{\partial P}{\partial q_1} = g(m_1 + m_2), \quad \phi_2 = \frac{\partial P}{\partial q_2} = 0 \quad (6.71)$$

Substituting into (6.60) gives the dynamical equations as

$$\begin{aligned} (m_1 + m_2) \ddot{q}_1 + g(m_1 + m_2) &= f_1 \\ m_2 \ddot{q}_2 &= f_2 \end{aligned} \quad (6.72)$$

### Planar Elbow Manipulator

Now consider the planar manipulator with two revolute joints shown in Figure 6.7. 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  $i$ ;  $\ell_i$  denotes the length of link  $i$ ;  $\ell_{ci}$  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 make effective use of the Jacobian expressions in Chapter 4 in computing the kinetic energy. Since we are using joint variables as the generalized coordinates, it follows that we can use the contents of Section 6.7. First,

$$v_{c1} = J_{v_{c1}} \dot{q} \quad (6.73)$$

where,

$$J_{v_{c1}} = \begin{bmatrix} -\ell_c \sin q_1 & 0 \\ \ell_{c1} \cos q_1 & 0 \\ 0 & 0 \end{bmatrix} \quad (6.74)$$

Similarly,

$$v_{c2} = J_{v_{c2}} \dot{q} \quad (6.75)$$

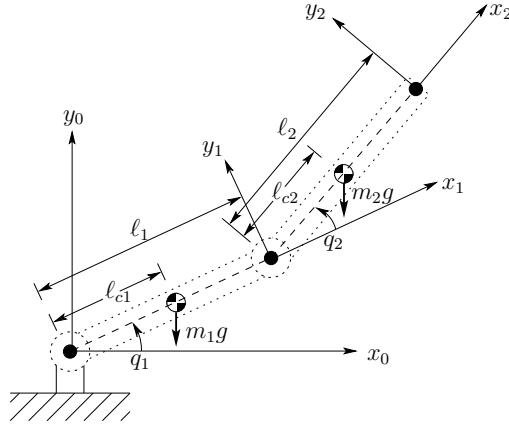


Fig. 6.7 Two-link revolute joint arm.

where

$$J_{v_{c2}} = \begin{bmatrix} -\ell_1 \sin q_1 - \ell_{c2} \sin(q_1 + q_2) & -\ell_{c2} \sin(q_1 + q_2) \\ \ell_1 \cos q_1 + \ell_{c2} \cos(q_1 + q_2) & \ell_{c2} \cos(q_1 + q_2) \\ 0 & 0 \end{bmatrix} \quad (6.76)$$

Hence the translational part of the kinetic energy is

$$\frac{1}{2}m_1 v_{c1}^T v_{c1} + \frac{1}{2}m_2 v_{c2}^T v_{c2} = \frac{1}{2}\dot{q} \{m_1 J_{v_{c1}}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}}\} \dot{q} \quad (6.77)$$

Next we deal with 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

$$\omega_1 = \dot{q}_1 k, \quad \omega_2 = (\dot{q}_1 + \dot{q}_2)k \quad (6.78)$$

when expressed in the base inertial frame. Moreover, since  $\omega_i$  is aligned with  $k$ , the triple product  $\omega_i^T I_i \omega_i$  reduces simply to  $(I_{33})_i$  times the square of the magnitude of the angular velocity. This quantity  $(I_{33})_i$  is indeed what we have labeled as  $I_i$  above. Hence the rotational kinetic energy of the overall system is

$$\frac{1}{2}\dot{q}^T \left\{ I_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + I_2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right\} \dot{q} \quad (6.79)$$

Now we are ready to form the inertia matrix  $D(q)$ . For this purpose, we merely have to add the two matrices in (6.77) and (6.79), respectively. Thus

$$D(q) = m_1 J_{v_{c1}}^T J_{v_{c1}} + m_2 J_{v_{c2}}^T J_{v_{c2}} + \begin{bmatrix} I_1 + I_2 & I_2 \\ I_2 & I_2 \end{bmatrix} \quad (6.80)$$

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

$$\begin{aligned} d_{11} &= m_1 \ell_{c1}^2 + m_2 (\ell_1^2 + \ell_{c2}^2 + 2\ell_1 \ell_{c2} + 2\ell_1 \ell_{c2} \cos q_2) + I_1 + I_2 \\ d_{12} &= d_{21} = m_2 (\ell_{c2}^2 + \ell_1 \ell_{c2} \cos q_2) + I_2 \\ d_{22} &= m_2 \ell_{c2}^2 + I_2 \end{aligned} \quad (6.81)$$

Now we can compute the Christoffel symbols using the definition (6.58). This gives

$$\begin{aligned} c_{111} &= \frac{1}{2} \frac{\partial d_{11}}{\partial q_1} = 0 \\ c_{121} &= c_{211} = \frac{1}{2} \frac{\partial d_{11}}{\partial q_2} = -m_2 \ell_1 \ell_{c2} \sin q_2 =: h \\ c_{221} &= \frac{\partial d_{12}}{\partial q_2} - \frac{1}{2} \frac{\partial d_{22}}{\partial q_1} = h \\ c_{112} &= \frac{\partial d_{21}}{\partial q_1} - \frac{1}{2} \frac{\partial d_{11}}{\partial q_2} = -h \\ c_{122} &= c_{212} = \frac{1}{2} \frac{\partial d_{22}}{\partial q_1} = 0 \\ c_{222} &= \frac{1}{2} \frac{\partial d_{22}}{\partial q_2} = 0 \end{aligned} \quad (6.82)$$

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

$$\begin{aligned} P_1 &= m_1 g \ell_{c1} \sin q_1 \\ P_2 &= m_2 g (\ell_1 \sin q_1 + \ell_{c2} \sin(q_1 + q_2)) \\ P &= P_1 + P_2 = (m_1 \ell_{c1} + m_2 \ell_1) g \sin q_1 + m_2 \ell_{c2} g \sin(q_1 + q_2) \end{aligned} \quad (6.83)$$

Hence, the functions  $\phi_k$  defined in (6.59) become

$$\phi_1 = \frac{\partial P}{\partial q_1} = (m_1 \ell_{c1} + m_2 \ell_1) g \cos q_1 + m_2 \ell_{c2} g \cos(q_1 + q_2) \quad (6.84)$$

$$\phi_2 = \frac{\partial P}{\partial q_2} = m_2 \ell_{c2} g \cos(q_1 + q_2) \quad (6.85)$$

Finally we can write down the dynamical equations of the system as in (6.60). Substituting for the various quantities in this equation and omitting zero terms leads to

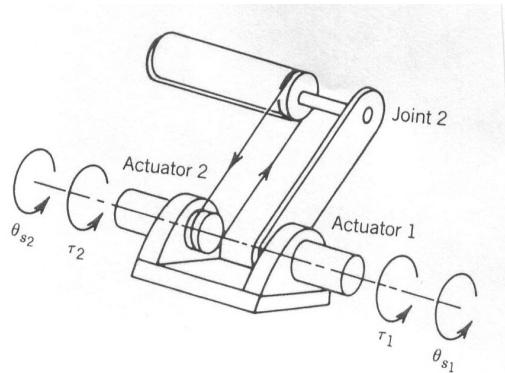
$$\begin{aligned} d_{11} \ddot{q}_1 + d_{12} \ddot{q}_2 + c_{121} \dot{q}_1 \dot{q}_2 + c_{211} \dot{q}_2 \dot{q}_1 + c_{221} \dot{q}_2^2 + \phi_1 &= \tau_1 \\ d_{21} \ddot{q}_1 + d_{22} \ddot{q}_2 + c_{112} \dot{q}_1^2 + \phi_2 &= \tau_2 \end{aligned} \quad (6.86)$$

In this case the matrix  $C(q, \dot{q})$  is given as

$$C = \begin{bmatrix} h \dot{q}_2 & h \dot{q}_2 + h \dot{q}_1 \\ -h \dot{q}_1 & 0 \end{bmatrix} \quad (6.87)$$

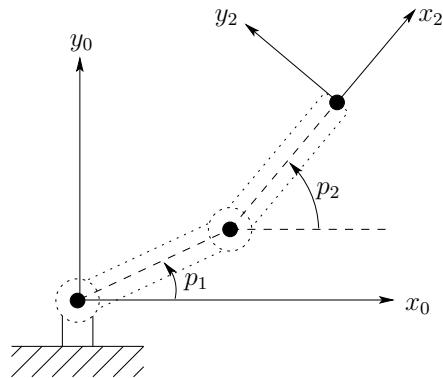
### Planar Elbow Manipulator with Remotely Driven Link

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 6.8). In this case one should choose the generalized coordinates



*Fig. 6.8* Two-link revolute joint arm with remotely driven link.

as shown in Figure 6.9, because the angle  $p_2$  is determined by driving motor



*Fig. 6.9* Generalized coordinates for robot of Figure 6.4.

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

$$\begin{aligned} v_{c1} &= \begin{bmatrix} -\ell_{c1} \sin p_1 \\ \ell_{c1} \cos p_1 \\ 0 \end{bmatrix} \dot{p}_1, \quad v_{c2} = \begin{bmatrix} \ell_1 \sin p_1 & -\ell_{c2} \sin p_2 \\ \ell_1 \cos p_1 & \ell_{c2} \cos p_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} \end{aligned} \quad (6.88)$$

$$\omega_1 = \dot{p}_1 k, \quad \omega_2 = \dot{p}_2 k \quad (6.89)$$

Hence the kinetic energy of the manipulator equals

$$K = \frac{1}{2} \dot{p}^T D(p) \dot{p} \quad (6.90)$$

where

$$D(p) = \begin{bmatrix} m_1 \ell_{c1}^2 + m_2 \ell_1^2 + I_1 & m_2 \ell_1 \ell_{c2} \cos(p_2 - p_1) \\ m_2 \ell_1 \ell_{c2} \cos(p_2 - p_1) & m_2 \ell_{c2}^2 + I_2 \end{bmatrix} \quad (6.91)$$

Computing the Christoffel symbols as in (6.58) gives

$$\begin{aligned} c_{111} &= \frac{1}{2} \frac{\partial d_{11}}{\partial p_1} = 0 \\ c_{121} &= c_{211} = \frac{1}{2} \frac{\partial d_{11}}{\partial p_2} = 0 \\ c_{221} &= \frac{\partial d_{12}}{\partial p_2} - \frac{1}{2} \frac{\partial d_{22}}{\partial p_1} = -m_2 \ell_1 \ell_{c2} \sin(p_2 - p_1) \\ c_{112} &= \frac{\partial d_{21}}{\partial p_1} - \frac{1}{2} \frac{\partial d_{11}}{\partial p_2} = m_2 \ell_1 \ell_{c2} \sin(p_2 - p_1) \\ c_{212} &= c_{122} = \frac{1}{2} \frac{\partial d_{22}}{\partial p_1} = 0 \\ c_{222} &= \frac{1}{2} \frac{\partial d_{22}}{\partial p_2} = 0 \end{aligned} \quad (6.92)$$

Next, the potential energy of the manipulator, in terms of  $p_1$  and  $p_2$ , equals

$$P = m_1 g \ell_{c1} \sin p_1 + m_2 g (\ell_1 \sin p_1 + \ell_{c2} \sin p_2) \quad (6.93)$$

Hence the gravitational generalized forces are

$$\begin{aligned} \phi_1 &= (m_1 \ell_{c1} + m_2 \ell_1) g \cos p_1 \\ \phi_2 &= m_2 \ell_{c2} g \cos p_2 \end{aligned}$$

Finally, the equations of motion are

$$\begin{aligned} d_{11} \ddot{p}_1 + d_{12} \ddot{p}_2 + c_{221} \dot{p}_2^2 + \phi_1 &= \tau_1 \\ d_{21} \ddot{p}_1 + d_{22} \ddot{p}_2 + c_{112} \dot{p}_1^2 + \phi_2 &= \tau_2 \end{aligned} \quad (6.94)$$

Comparing (6.94) and (6.86), 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.

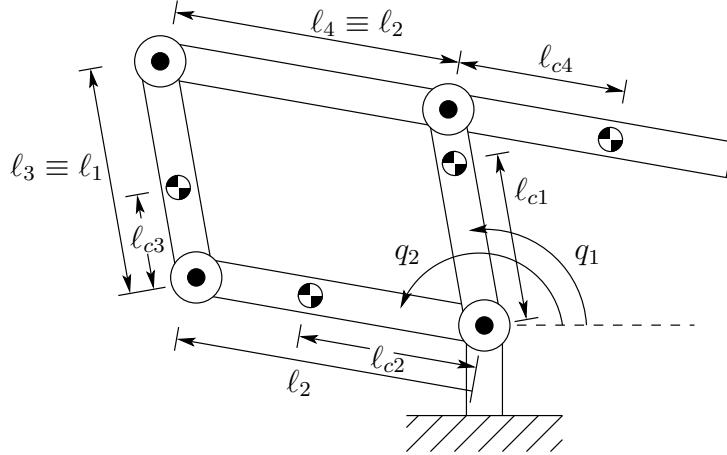


Fig. 6.10 Five-bar linkage

### Five-Bar Linkage

Now consider the manipulator shown in Figure 6.10. 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 6.10 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. In Figure 6.10, it is assumed that the lengths of links 3 and 4 are the same, and that the two lengths marked  $\ell_2$  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  $\ell_{c1}$ , and  $\ell_{c3}$  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 generalized coordinates. This gives

$$\begin{bmatrix} x_{c1} \\ y_{c1} \end{bmatrix} = \begin{bmatrix} \ell_{c1} \cos q_1 \\ \ell_{c1} \sin q_1 \end{bmatrix} \quad (6.95)$$

$$\begin{bmatrix} x_{c2} \\ y_{c2} \end{bmatrix} = \begin{bmatrix} \ell_{c2} \cos q_2 \\ \ell_{c2} \sin q_2 \end{bmatrix} \quad (6.96)$$

$$\begin{bmatrix} x_{c3} \\ y_{c3} \end{bmatrix} = \begin{bmatrix} \ell_{c2} \cos q_1 \\ \ell_{c2} \sin q_2 \end{bmatrix} + \begin{bmatrix} \ell_{c3} \cos q_1 \\ \ell_{c3} \sin q_1 \end{bmatrix} \quad (6.97)$$

$$\begin{aligned} \begin{bmatrix} x_{c4} \\ y_{c4} \end{bmatrix} &= \begin{bmatrix} \ell_1 \cos q_1 \\ \ell_1 \sin q_1 \end{bmatrix} + \begin{bmatrix} \ell_{c4} \cos(q_2 - \pi) \\ \ell_{c4} \sin(q_2 - \pi) \end{bmatrix} \\ &= \begin{bmatrix} \ell_1 \cos q_1 \\ \ell_1 \sin q_1 \end{bmatrix} - \begin{bmatrix} \ell_{c4} \cos q_2 \\ \ell_{c4} \sin q_2 \end{bmatrix} \end{aligned} \quad (6.98)$$

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

$$\begin{aligned} v_{c1} &= \begin{bmatrix} -\ell_{c1} \sin q_1 & 0 \\ \ell_{c1} \cos q_1 & 0 \end{bmatrix} \dot{q} \\ v_{c2} &= \begin{bmatrix} 0 & -\ell_{c2} \sin q_2 \\ 0 & \ell_{c2} \cos q_2 \end{bmatrix} \dot{q} \\ v_{c3} &= \begin{bmatrix} -\ell_{c3} \sin q_1 & -\ell_2 \sin q_2 \\ \ell_{c3} \cos q_1 & \ell_2 \cos q_2 \end{bmatrix} \dot{q} \\ v_{c4} &= \begin{bmatrix} -\ell_1 \sin q_1 & \ell_{c4} \sin q_2 \\ \ell_1 \cos q_1 & \ell_{c4} \cos q_2 \end{bmatrix} \dot{q} \end{aligned} \quad (6.99)$$

Let us define the velocity Jacobians  $J_{v_{ci}}$ ,  $i \in \{1, \dots, 4\}$  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

$$\omega_1 = \omega_3 = q_1 k, \omega_2 = \omega_4 = \dot{q}_2 k. \quad (6.100)$$

Thus the inertia matrix is given by

$$D(q) = \sum_{i=1}^4 m_i J_{vc}^T J_{vc} + \begin{bmatrix} I_1 + I_3 & 0 \\ 0 & I_2 + I_4 \end{bmatrix} \quad (6.101)$$

If we now substitute from (6.99) into the above equation and use the standard trigonometric identities, when the dust settles we are left with

$$\begin{aligned} d_{11}(q) &= m_1 \ell_{c1}^2 + m_3 \ell_{c3}^2 + m_4 \ell_1^2 + I_1 + I_3 \\ d_{12}(q) &= d_{21}(q) = (m_3 \ell_2 \ell_{c3} - m_4 \ell_1 \ell_{c4}) \cos(q_2 - q_1) \\ d_{22}(q) &= m_2 \ell_{c2}^2 + m_3 \ell_2^2 + m_4 \ell_{c4}^2 + I_2 + I_4 \end{aligned} \quad (6.102)$$

Now we note from the above expressions that if

$$m_3\ell_2\ell_{c3} = m_4\ell_1\ell_{c4} \quad (6.103)$$

then  $d_{12}$  and  $d_{21}$  are zero, i.e. 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

$$\begin{aligned} P &= g \sum_{i=1}^4 y_{ci} \\ &= g \sin q_1(m_1\ell_{c1} + m_3\ell_{c3} + m_4\ell_1) \\ &\quad + g \sin q_2(m_2\ell_{c2} + m_3\ell_2 - m_4\ell_{c4}) \end{aligned} \quad (6.104)$$

Hence

$$\begin{aligned} \phi_1 &= g \cos q_1(m_1\ell_{c1} + m_3\ell_{c3} + m_4\ell_1) \\ \phi_2 &= g \cos q_2(m_2\ell_{c2} + m_3\ell_2 - m_4\ell_{c4}) \end{aligned} \quad (6.105)$$

Notice that  $\phi_1$  depends only on  $q_1$  but not on  $q_2$  and similarly that  $\phi_2$  depends only on  $q_2$  but not on  $q_1$ . Hence, if the relationship (6.103) is satisfied, then the rather complex-looking manipulator in Figure 6.10 is described by the decoupled set of equations

$$d_{11}\ddot{q}_1 + \phi_1(q_1) = \tau_1, \quad d_{22}\ddot{q}_2 + \phi_2(q_2) = \tau_2 \quad (6.106)$$

This discussion helps to explain the popularity of the parallelogram configuration in industrial robots. If the relationship (6.103) 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 which can be exploited to good advantage in particular 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 The Skew Symmetry and Passivity Properties

The **Skew Symmetry** property refers to an important relationship between the inertia matrix  $D(q)$  and the matrix  $C(q, \dot{q})$  appearing in (6.61) that will be of fundamental importance for the problem of manipulator control considered in later chapters.

**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 (6.62). 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  $kj$ -th component of  $\dot{D}(q)$  is given by the chain rule as

$$\dot{d}_{kj} = \sum_{i=1}^n \frac{\partial d_{kj}}{\partial q_i} \dot{q}_i \quad (6.107)$$

Therefore, the  $kj$ -th component of  $N = \dot{D} - 2C$  is given by

$$\begin{aligned} n_{kj} &= \dot{d}_{kj} - 2c_{kj} \\ &= \sum_{i=1}^n \left[ \frac{\partial d_{kj}}{\partial q_i} - \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \right] \dot{q}_i \\ &= \sum_{i=1}^n \left[ \frac{\partial d_{ij}}{\partial q_k} - \frac{\partial d_{ki}}{\partial q_j} \right] \dot{q}_i \end{aligned} \quad (6.108)$$

Since the inertia matrix  $D(q)$  is symmetric, that is,  $d_{ij} = d_{ji}$ , it follows from (6.108) by interchanging the indices  $k$  and  $j$  that

$$n_{jk} = -n_{kj} \quad (6.109)$$

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 (6.62). 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

**The Passivity Property**

$$\int_0^T \dot{q}^T(\zeta) \tau(\zeta) d\zeta \geq -\beta, \quad \forall T > 0 \quad (6.110)$$

The term  $\dot{q}^T \tau$  has units of power and therefore the expression  $\int_0^T \dot{q}^T(\zeta) \tau(\zeta) d\zeta$  is the energy produced by the system over the time interval  $[0, T]$ . Passivity

therefore 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, i.e., the sum of the kinetic and potential energies,

$$H = \frac{1}{2}\dot{q}^T D(q)\dot{q} + P(q) \quad (6.111)$$

Then, the derivative  $\dot{H}$  satisfies

$$\begin{aligned} \dot{H} &= \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T \frac{\partial P}{\partial q} \\ &= \dot{q}^T \{\tau - C(q, \dot{q}) - g(q)\} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T \frac{\partial P}{\partial q} \end{aligned} \quad (6.112)$$

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

$$\begin{aligned} \dot{H} &= \dot{q}^T \tau + \frac{1}{2}\dot{q}^T \{\dot{D}(q) - 2C(q, \dot{q})\}\dot{q} \\ &= \dot{q}^T \tau \end{aligned} \quad (6.113)$$

the latter equality following from the skew-symmetry property. Integrating both sides of (6.113) with respect to time gives,

$$\int_0^T \dot{q}^T(\zeta) \tau(\zeta) d\zeta = H(T) - H(0) \geq -H(0) \quad (6.114)$$

since the total energy  $H(T)$  is non-negative, 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  $0 < \lambda_1(q) \leq \dots \leq \lambda_n(q)$  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

#### Bounds on the Inertia Matrix

$$\lambda_1(q)I_{n \times n} \leq D(q) \leq \lambda_n(q)I_{n \times n} \quad (6.115)$$

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 bounded functions of the joint variables, i.e., terms containing sine and cosine functions. As a result one can find constants  $\lambda_m$  and  $\lambda_M$  that provide uniform (independent of  $q$ ) bounds in the inertia matrix

$$\lambda_m I_{n \times n} \leq D(q) \leq \lambda_M I_{n \times n} < \infty \quad (6.116)$$

### 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,  $Y(q, \dot{q}, \ddot{q})$ , which we assume is completely known, and an  $\ell$ -dimensional vector  $\Theta$  such that the Euler-Lagrange equations can be written

#### The Linearity in the Parameters Property

$$D(q) + C(q, \dot{q})\dot{q} + g(q) =: Y(q, \dot{q}, \ddot{q})\Theta = \tau \quad (6.117)$$

The function,  $Y(q, \dot{q}, \ddot{q})$ , is called the **Regressor** and  $\Theta$  is the **Parameter** vector. The dimension of the parameter space,  $\mathbb{R}^\ell$ , i.e., 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 parametrize the dynamic equations is, however, difficult in general.

#### Example: 6.3 Two Link Planar Robot

Consider the two link, revolute joint, planar robot from section 6.4 above. If we group the inertia terms appearing in Equation 6.81 as

$$\Theta_1 = m_1 \ell_{c1}^2 + m_2 (\ell_1^2 + \ell_{c2}^2) + I_1 + I_2 \quad (6.118)$$

$$\Theta_2 = m_2 \ell_1 \ell_{c2} \quad (6.119)$$

$$\Theta_3 = m_2 \ell_1 \ell_{c2} \quad (6.120)$$

then we can write the inertia matrix elements as

$$d_{11} = \Theta_1 + 2\Theta_2 \cos(q_2) \quad (6.121)$$

$$d_{12} = d_{21} = \Theta_3 + \Theta_2 \cos(q_2) \quad (6.122)$$

$$d_{22} = \Theta_3 \quad (6.123)$$

No additional parameters are required in the Christoffel symbols as these are functions of the elements of the inertia matrix. The gravitational torques require additional parameters, in general. Setting

$$\Theta_4 = m_1\ell_{c1} + m_2\ell_1 \quad (6.124)$$

$$\Theta_5 = m_2\ell_2 \quad (6.125)$$

we can write the gravitational terms,  $\phi_1$  and  $\phi_2$  as

$$\phi_1 = \Theta_4 g \cos(q_1) + \Theta_5 g \cos(q_1 + q_2) \quad (6.126)$$

$$\phi_2 = \Theta_5 \cos(q_1 + q_2) \quad (6.127)$$

Substituting these into the equations of motion it is straightforward to write the dynamics in the form (6.117) where

$$Y(q, \dot{q}, \ddot{q}) = \begin{bmatrix} \ddot{q}_1 & \cos(q_2)(2\ddot{q}_1 + \ddot{q}_2) + \sin(q_2)(\dot{q}_1^2 - 2\dot{q}_1\dot{q}_2) & \ddot{q}_2 & g \cos(q_1) & g \cos(q_1 + q_2) \\ 0 & \cos(q_2)\ddot{q}_1 + \sin(q_2)\dot{q}_1^2 & \ddot{q}_2 & 0 & g \cos(q_1 + q_2) \end{bmatrix} \quad (6.128)$$

and the parameter vector  $\Theta$  is given by

$$\Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \\ \Theta_4 \\ \Theta_5 \end{bmatrix} = \begin{bmatrix} m_1\ell_{c1}^2 + m_2(\ell_{c1}^2 + \ell_{c2}^2) + I_1 + I_2 \\ m_2\ell_1\ell_{c2} \\ m_2\ell_1\ell_{c2} \\ m_1\ell_{c1} + m_2\ell_1 \\ m_2\ell_2 \end{bmatrix} \quad (6.129)$$

Thus, we have parameterized the dynamics using a five dimensional parameter space. Note that in the absence of gravity, as in a SCARA configuration, 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, and the answer is not clear. 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. Thus 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  $\tau$ ). Analyzing the dynamics of a system means finding the relationship between  $q$  and  $\tau$ . 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 (6.87) for example. 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 clearly 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 (6.86).

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  $\tau$  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

$$\frac{d(mv)}{dt} = f \quad (6.130)$$

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, (6.130) can be simplified to the familiar relationship

$$ma = f \quad (6.131)$$

where  $a = \dot{v}$  is the acceleration of the center of mass.

Applying the third fact to the angular motion of a body gives

$$\frac{d(I_0\omega_0)}{dt} = \tau_0 \quad (6.132)$$

where  $I_0$  is the moment of inertia of the body about an inertial frame whose origin is at the center of mass,  $\omega_0$  is the angular velocity of the body, and  $\tau_0$  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

$$I_0 = RIR^T \quad (6.133)$$

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

$$I\dot{\omega} + \omega \times (I\omega) = \tau \quad (6.134)$$

where  $I$  is the (constant) inertia matrix of the body with respect to the body attached frame,  $\omega$  is the angular velocity, but expressed in the body attached frame, and  $\tau$  is the total torque on the body, again expressed in the body attached frame. Let us now give a derivation of (6.134) 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 (6.133) gives the relation between  $I$  and  $I_0$ . Now by the definition of the angular velocity, we know that

$$\dot{R}R^T = S(\omega_0) \quad (6.135)$$

In other words, the angular velocity of the body, expressed in an inertial frame, is given by (6.135). Of course, the same vector, expressed in the body attached frame, is given by

$$\omega_0 = R\omega, \omega = R^T\omega_0 \quad (6.136)$$

Hence the angular momentum, expressed in the inertial frame, is

$$h = RIR^T R\omega = RI\omega \quad (6.137)$$

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:

$$\dot{h} = \dot{R}I\omega + RI\dot{\omega} \quad (6.138)$$

Now

$$S(\omega_0) = \dot{R}R^T, \quad \dot{R} = S(\omega)R \quad (6.139)$$

Hence, with respect to the inertial frame,

$$\dot{h} = S(\omega_0)RI\omega + RI\dot{\omega} \quad (6.140)$$

With respect to the frame rigidly attached to the body, the rate of change of the angular momentum is

$$\begin{aligned} R^T\dot{h} &= R^TS(\omega_0)RI\omega + I\dot{\omega} \\ &= S(R^T\omega_0)I\omega + I\dot{\omega} \\ &= S(\omega)I\omega + I\dot{\omega} = \omega \times (I\omega) + I\dot{\omega} \end{aligned} \quad (6.141)$$

This establishes (6.134). Of course we can, if we wish, 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  $0, \dots, n$ , 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 pertain to the velocities and accelerations of various parts of the manipulator.

- $a_{c,i}$  = the acceleration of the center of mass of link  $i$
- $a_{e,i}$  = the acceleration of the end of link  $i$  (i.e., joint  $i+1$ )
- $\omega_i$  = the angular velocity of frame  $i$  w.r.t. frame 0
- $\alpha_i$  = the angular acceleration of frame  $i$  w.r.t. frame 0

The next several vectors pertain to forces and torques.

- $g_i$  = the acceleration due to gravity (expressed in frame  $i$ )
- $f_i$  = the force exerted by link  $i - 1$  on link  $i$
- $\tau_i$  = the torque exerted by link  $i - 1$  on link  $i$
- $R_i^{i+1}$  = the rotation matrix from frame  $i + 1$  to frame  $i$

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.

- $m_i$  = the mass of link  $i$
- $I_i$  = the inertia matrix of link  $i$  about a frame parallel to frame  $i$  whose origin is at the center of mass of link  $i$
- $r_{i,ci}$  = the vector from joint  $i$  to the center of mass of link  $i$
- $r_{i+1,ci}$  = the vector from joint  $i + 1$  to the center of mass of link  $i$
- $r_{i,i+1}$  = the vector from joint  $i$  to joint  $i + 1$

Now consider the free body diagram shown in Figure 6.11; this shows link  $i$

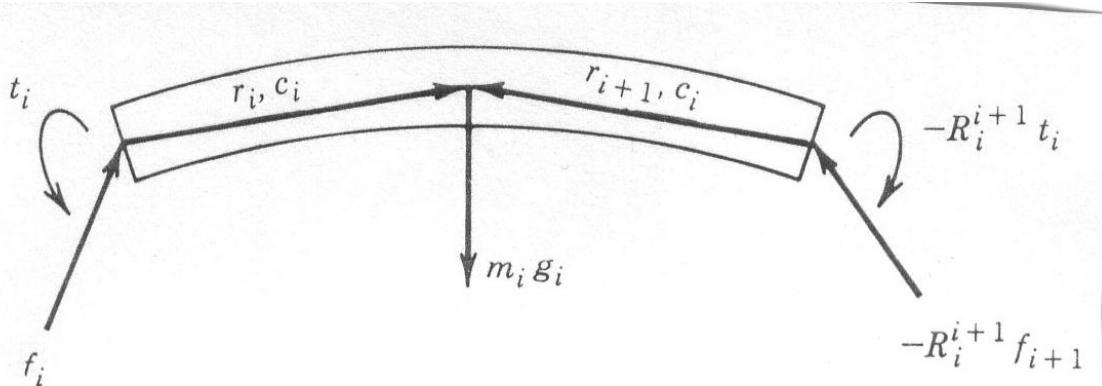


Fig. 6.11 Forces and moments on link  $i$

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  $t_i$  and  $-R_i^{i+1} \tau_{i+1}$ . The force  $m_i g_i$  is the gravitational force. Since all vectors in Figure 6.11 are expressed in frame  $i$ , the gravity vector  $g_i$  is in general a function of  $i$ .

Writing down the force balance equation for link  $i$  gives

$$f_i - R_i^{i+1} f_{i+1} + m_i g_i = m_i a_{c,i} \quad (6.142)$$

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

$$\begin{aligned} \tau_i - R_i^{i+1} \tau_{i+1} + f_i \times r_{i,ci} - (R_i^{i+1} f_{i+1}) \times r_{i+1,ci} \\ = \alpha_i + \omega_i \times (I_i \omega_i) \end{aligned} \quad (6.143)$$

Now we present the heart of the Newton-Euler formulation, which consists of finding the vectors  $f_1, \dots, f_n$  and  $\tau_1, \dots, \tau_n$  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  $\tau$  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  $a_{c,i}$ ,  $\omega_i$  and  $\alpha_i$ . Then we can solve (6.142) and (6.143) 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 (6.142) to obtain

$$f_i = R_i^{i+1} f_{i+1} + m_i a_{c,i} - m_i g_i \quad (6.144)$$

By successively substituting  $i = n, n-1, \dots, 1$  we find all forces. Similarly, we can solve (6.143) to obtain

$$\begin{aligned} \tau_i &= \\ R_i^{i+1} \tau_{i+1} - f_i \times r_{i,ci} + (R_i^{i+1} f_{i+1}) \times r_{i+1,ci} + \alpha_i + \omega_i \times (I_i \omega_i) \end{aligned} \quad (6.145)$$

By successively substituting  $i = n, n-1, \dots, 1$  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  $\alpha_i$ . 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,  $\omega_i$  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 from Section ?? we have that

$$\omega_i^{(0)} = \omega_{i-1}^{(0)} + z_{i-1}\dot{q}_i \quad (6.146)$$

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  $\omega_i$  and  $\omega_{i-1}$ , we need only express the above equation in frame  $i$  rather than the base frame, taking care to account for the fact that  $\omega_i$  and  $\omega_{i-1}$  are expressed in different frames. This leads to

$$\omega_i = (R_{i-1}^i)^T \omega_{i-1} + b_i \dot{q}_i \quad (6.147)$$

where

$$b_i = (R_0^i)^T z_{i-1} \quad (6.148)$$

is the axis of rotation of joint  $i$  expressed in frame  $i$ .

Next let us work on the angular acceleration  $\alpha_i$ . It is vitally important to note here that

$$\alpha_i = (R_0^i)^T \dot{\omega}_i^{(0)} \quad (6.149)$$

In other words,  $\alpha_i$  is the derivative of the angular velocity of frame  $i$ , but 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 (6.146) that

$$\dot{\omega}_i^{(0)} = \dot{\omega}_{i-1}^{(0)} + z_{i-1}\ddot{q}_i + \omega_i^{(0)} \times z_{i-1}\dot{q}_i \quad (6.150)$$

Expressing the same equation in frame  $i$  gives

$$\alpha_i = (R_{i-1}^i)^T \alpha_{i-1} + b_i \ddot{q}_i + \omega_i \times b_i \dot{q}_i \quad (6.151)$$

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 ??, we get that the velocity of the center of mass of link  $i$  is given by

$$v_{c,i}^{(0)} = v_{e,i-1}^{(0)} + \omega_i^{(0)} \times r_{i,ci}^{(0)} \quad (6.152)$$

To obtain an expression for the acceleration, we use (??), and note that the vector  $r_{i,ci}^{(0)}$  is constant in frame  $i$ . Thus

$$a_{c,i}^{(0)} = a_{e,i-1}^{(0)} \times r_{i,ci}^{(0)} + \omega_i^{(0)} \times (\omega_i^{(0)} \times r_{i,ci}^{(0)}) \quad (6.153)$$

Now

$$a_{c,i} = (R_0^i)^T a_{c,i}^{(0)} \quad (6.154)$$

Let us carry out the multiplication and use the familiar property

$$R(a \times b) = (Ra) \times (Rb) \quad (6.155)$$

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

$$a_{c,i} = (R_{i-1}^i)^T a_{e,i-1} + \dot{\omega}_i \times r_{i,ci} + \omega_i \times (\omega_i \times r_{i,ci}) \quad (6.156)$$

Now to find the acceleration of the end of link  $i$ , we can use (6.156) with  $r_{i,i+1}$  replacing  $r_{i,ci}$ . Thus

$$a_{e,i} = (R_{i-1}^i)^T a_{e,i-1} + \dot{\omega}_i \times r_{i,i+1} + \omega_i \times (\omega_i \times r_{i,i+1}) \quad (6.157)$$

Now the recursive formulation is complete. We can now state the Newton-Euler formulation as follows.

1. Start with the initial conditions

$$\omega_0 = 0, \alpha_0 = 0, a_{c,0} = 0, a_{e,0} = 0 \quad (6.158)$$

and solve (6.147), (6.151), (6.157) and (6.156) (in that order!) to compute  $\omega_i$ ,  $\alpha_i$  and  $a_{c,i}$  for  $i$  increasing from 1 to  $n$ .

2. Start with the terminal conditions

$$f_{n+1} = 0, \quad \tau_{n+1} = 0 \quad (6.159)$$

and use (6.144) and (6.145) to compute  $f_i$  and  $\tau_i$  for  $i$  decreasing from  $n$  to 1.

## 6.7 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 6.8, and show that the Newton-Euler method leads to the same equations as the Lagrangian method, namely (6.86).

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

$$\omega_1 = \dot{q}_1 k, \quad \alpha_1 = \ddot{q}_1 k, \quad \omega_2 = (q_1 + q_2)k, \quad \alpha_2 = (\ddot{q}_1 + \ddot{q}_2)k \quad (6.160)$$

so that there is no need to use (6.147) and (6.151). Also, the vectors that are independent of the configuration are as follows:

$$r_{1,c1} = \ell_{c1} i, \quad r_{2,c1} = (\ell_1 - \ell_{c1})i, \quad r_{1,2} = \ell_1 i \quad (6.161)$$

$$r_{2,c2} = \ell_{c2} i, \quad r_{3,c2} = (\ell_2 - \ell_{c2})i, \quad r_{2,3} = \ell_2 i \quad (6.162)$$

### Forward Recursion link 1

Using (6.156) with  $i = 1$  and noting that  $a_{e,0} = 0$  gives

$$\begin{aligned} a_{c,1} &= \ddot{q}_1 k \times \ell_{c1} i + \dot{q}_1 k \times (\dot{q}_1 k \times \ell_{c1} i) \\ &= \ell_{c1} \ddot{q}_1 j - \ell_{c1} \dot{q}_1^2 i = \begin{bmatrix} -\ell_{c1} \dot{q}_1^2 \\ \ell_{c1} \ddot{q}_1 \\ 0 \end{bmatrix} \end{aligned} \quad (6.163)$$

Notice how simple this computation is when we do it with respect to frame 1. Compare with the same computation in frame 0! Finally, we have

$$g_1 = -(R_0^1)^T g j = g \begin{bmatrix} \sin q_1 \\ -\cos q_1 \\ 0 \end{bmatrix} \quad (6.164)$$

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 obviously always 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. Clearly, this is obtained from (6.163) by replacing  $\ell_{c1}$  by  $\ell_1$ . Thus

$$a_{e,1} = \begin{bmatrix} -\ell_1 \dot{q}_1^2 \\ \ell_1 \ddot{q}_1 \end{bmatrix} \quad (6.165)$$

### Forward Recursion: Link 2

Once again we use (6.156) and substitute for  $(o_2$  from (6.160); this yields

$$\alpha_{c,2} = (R_1^2)^T a_{e,1} + [(\ddot{q}_1 + \ddot{q}_2)k] \times \ell_{c2} i + (\dot{q}_1 + \dot{q}_2)k \times [(\dot{q}_1 + \dot{q}_2)k \times \ell_{c2} i] \quad (6.166)$$

The only quantity in the above equation which is configuration dependent is the first one. This can be computed as

$$\begin{aligned} (R_1^2)^T a_{e,1} &= \begin{bmatrix} \cos q_2 & \sin q_2 \\ -\sin q_2 & \cos q_2 \end{bmatrix} \begin{bmatrix} -\ell_1 \dot{q}_1^2 \\ \ell_1 \ddot{q}_1 \end{bmatrix} \\ &= \begin{bmatrix} -\ell_1 \dot{q}_1^2 \cos q_2 + \ell_1 \ddot{q}_1 \sin q_2 \\ \ell_1 \dot{q}_1^2 \sin q_2 + \ell_1 \ddot{q}_1 \cos q_2 \end{bmatrix} \end{aligned} \quad (6.167)$$

Substituting into (6.166) gives

$$a_{c,2} = \begin{bmatrix} -\ell_1 \dot{q}_1^2 \cos q_2 + \ell_1 \ddot{q}_1 \sin q_2 - \ell_{c2}(\dot{q}_1 + \dot{q}_2)^2 \\ \ell_1 \dot{q}_1^2 \sin q_2 + \ell_1 \ddot{q}_1 \cos q_2 - \ell_{c2}(\ddot{q}_1 + \ddot{q}_2) \end{bmatrix} \quad (6.168)$$

The gravitational vector is

$$g_2 = g \begin{bmatrix} \sin(q_1 + q_2) \\ -\cos(q_1 + q_2) \end{bmatrix} \quad (6.169)$$

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 (6.144) with  $i = 2$  and note that  $f_3 = 0$ . This results in

$$f_2 = m_2 a_{c,2} - m_2 g_2 \quad (6.170)$$

$$\tau_2 = I_2 \alpha_2 + \omega_2 \times (I_2 \omega_2) - f_2 \times \ell_{c2} i \quad (6.171)$$

Now we can substitute for  $\omega_2, \alpha_2$  from (6.160), and for  $a_{c,2}$  from (6.168). We also note that the gyroscopic term equals zero, since both  $\omega_2$  and  $I_2 \omega_2$  are aligned with  $k$ . Now the cross product  $f_2 \times \ell_{c2} i$  is clearly aligned with  $k$  and its magnitude is just the second component of  $f_2$ . The final result is

$$\begin{aligned} \tau_2 = & I_2(\ddot{q}_1 + \ddot{q}_2)k + [m_2 \ell_1 \ell_{c2} \sin q_2 \dot{q}_1^2 + m_2 \ell_1 \ell_{c2} \cos q_2 \ddot{q}_1 \\ & + m_2 \ell_{c2}^2 (\ddot{q}_1 + \ddot{q}_2) + m_2 2\ell_{c2} g \cos(q_1 + q_2)]k \end{aligned} \quad (6.172)$$

Since  $\tau_2 = \tau_2 k$ , we see that the above equation is the same as the second equation in (6.87).

### Backward Recursion: Link 1

To complete the derivation, we apply (6.144) and (6.145) with  $i = 1$ . First, the force equation is

$$f_1 = m_1 a_{c,1} + R_1^2 f_2 - m_1 g_1 \quad (6.173)$$

and the torque equation is

$$\begin{aligned} \tau_1 = & R_1^2 \tau_2 - f_1 \times \ell_{c,1} i - (R_1^2 f_2) \times (\ell_1 - \ell_{c1}) i \\ & + I_1 \alpha_1 + \omega_1 \times (I_1 \omega_1) \end{aligned} \quad (6.174)$$

Now we can simplify things a bit. First,  $R_1^2 \tau_2 = \tau_2$ , since the rotation matrix does not affect the third components of vectors. Second, the gyroscopic term is the again equal to zero. Finally, when we substitute for  $f_1$  from (6.173) into (6.174), a little algebra gives

$$\begin{aligned} \tau_1 = & \tau_2 - m_1 a_{c,1} \times \ell_{c1} i + m_1 g_1 \times \ell_{c1} i - (R_1^2 f_2) \\ & \times \ell_1 i + I_1 i + I_1 \alpha_1 \end{aligned} \quad (6.175)$$

Once again, all these products are quite straightforward, and the only difficult calculation is that of  $R_1^2 f_2$ . The final result is:

$$\begin{aligned} \tau_1 = & \tau_2 + m_1 \ell_{c1}^2 + m_1 \ell_{c1} g \cos q_1 + m_2 \ell_1 g \cos q_1 + I_1 \ddot{q}_1 \\ & + m_2 \ell_1^2 \ddot{q}_1 - m_1 \ell_1 \ell_{c2} (\dot{q}_1 + \dot{q}_2)^2 \sin q_2 + m_2 \ell_1 \ell_{c2} (\dot{q}_1 + \dot{q}_2) \cos q_2 \end{aligned} \quad (6.176)$$

If we now substitute for  $\tau_1$  from (6.172) and collect terms, we will get the first equation in (6.87); the details are routine and are left to the reader.

---

## Problems

---

- 6-1 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 located at the center of mass and whose coordinate axes are principal axes. Take as generalized coordinates the Euler angles  $\phi, \theta, \psi$  and show that the Euler-Lagrange equations of motion of the rotating body are

$$\begin{aligned} I_{xx}\dot{\omega}_x + (I_{zz} - I_{yy})\omega_y\omega_z &= 0 \\ I_{yy}\dot{\omega}_y + (I_{xx} - I_{zz})\omega_z\omega_x &= 0 \\ I_{zz}\dot{\omega}_z + (I_{yy} - I_{xx})\omega_x\omega_y &= 0. \end{aligned}$$

- 6-2 Verify the expression (??).

- 6-3 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.

- 6-4 Given the cylindrical shell shown, show that

$$\begin{aligned} I_{xx} &= \frac{1}{2}mr^2 + \frac{1}{12}m\ell^2 \\ I_{x_1x_1} &= \frac{1}{2}mr^2 + \frac{1}{3}m\ell^2 \\ I_z &= mr^2. \end{aligned}$$

- 6-5 Given the inertia matrix  $D(q)$  defined by (6.81) show that  $\det D(q) \neq 0$  for all  $q$ .

- 6-6 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  $3 \times 3$  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:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = u.$$

- 6-7 Derive the Euler-Lagrange equations for the planar RP robot in Figure ??.
- 6-8 Derive the Euler-Lagrange equations for the planar PR robot in Figure ??.
- 6-9 Derive the Euler-Lagrange equations of motion for the three-link RRR robot of Figure ?? . Explore the use of symbolic software, such as Maple or Mathematica, for this problem. See, for example, the *Robotica* package [?].
- 6-10 For each of the robots above, define a parameter vector,  $\Theta$ , compute the Regressor,  $Y(q, \dot{q}, \ddot{q})$  and express the equations of motion as

$$Y(q, \dot{q}, \ddot{q})\Theta = \tau \quad (6.177)$$

- 6-11 Recall for a particle with kinetic energy  $K = \frac{1}{2}m\dot{x}^2$ , the **momentum** is defined as

$$p = m\dot{x} = \frac{dK}{dx}.$$

Therefore for a mechanical system with generalized coordinates  $q_1, \dots, q_n$ , we define the **generalized momentum**  $p_k$  as

$$p_k = \frac{\partial L}{\partial \dot{q}_k}$$

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

$$\sum_{k=1}^n \dot{q}_k p_k = 2K.$$

- 6-12 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

$$H = \sum_{k=1}^n \dot{q}_k p_k - L.$$

- a) Show that  $H = K + V$ .  
 b) Using Lagrange's 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  $\tau_k$  is the input generalized force.

- c) For two-link manipulator of Figure 6.7 compute Hamiltonian equations in matrix form. Note that Hamilton's equations are a system of first order differential equations as opposed to second order system given by Lagrange's equations.

6-13 Given the Hamiltonian  $H$  for a rigid robot, show that

$$\frac{dH}{dt} = \dot{q}^T \tau$$

where  $\tau$  is the external force applied at the joints. What are the units of  $\frac{dH}{dt}$ ?



# 7

---

## INDEPENDENT JOINT CONTROL

### 7.1 INTRODUCTION

The control problem for robot manipulators is the problem of determining 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 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 chosen as well as the manner in which it is implemented 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 control.

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 fundamentally different from those encountered with an elbow type manipulator. This creates a so-called *hardware/software trade-off* between the mechanical structure of the system and the architecture/programming of the controller.

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, 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 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 is treated as a disturbance. We assume, in this chapter, that the reader has had an introduction to the theory of feedback control systems up to the level of say, Kuo [?].

The basic structure of a single-input/single-output feedback control system is shown in Figure 7.1. The design objective is to choose the compensator in

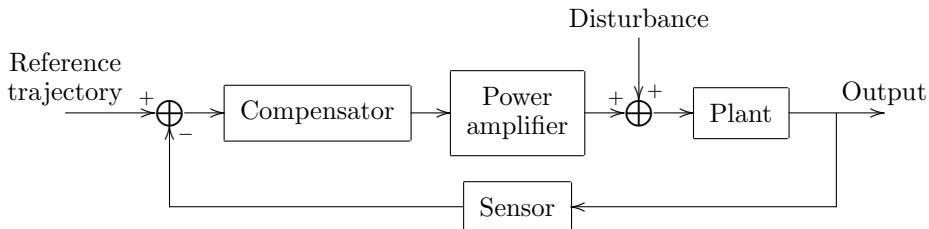


Fig. 7.1 Basic structure of a feedback control system.

such a way that the plant output “tracks” or follows 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.

## 7.2 ACTUATOR DYNAMICS

In Chapter 6 we obtained the following set of differential equations describing the motion of an  $n$  degree of freedom robot (cf. Equation (6.61))

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (7.1)$$

It is important to understand exactly what this equation represents. Equation (7.1) represents the dynamics of an interconnected chain of ideal rigid bodies, supposing that there is a generalized force  $\tau$  acting at the joints. We can assume that the  $k$ -th component  $\tau_k$  of the generalized force vector  $\tau$  is a torque about the joint axis  $z_{k-1}$  if joint  $k$  is revolute and is a force along the joint axis  $z_{k-1}$  if joint  $k$  is prismatic. This generalized force is produced by an actuator, which may be electric, hydraulic or pneumatic. Although Equation (7.1) is extremely complicated for all but the simplest manipulators, it nevertheless is an idealization, and there are a number of dynamic effects that are *not* included in Equation (7.1). For example, friction at the joints is not accounted for in these equations and may be significant for some manipulators. Also, no physical body is completely rigid. A more detailed analysis of robot dynamics would include various sources of flexibility, such as elastic deformation of bearings and gears, deflection of the links under load, and vibrations. In this section we are interested mainly in the dynamics of the actuators producing the generalized force  $\tau$ . We treat only the dynamics of *permanent magnet DC-motors*, as these are the simplest actuators to analyze. Other types of motors, in particular *AC-motors* and so-called *Brushless DC-motors* are increasingly common in robot applications (See [?] for details on the dynamics of AC drives.

A DC-motor works on the principle that a current carrying conductor in a magnetic field experiences a force  $F = i \times \phi$ , where  $\phi$  is the magnetic flux and  $i$  is the current in the conductor. The motor itself consists of a fixed *stator* and a movable *rotor* that rotates inside the stator, as shown in Figure 7.2. If the stator produces a radial magnetic flux  $\phi$  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

$$\tau_m = K_1 \phi i_a \quad (7.2)$$

where  $\tau_m$  is the motor torque ( $N - m$ ),  $\phi$  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  $\tau_m$  in (7.2), we have the back emf relation

$$V_b = K_2 \phi \omega_m \quad (7.3)$$

where  $V_b$  denotes the back emf (Volts),  $\omega_m$  is the angular velocity of the rotor (rad/sec), and  $K_2$  is a proportionality constant.

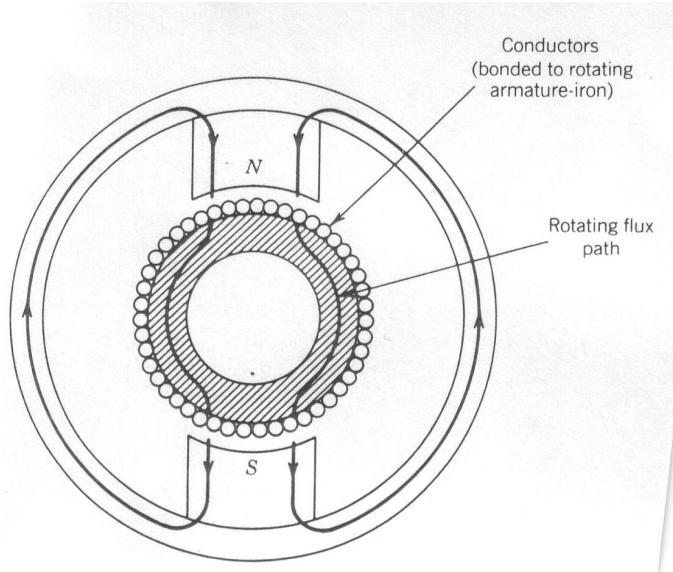


Fig. 7.2 Cross-sectional view of a surface-wound permanent magnet DC motor.

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,  $\phi$ , 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 7.3 where

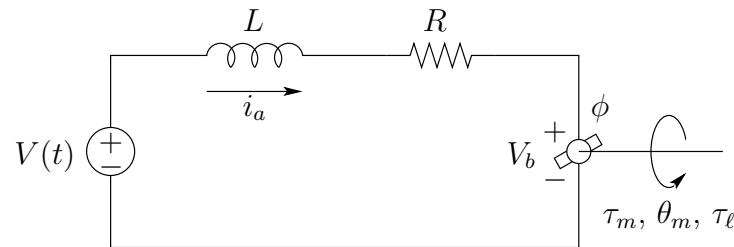


Fig. 7.3 Circuit diagram for armature controlled DC motor.

- $V$  = armature voltage  
 $L$  = armature inductance  
 $R$  = armature resistance  
 $V_b$  = back emf  
 $i_a$  = armature current  
 $\theta_m$  = rotor position (radians)  
 $\tau_m$  = generated torque  
 $\tau_\ell$  = load torque  
 $\phi$  = magnetic flux due to stator

The differential equation for the armature current is then

$$L \frac{di_a}{dt} + Ri_a = V - V_b. \quad (7.4)$$

Since the flux  $\phi$  is constant the torque developed by the motor is

$$\tau_m = K_1 \phi i_a = K_m i_a \quad (7.5)$$

where  $K_m$  is the torque constant in  $N \cdot m/\text{amp}$ . From (7.3) we have

$$V_b = K_2 \phi \omega_m = K_b \omega_m = K_b \frac{d\theta_m}{dt} \quad (7.6)$$

where  $K_b$  is the back emf constant.

We can determine the torque constant of the DC motor using a set of torque-speed curves as shown in Figure 7.4 for various values of the applied voltage

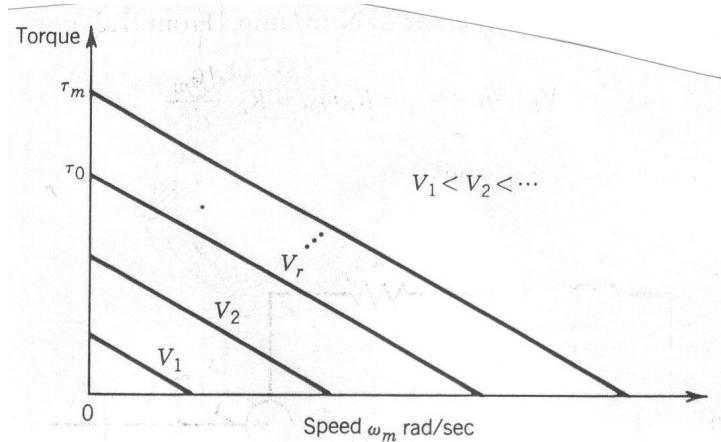


Fig. 7.4 Typical torque-speed curves of a DC motor.

V. When the motor is stalled, the blocked-rotor torque at the rated voltage is

denoted  $\tau_0$ . Using Equation (7.4) with  $V_b = 0$  and  $di_a/dt = 0$  we have

$$V_r = Ri_a = \frac{R\tau_0}{K_m} \quad (7.7)$$

Therefore the torque constant is

$$K_m = \frac{R\tau_0}{V_r} \quad (7.8)$$

The remainder of the discussion refers to Figure 7.5 consisting of the DC-motor

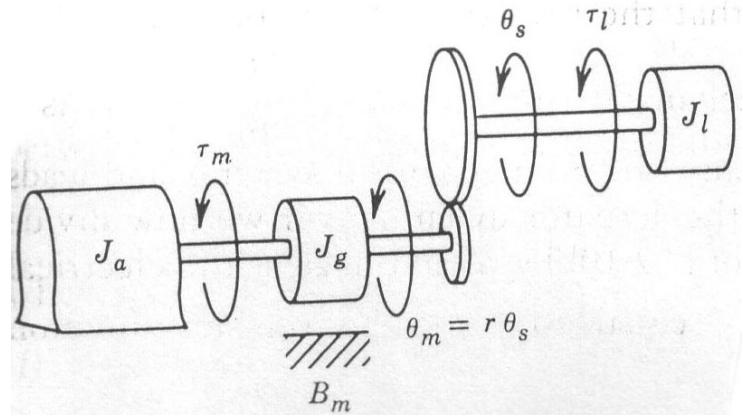


Fig. 7.5 Lumped model of a single link with actuator/gear train.

in series with a gear train with gear ratio  $r : 1$  and connected to a link of the manipulator. The gear ratio  $r$  typically has values in the range 20 to 200 or more. Referring to Figure 7.5, we set  $J_m = J_a + J_g$ , the sum of the actuator and gear inertias. The equation of motion of this system is then

$$\begin{aligned} J_m \frac{d^2\theta_m}{dt^2} + B_m \frac{d\theta_m}{dt} &= \tau_m - \tau_\ell/r \\ &= K_m i_a - \tau_\ell/r \end{aligned} \quad (7.9)$$

the latter equality coming from (7.5). In the Laplace domain the three equations (7.4), (7.6) and (7.9) may be combined and written as

$$(Ls + R)I_a(s) = V(s) - K_b s \Theta_m(s) \quad (7.10)$$

$$(J_m s^2 + B_m s) \Theta_m(s) = K_i I_a(s) - \tau_\ell(s)/r \quad (7.11)$$

The block diagram of the above system is shown in Figure 7.6. The transfer function from  $V(s)$  to  $\Theta_m(s)$  is then given by (with  $\tau_\ell = 0$ )

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m}{s[(Ls + R)(J_m s + B_m) + K_b K_m]} \quad (7.12)$$

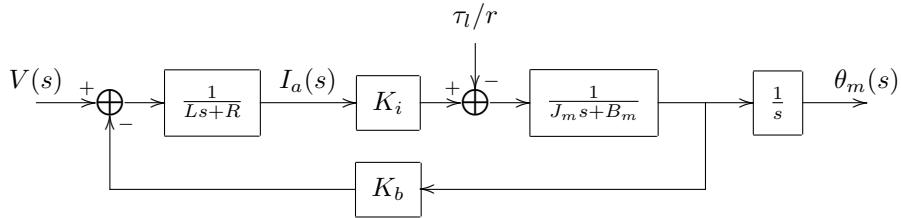


Fig. 7.6 Block diagram for a DC motor system

The transfer function from the load torque  $\tau_\ell(s)/r$  to  $\Theta_m(s)$  is given by (with  $V = 0$ )

$$\frac{\Theta_m(s)}{\tau_\ell(s)} = \frac{-(Ls + R)}{s[(Ls + R)(J_m s + B_m) + K_b K_m]} \quad (7.13)$$

Frequently it is assumed that the “electrical time constant”  $\frac{L}{R}$  is much smaller than the “mechanical time constant”  $\frac{J_m}{B_m}$ . This is a reasonable assumption for many electro-mechanical systems and leads to a reduced order model of the actuator dynamics. If we now divide numerator and denominator of Equations (7.12) and (7.13) by  $R$  and neglect the electrical time constant by setting  $\frac{L}{R}$  equal to zero, the transfer functions in Equations (7.12) and (7.13) become, respectively,

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m/R}{s(J_m s + B_m + K_b K_m/R)}. \quad (7.14)$$

and

$$\frac{\Theta_m(s)}{\tau_\ell(s)} = -\frac{1}{s(J_m(s) + B_m + K_b K_m/R)} \quad (7.15)$$

In the time domain Equations (7.14) and (7.15) represent, by superposition, the second order differential equation

$$J_m \ddot{\theta}_m(t) + (B_m + K_b K_m/R) \dot{\theta}_m(t) = (K_m/R)V(t) - \tau_\ell(t)/r \quad (7.16)$$

The block diagram corresponding to the reduced order system (7.16) is shown in Figure 7.7.

If the output side of the gear train is directly coupled to the link, then the joint variables and the motor variables are related by

$$\theta_{m_k} = r_k q_k ; \quad k = 1, \dots, n \quad (7.17)$$

where  $r_k$  is the  $k$ -th gear ratio. Similarly, the joint torques  $\tau_k$  given by (7.1) and the actuator load torques  $\tau_{\ell_k}$  are related by

$$\tau_{\ell_k} = \tau_k ; \quad k = 1, \dots, n. \quad (7.18)$$

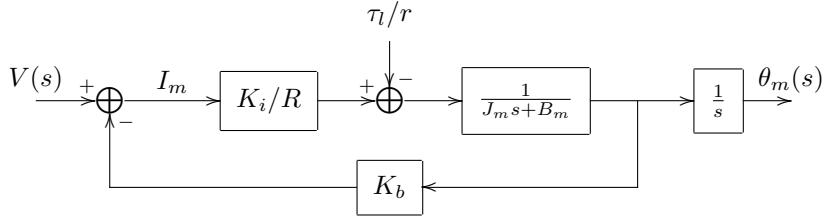


Fig. 7.7 Block diagram for reduced order system.

However, in manipulators incorporating other types of drive mechanisms such as belts, pulleys, chains, etc.,  $\theta_{m_k}$  need not equal  $r_k q_k$ . In general one must incorporate into the dynamics a transformation between joint space variables and actuator variables of the form

$$q_k = f_k(\theta_{s_1}, \dots, \theta_{s_n}) ; \quad \tau_{\ell_k} = f_k(\tau_1, \dots, \tau_n) \quad (7.19)$$

where  $\theta_{s_k} = \theta_{m_k}/r_k$ .

### Example 7.2.1

Consider the two link planar manipulator shown in Figure 7.8, whose actuators

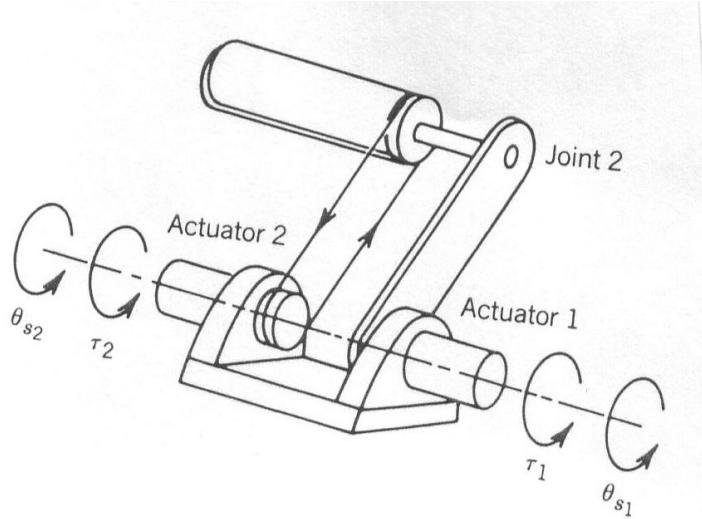


Fig. 7.8 Two-link manipulator with remotely driven link.

are both located on link 1. In this case we have,

$$q_1 = \theta_{s_1} ; \quad q_2 = \theta_{s_1} + \theta_{s_2}. \quad (7.20)$$

Similarly, the joint torques  $\tau_i$  and the actuator load torques  $\tau_\ell$ , are related by

$$\tau_{\ell_1} = \tau_1 \quad ; \quad \tau_{\ell_2} = \tau_1 + \tau_2. \quad (7.21)$$

The inverse transformation is then

$$\theta_{s_1} = q_1 \quad ; \quad \theta_{s_2} = q_2 - q_1 \quad (7.22)$$

and

$$\tau_1 = \tau_{\ell_1} \quad ; \quad \tau_2 = \tau_{\ell_2} - \tau_{\ell_1}. \quad (7.23)$$

### 7.3 SET-POINT TRACKING

In this section we discuss set-point tracking using a PD or PID compensator. This type of control is adequate for applications not involving very fast motion, especially in robots with large gear reduction between the actuators and the links. The analysis in this section follows typical engineering practice rather than complete mathematical rigor. For the following discussion, assume for simplicity that

$$\begin{aligned} q_k &= \theta_{s_k} &= \theta_{m_k}/r_k \text{ and} \\ \tau_{\ell_k} &= \tau_k. \end{aligned} \quad (7.24)$$

Then, for  $k = 1, \dots, n$ , the equations of motion of the manipulator can be written as

$$\sum_{j=1}^n d_{jk}(q) \ddot{q}_j + \sum_{i,j=1}^n c_{ijk}(q) \dot{q}_i \dot{q}_j + g_k(q) = \tau_k \quad (7.25)$$

$$J_{m_k} \ddot{\theta}_{m_k} + (B_{m_k} + K_{b_k} K_{m_k}/R_k) \dot{\theta}_{m_k} = K_{m_k}/R_k V_k - \tau_k/r_k \quad (7.26)$$

Combining these equations yields

$$(J_{m_k} + \frac{1}{r_k^2} d_{kk}(q)) \ddot{\theta}_{m_k} + (B_{m_k} + K_{b_k} K_{m_k}/R_k) \dot{\theta}_{m_k} = K_{m_k}/R_k V_k - d_k \quad (7.27)$$

where  $d_k$  is defined by

$$d_k := \frac{1}{r_k} \sum_{j \neq k} \ddot{q}_j + \sum_{i,j} c_{ijk} \dot{q}_i \dot{q}_j + g_k. \quad (7.28)$$

Note that the coefficient of  $\ddot{\theta}_{m_k}$  in the above expression is a nonlinear function of the manipulator configuration,  $q$ . However, large gear reduction, i.e. large values of  $r_k$ , mitigates the influence of this term and one often defines a constant *average*, or *effective inertia*  $J_{eff,k}$  as an approximation of the exact expression  $J_{m_k} + \frac{1}{r_k^2} d_{kk}(q)$ . If we further define

$$B_{eff,k} = B_{m_k} + K_{b_k} K_{m_k}/R_k \quad \text{and} \quad u_k = K_{m_k}/R_k V_k \quad (7.29)$$

we may write (7.26) as

$$J_{eff_k} \ddot{\theta}_{m_k} + B_{eff_k} \dot{\theta}_{m_k} = u_k - d_k \quad (7.30)$$

The advantage of this model is its simplicity since the motor dynamics represented by (7.26) are linear. The effect of the nonlinear coupling terms is treated as a disturbance  $d_k$ , which may be small for large gear reduction provided the velocities and accelerations of the joints are also small.

Henceforth we suppress the subscript  $k$  representing the particular joint and represent (7.30) in the Laplace domain by the block diagram of Figure 7.9. The

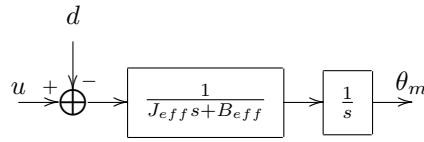


Fig. 7.9 Block diagram of simplified open loop system with effective inertia and damping.

*set-point tracking problem* is now the problem of tracking a constant or step reference command  $\theta^d$  while rejecting a constant disturbance,  $d$ .

### 7.3.1 PD Compensator

As a first illustration, we choose a so-called PD-compensator. The resulting closed loop system is shown in Figure 7.10. The input  $U(s)$  is given by

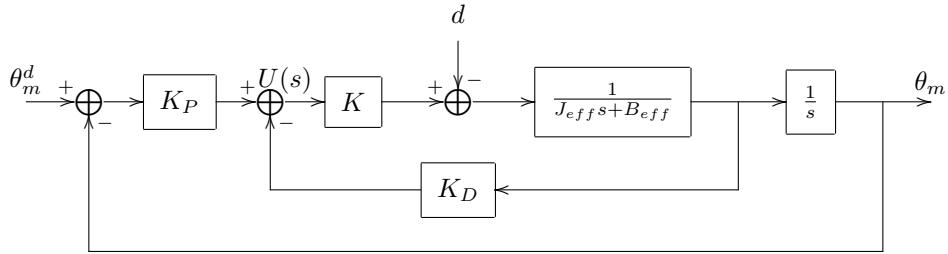


Fig. 7.10 Closed loop system with PD-control.

$$U(s) = K_p(\Theta^d(s) - \Theta(s)) - K_D s \Theta(s) \quad (7.31)$$

where  $K_p$ ,  $K_D$  are the proportional (P) and derivative (D) gains, respectively.

Taking Laplace transforms of both sides of (7.30) and using the expression (7.31) for the feedback control  $V(s)$ , leads to the closed loop system

$$\Theta_m(s) = \frac{KK_p}{\Omega(s)} \Theta^d(s) - \frac{1}{\Omega(s)} D(s) \quad (7.32)$$

where  $\Omega(s)$  is the closed loop characteristic polynomial

$$\Omega(s) = J_{eff}s^2 + (B_{eff} + KK_D)s + KK_p \quad (7.33)$$

The closed loop system will be stable for all positive values of  $K_p$  and  $K_D$  and bounded disturbances, and the tracking error is given by

$$\begin{aligned} E(s) &= \Omega^d(s) - \Theta_m(s) \\ &= \frac{J_{eff}s^2 + (B_{eff} + KK_D)s}{\Omega(s)} \Theta^d(s) + \frac{1}{\Omega(s)} D(s) \end{aligned} \quad (7.34)$$

For a step reference input

$$\Theta^d(s) = \frac{\Omega^d}{s} \quad (7.35)$$

and a constant disturbance

$$D(s) = \frac{D}{s} \quad (7.36)$$

it now follows directly from the final value theorem [4] that the steady state error  $e_{ss}$  satisfies

$$e_{ss} = \lim_{s \rightarrow 0} sE(s) \quad (7.37)$$

$$= \frac{-D}{KK_p} \quad (7.38)$$

Since the magnitude  $D$  of the disturbance is proportional to the gear reduction  $\frac{1}{r}$  we see that the steady state error is smaller for larger gear reduction 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.

We know, of course, from (7.28) that the disturbance term  $D(s)$  in (7.34) is not constant. However, in the steady state this disturbance term is just the gravitational force acting on the robot, which is constant. The above analysis therefore, while only approximate, nevertheless gives a good description of the actual steady state error using a PD compensator assuming stability of the closed loop system.

### 7.3.2 Performance of PD Compensators

For the PD-compensator given by (7.31) the closed loop system is second order and hence the step response is determined by the closed loop natural frequency  $\omega$  and damping ratio  $\zeta$ . Given a desired value for these quantities, the gains  $K_D$  and  $K_p$  can be found from the expression

$$s^2 + \frac{(B_{eff} + KK_D)}{J_{eff}}s + \frac{KK_p}{J_{eff}} = s^2 + 2\zeta\omega s + \omega^2 \quad (7.39)$$

*Table 7.1* Proportional and Derivative Gains for the System 7.11 for Various Values of Natural Frequency  $\omega$

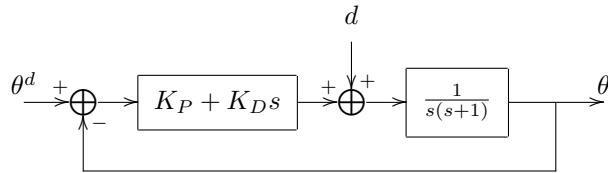
| Natural Frequency ( $\omega$ ) | Proportional Gain $K_P$ | Derivative Gain $K_D$ |
|--------------------------------|-------------------------|-----------------------|
| 4                              | 16                      | 7                     |
| 8                              | 64                      | 15                    |
| 12                             | 144                     | 23                    |

as

$$K_p = \frac{\omega^2 J_{eff}}{K}, \quad K_D = \frac{2\zeta\omega J_{eff} - B_{eff}}{K} \quad (7.40)$$

It is customary in robotics applications to take  $\zeta = 1$  so that the response is critically damped. This produces the fastest non-oscillatory response. In this context  $\omega$  determines the speed of response.

**Example 7.1** Consider the second order system of Figure 7.11. The closed



*Fig. 7.11* Second Order System with PD Compensator

loop characteristic polynomial is

$$p(s) = s^2 + (1 + K_D)s + K_p \quad (7.41)$$

Suppose  $\theta^d = 10$  and there is no disturbance ( $d = 0$ ). With  $\zeta = 1$ , the required PD gains for various values of  $\omega$  are shown in Table 7.1. The corresponding step responses are shown in Figure 7.12.

Now suppose that there is a constant disturbance  $d = 40$  acting on the system. The response of the system with the PD gains of Table 7.1 are shown in Figure 7.13. We see that the steady state error due to the disturbance is smaller for large gains as expected. ◇

### 7.3.3 PID Compensator

In order to reject a constant disturbance using PD control we have seen that large gains are required. By using integral control we may achieve zero steady

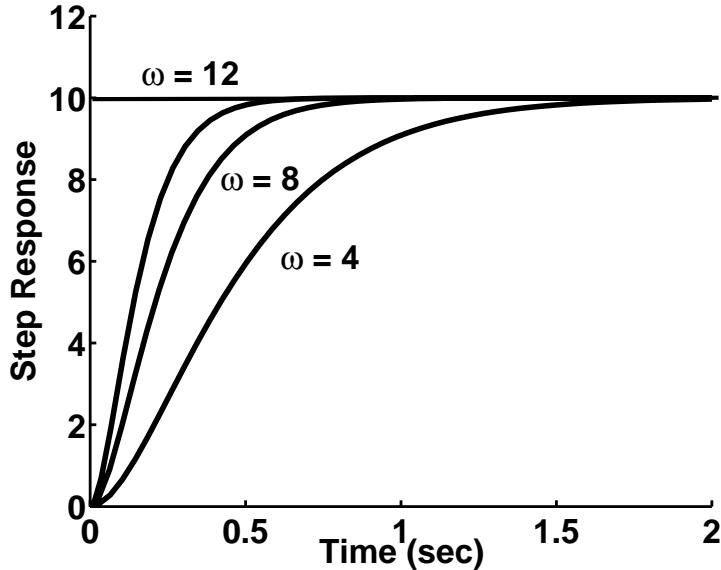


Fig. 7.12 Critically damped second order step responses.

state error while keeping the gains small. Thus, let us add an integral term  $\frac{K_I}{s}$  to the above PD compensator. This leads to the so-called PID control law, as shown in Figure 7.14. The system is now Type 2 and the PID control achieves exact steady tracking of step (and ramp) inputs while rejecting step disturbances, provided of course that the closed loop system is stable.

With the PID compensator

$$C(s) = K_p + K_D s + \frac{K_I}{s} \quad (7.42)$$

the closed loop system is now the third order system

$$\Theta_m(s) = \frac{(K_D s^2 + K_p s + K_I)}{\Omega_2(s)} \Theta^d(s) - \frac{rs}{\Omega_2(s)} D(s) \quad (7.43)$$

where

$$\Omega_2 = J_{eff} s^3 + (B_{eff} + KK_D) s^2 + KK_p s + KK_I \quad (7.44)$$

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,

$$K_I < \frac{(B_{eff} + KK_D) K_p}{J_{eff}} \quad (7.45)$$

**Example 7.2** To the previous system we have added a disturbance and an

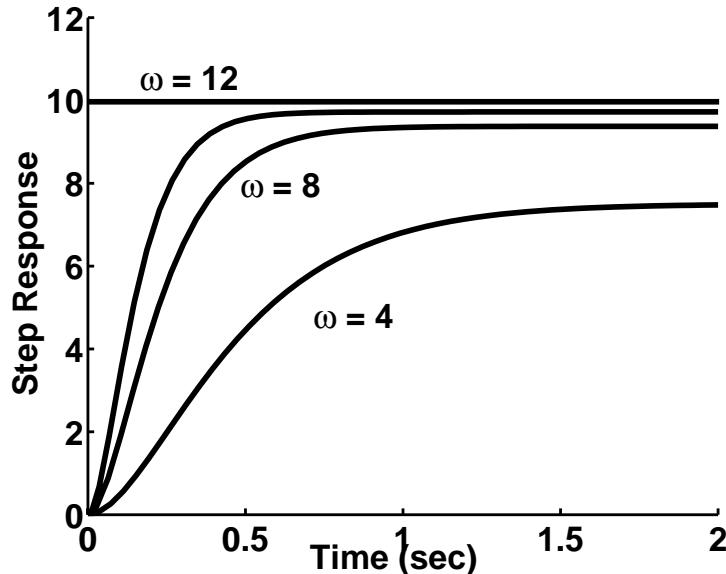


Fig. 7.13 Second order system response with disturbance added.

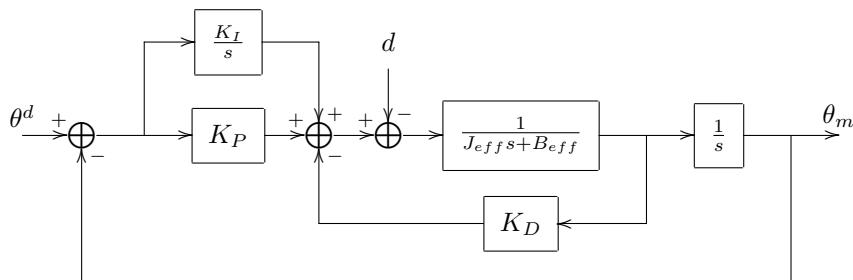


Fig. 7.14 Closed loop system with PID control.

integral control term in the compensator. The step responses are shown in Figure 7.15. We see that the steady state error due to the disturbance is removed.  $\diamond$

#### 7.3.4 Saturation

In theory, one could achieve arbitrarily fast response and arbitrarily small steady state error to a constant disturbance by simply increasing the gains in the PD or PID compensator. In practice, however, there is a maximum speed of response achievable from the system. Two major factors, heretofore neglected, limit the achievable performance of the system. The first factor, *saturation*, is due to

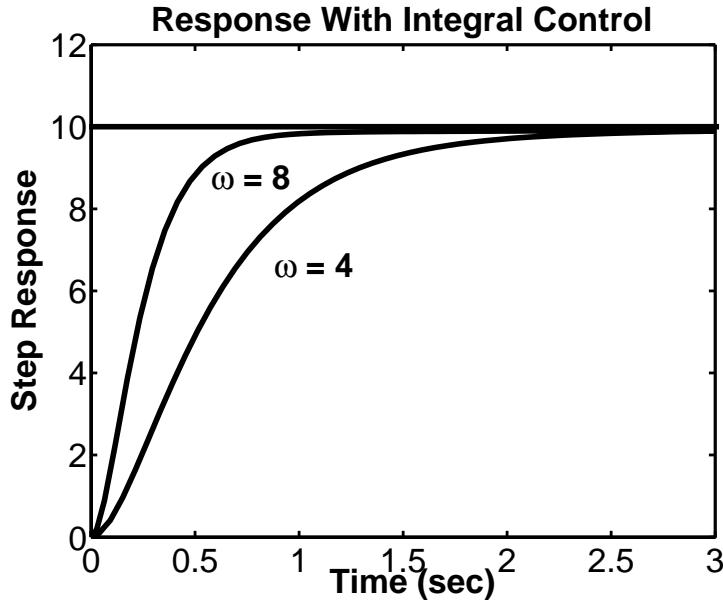


Fig. 7.15 Response with integral control action.

limits on the maximum torque (or current) input. Many manipulators, in fact, incorporate current limiters in the servo-system to prevent damage that might result from overdriving current. The second effect is flexibility in the motor shaft and/or drive train. We illustrate the effects of saturation below and drive train flexibility in section 7.5.

**Example 7.3** Consider the block diagram of Figure 7.16, where the saturation

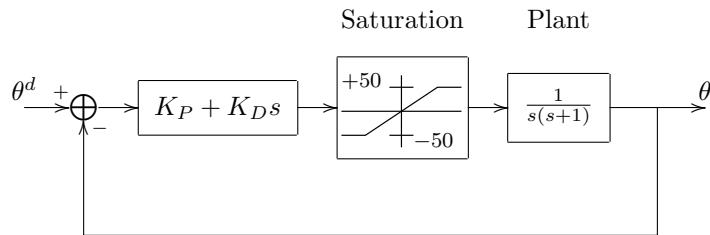


Fig. 7.16 Second order system with input saturation.

function represents the maximum allowable input. With PD control and saturation the response is below.  $\diamond$

The second effect to consider is the joint flexibility. Let  $k_r$  be the effective stiffness at the joint. The joint resonant frequency is then  $\omega_4 = \sqrt{k_r/J_{eff}}$ . It is common engineering practice to limit  $\omega$  in (7.40) to no more than half of  $\omega_r$  to

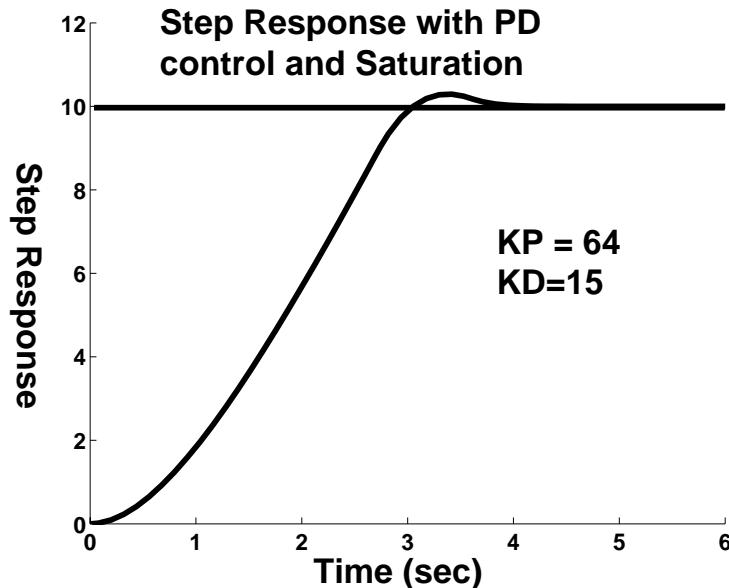


Fig. 7.17 Response with Saturation and PD Control

avoid excitation of the joint resonance. We will discuss the effects of the joint flexibility in more detail in section 7.5.

These examples clearly show the limitations of PID-control when additional effects such as input saturation, disturbances, and unmodeled dynamics must be considered.

#### 7.4 FEEDFORWARD CONTROL AND COMPUTED TORQUE

In this section we introduce the notion of *feedforward control* as a method to track time varying trajectories and reject time varying disturbances.

Suppose that  $r(t)$  is an arbitrary reference trajectory and consider the block diagram of Figure 7.18, where  $G(s)$  represents the forward transfer function of

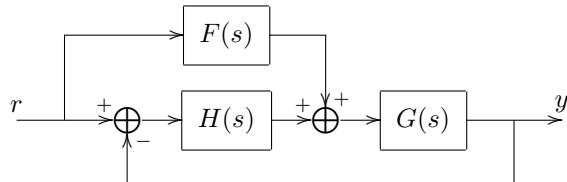


Fig. 7.18 Feedforward control scheme.

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

$$G(s) = \frac{q(s)}{p(s)} \quad H(s) = \frac{c(s)}{d(s)} \quad F(s) = \frac{a(s)}{b(s)} \quad (7.46)$$

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 7-9)

$$T(s) = \frac{q(s)(c(s)b(s) + a(s)d(s))}{b(s)(p(s)d(s) + q(s)c(s))} \quad (7.47)$$

The closed loop characteristic polynomial of the system is then  $b(s)(p(s)d(s) + q(s)c(s))$ . For stability of the closed loop system therefore 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  $\frac{1}{G(s)}$ , the inverse of the forward plant, that is,  $a(s) = p(s)$  and  $b(s) = q(s)$ , then the closed loop system becomes

$$q(s)(p(s)d(s) + q(s)c(s))Y(s) = q(s)(p(s)d(s) + q(s)c(s))R(s) \quad (7.48)$$

or, in terms of the tracking error  $E(s) = R(s) - Y(s)$ ,

$$q(s)(p(s)d(s) + q(s)c(s))E(s) = 0 \quad (7.49)$$

Thus, assuming stability, the output  $y(t)$  will track any reference trajectory  $r(t)$ . 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 7.19, then it is easily shown that the tracking error  $E(s)$  is given by

$$E(s) = \frac{q(s)d(s)}{p(s)d(s) + q(s)c(s)}D(s) \quad (7.50)$$

We have thus shown that, in the absence of disturbances the closed loop system will track any desired trajectory  $r(t)$  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 robot model of Section 7.3. Suppose that  $\theta^d(t)$  is an arbitrary trajectory that we wish the system to track. In this case

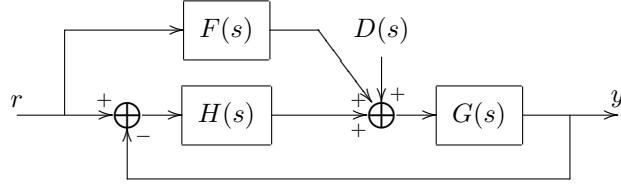


Fig. 7.19 Feedforward control with disturbance.

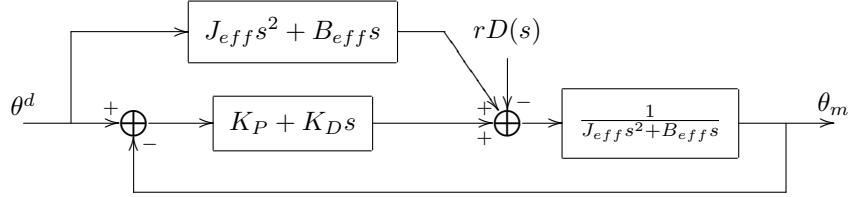


Fig. 7.20 Feedforward compensator for second order system.

we have from (7.30)  $G(s) = \frac{K}{J_{eff}s^2 + B_{eff}s}$  together with a PD compensator  $H(s) = K_p + K_Ds$ . The resulting system is shown in Figure 7.20. Note that  $G(s)$  has no zeros at all and hence is minimum phase. Note also that  $G(s)^{-1}$  is not a proper rational function. However, since the derivatives of the reference trajectory  $\theta^d$  are known and precomputed, the implementation of the above scheme does not require differentiation of an actual signal. It is easy to see from (7.50) that the steady state error to a step disturbance is now given by the same expression (7.37) 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 7.20 can be written as

$$\begin{aligned} V(t) &= \frac{J_{eff}}{K} \ddot{\theta}^d + \frac{B_{eff}}{K} \dot{\theta}^d + K_D(\dot{\theta}^d - \dot{\theta}_m) + K_p(\theta^d - \theta_m) \quad (7.51) \\ &= f(t) + K_D \dot{e}(t) + K_p e(t) \end{aligned}$$

where  $f(t)$  is the feedforward signal

$$f(t) = \frac{J_{eff}}{K} \ddot{\theta}^d + \frac{B_{eff}}{K} \dot{\theta}^d \quad (7.52)$$

and  $e(t)$  is the tracking error  $\theta^d(t) - \theta(t)$ . Since the forward plant equation is

$$J_{eff} \ddot{\theta}_m + B_{eff} \dot{\theta}_m = KV(t) - rd(t)$$

the closed loop error  $e(t) = \theta_m - \theta^d$  satisfies the second order differential equation

$$J_{eff} \ddot{e} + (B_{eff} + KK_D) \dot{e} + KK_p e(t) = -rd(t) \quad (7.53)$$

**Remark 7.6.1**

We note from (7.53) that the characteristic polynomial of the closed loop system is identical to (7.33). The system now however is 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  $d = 0$ .

**Computed Torque Disturbance Cancellation**

We see that the feedforward signal (7.52) 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  $d(t)$  in (7.53) represents a disturbance, it is not completely unknown since  $d$  satisfies (7.28). Thus we may consider adding to the above feedforward signal, a term to anticipate the effects of the disturbance  $d(t)$ . Consider the diagram of Figure 7.21. Given a desired trajectory, then we superimpose, as shown, the

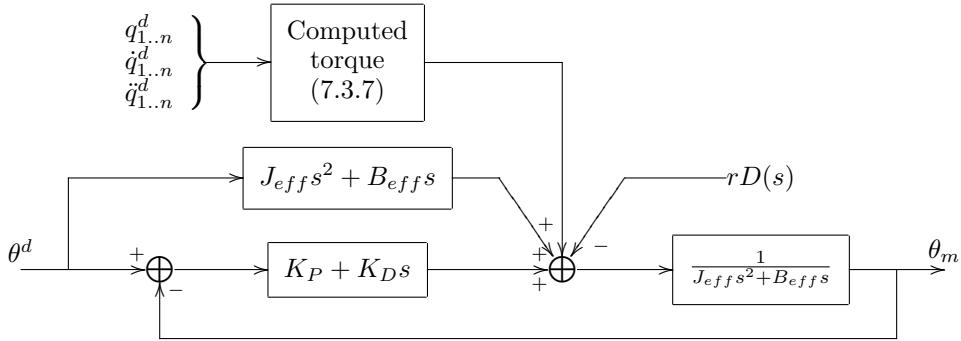


Fig. 7.21 Feedforward computed torque compensation.

term

$$d^d := \sum d_{jk}(q^d)\dot{q}_j^d + \sum c_{ijk}(q^d)\dot{q}_i^d\dot{q}_j^d + g_k(q^d) \quad (7.54)$$

since  $d^d$  has units of torque, the above feedforward disturbance cancellation control is called the *method of computed torque*. The expression (7.54) thus compensates in a feedforward manner the nonlinear coupling inertial, coriolis, centripetal, and gravitational forces arising due to the motion of the manipulator. Although the difference  $\Delta d := d^d - d$  is zero only in the ideal case of perfect tracking ( $\theta = \theta^d$ ) and perfect computation of (7.54), in practice, the goal is to reduce  $\Delta d$  to a value smaller than  $d$  (say, in the usual Euclidean norm sense). Hence the computed torque has the advantage of reducing the effects of  $d$ . Note that the expression (7.54) is in general extremely complicated so that the computational burden involved in computing (7.54) is of major concern. Since only the values of the desired trajectory need to be known, many of

these terms can be precomputed and stored off-line. Thus there is a trade-off between memory requirements and on-line computational requirements. This has led to the development of table look up schemes to implement (7.54) and also to the development of computer programs for the automatic generation and simplification of manipulator dynamic equations.

## 7.5 DRIVE TRAIN DYNAMICS

In this section we discuss in more detail the problem of joint flexibility. For many manipulators, particularly those using harmonic drives<sup>1</sup> 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.

Consider the idealized situation of Figure 7.22 consisting of an actuator con-

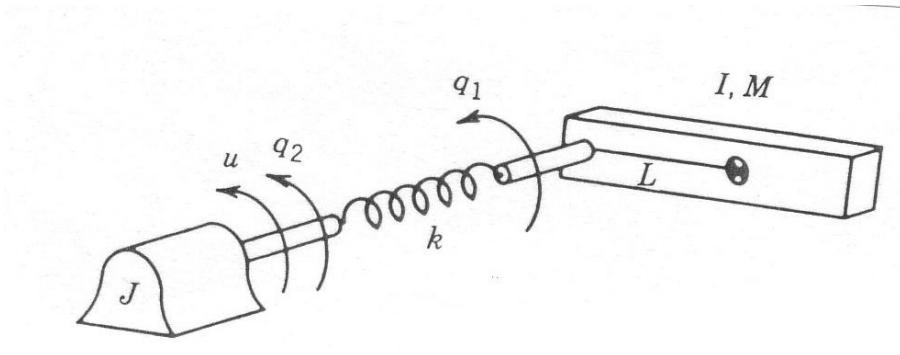


Fig. 7.22 Idealized model to represent joint flexibility.

nected to a load through a torsional spring which represents the joint flexibility. For simplicity we take the motor torque  $u$  as input. The equations of motion are easily derived using the techniques of Chapter 6, with generalized coordinates  $\theta_\ell$  and  $\theta_m$ , the link angle, and the motor angle, respectively, as

$$J_\ell \ddot{\theta}_\ell + B_\ell \dot{\theta}_\ell + k(\theta_\ell - \theta_m) = 0 \quad (7.55)$$

$$J_m \ddot{\theta}_m + B_m \dot{\theta}_m - k(\theta_\ell - \theta_m) = u \quad (7.56)$$

<sup>1</sup>Harmonic drives 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. However, they also introduce unwanted friction and flexibility at the joints.

where  $J_\ell$ ,  $J_m$  are the load and motor inertias,  $B_\ell$  and  $B_m$  are the load and motor damping constants, and  $u$  is the input torque applied to the motor shaft. In the Laplace domain we can write this as

$$p_\ell(s)\Theta_\ell(s) = k\Theta_m(s) \quad (7.57)$$

$$p_m(s)\Theta_m(s) = k\Theta_\ell(s) + U(s) \quad (7.58)$$

where

$$p_\ell(s) = J_\ell s^2 + B_\ell s + k \quad (7.59)$$

$$p_m(s) = J_m s^2 + B_m s + k \quad (7.60)$$

This system is represented by the block diagram of Figure 7.23.

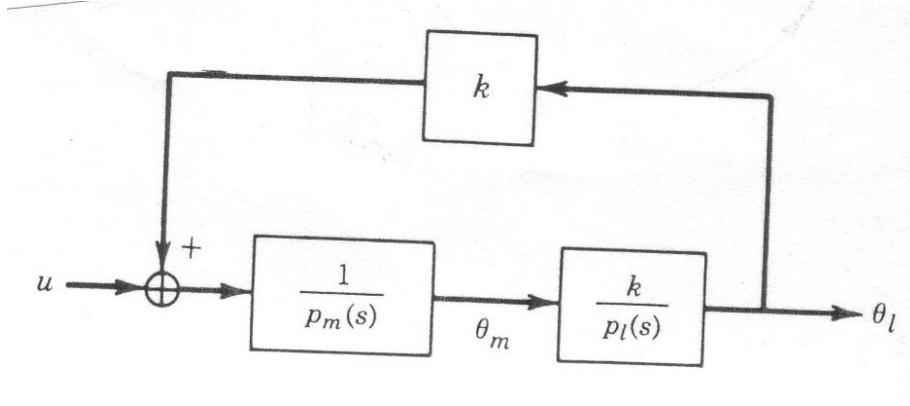


Fig. 7.23 Block diagram for the system (7.57)-(7.58).

The output to be controlled is, of course, the load angle  $\theta_\ell$ . The open loop transfer function between  $U$  and  $\Theta_\ell$  is given by

$$\frac{\Theta_\ell(s)}{U(s)} = \frac{k}{p_\ell(s)p_m(s) - k^2} \quad (7.61)$$

The open loop characteristic polynomial is

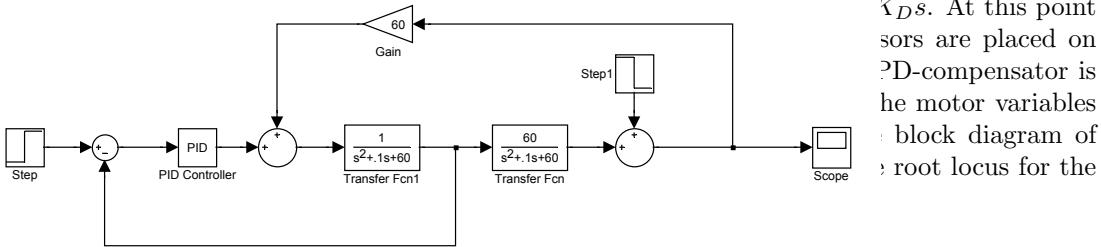
$$J_\ell J_m s^4 + (J_\ell B_m + J_m B_\ell)s^3 + (k(J_\ell + J_m) + B_\ell B_m)s^2 + k(B_\ell + B_m)s \quad (7.62)$$

If the damping constants  $B_\ell$  and  $B_m$  are neglected, the open loop characteristic polynomial is

$$J_\ell J_m s^4 + k(J_\ell + J_m)s^2 \quad (7.63)$$

which has a double pole at the origin and a pair of complex conjugate poles at  $s = \pm j\omega$  where  $\omega^2 = k \left( \frac{1}{J_\ell} + \frac{1}{J_m} \right)$ . Assuming that the open loop damping

constants  $B_\ell$  and  $B_m$  are small, then the open loop poles of the system (7.57)-(7.58) will be in the left half plane near the poles of the undamped system.



$K_D s$ . At this point zeros are placed on the  $s$ -axis. A D-compensator is used to improve the motor variables. The block diagram of the root locus for the

Fig. 7.24 PD-control with motor angle feedback.

closed loop system in terms of  $K_D$  is shown in Figure 7.25.

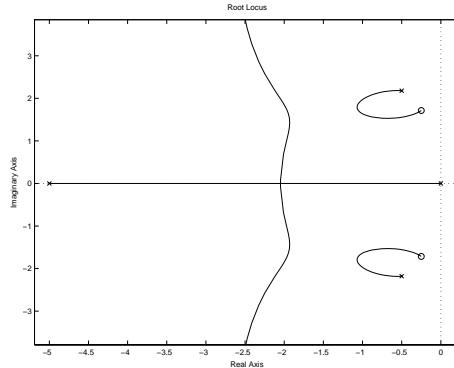


Fig. 7.25 Root locus for the system of Figure 7.24.

We see that the 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 poor overall performance, for example, undesirable oscillations with poor settling time. Also the poor relative stability means that disturbances and other unmodeled dynamics could render the system unstable.

If we measure instead the load angle  $\theta_\ell$ , the system with PD control is represented by the block diagram of Figure 7.26. The corresponding root locus is shown in Figure 7.27. In this case the 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

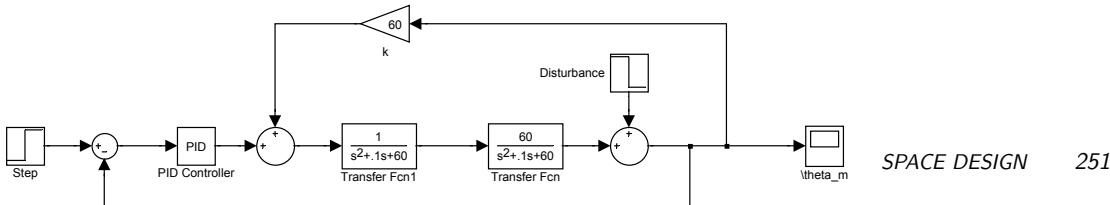


Fig. 7.26 PD-control with load angle feedback.

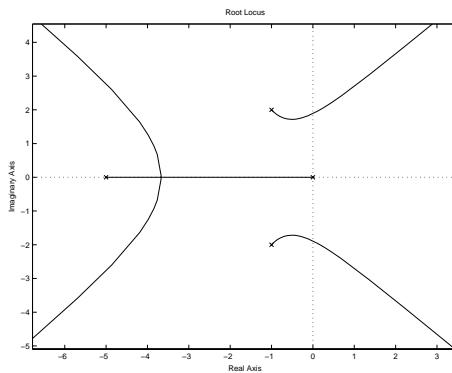


Fig. 7.27 Root locus for the system of Figure 7.22.

unstable, can be found from the Routh 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.

**Example 7.4** Suppose that the system (7.55)-(7.56) has the following parameters (see [1])

$$\begin{aligned} k &= 0.8 \text{ Nm/rad} & B_m &= 0.015 \text{ Nms/rad} \\ J_m &= 0.0004 \text{ Nms}^2/\text{rad} & B_\ell &= 0.0 \text{ Nms/rad} \\ J_\ell &= 0.0004 \text{ Nm}^2/\text{rad} \end{aligned} \quad (7.64)$$

If we implement a PD controller  $K_D(s + a)$  then the response of the system with motor (respectively, load) feedback is shown in Figure 7.28 (respectively, Figure 7.29). ◇

## 7.6 STATE SPACE DESIGN

In this section we consider the application of state space methods for the control of the flexible joint system above. The previous analysis has shown that PD

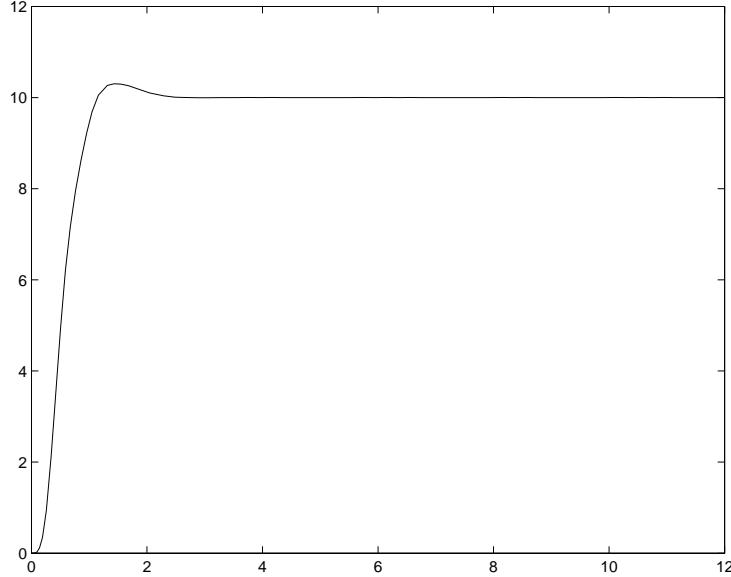


Fig. 7.28 Step response – PD-control with motor angle feedback.

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 lightly damped poles into the closed loop system that result in unacceptable oscillation of the transient response. We can write the system (7.55)-(7.56) in state space by choosing state variables

$$\begin{aligned} x_1 &= \theta_\ell & x_2 &= \dot{\theta}_\ell \\ x_3 &= \theta_m & x_4 &= \dot{\theta}_m. \end{aligned} \quad (7.65)$$

In terms of these state variables the system (7.55)-(7.56) becomes

$$\dot{x}_1 = x_2 \quad (7.66)$$

$$\begin{aligned} \dot{x}_2 &= -\frac{k}{J_\ell}x_1 - \frac{B_\ell}{J_\ell}x_2 + \frac{k}{J_\ell}x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{k}{J_m}x_1 - \frac{B_\ell}{J_m}x_4 - \frac{k}{J_m}x_3 + \frac{1}{J_m}u \end{aligned} \quad (7.67)$$

which, in matrix form, can be written as

$$\dot{x} = Ax + bu \quad (7.68)$$

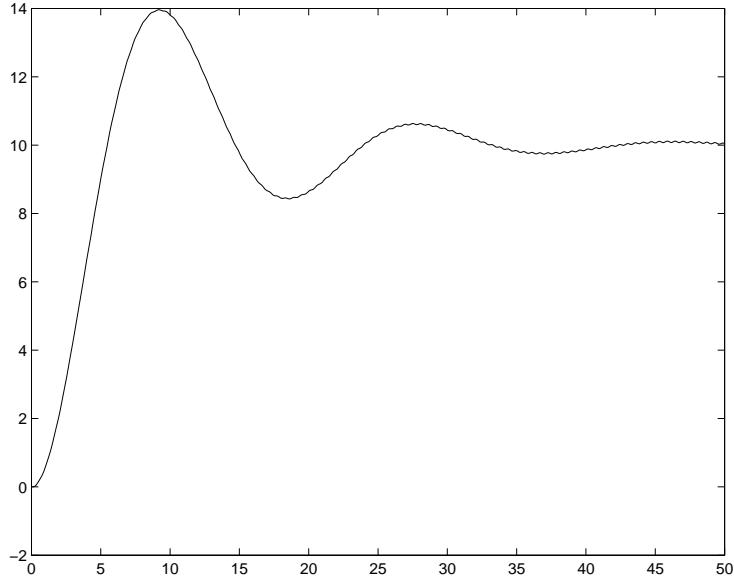


Fig. 7.29 Step response – PD control with load angle feedback.

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{J_\ell} & -\frac{B_\ell}{J_\ell} & \frac{k}{J_\ell} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J_m} & 0 & -\frac{k}{J_m} & \frac{B_m}{J_m} \end{bmatrix}; \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_m} \end{bmatrix}. \quad (7.69)$$

If we choose an output  $y(t)$ , say the measured load angle  $\theta_\ell(t)$ , then we have an output equation

$$y = x_1 = c^T x \quad (7.70)$$

where

$$c^T = [1, 0, 0, 0]. \quad (7.71)$$

The relationship between the state space form (7.68)–(7.70) and the transfer function defined by (7.61) is found by taking Laplace transforms of (7.68)–(7.70) with initial conditions set to zero. This yields

$$G(s) = \frac{\Theta_\ell(s)}{U(s)} = \frac{Y(s)}{U(s)} = c^T (sI - A)^{-1} b \quad (7.72)$$

where  $I$  is the  $n \times n$  identity matrix. The poles of  $G(s)$  are eigenvalues of the matrix  $A$ . In the system (7.68)–(7.70) 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].

### 7.6.1 State Feedback Compensator

Given a linear system in state space form, such as (7.68), a **linear state feedback control law** is an input  $u$  of the form

$$\begin{aligned} u(t) &= -k^T x + r \\ &= -\sum_{i=1}^4 k_i x_i + r \end{aligned} \quad (7.73)$$

where  $k_i$  are constants and  $r$  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-control, which was a function either of the motor position and velocity or of the load position and velocity, but not both. The coefficients  $k_i$  in (7.73) are the gains to be determined. If we substitute the control law (7.73) into (7.68) we obtain

$$\dot{x} = (A - bk^T)x + br. \quad (7.74)$$

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-design the closed loop pole locations were restricted to lie on the root locus shown in Figure 7.25 or 7.27. Since there are more free parameters to choose in (7.73) 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 (7.68) satisfies a property known as **controllability**.

#### (i) Definition 7.4.2

A linear system is said to be **completely state-controllable**, or **controllable** for short, if for each initial state  $x(t_0)$  and each final state  $x(t_f)$  there is a control input  $t \rightarrow u(t)$  that transfers the system from  $x(t_0)$  at time  $t = o$  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. To check whether a system is controllable we have the following simple test.

#### (ii) Lemma 7.4.3

A linear system of the form (7.68) is controllable if and only if

$$\det[b, Ab, A^2b, \dots, A^{n-1}b] \neq 0. \quad (7.75)$$

The  $n \times n$  matrix  $[b, Ab, \dots, A^{n-1}b]$  is called the **controllability matrix** for the linear system defined by the pair  $(A, b)$ . The fundamental importance of controllability of a linear system is shown by the following

**(iii) Theorem 7.4.4**

Let  $\alpha(x) = s^n + \alpha_n s^{n-1} + \cdots + \alpha_2 s + \alpha_1$  be an arbitrary polynomial of degree  $n$ . Then there exists a state feedback control law (7.73) such that

$$\det(sI - A + bk^T) = \alpha(s) \quad (7.76)$$

if and only if the system (7.68) is controllable.

This fundamental result says that, for a controllable linear system, we may achieve **arbitrary**<sup>2</sup> closed loop poles using state feedback. Returning to the specific fourth-order system (7.69) we see that the system is indeed controllable since

$$\det[b, Ab, A^2b, A^3b] = \frac{k^2}{J_m^4 J_\ell^2} \quad (7.77)$$

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.

There are many algorithms that can be used to determine the feedback gains in (7.73) to achieve a desired set of closed loop poles. This is known as the pole assignment problem. In this case most of the difficulty lies in choosing an appropriate set of closed loop poles based on the desired performance, the limits on the available torque, etc. We would like to achieve a fast response from the system without requiring too much torque from the motor. 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

$$J = \int_0^\infty (x^T Q x + R u^2) dt \quad (7.78)$$

where  $Q$  is a given symmetric, positive definite matrix and  $R > O$ .

Choosing a control law to minimize (7.78) frees us from having to decide beforehand what the closed loop poles should be as they are automatically dictated by the weighting matrices  $Q$  and  $R$  in (7.78). It is shown in optimal control texts that the optimum linear control law that minimizes (7.78) is given as

$$u = -k^T x \quad (7.79)$$

where

$$k = R^{-1} b^T P \quad (7.80)$$

<sup>2</sup>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.

and  $P$  is the (unique) symmetric, positive definite  $n \times n$  matrix satisfying the so-called **matrix Algebraic Riccati equation**

$$A^T P + PA - PbR^{-1}b^T P + Q = 0. \quad (7.81)$$

The control law (7.79) is referred to as a **Linear Quadratic (LQ) Optimal Control**, since the performance index is quadratic and the control system is linear.

#### (iv) Example 7.4.5

For illustration purposes, let  $Q$  and  $R$  in (7.78) be given as  $Q = \text{diag}\{100, 0.1, 100, 0.1\}$  and  $R = 100$ . This puts a relatively large weighting on the position variables and control input with smaller weighting on the velocities of the motor and load. Figure 7.30 shows the optimal gains that result and the response of this

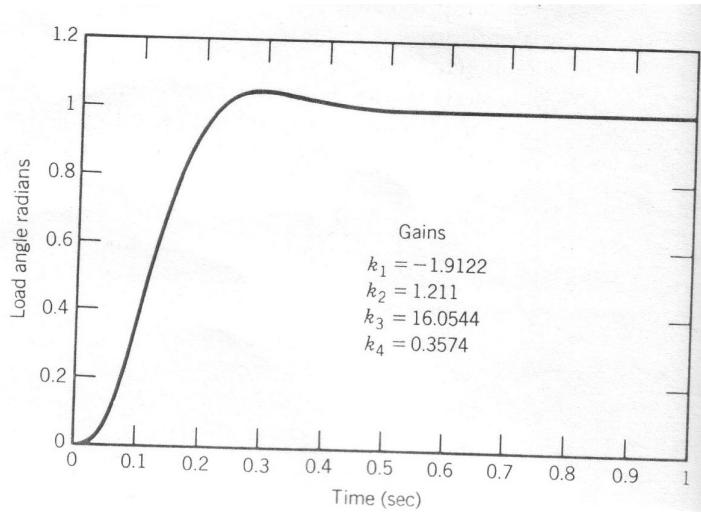


Fig. 7.30 Step response—Linear, Quadratic—Optimal (LQ) state feedback control.

LQ-optimal control for the system (7.66) with a unit step reference input  $r$ .

#### 7.6.2 Observers

The above result is remarkable; however, in order to achieve it, we have had to pay a price, namely, 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  $\theta_\ell$ , we need to introduce the concept of an **observer**. An observer is a state estimator. It is a dynamical system (constructed in software) that attempts to estimate the full state  $x(t)$  using only the system model (7.68)-7.70) 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 (7.68) 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. We could then use this simulated or estimated state, call it  $\hat{x}(t)$ , in place of the true state in (7.79). However, since the true initial condition  $x(t_0)$  for (7.68) will generally be unknown, this idea is not feasible. However the idea of using the model of the system (7.68) is a good starting point to construct a state estimator in software. Let us, therefore, consider an estimate  $\hat{x}(t)$  satisfying the system

$$\dot{\hat{x}} = A\hat{x} + bu + \ell(y - c^T\hat{x}). \quad (7.82)$$

Equation (7.82) is called an **observer** for (7.68) and represents a model of the system (7.68) 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 (7.82) 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 (7.79). The additional term  $\ell$  in (7.82) is to be designed so that  $\hat{x} \rightarrow x$  as  $t \rightarrow \infty$ , 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 (7.68) and (7.82), since  $y = c^T x$ , we see that the estimation error satisfies the system

$$\dot{e} = (A - \ell c^T)e. \quad (7.83)$$

From (7.83) we see that the dynamics of the estimation error are determined by the eigenvalues of  $A - \ell c^T$ . Since  $\ell$  is a design quantity we can attempt to choose it so that  $e(t) \rightarrow 0$  as  $t \rightarrow \infty$ , in which case the estimate  $\hat{x}$  converges to the true state  $x$ . In order to do this we obviously want to choose  $\ell$  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:

### (i) Definition 7.4.6

A linear system is **completely observable**, or **observable** for short, 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  $t_0 \leq t \leq t_f$ .

To check whether a system is observable we have the following

**(ii) Theorem 7.4.7**

The pair  $(A, c)$  is observable if and only if

$$\det [c, A^T c, \dots, A^{t^{n-1}} c] \neq 0. \quad (7.84)$$

The  $n \times n$  matrix  $[c^T, c^T A^T, \dots, c^T A^{t^n} c]$  is called the **observability matrix** for the pair  $(A, c^T)$ . In the system (7.68)-(7.70) above we have that

$$\det [c, A^T c, A^{T^2} c, A^{T^3} c] = \frac{k^2}{J_\ell^2} \quad (7.85)$$

and hence the system is observable. A result known as the **Separation Principle** says that if we use the estimated state in place of the true state in (7.79), then 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  $A - b k^T$ . As the name suggests the Separation Principle allows us to separate the design of the state feedback control law (7.79) from the design of the state estimator (7.82). A typical procedure is to place the observer poles to the left of the desired pole locations of  $A - b k^T$ . 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 (7.79).

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 (7.79) can result in saturation of the input, again resulting in poor performance. Also uncertainties in the system parameters, nonlinearities such as a nonlinear spring characteristic or 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.

---

## Problems

---

- 7-1 Using block diagram reduction techniques derive the transfer functions (7.12) and (7.13).
- 7-2 Derive the transfer functions for the reduced order model (7.14)-(7.15).
- 7-3 Derive Equations (7.32), (7.33) and (7.34).
- 7-4 Derive Equations (7.43)-(7.44).
- 7-5 Derive Equations (7.61), (7.62), and (7.63).
- 7-6 Given the state space model (7.68) show that the transfer function
- $$G(s) = c^T(sI - A)^{-1}b$$
- is identical to (7.61).
- 7-7 Search the control literature (e.g., [8]) and find two or more algorithms for the pole assignment problem for linear systems. Design a state feedback control law for (7.68) using the parameter values given in Example 7.4.1 so that the poles are at  $s = -10$ . Simulate the step response. How does it compare to the response of Example 7.4.1? How do the torque profiles compare?
- 7-8 Design an observer for the system (7.68) using the parameter values of Example 7.4.1. Choose reasonable locations for the observer poles. Simulate the combined observer/state feedback control law using the results of Problem 7-7.
- 7-9 Derive (7.77) and (7.85).
- 7-10 Given a three-link elbow type robot, a three-link SCARA robot and a three-link cartesian robot, discuss the differences in the dynamics of each type of robot as they impact the control problem. Discuss the nature of the coupling nonlinearities, the effect of gravity, and inertial variations as the robots move about. For which manipulator would you expect PD control to work best? worst?
- 7-11 Consider the two-link cartesian robot of Example 6.4.1. Suppose that each joint is actuated by a permanent magnet DC-motor. Write the complete dynamics of the robot assuming perfect gears.
- 7-12 Carry out the details of a PID control for the two-link cartesian robot of Problem 11. Note that the system is linear and the gravitational forces are configuration independent. What does this say about the validity of this approach to control?

- 7-13 Simulate the above PID control law. Choose reasonable numbers for the masses, inertias, etc. Also place reasonable limits on the magnitude of the control input. Use various methods, such as root locus, Bode plots, etc. to design the PID gains.
- 7-14 Search the control literature (e.g., [6]) to find out what is meant by **integrator windup**. Did you experience this problem with the PID control law of Problem 13? Find out what is meant by **anti-windup** (or anti-reset windup). Implement the above PID control with anti-reset windup. Is the response better?
- 7-15 Repeat the above analysis and control design (Problems 11 – 14 for the two-link elbow manipulator of Example 6.4.2. Note that you will have to make some assumptions to arrive at a value of the effective inertias  $J_{eff}$ .
- 7-16 Repeat Problem 15 for the two-link elbow manipulator with remote drive of Example 6.4.3.
- 7-17 Include the dynamics of a permanent magnet DC-motor for the system (7.55)-(7.56). What can you say now about controllability and observability of the system?
- 7-18 Choose appropriate state variables and write the system (7.10)-(7.11) in state space. What is the order of the state space?
- 7-19 Repeat Problem 7-18 for the reduced order system (7.16).
- 7-20 Suppose in the flexible joint system represented by (7.55)-(7.56) the following parameters are given

$$\begin{aligned} J_\ell &= 10 & B_\ell &= 1 & k &= 100 \\ J_m &= 2 & B_m &= 0.5 \end{aligned}$$

- (a) Sketch the open loop poles of the transfer functions (7.61).
- (b) Apply a PD compensator to the system (7.61). 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  $K$  when the root locus crosses the imaginary axis.
- 7-21 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 7.31, 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

$$\begin{aligned} J_1 \ddot{q}_1 &= \tau \\ J_2 \ddot{q}_2 &= \tau \end{aligned}$$

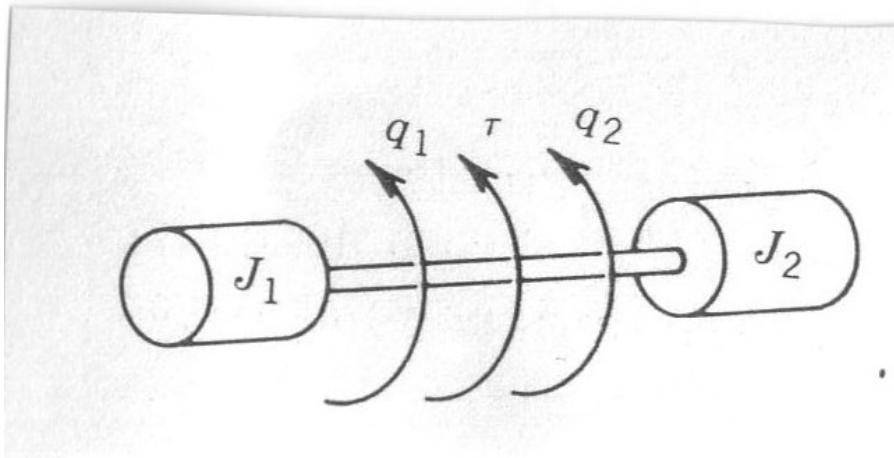


Fig. 7.31 Coupled Inertias in Free Space

Write this system in state space form and show that it is uncontrollable. Discuss the implications of this and suggest possible solutions.

7-22 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  $s = -2, 2$ .

7-23 Repeat the above if possible for the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

Can the closed loop poles be placed at -2?

Can this system be stabilized? Explain.

[**Remark:** The system of Problem 7-23 is said to be *stabilizable*, which is a weaker notion than controllability.]

7-24 Repeat the above for the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} +1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

7-25 Consider the block diagram of Figure 7.18. Suppose that  $G(s) = \frac{1}{2s^2+s'}$  and suppose that it is desired to track a reference signal  $r(t) = \sin(t) + \cos(2t)$ . 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  $F(s)$ .

# 8

---

# MULTIVARIABLE CONTROL

## 8.1 INTRODUCTION

In the previous chapter we discussed techniques to derive a control law 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 arbitrary trajectories.

We first reformulate the manipulator dynamic equations in a form more suitable for the discussion to follow. Recall the robot equations of motion (7.25) and (7.26)

$$\sum_{j=1}^n d_{jk}(q)\ddot{q}_j + \sum_{i,j=1}^n c_{ijk}(q)\dot{q}_i\dot{q}_j + \phi_k = \tau_k \quad (8.1)$$

$$J_{m_k}\ddot{\theta}_{m_k} + B_k\dot{\theta}_{m_k} = K_{m_k}/R_k V_k - \tau_k/r_k. \quad (8.2)$$

where  $B_k = B_{m_k} + K_{b_k}K_{m_k}/R_k$ . Multiplying (8.2) by  $r_k$  and using the fact that

$$\theta_{m_k} = r_k q_k \quad (8.3)$$

we write Equation (8.2) as

$$r_k^2 J_m \ddot{q}_k + r_k^2 B_k \dot{q}_k = r_k K_{m_k} / R V_k - \tau_k \quad (8.4)$$

Substituting (8.4) into (8.1) yields

$$r_k^2 J_{m_k} \ddot{q}_k + \sum_{j=1}^n d_{jk} \ddot{q}_j + \sum_{i,j=1}^n c_{ijk} \dot{q}_i \dot{q}_j + r_k^2 B_k \dot{q}_k + \phi_k = r_k \frac{K_m}{R} V_k. \quad (8.5)$$

In matrix form these equations of motion can be written as

$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + B \dot{q} + g(q) = u \quad (8.6)$$

where  $M(q) = D(q) + J$  where  $J$  is a diagonal matrix with diagonal elements  $r_k^2 J_{m_k}$ . The vector  $g(q)$  and the matrix  $C(q, \dot{q})$  are defined by (6.61) and (6.62), respectively, and the input vector  $u$  has components

$$u_k = r_k \frac{K_{m_k}}{R_k} V_k.$$

Note that  $u_k$  has units of torque.

Henceforth, we will take  $B = 0$  for simplicity in Equation (8.6) and use this equation for all of our subsequent development. We leave it as an exercise for the reader (cf:Problem X) 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 (8.6).

## 8.2 PD CONTROL REVISITED

It is rather remarkable that the simple PD-control scheme for set-point control of rigid robots that we discussed in Chapter 7 can be rigorously shown to work in the general case.<sup>1</sup> An independent joint PD-control scheme can be written in vector form as

$$u = K_P \tilde{q} - K_D \dot{q} \quad (8.7)$$

where  $\tilde{q} = q^d - q$  is the error between the desired joint displacements  $q^d$  and the actual joint displacements  $q$ , 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  $g$  is zero in (8.6), the PD control law (8.7) achieves 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 equations of motion (8.1) are not approximated by a constant disturbance.

<sup>1</sup>The reader should review the discussion on Lyapunov Stability in Appendix C.

To show that the above control law achieves zero steady state error consider the Lyapunov function candidate

$$V = 1/2\dot{q}^T M(q)\dot{q} + 1/2\tilde{q}^T K_P \tilde{q}. \quad (8.8)$$

The first term in (8.8) 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 kinetic energy that would result if the joint actuators were to be replaced by springs with stiffnesses represented by  $K_P$  and with equilibrium positions at  $q^d$ . Thus  $V$  is a positive function except at the “goal”  $q = q^d$ ,  $\dot{q} = 0$ , 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  $J$  and  $q^d$  are constant, the time derivative of  $V$  is given by

$$\dot{V} = \dot{q}^T M(q)\ddot{q} + 1/2\dot{q}^T \dot{D}(q)\dot{q} - \dot{q}^T K_P \tilde{q}. \quad (8.9)$$

Solving for  $M(q)\ddot{q}$  in (8.6) with  $g(q) = 0$  and substituting the resulting expression into (8.9) yields

$$\begin{aligned} \dot{V} &= \dot{q}^T (u - C(q, \dot{q})\dot{q}) + 1/2\dot{q}^T \dot{D}(q)\dot{q} - \dot{q}^T K_P \tilde{q} \\ &= \dot{q}^T (u - K_P \tilde{q}) + 1/2\dot{q}^T (\dot{D}(q) - 2C(q, \dot{q}))\dot{q} \\ &= \dot{q}^T (u - K_P \tilde{q}) \end{aligned} \quad (8.10)$$

where in the last equality we have used the fact (Theorem 6.3.1) that  $\dot{D} - 2C$  is skew symmetric. Substituting the PD control law (8.7) for  $u$  into the above yields

$$\dot{V} = -\dot{q}^T K_D \dot{q} \leq 0. \quad (8.11)$$

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 can 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  $\dot{V} \equiv 0$ . Then (8.11) implies that  $\dot{q} \equiv 0$  and hence  $\ddot{q} \equiv 0$ . From the equations of motion with PD-control

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = -K_P \tilde{q} - K_D \dot{q}$$

must then have

$$0 = -K_P \tilde{q}$$

which implies that  $\tilde{q} = 0$ ,  $\dot{q} = 0$ . LaSalle’s Theorem then implies that the equilibrium is asymptotically stable.

In case there are gravitational terms present in (8.6) Equation (8.10) must be modified to read

$$\dot{V} = \dot{q}^T(u - g(q) - K_P \tilde{q}). \quad (8.12)$$

The presence of the gravitational term in (8.12) 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 the robot configuration  $q$  that is achieved will satisfy

$$K_P(q^d - q) = g(q). \quad (8.13)$$

The physical interpretation of (8.13) is that the configuration  $q$  must be such that the motor generates a steady state “holding torque”  $K_P(q^d - q)$  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

$$u = K_P \tilde{q} - K_D \dot{q} + g(q). \quad (8.14)$$

The modified control law (8.14), in effect, cancels the effect of the gravitational terms and we achieve the same Equation (8.11) as before. The control law (8.14) requires microprocessor implementation to compute at each instant the gravitational terms  $g(q)$  from the Lagrangian equations. In the case that these terms are unknown the control law (8.14) cannot be implemented. We will say more about this and related issues later.

### 8.3 INVERSE DYNAMICS

We now consider the application of more complex nonlinear control techniques for trajectory tracking of rigid manipulators. Consider again the dynamic equations of an  $n$ -link robot in matrix form from (8.6)

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = u. \quad (8.15)$$

The idea of inverse dynamics is to seek a nonlinear feedback control law

$$u = f(q, \dot{q}, t) \quad (8.16)$$

which, when substituted into (8.15), 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 dynamic equations (8.15), however, the problem is actually easy. By inspecting (8.15) we see that if we choose the control  $u$  according to the equation

$$u = M(q)a_q + C(q, \dot{q})\dot{q} + g(q) \quad (8.17)$$

then, since the inertia matrix  $M$  is invertible, the combined system (8.15)-(8.17) reduces to

$$\ddot{q} = a_q \quad (8.18)$$

The term  $a_q$  represents a new input to the system which is yet to be chosen. Equation (8.18) is known as the *double integrator system* as it represents  $n$  uncoupled double integrators. The nonlinear control law (8.17) is called the *inverse dynamics control*<sup>2</sup> and achieves a rather remarkable result, namely that the “new” system (8.18) is linear, and decoupled. This means that each input  $a_{q_k}$  can be designed to control a scalar linear system. Moreover, assuming that  $a_{q_k}$  is a function only of  $q_k$  and its derivatives, then  $a_{q_k}$  will affect  $q_k$  independently of the motion of the other links.

Since  $a_{q_k}$  can now be designed to control a linear second order system, the obvious choice is to set

$$a_q = -K_0 q - K_1 \dot{q} + r \quad (8.19)$$

where  $K_0$  and  $K_1$  are diagonal matrices with diagonal elements consisting of position and velocity gains, respectively. The closed loop system is then the linear system

$$\ddot{q} + K_1 \dot{q} + K_0 q = r. \quad (8.20)$$

Now, given a desired trajectory

$$t \rightarrow (q^d(t), \dot{q}^d(t)). \quad (8.21)$$

if one chooses the reference input  $r(t)$  as<sup>3</sup>

$$r(t) = \ddot{q}^d(t) + K_0 q^d(t) + K_1 \dot{q}^d(t) \quad (8.22)$$

then the tracking error  $e(t) = q - q^d$  satisfies

$$\ddot{e}(t) + K_1 e(t) + K_0 e(t) = 0. \quad (8.23)$$

A simple choice for the gain matrices  $K_0$  and  $K_1$  is

$$\begin{aligned} K_0 &= \text{diag}\{\omega_1^2, \dots, \omega_n^2\} \\ K_1 &= \text{diag}\{2\omega_1, \dots, 2\omega_n\} \end{aligned} \quad (8.24)$$

which results in a closed loop system which is globally decoupled, with each joint response equal to the response of a critically damped linear second order system with natural frequency  $\omega_i$ . As before, the natural frequency  $\omega_i$  determines the

<sup>2</sup>We should point out that in the research literature the control law (8.17) is frequently called *computed torque* as well.

<sup>3</sup>Compare this with the feedforward expression (7.51).

speed of response of the joint, or equivalently, the rate of decay of the tracking error.

The inverse dynamics approach is extremely important as a basis for control of robot manipulators and it is worthwhile trying to see it from alternative viewpoints. We can give a second interpretation of the control law (8.17) as follows. Consider again the manipulator dynamic equations (8.15). Since  $M(q)$  is invertible for  $q \in \mathbb{R}^n$  we may solve for the acceleration  $\ddot{q}$  of the manipulator as

$$\ddot{q} = M^{-1}\{u - C(q, \dot{q})\dot{q} - g(q)\}. \quad (8.25)$$

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

$$\ddot{q}(t) = a_q(t) \quad (8.26)$$

where  $a_q(t)$  is the input acceleration vector. This is again the familiar double integrator system. Note that (8.26) 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 (8.26) is now easy and the acceleration input  $a_q$  can be chosen as before according to (8.19).

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 (8.25) and (8.26) we see that the torque  $u$  and the acceleration  $a_q$  of the manipulator are related by

$$M^{-1}\{u(t) - C(q, \dot{q})\dot{q} - g(q)\} = a_q \quad (8.27)$$

By the invertibility of the inertia matrix we may solve for the input torque  $u(t)$  as

$$u = M(q)a_q + C(q, \dot{q})\dot{q} + g(q) \quad (8.28)$$

which is the same as the previously derived expression (8.17). Thus the inverse dynamics can be viewed as an input transformation which transforms the problem from one of choosing torque input commands, which is difficult, to one of choosing acceleration input commands, which is easy.

Note that the implementation of this control scheme requires the computation at each sample instant of the inertia matrix  $M(q)$  and the vector of Coriolis, centrifugal, and gravitational. Unlike the computed torque scheme (7.53), however, the inverse dynamics *must* be computed on-line. In other words, as a feedback control law, it cannot be precomputed off-line and stored as can the computed torque (7.54). An important issue therefore in the control system

implementation is the design of the computer architecture for the above computations. As processing power continues to increase the computational issues of real-time implementation become less important. An attractive method to implement this scheme is to use a dedicated hardware interface, such as a DSP chip, to perform the required computations in real time. Such a scheme is shown in Figure 8.1.

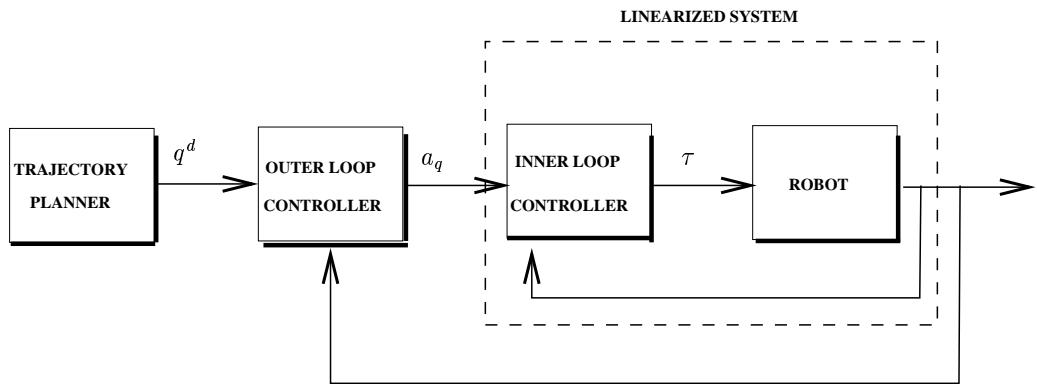


Fig. 8.1 Inner loop/outer-loop control architecture.

Figure 8.1 illustrates the notion of *inner-loop/outer-loop* control. By this we mean that the computation of the nonlinear control (8.17) is performed in an inner loop, perhaps with a dedicated hardwire interface, 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 8.1, which is now a linear or nearly linear system.

### 8.3.1 Task Space Inverse Dynamics

As an illustration of the importance of the inner loop/outer loop paradigm, we will show that tracking in task space can be achieved by modifying our choice of outer loop control  $\ddot{q}$  in (8.18) while leaving the inner loop control unchanged. Let  $X \in R^6$  represent the end-effector pose using any minimal representation of  $SO(3)$ . Since  $X$  is a function of the joint variables  $q \in \mathcal{C}$  we have

$$\dot{X} = J(q)\dot{q} \quad (8.29)$$

$$\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q}. \quad (8.30)$$

where  $J = J_a$  is the analytical Jacobian of section 4.8. Given the double integrator system, (8.18), in joint space we see that if  $a_q$  is chosen as

$$a_q = J^{-1} \{ a_X - \dot{J}\dot{q} \} \quad (8.31)$$

the result is a double integrator system in task space coordinates

$$\ddot{X} = a_X \quad (8.32)$$

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

$$a_X = \ddot{X}^d + K_P(X^d - X) + K_D(\dot{X}^d - \dot{X}) \quad (8.33)$$

so that the Cartesian space tracking error,  $\tilde{X} = X - X^d$ , satisfies

$$\ddot{\tilde{X}} + K_D\dot{\tilde{X}} + K_P\tilde{X} = 0. \quad (8.34)$$

Therefore, a modification of the outer loop control 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  $J$ . In this case

$$V = \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \omega \end{pmatrix} = J(q)\dot{q} \quad (8.35)$$

and the outer loop control

$$a_q = J^{-1}(q)\{ \begin{pmatrix} a_x \\ a_\omega \end{pmatrix} - \dot{J}(q)\dot{q} \} \quad (8.36)$$

applied to (8.18) results in the system

$$\ddot{x} = a_x \in \mathbb{R}^3 \quad (8.37)$$

$$\dot{\omega} = a_\omega \in \mathbb{R}^3 \quad (8.38)$$

$$\dot{R} = S(\omega)R, \quad R \in SO(3), \quad S \in so(3). \quad (8.39)$$

Although, in this latter case, the dynamics have not been linearized to a double integrator, the outer loop terms  $a_v$  and  $a_\omega$  may still be used to defined control laws to track end-effector trajectories in  $SE(3)$ .

In both cases we see that non-singularity of the Jacobian is necessary to implement the outer loop control. 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 pseudoinverse in place of the inverse of the Jacobian. See [?] for details.

## 8.4 ROBUST AND ADAPTIVE MOTION CONTROL

A drawback to the implementation of the inverse dynamics control methodology described in the previous section is the requirement that the parameters of the system be known exactly. If the parameters are not known precisely, for example, when the manipulator picks up an unknown load, then the ideal performance of the inverse dynamics controller is no longer guaranteed. This section is concerned with robust and adaptive motion control of manipulators. The goal of both robust and adaptive control to maintain performance in terms of stability, tracking error, or other specifications, despite parametric uncertainty, external disturbances, unmodeled dynamics, or other uncertainties present in the system. In distinguishing between robust control and adaptive control, we follow the commonly accepted notion that a robust controller is a fixed controller, static or dynamic, designed to satisfy performance specifications over a given range of uncertainties whereas an adaptive controller incorporates some sort of on-line 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. An understanding of the trade-offs involved is therefore important in deciding whether to employ robust or adaptive control design methods in a given situation.

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-1990's during which time researchers first began to exploit fundamental structural properties of manipulator dynamics such as feedback linearizability, passivity, multiple time-scale behavior, and other properties that we discuss below.

### 8.4.1 Robust Feedback Linearization

The feedback linearization approach relies on exact cancellation of nonlinearities in the robot equations of motion. Its practical implementation 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

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = u \quad (8.40)$$

and write the inverse dynamics control input  $u$  as

$$u = \hat{M}(q)a_q + \hat{C}(q, \dot{q})\dot{q} + \hat{g}(q) \quad (8.41)$$

where the notation  $(\hat{\cdot})$  represents the computed or nominal value of  $(\cdot)$  and indicates that the theoretically exact feedback linearization cannot be achieved in practice due to the uncertainties in the system. The error or mismatch  $(\tilde{\cdot}) = (\cdot) - (\hat{\cdot})$  is a measure of one's knowledge of the system dynamics.

If we now substitute (8.41) into (8.40) we obtain, after some algebra,

$$\ddot{q} = a_q + \eta(q, \dot{q}, a_q) \quad (8.42)$$

where

$$\eta = M^{-1}(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{g}) \quad (8.43)$$

is called the *Uncertainty*. We note that

$$M^{-1}\tilde{M} = M^{-1}\hat{M} - I =: E \quad (8.44)$$

and so we may decompose  $\eta$  as

$$\eta = Ea_q + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \quad (8.45)$$

We note that the system (8.42) is still nonlinear and coupled due to the uncertainty  $\eta(q, \dot{q}, a_q)$ . Thus we have no guarantee that the outer loop control given by Equation (8.19) will satisfy desired tracking performance specifications. In this chapter we discuss several design methods to modify the outer loop control (??) to guarantee global convergence of the tracking error for the system (8.42).

**8.4.1.1 Outer Loop Design via Lyapunov's Second Method** There are several approaches to treat the robust feedback linearization problem outlined above. We will discuss only one method, namely the so-called theory of **guaranteed stability of uncertain systems**, which is based on Lyapunov's second method. In this approach we set the outer loop control  $a_q$  as

$$\ddot{q} = \ddot{q}^d(t) + K_P(q^d - q) + K_D(\dot{q}^d - \dot{q}) + \delta a \quad (8.46)$$

In terms of the tracking error

$$e = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix} = \begin{bmatrix} q - q^d \\ \dot{q} - \dot{q}^d \end{bmatrix} \quad (8.47)$$

we may write

$$\dot{e} = Ae + B\{\delta a + \eta\} \quad (8.48)$$

where

$$A = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix}; \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix}. \quad (8.49)$$

Thus the double integrator is first stabilized by the linear feedback,  $-K_P e - K_D \dot{e}$ , and  $\delta a$  is an additional control input that must be designed to overcome

the potentially destabilizing effect of the uncertainty  $\eta$ . The basic idea is to compute a time-varying scalar bound,  $\rho(e, t) \geq 0$ , on the uncertainty  $\eta$ , i.e.,

$$\|\eta\| \leq \rho(e, t) \quad (8.50)$$

and design the additional input term  $\delta a$  to guarantee ultimate boundedness of the state trajectory  $x(t)$  in (8.48).

Returning to our expression for the uncertainty

$$\eta = E\ddot{q} + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \quad (8.51)$$

$$= E\delta a + E(\ddot{q}^d - K_P e - K_D \dot{e}) + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \quad (8.52)$$

we assume a bound of the form

$$\|\eta\| \leq \alpha\|\delta a\| + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3 \quad (8.53)$$

where  $\alpha = \|E\| = \|M^{-1}\hat{M} - I\|$  and  $\gamma_i$  are nonnegative constants. Assuming for the moment that  $\|\delta a\| \leq \rho(e, t)$ , which must then be checked a posteriori, we have

$$\|\eta\| \leq \alpha\rho(e, t) + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3 =: \rho(e, t) \quad (8.54)$$

which defines  $\rho$  as

$$\rho(e, t) = \frac{1}{1-\alpha}(\gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3) \quad (8.55)$$

Since  $K_P$  and  $K_D$  are chosen in so that  $A$  in (8.48) is a Hurwitz matrix, we choose  $Q > 0$  and let  $P > 0$  be the unique symmetric positive definite matrix satisfying the Lyapunov equation,

$$A^T P + P A = -Q. \quad (8.56)$$

Defining the control  $\delta a$  according to

$$\delta a = \begin{cases} -\rho(e, t) \frac{B^T P e}{\|B^T P e\|} & ; \text{ if } \|B^T P e\| \neq 0 \\ 0 & ; \text{ if } \|B^T P e\| = 0 \end{cases} \quad (8.57)$$

it follows that the Lyapunov function  $V = e^T P e$  satisfies  $\dot{V} \leq 0$  along solution trajectories of the system (8.48). To show this result, we compute

$$\dot{V} = -e^T Q e + 2e^T P B \{\delta a + \eta\} \quad (8.58)$$

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 for  $w \neq 0$ , we have

$$\delta a = -\rho \frac{w}{\|w\|} \quad (8.59)$$

and (8.58) becomes, using the Cauchy-Schwartz inequality,

$$w^T(-\rho \frac{w}{\|w\|} + \eta) \leq -\rho \|w\| + \|w\| \|\eta\| \quad (8.60)$$

$$= \|w\|(-\rho + \|\eta\|) \leq 0 \quad (8.61)$$

since  $\|\eta\| \leq \rho$  and hence

$$\dot{V} < -e^T Q e \quad (8.62)$$

and the result follows. Note that  $\|\delta a\| \leq \rho$  as required.

Since the above control term  $\delta a$  is discontinuous on the manifold defined by  $B^T Pe = 0$ , solution trajectories on this manifold are not well defined in the usual sense. One may define solutions in a generalized sense, the so-called **Filipov solutions** [?]. A detailed treatment of discontinuous control systems is beyond the scope of this text. In practice, the discontinuity in the control results in the phenomenon of *chattering*, as the control switches rapidly across the manifold  $B^T Pe = 0$ . One may implement a continuous approximation to the discontinuous control as

$$\delta a = \begin{cases} -\rho(e, t) \frac{B^T Pe}{\|B^T Pe\|} & ; \text{ if } \|B^T Pe\| > \epsilon \\ -\frac{\rho(e, t)}{\epsilon} B^T Pe & ; \text{ if } \|B^T Pe\| \leq \epsilon \end{cases} \quad (8.63)$$

In this case, since the control signal (8.63), a solution to the system (8.48) exists and is *uniformly ultimately bounded* (u.u.b). See Appendix C for the definition of uniform ultimate boundedness.

**Theorem 1** *The origin of the system (8.48) is u.u.b. with respect to the set  $S$ , defined below, using the continuous control law (8.63).*

*Proof:* As before, choose  $V(e) = e^T Pe$  and compute

$$\dot{V} = -e^T Q e + 2w^T(\delta a + \eta) \quad (8.64)$$

$$\leq -e^T Q e + 2w^T(\delta a + \rho \frac{w}{\|w\|}) \quad (8.65)$$

with  $\|w\| = \|B^T Pe\|$  as above.

For  $\|w\| \geq \epsilon$  the argument proceeds as above and  $\dot{V} < 0$ . For  $\|w\| \leq \epsilon$  the second term above becomes

$$\begin{aligned} & 2w^T(-\frac{\rho}{\epsilon} w + \rho \frac{w}{\|w\|}) \\ &= -2\frac{\rho}{\epsilon} \|w\|^2 + 2\rho \|w\| \end{aligned}$$

This expression attains a maximum value of  $\epsilon \frac{\rho}{2}$  when  $\|w\| = \frac{\epsilon}{2}$ . Thus we have

$$\dot{V} = -e^T Q e + \epsilon \frac{\rho}{2} < 0 \quad (8.66)$$

provided

$$-e^T Q e > \epsilon \frac{\rho}{2} \quad (8.67)$$

Using the relationship

$$\lambda_{\min}(Q) \|e\|^2 \leq e^T Q e \leq \lambda_{\max}(Q) \|e\|^2 \quad (8.68)$$

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

$$\lambda_{\min}(Q) \|e\|^2 \geq \epsilon \frac{\rho}{2} \quad (8.69)$$

or, equivalently

$$\|e\| \geq \left( \frac{\epsilon \rho}{2 \lambda_{\min}(Q)} \right)^{\frac{1}{2}} =: \delta \quad (8.70)$$

Let  $S_\delta$  denote the smallest level set of  $V$  containing  $B(\delta)$ , the ball of radius  $\delta$  and let  $B_r$  denote the smallest ball containing  $S_\delta$ . Then all solutions of the closed loop system are u.u.b. with respect to  $S := B_r$ . The situation is shown in Figure 8.2. All trajectories will eventually enter the ball,  $B_r$ ; in fact, all trajectories will reach the boundary of  $S_\delta$  since  $\dot{V}$  is negative definite outside of  $S_\delta$ .

*Fig. 8.2 Uniform Ultimate Boundedness Set*

#### 8.4.2 Passivity Based Robust Control

In this section we derive an alternative robust control algorithm which exploits the passivity and linearity in the parameters of the rigid robot dynamics. This methods are qualitatively different from the previous methods which were based on feedback linearization. In the passivity based approach we modify the inner loop control as

$$u = \hat{M}(q)a + \hat{C}(q, \dot{q})v + \hat{g}(q) - Kr. \quad (8.71)$$

where  $v$ ,  $a$ , and  $r$  are given as

$$\begin{aligned} v &= \dot{q}^d - \Lambda \tilde{q} \\ a &= \dot{v} = \ddot{q}^d - \Lambda \dot{\tilde{q}} \\ r &= \dot{q}^d - v = \dot{\tilde{q}} + \Lambda \tilde{q} \end{aligned}$$

with  $K$ ,  $\Lambda$  diagonal matrices of positive gains. In terms of the linear parametrization of the robot dynamics, the control (8.71) becomes

$$u = Y(q, \dot{q}, a, v)\hat{\theta} - Kr \quad (8.72)$$

and the combination of (8.71) with (8.40) yields

$$M(q)\dot{r} + C(q, \dot{q})r + Kr = Y(\theta - \theta_0). \quad (8.73)$$

Note that, unlike the inverse dynamics control, the modified inner loop control (8.40) does not achieve a linear, decoupled system, even in the known parameter case  $\hat{\theta} = \theta$ .

In the robust passivity based approach of [?], the term  $\hat{\theta}$  in (8.72) is chosen as

$$\hat{\theta} = \theta_0 + u \quad (8.74)$$

where  $\theta_0$  is a fixed nominal parameter vector and  $u$  is an additional control term. The system (8.73) then becomes

$$M(q)\dot{r} + C(q, \dot{q})r + Kr = Y(a, v, q, \dot{q})(\tilde{\theta} + u) \quad (8.75)$$

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

$$\|\tilde{\theta}\| = \|\theta_0 - \theta\| \leq \rho, \quad (8.76)$$

then the additional term  $u$  can be designed according to the expression,

$$u = \begin{cases} -\rho \frac{Y^T r}{\|Y^T r\|} & ; \text{ if } \|Y^T r\| > \epsilon \\ -\frac{\rho}{\epsilon} Y^T r & ; \text{ if } \|Y^T r\| \leq \epsilon \end{cases} \quad (8.77)$$

The Lyapunov function

$$V = \frac{1}{2} r^T M(q)r + \tilde{q}^T \Lambda K \tilde{q} \quad (8.78)$$

is used to show uniform ultimate boundedness of the tracking error. Calculating  $\dot{V}$  yields

$$\dot{V} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + 2\tilde{q}^T \Lambda K \dot{q} \quad (8.79)$$

$$= -r^T Kr + 2\tilde{q}^T \Lambda K \dot{q} + \frac{1}{2} r^T (\dot{M} - 2C)r + r^T Y(\tilde{\theta} + u) \quad (8.80)$$

Using the passivity property and the definition of  $r$ , this reduces to

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \tilde{q}^T K \dot{q} + r^T Y(\tilde{\theta} + u) \quad (8.81)$$

Defining  $w = Y^T r$  and

$$Q = \begin{bmatrix} \Lambda^T K \Lambda & 0 \\ 0 & \Lambda K \end{bmatrix} \quad (8.82)$$

and mimicking the argument in the previous section, we have

$$\dot{V} = -e^T Q e + w^T (u + \tilde{\theta}) \quad (8.83)$$

$$= -e^T Q e + w^T (u + \rho \frac{w}{\|w\|}) \quad (8.84)$$

Uniform ultimate boundedness of the tracking error follows with the control  $u$  from (8.77) exactly as in the proof of Theorem 1.

Comparing this approach with the approach in the section (8.4.1), we see that finding a constant bound  $\rho$  for the constant vector  $\tilde{\theta}$  is much simpler than finding a time-varying bound for  $\eta$  in (8.43). The bound  $\rho$  in this case depends only on the inertia parameters of the manipulator, while  $\rho(x, t)$  in (8.50) depends on the manipulator state vector, the reference trajectory and, in addition, requires some assumptions on the estimated inertia matrix  $\hat{M}(q)$ .

#### 8.4.3 Passivity Based Adaptive Control

In the adaptive approach the vector  $\hat{\theta}$  in (8.72) is taken to be a time-varying estimate of the true parameter vector  $\theta$ . Combining the control law (8.71) with (8.40) yields

$$M(q)\dot{r} + C(q, \dot{q})r + Kr = Y\tilde{\theta} \quad (8.85)$$

The parameter estimate  $\hat{\theta}$  may be computed using standard methods such as gradient or least squares. For example, using the gradient update law

$$\dot{\hat{\theta}} = -\Gamma^{-1}Y^T(q, \dot{q}, a, v)r \quad (8.86)$$

together with the Lyapunov function

$$V = \frac{1}{2}r^T M(q)r + \tilde{q}^T \Lambda K \tilde{q} + \frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta} \quad (8.87)$$

results in global convergence of the tracking errors to zero and boundedness of the parameter estimates.

To show this, we first note an important difference between the adaptive control approach and the robust control approach from the previous section. In the robust approach the state vector of the system is  $(\tilde{q}, \dot{\tilde{q}})^T$ . In the adaptive control approach, the fact that  $\hat{\theta}$  satisfies the differential equation (8.86)<sup>4</sup> means that the complete state vector now includes  $\tilde{\theta}$  and the state equations are given by the couple system (8.85)-(8.86). For this reason we included the positive definite term  $\frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta}$  in the Lyapunov function (8.87).

If we now compute  $\dot{V}$  along trajectories of the system (8.85), we obtain

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \tilde{q}^T K \dot{\tilde{q}} + \tilde{\theta}^T \{\Gamma \dot{\tilde{\theta}} + Y^T r\} \quad (8.88)$$

<sup>4</sup>Note that  $\dot{\tilde{\theta}} = \dot{\hat{\theta}}$  since the parameter vector  $\theta$  is constant

Substituting the expression for  $\dot{\hat{\theta}}$  from gradient update law (8.86) into (8.88) yields

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} = -e^T Q e \leq 0 \quad (8.89)$$

where  $e$  and  $Q$  are defined as before, showing that the closed loop system is stable in the sense of Lyapunov.

**Remark 8.1** Note that we have claimed only that the Lyapunov function is negative semi-definite, not negative definite since  $\dot{V}$  does not contain any terms that are negative definite in  $\hat{\theta}$ . In fact, this situation is common in most gradient based adaptive control schemes and is a fundamental reason for several difficulties that arise in adaptive control such as lack of robustness to external disturbances and lack of parameter convergence. A detailed discussion of these and other problems in adaptive control is outside the scope of this text.

Returning to the problem at hand, although we conclude only stability in the sense of Lyapunov for the closed loop system (8.85)–(8.86), further analysis will allow to reach stronger conclusions. First, note that since  $\dot{V}$  is nonincreasing, 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  $\hat{\theta}$  are bounded as functions of time,  $t$ . Thus we immediately conclude that the parameter estimation error is bounded.

With regard to the tracking error,  $\tilde{q}$ , we also note that,  $\dot{V}$  is not simply negative but also quadratic in the error vector  $e(t)$ . Integrating both sides of Equation (8.89) gives

$$V(t) - V(0) = - \int_0^t e^t(\sigma) Q e(\sigma) d\sigma < \infty \quad (8.90)$$

As a consequence,  $V$  is a so-called *square integrable function*. Such functions, under some mild additional restrictions must tend to zero as  $t \rightarrow \infty$ . Specifically, we may appeal to the following lemma[?]

**Lemma 8.1** Suppose  $f : \mathbb{R} \mapsto \mathbb{R}$  is a square integrable function and further suppose that its derivative  $\dot{f}$  is bounded. Then  $f(t) \rightarrow 0$  as  $t \rightarrow \infty$ .

We note that, 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  $\tilde{q} \rightarrow 0$  as  $t \rightarrow \infty$ .

To show that the velocity tracking error also converges to zero, one must appeal to the equations of motion (8.85). From Equation (8.85) one may argue (Problem ??) that the acceleration  $\ddot{q}$  is bounded. The result will follow assuming that the reference acceleration  $\ddot{q}^d(t)$  is also bounded (Problem ??).

---

## Problems

---

- 8-1 Form the Lagrangian for an  $n$ -link manipulator with joint flexibility using (??)-(??). From this derive the dynamic equations of motion (??)-(??).
- 8-2 Complete the proof of stability of PD-control for the flexible joint robot without gravity terms using (??) and LaSalle's Theorem.
- 8-3 Suppose that the PD control law (??) is implemented using the link variables, that is,

$$u = K_p q_1 - K_D \dot{q}_1.$$

What can you say now about stability of the system? **Note:** This is a difficult problem. Try to prove the following conjecture: Suppose that  $B_2 = 0$  in (??). Then with the above PD control law, the system is unstable. [Hint: Use Lyapunov's First Method, that is, show that the equilibrium is unstable for the linearized system.]

- 8-4 Using the control law (??) for the system (??)-(??), what is the steady state error if the gravity terms are present?
- 8-5 Simulate an inverse dynamics control law for a two-link elbow manipulator whose equations of motion were derived in Chapter ???. Investigate what happens if there are bounds on the maximum available input torque.
- 8-6 For the system of Problem 8-5 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.
- 8-7 Add an outer loop correction term  $\Delta v$  to the control law of Problem 8-6 to overcome the effects of uncertainty. Base your design on the Second Method of Lyapunov as in Section ??.
- 8-8 Consider the coupled nonlinear system

$$\begin{aligned} \ddot{y}_1 + 3y_1 y_2 + y_2^2 &= u_1 + y_2 u_2 \\ \ddot{y}_2 + \cos y_1 \dot{y}_2 + 3(y_1 - y_2) &= u_2 - 3(\cos y_1)^2 y_2 u_1 \end{aligned}$$

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

---

# FORCE CONTROL

## 9.1 INTRODUCTION

In previous chapters we considered the problem of tracking motion trajectories using a variety of basic and advanced control methods. Such position control schemes are adequate for tasks such as materials transfer 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*<sup>1</sup> 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.

<sup>1</sup>Hereafter we use *force* to mean force and/or torque and *position* to mean position and/or orientation.

A force control strategy is one that modifies position trajectories based on the sensed force. 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 9.1 usually consists of an array of strain

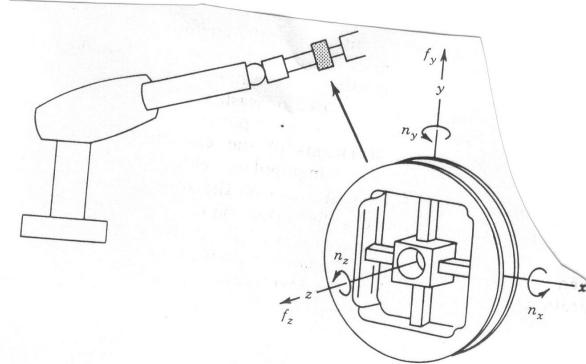


Fig. 9.1 A Wrist Force Sensor.

gauges and can delineate the three components of the vector force along the three axes of the sensor coordinate frame, and the three 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.

## 9.2 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 9.2, 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.

In order to describe the robot/environment interaction, let  $V = (v, \omega)$  represent the instantaneous linear and angular velocity of the end-effector and let  $F = (f, n)$  represent the instantaneous force and moment. The vectors  $V$  and  $F$

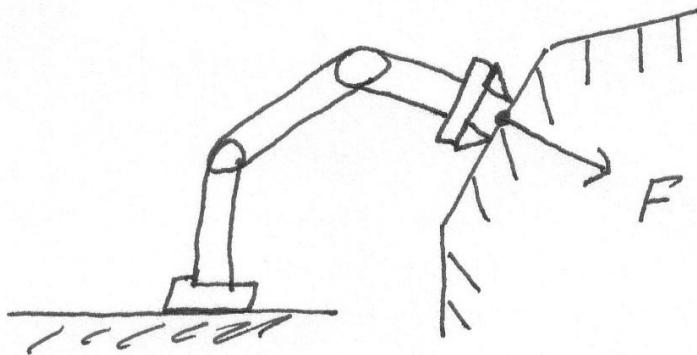


Fig. 9.2 Robot End-Effector in contact with a Rigid Surface

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  $V$  and  $F$  are called *Twists* and *Wrenches* in more advanced texts [?] although we will continue to refer to them simply as velocity and force for simplicity. If  $e_1, \dots, e_6$  is a basis for the vector space  $\text{so}(3)$ , and  $f_1, \dots, f_6$  is a basis for  $\text{so}^*(3)$ , we say that these basis vectors are *reciprocal* provided

$$\begin{aligned} e_i f_j &= 0 \quad \text{if } i \neq j \\ e_i f_j &= 1 \quad \text{if } i = j \end{aligned}$$

We can define the product of  $V \in \text{so}(3)$  and  $F \in \text{so}^*(3)$  in the usual way as

$$V \cdot F = V^T F = v^T f + \omega^T n \quad (9.1)$$

The advantage of using reciprocal basis vectors is that the product  $V^T F$  is then invariant with respect to a linear change of basis from one reciprocal coordinate system to another. We note that expressions such as  $V_1^T V_2$  or  $F_1^T F_2$  for vectors  $V_i, F_i$  belonging to  $\text{so}(3)$  and  $\text{so}^*(3)$ , respectively, are not necessarily well defined. For example, the expression

$$V_1^T V_2 = v_1^T v_2 + \omega_1^T \omega_2 \quad (9.2)$$

is not invariant with respect to either choice of units or basis vectors in  $\text{so}(3)$ . It is possible to define inner product like operations, i.e. symmetric, bilinear forms on  $\text{so}(3)$  and  $\text{so}^*(3)$ , which have the necessary invariance properties. These are the so-called *Klein Form*,  $KL(V_1, V_2)$ , and *Killing Form*,  $KI(V_1, V_2)$ , defined according to

$$KL(V_1, V_2) = v_1^T \omega_2 + \omega_1^T v_2 \quad (9.3)$$

$$KI(V_1, V_2) = \omega_1^T \omega_2 \quad (9.4)$$

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.

**Example 9.1** [?] Suppose that

$$\begin{aligned} V_1 &= (1, 1, 1, 2, 2, 2)^T \\ V_2 &= (2, 2, 2, -1, -1, -1)^T \end{aligned}$$

where the linear velocity is in meters/sec and angular velocity is in radians/sec. The clearly,  $V_1^T V_2 = 0$  and so one could infer that  $V_1$  and  $V_2$  are orthogonal vectors in  $so(3)$ . However, suppose now that the linear velocity is represented in units of centimeters/sec. Then

$$\begin{aligned} V_1 &= (1 \times 10^2, 1 \times 10^2, 1 \times 10^2, 2, 2, 2)^T \\ V_2 &= (2 \times 10^2, 2 \times 10^2, 2 \times 10^2, -1, -1, -1)^T \end{aligned}$$

and clearly  $V_1^T V_2 \neq 0$ . Thus, usual notion of orthogonality is not meaningful in  $so(3)$ . It is easy to show that the equality  $KL(V_1, V_2) = 0$  (respectively,  $KI(V_1, V_2) = 0$ ) is independent of the units or the basis chosen to represent  $V_1$  and  $V_2$ . For example, the condition  $KI(V_1, V_2) = 0$  means that the axes of rotation defining  $\omega_1$  and  $\omega_2$  are orthogonal.  $\diamond$

We shall see in specific cases below that the reciprocity relation (9.1) may be used to design reference inputs to execute motion and force control tasks.

### 9.2.1 Natural and Artificial Constraints

In this section we discuss so-called *Natural Constraints* which are defined using the reciprocity condition (9.1). 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 would then be 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 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.

Figure 9.3 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  $so(3)$  and  $so^*(3)$ , in which case

$$V^T F = v_x f_x + v_y f_y + v_z f_z + \omega_x n_x + \omega_y n_y + \omega_z n_z \quad (9.5)$$

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

$$\begin{aligned} v_x &= 0 & v_y &= 0 & f_z &= 0 \\ \omega_x &= 0 & \omega_y &= 0 & n_z &= 0 \end{aligned} \quad (9.6)$$

and thus the reciprocity condition  $V^T F = 0$  is satisfied. These relationships (9.6) are termed *Natural Constraints*.

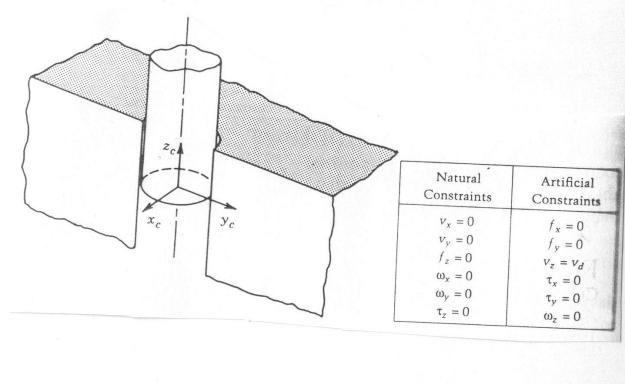


Fig. 9.3 Inserting a peg into a hole.

Examining Equation (9.5) we see that the variables

$$f_x \quad f_y \quad v_z \quad n_x \quad n_y \quad \omega_z \quad (9.7)$$

are unconstrained by the environment, i.e. given the natural constraints (9.6), the reciprocity condition  $V^T F = 0$  holds for all values of the above variables (9.7). We may therefore assign reference values, called *Artificial Constraints*, for these variables that may then be enforced by the control system to carry out the task at hand. For example, in the peg-in-hole task we may define artificial constraints as

$$\begin{aligned} f_x &= 0 & f_y &= 0 & v_z &= v^d \\ n_x &= 0 & n_y &= 0 & \omega_z &= 0 \end{aligned} \quad (9.8)$$

where  $v^d$  is the desired speed of insertion of the peg in the  $z$ -direction.

Figures 9.4 and 9.5 show natural and artificial constraints for two additional tasks, that of turning a crank and and turning a screw, respectively.

### 9.3 NETWORK MODELS AND IMPEDANCE

The reciprocity condition  $V^T F = 0$  means that the forces of constraint do no work in directions compatible with motion constraints and holds under the ideal

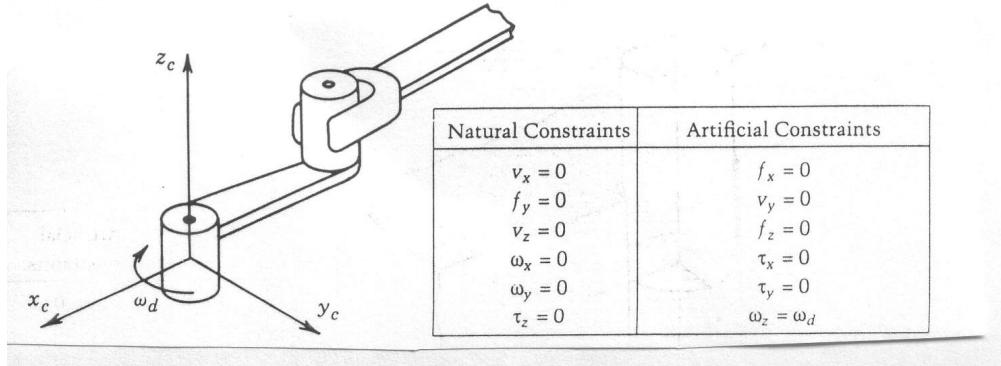


Fig. 9.4 Turning a crank

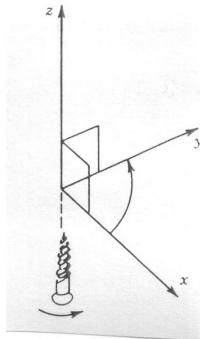


Fig. 9.5 Turning a screw.

conditions of no friction and perfect rigidity of both the robot and 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 9.6. Since the environment deforms in response to a force, there is clearly both motion and force in the same direction, i.e. normal to the surface. Thus the product  $V(t)F(t)$  along this direction will not be zero. Let  $k$  represent the stiffness of the surface so that  $f = kx$ . Then

$$\int_0^t V(u)F(u)du = \int_0^t \dot{x}(u)kx(u)du = k \int_0^t \frac{d}{du} \frac{1}{2}kx^2(u)du = \frac{1}{2}k(x^2(t) - x^2(0)) \quad (9.9)$$

is the change of the potential energy. 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.

In this section we introduce the notion of *Mechanical Impedance* which captures the relation between force and motion. We introduce so-called *Network*

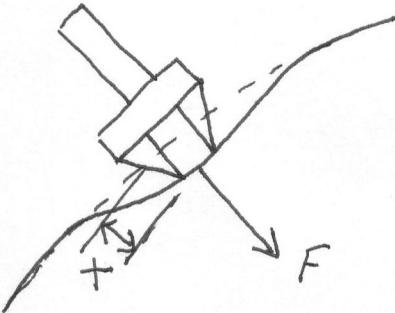


Fig. 9.6 Compliant Environment

*Models*, which are particularly useful for modeling the robot/environment interaction. We model the robot and environment as *One Port Networks* as shown in Figure 9.7. The dynamics of the robot and environment, respectively, determine the relation between the *Port Variables*,  $V_r$ ,  $F_r$ , and  $V_e$ ,  $F_e$ , respectively.  $F_r$ ,  $F_e$  are known as *Effort* or *Across* variables while  $V_r$ ,  $V_e$  are known as *Flow* or *Through* variables. In a mechanical system, such as a robot, force and velocity are the effort and flow variables while in an electrical system, voltage and current are the effort and flow variables, respectively. With this description,

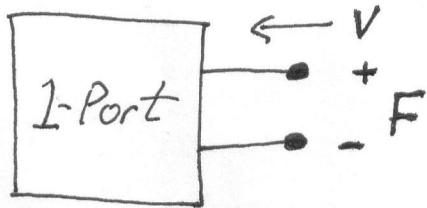


Fig. 9.7 One-Port Networks

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 9.8, which describes the energy exchange between the robot and the environment.

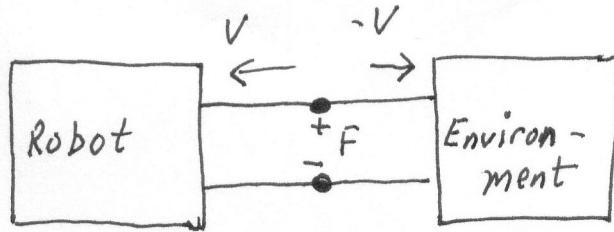


Fig. 9.8 Robot/Environment Interaction

### 9.3.1 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 9.1** Given the one-port network 9.7 the Impedance,  $Z(s)$  is defined as the ratio of the Laplace Transform of the effort to the Laplace Transform of the flow, i.e.

$$Z(s) = \frac{F(s)}{V(s)} \quad (9.10)$$

**Example 9.2** Suppose a mass-spring-damper system is described by the differential equation

$$M\ddot{x} + B\dot{x} + Kx = F \quad (9.11)$$

Taking Laplace Transforms of both sides (assuming zero initial conditions) it follows that

$$Z(s) = F(s)/V(s) = Ms + B + K/s \quad (9.12)$$

◊

### 9.3.2 Classification of Impedance Operators

**Definition 9.2** 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  $|Z(0)| = \infty$

Thus we classify impedance operators based on their low frequency or *DC-gain*, which will prove useful in the steady state analysis to follow.

**Example 9.3** Figure 9.9 shows examples of environment types. Figure 9.9(a) shows a mass on a frictionless surface. The impedance is  $Z(s) = Ms$ , which is inertial. Figure 9.9(b) shows a mass moving in a viscous medium with resistance  $B$ . Then  $Z(s) = Ms + B$ , which is resistive. Figure 9.9(c) shows a linear spring with stiffness  $K$ . Then  $Z(s) = K/s$ , which is capacitive.  $\diamond$

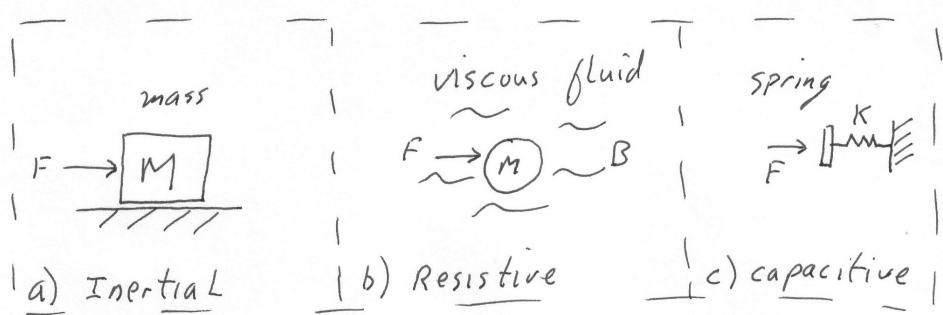


Fig. 9.9 Inertial, Resistive, and Capacitive Environment Examples

### 9.3.3 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 active 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

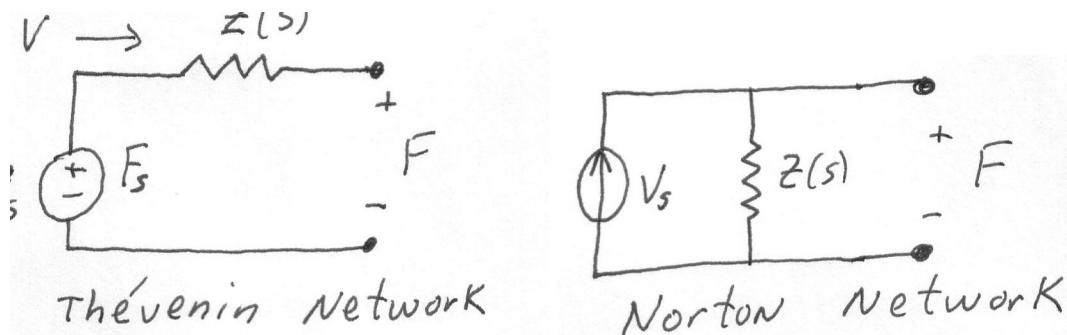


Fig. 9.10 Thévenin and Norton Equivalent Networks

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.

## 9.4 TASK SPACE DYNAMICS AND CONTROL

Since a manipulator task specification, such as grasping an object, or inserting a peg into a hole, is typically given relative to the end-effector frame, it is natural to derive the control algorithm directly in the task space rather than joint space coordinates.

### 9.4.1 Static Force/Torque Relationships

Interaction of the manipulator with the environment will produce forces and moments at the end-effector or tool. Let  $F = (F_x, F_y, F_z, n_x, n_y, n_z)^T$  represent the vector of forces and torques at the end-effector, expressed in the tool frame. Thus  $F_x, F_y, F_z$  are the components of the force at the end-effector, and  $n_x, n_y, n_z$  are the components of the torque at the end-effector.

Let  $\tau$  denote the vector of joint torques, and let  $\delta X$  represent a virtual end-effector displacement caused by the force  $F$ . Finally, let  $\delta q$  represent the corresponding virtual joint displacement. These virtual displacements are related through the manipulator Jacobian  $J(q)$  according to

$$\delta X = J(q)\delta q. \quad (9.13)$$

The virtual work  $\delta w$  of the system is

$$\delta w = F^T \delta X - \tau^T \delta q. \quad (9.14)$$

Substituting (9.13) into (9.14) yields

$$\delta w = (F^T J - \tau^T) \delta q \quad (9.15)$$

which is equal to zero if the manipulator is in equilibrium. Since the generalized coordinates  $q$  are independent we have the equality

$$\tau = J(q)^T F. \quad (9.16)$$

In other words the end-effector forces are related to the joint torques by the transpose of the manipulator Jacobian according to (9.16).

**Example 9.4** Consider the two-link planar manipulator of Figure 9.11, with a force  $F = (F_x, F_y)^T$  applied at the end of link two as shown. The Jacobian of this manipulator is given by Equation (4.86). The resulting joint torques

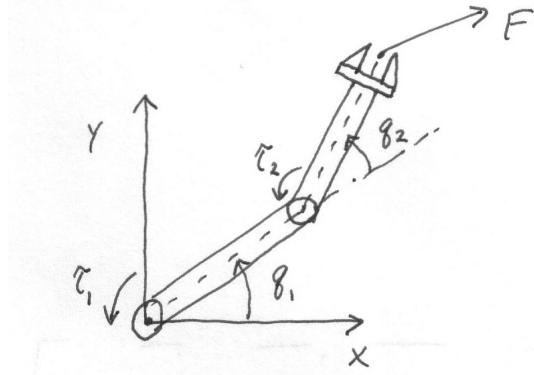


Fig. 9.11 Two-link planar robot.

$\tau = (\tau_1, \tau_2)$  are then given as

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} & 0 & 0 & 0 & 1 \\ -a_2 s_{12} & a_2 c_{12} & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \\ n_x \\ n_y \\ n_z \end{bmatrix} \quad (9.17)$$

◇

#### 9.4.2 Task Space Dynamics

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  $F_e$ . Thus the equations of motion of the manipulator in joint space are given by

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + J^T(q)F_e = u \quad (9.18)$$

Let us consider a modified inverse dynamics control law of the form

$$u = M(q)a_q + C(q, \dot{q})\dot{q} + g(q) + J^T(q)a_f \quad (9.19)$$

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 8

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \quad (9.20)$$

$$a_x = J(q)a_q + \dot{J}(q)\dot{q} \quad (9.21)$$

we substitute (9.19)-(9.21) into (9.18) to obtain

$$\ddot{x} = a_x + W(q)(F_e - a_f) \quad (9.22)$$

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,  $F_e$  and thus recover the task space double integrator system

$$\ddot{x} = a_x \quad (9.23)$$

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.

#### 9.4.3 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 9.5** Consider the one-dimensional system in Figure 9.12 consisting

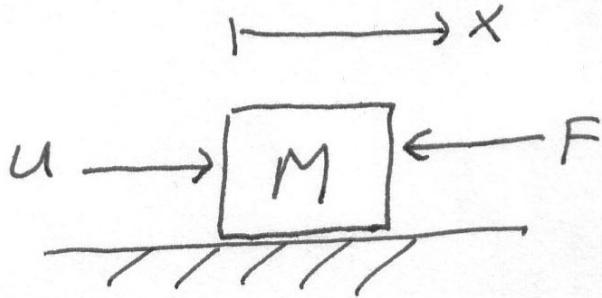


Fig. 9.12 One Dimensional System

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

$$M\ddot{x} = u - F \quad (9.24)$$

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

$$M\ddot{x} = -(1+m)F \implies \frac{M}{1+m}\ddot{x} = -F \quad (9.25)$$

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.  $\diamond$

The idea behind Impedance Control is to regulate the mechanical impedance, i.e., the apparent inertia, damping, and stiffness, through force feedback as in the above example. For example, 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.

We may formulate Impedance Control within our standard inner loop/outer loop control architecture by specifying the outer loop term,  $a_x$ , in Equation (9.23). Let  $x^d(t)$  be a reference trajectory defined in task space coordinates and let  $M_d$ ,  $B_d$ ,  $K_d$ , be  $6 \times 6$  matrices specifying desired inertia, damping, and stiffness, respectively. Let  $e(t) = x(t) - x^d(t)$  be the tracking error in task space and set

$$a_x = \ddot{x}^d - M_d^{-1}(B_d\dot{e} + K_d e + F) \quad (9.26)$$

where  $F$  is the measured environmental force. Substituting (9.26) into (9.23) yields the closed loop system

$$M_d\ddot{e} + B_d\dot{e} + K_d e = -F \quad (9.27)$$

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 necessarily achieved. We will address this difficulty in the next section.

#### 9.4.4 Hybrid Impedance Control

In this section we introduce the notion of *Hybrid Impedance Control* following the treatment of [?]. We again take as our starting point the linear, decoupled system (9.23). 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 regulate both position and impedance simultaneously which is not possible with the pure impedance control law (9.26).

We consider a one-dimensional system representing one component of the outer loop system (9.23)

$$\ddot{x}_i = a_{x_i} \quad (9.28)$$

and we henceforth drop the subscript,  $i$ , for simplicity. We assume that the impedance of the environment in this direction,  $Z_e$  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<sup>2</sup>.
2. Represent the robot impedance as the *Dual* to the environment impedance. Thévenin and Norton networks are considered dual to one another.
3. In the case that the environment impedance is capacitive we have the robot/environment interconnection as shown in Figure 9.13 where the

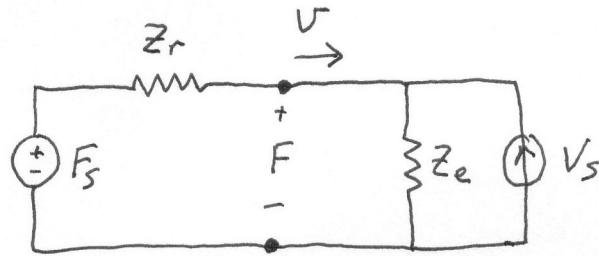


Fig. 9.13 Capacitive Environment Case

environment one-port is the Norton network and the robot on-port is the Thévenin network. Suppose that  $V_s = 0$ , i.e. there are no environmental disturbances, and that  $F_s$  represents a reference force. From the circuit diagram it is straightforward to show that

$$\frac{F}{F_s} = \frac{Z_e(s)}{Z_e(s) + Z_r(s)} \quad (9.29)$$

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

$$e_{ss} = \frac{-Z_r(0)}{Z_r(0) + Z_e(0)} = 0 \quad (9.30)$$

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 (9.28) using only position, velocity, and force feedback. This imposes a practical limitation on the achievable robot impedance functions,  $Z_r$ .

<sup>2</sup>In fact, for a resistive environment, either representation may be used

Suppose  $Z_r^{-1}$  has relative degree one. This means that

$$Z_r(s) = M_c s + Z_{rem}(s) \quad (9.31)$$

where  $Z_{rem}(s)$  is a proper rational function. If we now choose

$$a_x = -\frac{1}{M_c} Z_{rem} \dot{x} + \frac{1}{m_c} (F_s - F) \quad (9.32)$$

Substituting this into the double integrator system  $\ddot{x} = a_x$  yields

$$Z_r(s) \dot{x} = F_s - F \quad (9.33)$$

Thus we have shown that, for a capacitive environment, force feedback can be used to regulate contact force and specify a desired robot impedance.

4. In the case that the environment impedance is inertial we have the robot/environment interconnection as shown in Figure 9.14 where the environment one-port

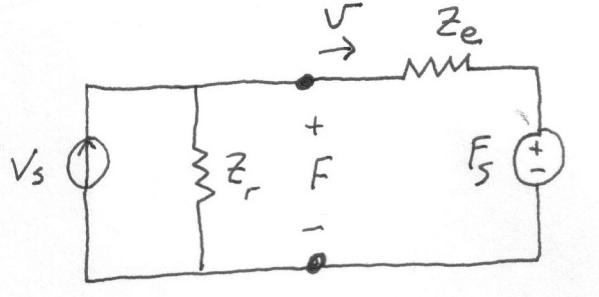


Fig. 9.14 Inertial Environment Case

is a Thévenin network and the robot on-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 show that

$$\frac{V}{V_s} = \frac{Z_r(s)}{Z_e(s) + Z_r(s)} \quad (9.34)$$

Then the steady state force error,  $e_{ss}$ , to a step reference velocity command,  $V_s = \frac{V^d}{s}$  is given by the *Final Value Theorem* as

$$e_{ss} = \frac{-Z_e(0)}{Z_r(0) + Z_e(0)} = 0 \quad (9.35)$$

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,

$$Z_r(s) = M_c s + Z_{rem}(s) \quad (9.36)$$

and set

$$a_x = \ddot{x}^d + \frac{1}{M_c} Z_{rem}(\dot{x}^d - \dot{x}) + \frac{1}{M_c} F \quad (9.37)$$

Then, substituting this into the double integrator equation,  $\ddot{x} = a_x$ , yields

$$Z_r(s)(\dot{x}^d - x) = F \quad (9.38)$$

Thus we have shown that, for a capacitive environment, position control can be used to regulate a motion reference and specify a desired robot impedance.

---

## Problems

---

- 9-1 Given the two-link planar manipulator of Figure 9.11, find the joint torques  $\tau_1$  and  $\tau_2$  corresponding to the end-effector force vector  $(-1, -1)^T$ .
- 9-2 Consider the two-link planar manipulator with remotely driven links shown in Figure 9.15. Find an expression for the motor torques needed to bal-

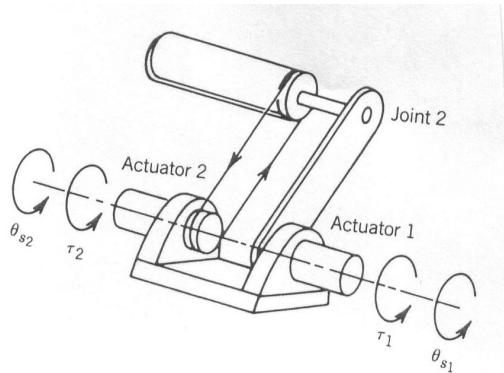


Fig. 9.15 Two-link manipulator with remotely driven link.

- ance a force  $F$  at the end-effector. Assume that the motor gear ratios are  $r_1, r_2$ , respectively.
- 9-3 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.
- 9-4 Describe the natural and artificial constraints associated with the task of opening a box with a hinged lid. Sketch the compliance frame.
- 9-5 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.
- 9-6 Given the following tasks, classify the environments as either *Inertial*, *Capacitive*, or *Resistive* according to Definition 9.2
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

# 10

---

## *GEOMETRIC NONLINEAR CONTROL*

### 10.1 INTRODUCTION

In this chapter we present some basic, but fundamental, ideas from **geometric nonlinear control theory**. We first discuss the notion of **feedback linearization of nonlinear systems**. This approach generalizes the concept of inverse dynamics of rigid manipulators discussed in Chapter 8. The basic idea of feedback linearization is to construct a nonlinear control law as a so-called **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 a second stage or **outer loop control** in the new coordinates to satisfy the traditional control design specifications such as tracking, disturbance rejection, and so forth.

In the case of rigid manipulators the inverse dynamics control of Chapter 8 and the feedback linearizing control are the same. However, 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.

We also give an introduction to modeling and controllability of nonholonomic systems. We treat systems such as mobile robots and other systems subject to constraints arising from conservation of angular momentum or rolling contact. We discuss the controllability of a particular class of such systems, known as **driftless systems**. We present a result known as **Chow's Theorem**, which gives a sufficient condition for local controllability of driftless systems.

## 10.2 BACKGROUND

In this section we give some background from differential geometry that is necessary to understand the results to follow. 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, etc. The reader is referred to [?] for a comprehensive treatment of the subject. 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.

The fundamental notion in differential geometry is that of a **differentiable manifold** (manifold for short) which is a topological space that is locally diffeomorphic<sup>1</sup> 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<sup>2</sup>  $h =: \mathbb{R}^n \rightarrow \mathbb{R}^p$ , for  $p < n$ , i.e.

$$\begin{aligned} h_1(x_1, \dots, x_n) &= 0 \\ &\vdots \\ h_p(x_1, \dots, x_n) &= 0 \end{aligned}$$

We assume that the differentials  $dh_1, \dots, dh_p$  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**,  $T_x M$ , which is an  $m$ -dimensional vector space specifying the set of possible velocities (directional derivatives) at  $x$ .

**Definition 10.1** *A smooth vector field on a manifold  $M$  is a function  $f : M \rightarrow T_x M$  which is infinitely differentiable, represented as a column vector,*

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

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  $T_x M$ , i.e., the space of functions from  $T_x M$  to  $\mathbb{R}$ .

<sup>1</sup>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  $C^\infty$  **diffeomorphisms**

<sup>2</sup>Our definition amounts to the special case of an **embedded submanifold** of dimension  $m = n - p$  in  $\mathbb{R}^n$

**Definition 10.2** A smooth covector field is a function  $w : M \rightarrow T_x^*M$  which is infinitely differentiable, represented as a row vector,

$$w(x) = [ w_1(x) \dots w_m(x) ]$$

Henceforth, whenever we use the term function, vector field, or covector field, it is assumed to be smooth. Since  $T_x M$  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 row vectors or column vectors.

**Example 10.1** 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,

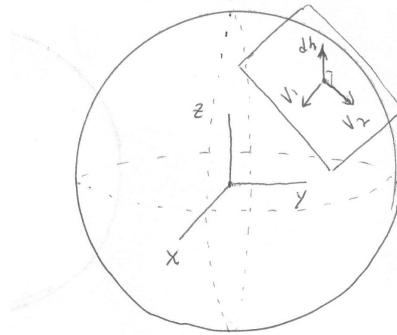


Fig. 10.1 The sphere as a manifold in  $\mathbb{R}^3$

$z = \sqrt{1 - x^2 - y^2}$ , the tangent space is spanned by the vectors

$$v_1 = (1, 0, -x/\sqrt{1 - x^2 - y^2})^T \quad (10.1)$$

$$v_2 = (0, 1, -y/\sqrt{1 - x^2 - y^2})^T \quad (10.2)$$

The differential of  $h$  is

$$dh = (2x, 2y, 2z) = (2x, 2y, 2\sqrt{1 - x^2 - y^2}) \quad (10.3)$$

which is easily shown to be normal to the tangent plane at  $x, y, z$ .  $\diamond$

We may also have multiple vector fields defined simultaneously on a given manifold. Such a set of vector fields will fill out 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 10.3**

1. Let  $X_1(x), \dots, X_k(x)$  be vector fields on  $M$ , which are linearly independent at each point. By a **Distribution**  $\Delta$  we mean the linear span (at each  $x \in M$ )

$$\Delta = \text{span} \{X_1(x), \dots, X_k(x)\} \quad (10.4)$$

2. Likewise, let  $w_1(x), \dots, w_k(x)$  be covector fields on  $M$ , which are linearly independent at each point. By a **Codistribution**  $\Omega$  we mean the linear span (at each  $x \in M$ )

$$\Omega = \text{span}\{w_1(x), \dots, w_k(x)\} \quad (10.5)$$

A distribution therefore assigns a vector space  $\Delta(x)$  at each point  $x \in M$ ; a  $k$ -dimensional subspace of the  $m$ -dimensional tangent space  $T_x M$ . A codistribution likewise defines a  $k$ -dimensional subspace at each  $x$  of the  $m$ -dimensional cotangent space  $T_x^* M$ . The reader may consult any text on differential geometry, for example [?], for more details.

Vector fields are used to define differential equations and their associated flows. We restrict our attention here to nonlinear systems of the form

$$\begin{aligned} \dot{x} &= f(x) + g_1(x)_1 u + \dots g_m(x)_m u \\ &=: f(x) + G(x)u \end{aligned} \quad (10.6)$$

where  $G(x) = [g_1(x), \dots, g_m(x)]$ ,  $u = (u_1, \dots, u_m)^T$ , and  $f(x), g_1(x), \dots, g_m(x)$  are vector fields on a manifold  $M$ . For simplicity we will assume that  $M = \mathbb{R}^n$ .

**Definition 10.4** 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

$$[f, g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g \quad (10.7)$$

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 10.2** Suppose that vector fields  $f(x)$  and  $g(x)$  on  $\mathbb{R}^3$  are given as

$$f(x) = \begin{bmatrix} x_2 \\ \sin x_1 \\ x_3^2 + x_1 \end{bmatrix} \quad g(x) = \begin{bmatrix} 0 \\ x_2^2 \\ 0 \end{bmatrix}$$

Then the vector field  $[f, g]$  is computed according to (10.7) 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

$$ad_f^k(g) = [f, ad_f^{k-1}(g)] \quad (10.8)$$

with  $ad_f^0(g) = g$ .

**Definition 10.5** Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  be a vector field on  $\mathbb{R}^n$  and let  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  be a scalar function. The **Lie Derivative** of  $h$ , with respect to  $f$ , denoted  $L_f h$ , is defined as

$$L_f h = \frac{\partial h}{\partial x} f(x) = \sum_{i=1}^n \frac{\partial h}{\partial x_i} f_i(x) \quad (10.9)$$

The Lie derivative is simply the directional derivative of  $h$  in the direction of  $f(x)$ , equivalently the inner product of the gradient of  $h$  and  $f$ . We denote by  $L_f^2 h$  the Lie Derivative of  $L_f h$  with respect to  $f$ , i.e.

$$L_f^2 h = L_f(L_f h) \quad (10.10)$$

In general we define

$$L_f^k h = L_f(L_f^{k-1} h) \quad \text{for } k = 1, \dots, n \quad (10.11)$$

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 10.1** Let  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  be a scalar function and  $f$  and  $g$  be vector fields on  $\mathbb{R}^n$ . Then we have the following identity

$$L_{[f,g]} h = L_f L_g h - L_g L_f h \quad (10.12)$$

*Proof:* Expand Equation (10.12) in terms of the coordinates  $x_1, \dots, x_n$  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 (10.12) 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 (10.12) 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 10-1).

### 10.2.1 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

$$\frac{\partial z}{\partial x} = f(x, y, z) \quad (10.13)$$

$$\frac{\partial z}{\partial y} = g(x, y, z) \quad (10.14)$$

In this example there are two partial differential equations in a single dependent variable  $z$ . A solution to (10.13)-(10.14) is a function  $z = \phi(x, y)$  satisfying

$$\frac{\partial \phi}{\partial x} = f(x, y, \phi(x, y)) \quad (10.15)$$

$$\frac{\partial \phi}{\partial y} = g(x, y, \phi(x, y)) \quad (10.16)$$

We can think of the function  $z = \phi(x, y)$  as defining a surface in  $\mathbb{R}^3$  as in Figure 10.2. The function  $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$  defined by

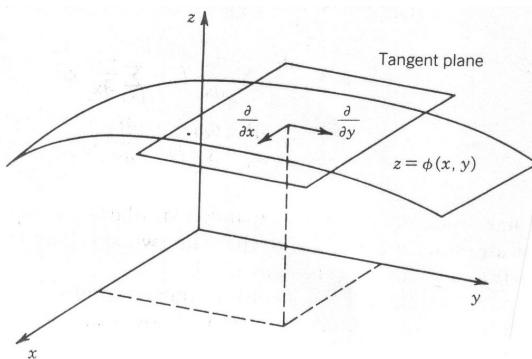


Fig. 10.2 Integral manifold in  $\mathbb{R}^3$

$$\Phi(x, y) = (x, y, \phi(x, y)) \quad (10.17)$$

then characterizes both the surface and the solution to the system of Equations (10.13)-(10.14). At each point  $(x, y)$  the tangent plane to the surface is spanned by two vectors found by taking partial derivatives of  $\Phi$  in the  $x$  and  $y$  directions, respectively, that is, by

$$\begin{aligned} X_1 &= (1, 0, f(x, y, \phi(x, y)))^T \\ X_2 &= (0, 1, g(x, y, \phi(x, y)))^T \end{aligned} \quad (10.18)$$

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 the system of Equations (10.13)-(10.14). Geometrically, one can now think of the problem of solving the system of first order partial differential Equations (10.13)-(10.14) 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 the system (10.13)-(10.14). 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 (10.13)-(10.14). Then it is a simple computation (Problem 10-2) to check that the function

$$h(x, y, z) = z - \phi(x, y) \quad (10.19)$$

satisfies the system of partial differential equations

$$\begin{aligned} L_{X_1} h &= 0 \\ L_{X_2} h &= 0 \end{aligned} \quad (10.20)$$

Conversely, suppose a scalar function  $h$  can be found satisfying (10.20), and suppose that we can solve the equation

$$h(x, y, z) = 0 \quad (10.21)$$

for  $z$ , as  $z = \phi(x, y)$ .<sup>3</sup> Then it can be shown that  $\phi$  satisfies (10.13)-(10.14). (Problem 10-3) Hence, complete integrability of the set of vector fields  $(X_1, X_2)$  is equivalent to the existence of  $h$  satisfying (10.20). With the preceding discussion as background we state the following

**Definition 10.6** A distribution  $\delta = \text{span}\{X_1, \dots, X_m\}$  on  $\mathbb{R}^n$  is said to be **completely integrable** if and only if there are  $n - m$  linearly independent functions  $h_1, \dots, h_{n-m}$  satisfying the system of partial differential equations

$$L_{X_i} h_j = 0 \quad \text{for } 1 \leq i \leq n ; 1 \leq j \leq m \quad (10.22)$$

<sup>3</sup>The so-called Implicit Function Theorem states that (10.21) can be solved for  $z$  as long as  $\frac{\partial h}{\partial z} \neq 0$ .

**Definition 10.7** A distribution  $\delta = \text{span}\{X_1, \dots, X_m\}$  is said to be **involutive** if and only if there are scalar functions  $\alpha_{ijk} : \mathbb{R}^n \rightarrow \mathbb{R}$  such that

$$[X_i, X_j] = \sum_{k=1}^m \alpha_{ijk} X_k \text{ for all } i, j, k \quad (10.23)$$

Involutivity simply means that if one forms the Lie Bracket of any pair of vector fields in  $\Delta$  then the resulting vector field can be expressed as a linear combination of the original vector fields  $X_1, \dots, X_m$ . Note that the coefficients in this linear combination are allowed to be smooth functions on  $\mathbb{R}^n$ . In the simple case of (10.13)-(10.14) one can show that if there is a solution  $z = \phi(x, y)$  of (10.13)-(10.14) then involutivity of the set  $\{X_1, X_2\}$  defined by (10.22) is equivalent to interchangeability of the order of partial derivatives of  $\phi$ , that is,  $\frac{\partial^2 \phi}{\partial x \partial y} = \frac{\partial^2 \phi}{\partial y \partial x}$ . The Frobenius Theorem, stated next, gives the conditions for the existence of a solution to the system of partial differential Equations (10.22).

**Theorem 2** A distribution  $\Delta$  is completely integrable if and only if it is involutive.

*Proof:* See, for example, Boothby [?].

### 10.3 FEEDBACK LINEARIZATION

To introduce the idea of feedback linearization consider the following simple system,

$$\dot{x}_1 = a \sin(x_2) \quad (10.24)$$

$$\dot{x}_2 = -x_1^2 + u \quad (10.25)$$

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

$$y_1 = x_1 \quad (10.26)$$

$$y_2 = a \sin(x_2) = \dot{x}_1 \quad (10.27)$$

then, by the chain rule,  $y_1$  and  $y_2$  satisfy

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= a \cos(x_2)(-x_1^2 + u) \end{aligned} \quad (10.28)$$

We see that the nonlinearities can now be cancelled by the input

$$u = \frac{1}{a \cos(x_2)} v + x_1^2 \quad (10.29)$$

which result in the linear system in the  $(y_1, y_2)$  coordinates

$$\begin{aligned}\dot{y}_1 &= y_2 \\ \dot{y}_2 &= v\end{aligned}\quad (10.30)$$

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 (10.30) in the coordinates  $(y_1, y_2)$ . For example the outer loop control

$$v = -k_1 y_1 - k_2 y_2 \quad (10.31)$$

applied to (10.30) results in the closed loop system

$$\begin{aligned}\dot{y}_1 &= y_2 \\ \dot{y}_2 &= -k_1 y_1 - k_2 y_2\end{aligned}\quad (10.32)$$

which has characteristic polynomial

$$p(s) = s^2 + k_2 s + k_1 \quad (10.33)$$

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 10.3 illustrates the inner loop/outer loop implementation of the above control strat-

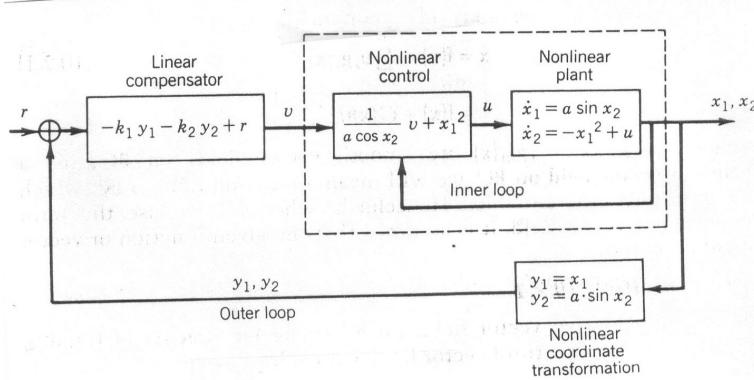


Fig. 10.3 Feedback linearization control architecture

egy. 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 (10.26)-(10.27), in this case

$$\begin{aligned}x_1 &= y_1 \\ x_2 &= \sin^{-1}(y_2/a)\end{aligned}\quad -a < y_2 < +a \quad (10.34)$$

This example illustrates several important features of feedback linearization. The first thing to note is the local nature of the result. We see from (10.26)

and (10.27) 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 (10.30), 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 (10.26)-(10.27). In the latter case the parameter  $a$  must be known precisely.

In Section 10.4 give necessary and sufficient conditions under which a general single-input nonlinear system can be transformed into a linear system in the above fashion, using a nonlinear change of variables and nonlinear feedback as in the above example.

#### 10.4 SINGLE-INPUT SYSTEMS

The idea of feedback linearization is easiest to understand in the context of single-input systems. In this section we derive the feedback linearization result of Su [?] for single-input nonlinear systems. As an illustration we apply this result to the control of a single-link manipulator with joint elasticity.

**Definition 10.8** *A single-input nonlinear system*

$$\dot{x} = f(x) + g(x)u \quad (10.35)$$

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  $T : U \rightarrow \mathbb{R}^n$ , defined on an open region  $U$  in  $\mathbb{R}^n$  containing the origin, and nonlinear feedback

$$u = \alpha(x) + \beta(x)v \quad (10.36)$$

with  $\beta(x) \neq 0$  on  $U$  such that the transformed variables

$$y = T(x) \quad (10.37)$$

satisfy the linear system of equations

$$\dot{y} = Ay + bv \quad (10.38)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & & & 0 \\ 0 & 0 & 1 & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & 1 \\ 0 & 0 & \cdot & \cdot & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \quad (10.39)$$

**Remark 10.1** The nonlinear transformation (10.37) and the nonlinear control law (10.36), when applied to the nonlinear system (10.35), result in a linear

controllable system (10.38). The diffeomorphism  $T(x)$  can be thought of as a nonlinear change of coordinates in the state space. The idea of feedback linearization is then 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 (10.35) for the existence of such a transformation. Let us set

$$y = T(x) \quad (10.40)$$

and see what conditions the transformation  $T(x)$  must satisfy. Differentiating both sides of (10.40) with respect to time yields

$$\dot{y} = \frac{\partial T}{\partial x} \dot{x} \quad (10.41)$$

where  $\frac{\partial T}{\partial x}$  is the Jacobian matrix of the transformation  $T(x)$ . Using (10.35) and (10.38), Equation (10.41) can be written as

$$\frac{\partial T}{\partial x}(f(x) + g(x)u) = Ay + bv \quad (10.42)$$

In component form with

$$T = \begin{bmatrix} T_1 \\ \cdot \\ \cdot \\ T_n \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & 1 \\ 0 & 0 & \cdot & \cdot & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \quad (10.43)$$

we see that the first equation in (10.42) is

$$\frac{\partial T_1}{\partial x_1} \dot{x}_1 + \cdots + \frac{\partial T_1}{\partial x_n} \dot{x}_n = T_2 \quad (10.44)$$

which can be written compactly as

$$L_f T_1 + L_g T_1 u = T_2 \quad (10.45)$$

Similarly, the other components of  $T$  satisfy

$$\begin{aligned} L_f T_2 + L_g T_2 u &= T_3 \\ &\vdots \\ L_f T_n + L_g T_n u &= v \end{aligned} \quad (10.46)$$

Since we assume that  $T_1, \dots, T_n$  are independent of  $u$  while  $v$  is not independent of  $u$  we conclude from (10.46) that

$$L_g T_1 = L_g T_2 = \dots = L_g T_{n-1} = 0 \quad (10.47)$$

$$L_g T_n \neq 0 \quad (10.48)$$

This leads to the system of partial differential equations

$$L_f T_i = T_{i+1} \quad i = 1, \dots, n-1 \quad (10.49)$$

together with

$$L_f T_n + L_g T_n u = v \quad (10.50)$$

Using Lemma 10.1 and the conditions (10.47) and (10.48) we can derive a system of partial differential equations in terms of  $T_1$  alone as follows. Using  $h = T_1$  in Lemma 10.1 we have

$$L_{[f,g]} T_1 = L_f L_g T_1 - L_g L_f T_1 = 0 - L_g T_2 = 0 \quad (10.51)$$

Thus we have shown

$$L_{[f,g]} T_1 = 0 \quad (10.52)$$

By proceeding inductively it can be shown (Problem 10-4) that

$$L_{ad_f^k g} T_1 = 0 \quad k = 0, 1, \dots, n-2 \quad (10.53)$$

$$L_{ad_f^{n-1} g} T_1 \neq 0 \quad (10.54)$$

If we can find  $T_1$  satisfying the system of partial differential Equations (10.53), then  $T_2, \dots, T_n$  are found inductively from (10.49) and the control input  $u$  is found from

$$L_f T_n + L_g T_n u = v \quad (10.55)$$

as

$$u = \frac{1}{L_g T_n} (v - L_f T_n) \quad (10.56)$$

We have thus reduced the problem to solving the system (10.53) for  $T_1$ . When does such a solution exist?

First note that the vector fields  $g, ad_f(g), \dots, ad_f^{n-1}(g)$  must be linearly independent. If not, that is, if for some index  $i$

$$ad_f^i(g) = \sum_{k=0}^{i-1} \alpha_k ad_f^k(g) \quad (10.57)$$

then  $ad_f^{n-1}(g)$  would be a linear combination of  $g, ad_f(g), \dots, ad_f^{n-2}(g)$  and (10.54) could not hold. Now by the Frobenius Theorem (10.53) has a solution if and only if the distribution  $\Delta = \text{span}\{g, ad_f(g), \dots, ad_f^{n-2}(g)\}$  is involutive. Putting this together we have shown the following

**Theorem 3** *The nonlinear system*

$$\dot{x} = f(x) + g(x)u \quad (10.58)$$

with  $f(x), g(x)$  vector fields, and  $f(0) = 0$  is feedback linearizable if and only if there exists a region  $U$  containing the origin in  $\mathbb{R}^n$  in which the following conditions hold:

1. The vector fields  $\{g, ad_f(g), \dots, ad_f^{n-1}(g)\}$  are linearly independent in  $U$
2. The distribution  $\text{span}\{g, ad_f(g), \dots, ad_f^{n-2}(g)\}$  is involutive in  $U$

**Example 10.3** [?] Consider the single link manipulator with flexible joint shown in Figure 10.4. Ignoring damping for simplicity, the equations of motion

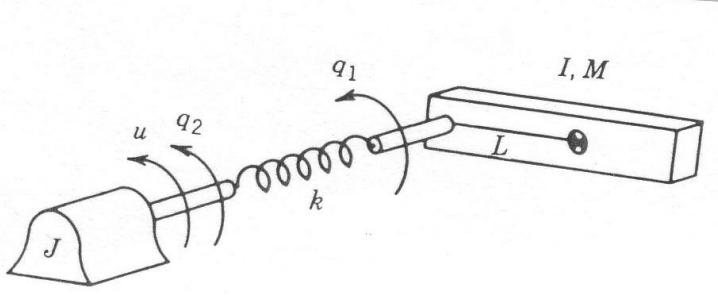


Fig. 10.4 Single-Link Flexible Joint Robot

are

$$\begin{aligned} I\ddot{q}_1 + Mgl \sin(q_1) + k(q_1 - q_2) &= 0 \\ J\ddot{q}_2 + k(q_2 - q_1) &= u \end{aligned} \quad (10.59)$$

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

$$\begin{aligned} x_1 &= q_1 & x_2 &= \dot{q}_1 \\ x_3 &= q_2 & x_4 &= \dot{q}_2 \end{aligned} \quad (10.60)$$

and write the system (10.59) as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{MgL}{I} \sin(x_1) - \frac{k}{I}(x_1 - x_3) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{k}{J}(x_1 - x_3) + \frac{1}{J}u\end{aligned}\tag{10.61}$$

The system is thus of the form (10.35) with

$$f(x) = \begin{bmatrix} x_2 \\ -\frac{MgL}{I} \sin(x_1) - \frac{k}{I}(x_1 - x_3) \\ x_4 \\ \frac{k}{J}(x_1 - x_3) \end{bmatrix} \quad g(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J} \end{bmatrix} \tag{10.62}$$

Therefore  $n = 4$  and the necessary and sufficient conditions for feedback linearization of this system are that

$$\text{rank } [g, ad_f(g), ad_f^2(g), ad_f^3(g)] = 4 \tag{10.63}$$

and that the set

$$\{g, ad_f(g), ad_f^2(g)\} \tag{10.64}$$

be involutive. Performing the indicated calculations it is easy to check that (Problem 10-6)

$$[g, ad_f(g), ad_f^2(g), ad_f^3(g)] = \begin{bmatrix} 0 & 0 & 0 & \frac{k}{IJ} \\ 0 & 0 & \frac{k}{IJ} & 0 \\ 0 & \frac{1}{J} & 0 & -\frac{k}{J^2} \\ \frac{1}{J} & 0 & -\frac{k}{J^2} & 0 \end{bmatrix} \tag{10.65}$$

which has rank 4 for  $k > 0$ ,  $I, J < \infty$ . Also, since the vector fields  $\{g, ad_f(g), ad_f^2(g)\}$  are constant, they form an involutive set. 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 (10.64) is zero which is trivially a linear combination of the vector fields themselves. It follows that the system (10.59) is feedback linearizable. The new coordinates

$$y_i = T_i \quad i = 1, \dots, 4 \tag{10.66}$$

are found from the conditions (10.53), with  $n = 4$ , that is

$$\begin{aligned}L_g T_1 &= 0 \\ L_{[f,g]} T_1 &= 0 \\ L_{ad_f^2 g} T_1 &= 0 \\ L_{ad_f^3 g} T_1 &= 0\end{aligned}\tag{10.67}$$

Carrying out the above calculations leads to the system of equations (Problem 10-9)

$$\frac{\partial T_1}{\partial x_2} = 0 ; \quad \frac{\partial T_1}{\partial x_3} = 0 ; \quad \frac{\partial T_1}{\partial x_4} = 0 \quad (10.68)$$

and

$$\frac{\partial T_1}{\partial x_1} \neq 0 \quad (10.69)$$

From this we see that the function  $T_1$  should be a function of  $x_1$  alone. Therefore, we take the simplest solution

$$y_1 = T_1 = x_1 \quad (10.70)$$

and compute from (10.49) (Problem 10-10)

$$\begin{aligned} y_2 &= T_2 = L_f T_1 = x_2 \\ y_3 &= T_3 = L_f T_2 = -\frac{MgL}{I} \sin(x_1) - \frac{k}{I}(x_1 - x_3) \\ y_4 &= T_4 = L_f T_3 = -\frac{MgL}{I} \cos(x_1) - \frac{k}{I}(x_2 - x_4) \end{aligned} \quad (10.71)$$

The feedback linearizing control input  $u$  is found from the condition

$$u = \frac{1}{L_g T_4} (v - L_f T_4) \quad (10.72)$$

as (Problem 10-11)

$$u = \frac{IJ}{k} (v - a(x)) = \beta(x)v + \alpha(x) \quad (10.73)$$

where

$$\begin{aligned} a(x) := & \frac{MgL}{I} \sin(x_1) \left( x_2^2 + \frac{MgL}{I} \cos(x_1) + \frac{k}{I} \right) \\ & + \frac{k}{I} (x_1 - x_3) \left( \frac{k}{I} + \frac{k}{J} + \frac{MgL}{I} \cos(x_1) \right) \end{aligned} \quad (10.74)$$

◇

Therefore in the coordinates  $y_1, \dots, y_4$  with the control law (10.73) the system becomes

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= y_3 \\ \dot{y}_3 &= y_4 \\ \dot{y}_4 &= v \end{aligned} \quad (10.75)$$

or, in matrix form,

$$\dot{y} = Ay + bv \quad (10.76)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (10.77)$$

**Remark 10.2** The above feedback linearization is actually global. In order to see this we need only compute the inverse of the change of variables (10.70)-(10.71). Inspecting (10.70)-(10.71) we see that

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 \\ x_3 &= y_1 + \frac{I}{k} \left( y_3 + \frac{MgL}{I} \sin(y_1) \right) \\ x_4 &= y_2 + \frac{I}{k} \left( y_4 + \frac{MgL}{I} \cos(y_1)y_2 \right) \end{aligned} \quad (10.78)$$

The inverse transformation is well defined and differentiable everywhere and, hence, the feedback linearization for the system (10.59) holds globally. The transformed variables  $y_1, \dots, y_4$  are themselves physically meaningful. We see that

$$\begin{aligned} y_1 = x_1 &= \text{link position} \\ y_2 = x_2 &= \text{link velocity} \\ y_3 = \dot{y}_2 &= \text{link acceleration} \\ y_4 = \dot{y}_3 &= \text{link jerk} \end{aligned} \quad (10.79)$$

Since the motion trajectory of the link is typically specified in terms of these quantities they are natural variables to use for feedback.

**Example 10.4** One way to execute a step change in the link position while keeping the manipulator motion smooth would be to require a constant jerk during the motion. This can be accomplished by a cubic polynomial trajectory using the methods of Chapter 5.6. Therefore, let us specify a trajectory

$$\theta_\ell^d(t) = y_1^d = a_1 + a_2t + a_3t^2 + a_4t^3 \quad (10.80)$$

so that

$$\begin{aligned} y_2^d &= \dot{y}_1^d = a_2 + 2a_3t + 3a_4t^2 \\ y_3^d &= \ddot{y}_2^d = 2a_3 + 6a_4t \\ y_4^d &= \dddot{y}_3^d = 6a_4 \end{aligned}$$

Then a linear control law that tracks this trajectory and that is essentially equivalent to the feedforward/feedback scheme of Chapter 8 is given by

$$v = \dot{y}_4^d - k_1(y_1 - y_1^d) - k_2(y_2 - y_2^d) - k_3(y_3 - y_3^d) - k_4(y_4 - y_4^d) \quad (10.81)$$

Applying this control law to the fourth order linear system (10.73) we see that the tracking error  $e(t) = y_1 - y_1^d$  satisfies the fourth order linear equation

$$\frac{d^4 e}{dt^4} + k_4 \frac{d^3 e}{dt^3} + k_3 \frac{d^2 e}{dt^2} + k_2 \frac{de}{dt} + k_1 e = 0 \quad (10.82)$$

and, hence, the error dynamics are completely determined by the choice of gains  $k_1, \dots, k_4$ .  $\diamond$

Notice that the feedback control law (10.81) is stated in terms of the variables  $y_1, \dots, y_4$ . 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  $x_1, \dots, x_4$  which represent the motor and link positions and velocities, and compute  $y_1, \dots, y_4$  using the transformation Equations (10.70)-(10.71). In this case the parameters appearing in the transformation equations would have to be known precisely. Another, and perhaps more promising, approach is to construct a dynamic observer to estimate the state variables  $y_1, \dots, y_4$ .

## 10.5 FEEDBACK LINEARIZATION FOR $N$ -LINK ROBOTS

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 systems in which the nonlinearities can be exactly canceled 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 directly.

**Example 10.5** 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 8. To see this, consider the rigid equations of motion (8.6), which we write in state space as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -M(x_1)^{-1}(C(x_1, x_2)x_2 + g(x_1)) + M(x_1)^{-1}u \end{aligned} \quad (10.83)$$

with  $x_1 = q$ ;  $x_2 = \dot{q}$ . In this case a feedback linearizing control is found by simply inspecting (10.83) as

$$u = M(x_1)v + C(x_1, x_2)x_2 + g(x_1) \quad (10.84)$$

Substituting (10.84) into (10.83) yields

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= v \end{aligned} \quad (10.85)$$

Equation (10.85) represents a set of  $n$ -second order systems of the form

$$\begin{aligned} \dot{x}_{1i} &= x_{2i} \\ \dot{x}_{2i} &= v_i, \quad i = 1, \dots, n \end{aligned} \quad (10.86)$$

Comparing (10.84) with (8.17) we see indeed that the feedback linearizing control for a rigid manipulator is precisely the inverse dynamics control of Chapter 8.  
◇

**Example 10.6** If the joint flexibility is included in the dynamic description of an  $n$ -link robot the equations of motion can be written as [?]

$$\begin{aligned} D(q_1)\ddot{q}_1 + C(q_1, \dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 - K(q_1 - q_2) &= u \end{aligned} \quad (10.87)$$

In state space, which is now  $\mathbb{R}^{4n}$ , we define state variables in block form

$$\begin{aligned} \dot{x}_1 &= q_1 & x_2 &= \dot{q}_1 \\ \dot{x}_3 &= q_2 & x_4 &= \dot{q}_2 \end{aligned} \quad (10.88)$$

Then from (10.87) we have:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -D(x_1)^{-1}\{h(x_1, x_2) + K(x_1 - x_3)\} \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= J^{-1}K(x_1 - x_3) + J^{-1}u \end{aligned} \quad (10.89)$$

Here we define  $h(x_1, x_2) = C(x_1, x_2)x_2 + g(x_1)$  for simplicity. This system is then of the form

$$\dot{x} = f(x) + G(x)u \quad (10.90)$$

In the single-link case we saw that the appropriate state variables with which to define the system so that it could be linearized by nonlinear feedback were the link position, velocity, acceleration, and jerk. Following the single-input example, then, we can attempt to do the same thing in the multi-link case and

derive a feedback linearizing transformation blockwise as follows: Set

$$\begin{aligned}
 y_1 &= T_1(x_1) := x_1 \\
 y_2 &= T_2(x) := \dot{y}_1 = \dot{x}_2 \\
 y_3 &= T_3(x) := \dot{y}_2 = \dot{x}_2 \\
 &= -D(x_1)^{-1}\{h(x_1, x_2) + K(x_1 - x_3)\} \\
 y_4 &= T_4(x) := \dot{y}_3 \\
 &= -\frac{d}{dt}[D(x_1)^{-1}]\{h(x_1, x_2) + K(x_1 - x_3)\} - D(x_1)^{-1}\left\{\frac{\partial h}{\partial x_1}x_2\right. \\
 &\quad \left.+\frac{\partial h}{\partial x_2}[-D(x_1)^{-1}(h(x_1, x_2) + K(x_1 - x_3))] + K(x_2 - x_4)\right\} \\
 &:= a_4(x_1, x_2, x_3) + D(x_1)^{-1}Kx_4
 \end{aligned} \tag{10.91}$$

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

$$\begin{aligned}
 x_1 &= y_1 \\
 x_2 &= y_2 \\
 x_3 &= y_1 + K^{-1}(D(y_1)y_3 + h(y_1, y_2)) \\
 x_4 &= K^{-1}D(y_1)(y_4 - a_4(y_1, y_2, y_3))
 \end{aligned} \tag{10.92}$$

The linearizing control law can now be found from the condition

$$\dot{y}_4 = v \tag{10.93}$$

where  $v$  is a new control input. Computing  $\dot{y}_4$  from (10.91) and suppressing function arguments for brevity yields

$$\begin{aligned}
 v &= \frac{\partial a_4}{\partial x_1}x_2 - \frac{\partial a_4}{\partial x_2}D^{-1}(h + K(x_1 - x_3)) \\
 &+ \frac{\partial a_4}{\partial x_3}x_4 + \frac{d}{dt}[D^{-1}]Kx_4 + D^{-1}K(J^{-1}K(x_1 - x_3) + J^{-1}u) \\
 &=: a(x) + b(x)u
 \end{aligned} \tag{10.94}$$

where  $a(x)$  denotes all the terms in (10.94) but the last term, which involves the input  $u$ , and  $b(x) := D^{-1}(x)KJ^{-1}$ .

Solving the above expression for  $u$  yields

$$u = b(x)^{-1}(v - a(x)) =: \alpha(x) + \beta(x)v \tag{10.95}$$

where  $\beta(x) = JK^{-1}D(x)$  and  $\alpha(x) = -b(x)^{-1}a(x)$ .

With the nonlinear change of coordinates (10.91) and nonlinear feedback (10.95) the transformed system now has the linear block form

$$\begin{aligned}
 \dot{y} &= \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{bmatrix} y + \begin{bmatrix} 0 \\ 0 \\ 0 \\ I \end{bmatrix} v \\
 &=: Ay + Bv
 \end{aligned} \tag{10.96}$$

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 (10.96) 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 (10.75).  $\diamond$

## 10.6 NONHOLONOMIC SYSTEMS

In this section we return to a discussion of systems subject to constraints. A constraint on a mechanical system restricts its motion by limiting the set of paths that the system can follow. We briefly discussed so-called **holonomic constraints** in Chapter 6 when we derived the Euler-Lagrange equations of motion. Our treatment of force control in Chapter 9 dealt with unilateral constraints defined by the environmental contact. In this section we expand upon the notion of systems subject to constraints and discuss **nonholonomic systems**.

Let  $Q$  denote the configuration space of a given system and let  $q = (q_1, \dots, q_n)^T \in Q$  denote the vector of generalized coordinates defining the system configuration. We recall the following definition.

**Definition 10.9** *A set of  $k < n$  constraints*

$$h_i(q_1, \dots, q_n) = 0 \quad i = 1, \dots, k \quad (10.97)$$

*is called holonomic, where  $h_i$  is a smooth mapping from  $Q \mapsto \mathbb{R}$ .*

We assume that the constraints are independent so that the differentials

$$\begin{aligned} dh_1 &= \left[ \frac{\partial h_1}{\partial q_1}, \dots, \frac{\partial h_1}{\partial q_n} \right] \\ &\vdots \\ dh_k &= \left[ \frac{\partial h_k}{\partial q_1}, \dots, \frac{\partial h_k}{\partial q_n} \right] \end{aligned}$$

are linearly independent covectors. Note that, in order to satisfy these constraints, the motion of the system must lie on the hypersurface defined by the functions  $h_1, \dots, h_k$ , i.e.

$$h_i(q(t)) = 0 \quad \text{for all } t > 0 \quad (10.98)$$

As a consequence, by differentiating (10.98), we have

$$\langle dh_i, \dot{q} \rangle = 0 \quad i = 1, \dots, k \quad (10.99)$$

where  $\langle \cdot, \cdot \rangle$  denotes the standard inner product.

It frequently happens that constraints are expressed, not as constraints on the configuration, as in (10.97), but as constraints on the velocity

$$\langle w_i, \dot{q} \rangle = 0 \quad i = 1, \dots, k \quad (10.100)$$

where  $w_i(q)$  are covectors. Constraints of the form (10.100) are known as **Pfaffian** constraints. The crucial question in such cases is, therefore, when can the covectors  $w_1, \dots, w_k$  be expressed as differentials of smooth functions,  $h_1, \dots, h_k$ ? We express this as

**Definition 10.10** *Constraints of the form*

$$\langle w_i, \dot{q} \rangle = 0 \quad i = 1, \dots, k \quad (10.101)$$

*are called holonomic if there exists smooth functions  $h_1, \dots, h_k$  such that*

$$w_i(q) = dh_i(q) \quad i = 1, \dots, k \quad (10.102)$$

*and nonholonomic otherwise, i.e. if no such functions  $h_1, \dots, h_k$  exist.*

We can begin to see a connection with our earlier discussion of integrability and the Frobenius Theorem if we think of Equation (10.102) as a set of partial differential equations in the (unknown) functions  $h_i$ . Indeed, the term **integrable constraint** is frequently used interchangeably with holonomic constraint for this reason.

### 10.6.1 Involutivity and Holonomy

Now, given a set of Pfaffian constraints  $\langle w_i(q), \dot{q} \rangle = 0$  for  $i = 1, \dots, k$ , let  $\Omega$  be the codistribution defined by the covectors  $w_1, \dots, w_k$  and let  $\{g_1, \dots, g_m\}$  for  $m = n - k$  be a basis for the distribution  $\Delta$  that **annihilates**  $\Omega$ , i.e. such that

$$\langle w_i, g_j \rangle = 0 \quad \text{for each } i, j \quad (10.103)$$

We use the notation  $\Delta = \Omega^\perp$  (This is pronounced “ $\Omega$  perp”). Notice from Equation (10.102) that

$$0 = \langle w_i, g_j \rangle = \langle dh_i, g_j \rangle \quad \text{for each } i, j \quad (10.104)$$

Using our previous notation for Lie Derivative, the above system of equations may be written as

$$L_{g_j} h_i = 0 \quad i = 1, \dots, k; j = 1, \dots, m \quad (10.105)$$

The following theorem thus follows immediately from the Frobenius Theorem

**Theorem 4** *Let  $\Omega$  be the codistribution defined by covectors  $w_1, \dots, w_k$ . Then the constraints  $\langle w_i, \dot{q} \rangle = 0$  for  $i = 1, \dots, k$  are holonomic if and only if the distribution  $\Delta = \Omega^\perp$  is involutive.*

### 10.6.2 Driftless Control Systems

It is important to note that the velocity vector  $\dot{q}$  of the system is orthogonal to the covectors  $w_i$  according to (10.101) and hence lies in the distribution  $\Delta = \Omega^\perp$ . In other words the system velocity satisfies

$$\dot{q} = g_1(q)u_1 + \cdots + g_m(q)u_m \quad (10.106)$$

for suitable coefficients  $u_1, \dots, u_m$ . In many systems of interest, the coefficients  $u_i$  in (10.106) have the interpretation of control inputs and hence Equation (10.106) defines a useful model for control design in such cases. The system (10.106) is called **driftless** because  $\dot{q} = 0$  when the control inputs  $u_1, \dots, u_m$  are zero. In the next section we give some examples of driftless systems arising from nonholonomic constraints, followed by a discussion of controllability of driftless systems and Chow's Theorem in Section 10.7.

### 10.6.3 Examples of Nonholonomic Systems

Nonholonomic constraints arise in two primary ways:

1. In so-called **rolling without slipping** constraints. For example, the translational and rotational velocities of a rolling wheel are not independent if the wheel rolls without slipping. Examples include
  - A unicycle
  - An automobile, tractor/trailer, or wheeled mobile robot
  - Manipulation of rigid objects
2. In systems where angular momentum is conserved. Examples include
  - Space robots
  - Satellites
  - Gymnastic robots

**Example: 10.7 The Unicycle.** *The unicycle is equivalent to a wheel rolling on a plane and is thus the simplest example of a nonholonomic system. Referring to Figure 10.5 we see that the configuration of the unicycle is defined by the variables  $x, y, \theta$  and  $\phi$ , where  $x$  and  $y$  denote the Cartesian position of the ground contact point,  $\theta$  denotes the heading angle and  $\phi$  denotes the angle of the wheel measured from the vertical. The rolling without slipping condition means that*

$$\begin{aligned} \dot{x} - r \cos \theta \dot{\phi} &= 0 \\ \dot{y} - r \sin \theta \dot{\phi} &= 0 \end{aligned} \quad (10.107)$$

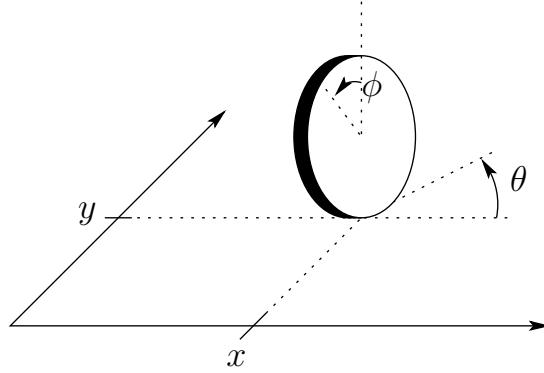


Fig. 10.5 The Unicycle

where  $r$  is the radius of the wheel. These constraints can be written in the form (10.101) with  $q = (x, y, \theta, \phi)$  and

$$\begin{aligned} w_1 &= [1 \ 0 \ 0 \ -r \cos \theta] \\ w_2 &= [1 \ 0 \ 0 \ -r \sin \theta] \end{aligned} \quad (10.108)$$

Since the dimension of the configuration space is  $n = 4$  and there are two constraint equations, we need to find two functions  $g_1, g_2$  orthogonal to  $w_1, w_2$ . It is easy to see that

$$g_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; \quad g_2 = \begin{bmatrix} r \cos \theta \\ r \sin \theta \\ 0 \\ 1 \end{bmatrix} \quad (10.109)$$

are both orthogonal to  $w_1$  and  $w_2$ . Thus we can write

$$\dot{q} = g_1(q)u_1 + g_2(q)u_2 \quad (10.110)$$

where  $u_1$  is the turning rate and  $u_2$  is the rate of rolling.

We can now check to see if rolling without slipping constraints on the unicycle is holonomic or nonholonomic using Theorem 4. It is easy to show (Problem 10-18) that the Lie Bracket

$$[g_1, g_2] = \begin{bmatrix} -r \sin \theta \\ r \cos \theta \\ 0 \\ 0 \end{bmatrix} \quad (10.111)$$

which is not in the distribution  $\Delta = \text{span}\{g_1, g_2\}$ . Therefore the constraints on the unicycle are nonholonomic. We shall see the consequences of this fact in the next section when we discuss controllability of driftless systems.

**Example: 10.8 The Kinematic Car.** Figure 10.6 shows a simple representation of a car, or mobile robot with steerable front wheels. The configuration

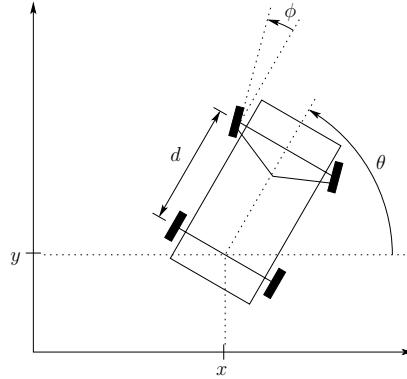


Fig. 10.6 The Kinematic Car

of the car can be described by  $q = [x, y, \theta, \phi]^T$ , where  $x$  and  $y$  is the point at the center of the rear axle,  $\theta$  is the heading angle, and  $\phi$  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

$$\begin{aligned} \sin \theta \dot{x} - \cos \theta \dot{y} &= 0 \\ \sin(\theta + \phi) \dot{x} - \cos(\theta + \phi) \dot{y} - d \cos \phi \dot{\theta} &= 0 \end{aligned} \quad (10.112)$$

which can be written as

$$\begin{bmatrix} \sin \theta & \cos \theta & 0 & 0 \\ \sin(\theta + \phi) & -\cos(\theta + \phi) & -d \cos \phi & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \langle w_1, \dot{q} \rangle \\ \langle w_2, \dot{q} \rangle \end{bmatrix} = 0 \quad (10.113)$$

It is thus straightforward to find vectors

$$g_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}; \quad g_2 = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \frac{1}{d} \tan \phi \\ 0 \end{bmatrix} \quad (10.114)$$

orthogonal to  $w_1$  and  $w_2$  and write the corresponding control system in the form (10.106). It is left as an exercise (Problem 19) to show that the above constraints are nonholonomic.

**Example: 10.9 A Hopping Robot.** Consider the hopping robot in Figure 10.7. The configuration of this robot is defined by  $q = (\psi, \ell, \theta)$ , where

- $\psi$  = the leg angle
- $\theta$  = the body angle
- $\ell$  = the leg extension

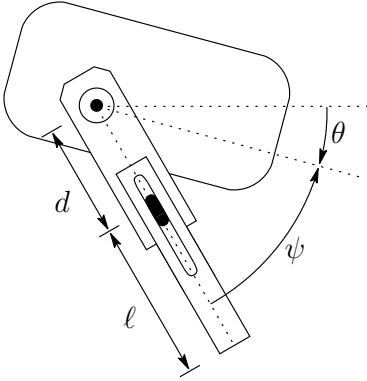


Fig. 10.7 Hopping Robot

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

$$I\dot{\theta} + m(\ell + d)^2(\dot{\theta} + \dot{\psi}) = 0 \quad (10.115)$$

assuming the initial angular momentum is zero. This constraint may be written as

$$\langle w, \dot{q} \rangle = 0 \quad (10.116)$$

where  $w = [ m(\ell + d)^2 \ 0 \ I + m(\ell + d)^2 ]$ . 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  $\Delta = \Omega^\perp$ , where  $\Omega = \text{span } \{w\}$ . It is easy to see that

$$g_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad g_2 = \begin{bmatrix} 1 \\ 0 \\ -\frac{m(\ell+d)^2}{I+m(\ell+d)^2} \end{bmatrix} \quad (10.117)$$

are linearly independent at each point and orthogonal to  $w$ . Checking involutivity of  $\Delta$  we find that

$$[g_1, g_2] = \begin{bmatrix} 0 \\ 0 \\ \frac{-2Im(\ell+d)}{[I+m(\ell+d)^2]^2} \end{bmatrix} =: g_3 \quad (10.118)$$

Since  $g_3$  is not a linear combination of  $g_1$  and  $g_2$  it follows that  $\Delta$  is not an involutive distribution and hence the constraint is nonholonomic.

## 10.7 CHOW'S THEOREM AND CONTROLLABILITY OF DRIFTLESS SYSTEMS

In this section we discuss the controllability properties of driftless systems of the form

$$\dot{x} = g_1(x)u_1 + \cdots + g_m(x)u_m \quad (10.119)$$

with  $x \in \mathbb{R}^n$ ,  $u \in \mathbb{R}^m$ . We assume that the vector fields  $g_1(x), \dots, g_m(x)$  are smooth, complete<sup>4</sup>, and linearly independent at each  $x \in \mathbb{R}^n$ .

We have seen previously that if the  $k < n$  Pfaffian constraints on the system are holonomic then the trajectory of the system lies on a  $m = n - k$ -dimensional surface (an integral manifold) found by integrating the constraints. In fact, at each  $x \in \mathbb{R}$  the tangent space to this manifold is spanned by the vectors  $g_1(x), \dots, g_m(x)$ . If we examine Equation (10.119) we see that any instantaneous direction in this tangent space, i.e. any linear combination of  $g_1, \dots, g_m$ , is achievable by suitable choice of the control input terms  $u_i$ ,  $i = 1, \dots, m$ . Thus every point on the manifold may be reached from any other point on the manifold by suitable control input. However, points not lying on the manifold cannot be reached no matter what control input is applied. Thus, for an initial condition  $x_0$ , only points on the particular integral manifold through  $x_0$  are reachable.

What happens if the constraints are nonholonomic? Then no such integral manifold of dimension  $m$  exists. Thus it might be possible to reach a space (manifold) of dimension larger than  $m$  by suitable application of the control inputs  $u_i$ . It turns out that this interesting possibility is true. In fact, by suitable combinations of two vector fields  $g_1$  and  $g_2$  it is possible to move in the direction defined by the Lie Bracket  $[g_1, g_2]$ . If the distribution  $\Delta = \text{span}\{g_1, g_2\}$  is not involutive, then the Lie Bracket vector field  $[g_1, g_2]$  defines a direction not in the span of  $g_1$  and  $g_2$ . Therefore, given vector fields  $g_1, \dots, g_m$  one may reach points not only in the span of these vector field but in the span of the distribution obtained by augmenting  $g_1, \dots, g_m$  with various Lie Bracket directions.

### Definition 10.11 Involutive Closure of a Distribution

*The involutive closure,  $\bar{\Delta}$ , of a distribution  $\Delta = \text{span}\{g_1, \dots, g_m\}$  is the smallest involutive distribution containing  $\Delta$ . In other words,  $\bar{\Delta}$  is an involutive distribution such that if  $\Delta_0$  is any involutive distribution satisfying  $\Delta \subset \Delta_0$  then  $\bar{\Delta} \subset \Delta_0$ .*

Conceptually, the involutive closure of  $\Delta$  can be found by forming larger and larger distributions by repeatedly computing Lie Brackets until an involutive

<sup>4</sup>A complete vector field is one for which the solution of the associated differential equation exists for all time  $t$

distribution is found, i.e.

$$\bar{\Delta} = \text{span}\{g_1, \dots, g_m, [g_i, g_j], [g_k, [g_i, g_j]], \dots\} \quad (10.120)$$

The involutive closure  $\bar{\Delta}$  in (10.120) is also called the **Control Lie Algebra** for the driftless control system (10.119). Intuitively, if  $\dim \bar{\Delta} = n$  then all points in  $\mathbb{R}^n$  should be reachable from  $x_0$ . This is essentially the conclusion of **Chow's Theorem**.

### Definition 10.12 Controllability

A driftless system of the form (10.106) is said to be **Controllable** if, for any  $x_0$  and  $x_1 \in \mathbb{R}^n$ , there exists a time  $T > 0$  and a control input  $u = [u_1, \dots, u_m]^T : [0, T] \rightarrow \mathbb{R}^m$  such that the solution  $x(t)$  of (10.106) satisfies  $x(0) = x_0$  and  $x(T) = x_1$ .

Given an open set  $U \subset \mathbb{R}^n$ , we let  $R_V^\epsilon(x_0)$  denote the set of states  $x$  such that there exists a control  $u : [0, \epsilon] \mapsto U$  with  $x(0) = x_0$ ,  $x(\epsilon) = x$  and  $x(t) \in V$  for  $0 \leq t \leq \epsilon$ . We set

$$R_{V,T}(x_0) = \cup_{0 < \epsilon \leq T} R_V^\epsilon(x_0) \quad (10.121)$$

$R_{V,T}$  is the set of states reachable up to time  $T > 0$ .

**Definition 10.13** We say that the system (10.106) is **Locally Controllable at  $x_0$**  if  $R_{V,T}(x_0)$  contains an open neighborhood of  $x_0$  for all neighborhoods  $V$  of  $x_0$  and  $T > 0$ .

The next result, known as **Chow's Theorem**, gives a sufficient condition for the system (10.106) to be locally controllable.

**Theorem 5** The driftless system

$$\dot{x} = g_1(x)u_1 + \dots + g_m(x)u_m \quad (10.122)$$

is locally controllable at  $x_0 \in \mathbb{R}^n$  if  $\text{rank } \bar{\Delta}(x_0) = n$ .

The proof of Chow's Theorem is beyond the scope of this text. The condition  $\text{rank } \bar{\Delta}(x_0) = n$  is called the **Controllability Rank Condition**. Note that Chow's theorem tells us when a driftless system is locally controllable but does not tell us how to find the control input to steer the system from  $x_0$  to  $x_1$ .

**Example: 10.10** Consider the system on  $\mathbb{R}^3 - \{0\}$ .

$$\begin{aligned} \dot{x} &= \begin{bmatrix} x_3 \\ x_2 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ x_1 \end{bmatrix} u_2 \\ &= g_1(x)u_1 + g_2(x)u_2 \end{aligned} \quad (10.123)$$

For  $x \neq 0$  the distribution  $\Delta = \text{span}\{g_1, g_2\}$  has rank two. It is easy to compute the Lie Bracket  $[g_1, g_2]$  as

$$[g_1, g_2] = \begin{bmatrix} -x_1 \\ 0 \\ x_3 \end{bmatrix}$$

and therefore

$$\text{rank}[g_1, g_2, [g_1, g_2]] = \text{rank} \begin{bmatrix} x_3 & 0 & -x_1 \\ x_2 & 0 & 0 \\ 0 & x_1 & x_3 \end{bmatrix}$$

which has rank three for  $x \neq 0$ . Therefore the system is locally controllable on  $\mathbb{R}^3 - \{0\}$ . Note that the origin is an equilibrium for the system independent of the control input, which is why we must exclude the origin from the above analysis.

### Example 10.11 Attitude Control of a Satellite

Consider a cylindrical satellite equipped with reaction wheels for control as shown in Figure 10.8 Suppose we can control the angular velocity about the

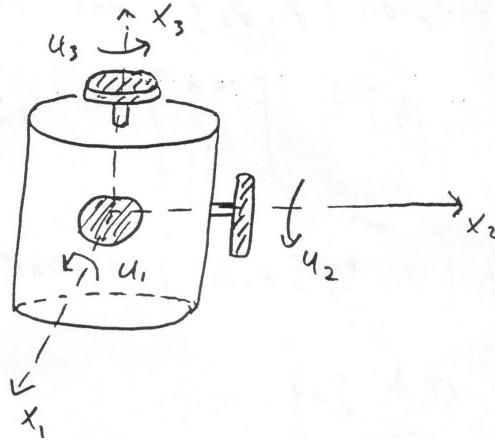


Fig. 10.8 Satellite with Reaction Wheels

$x_1$ ,  $x_2$ , and  $x_3$  axes with controls  $u_1$ ,  $u_2$ , and  $u_3$ , respectively. The equations of motion are then given by

$$\dot{\omega} = \omega \times u$$

with

$$w = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

Carrying out the above calculation, it is readily shown (Problem 10-20) that

$$\begin{aligned}\dot{\omega} &= \begin{bmatrix} 0 \\ \omega_3 \\ -\omega_2 \end{bmatrix} u_1 + \begin{bmatrix} -\omega_3 \\ 0 \\ \omega_1 \end{bmatrix} u_2 + \begin{bmatrix} \omega_2 \\ -\omega_1 \\ 0 \end{bmatrix} u_3 \\ &= g_1(\omega)u_1 + g_2(\omega)u_2 + g_3(\omega)u_3\end{aligned}\quad (10.124)$$

It is easy to show (Problem 10-21) that the distribution  $\Delta = \text{span}\{g_1, g_2, g_3\}$  is involutive of rank 3 on  $\mathbb{R}^3 - \{0\}$ . A more interesting property is that the satellite is controllable as long as any two of the three reaction wheels are functioning. The proof of this strictly nonlinear phenomenon is left as a exercise (Problem 10-22).  $\diamond$

---

## Problems

---

- 10-1 Complete the proof of Lemma 10.1 by direct calculation.
- 10-2 Show that the function  $h = z - \phi(x, y)$  satisfies the system (10.20) if  $\phi$  is a solution of (10.13)-(10.14) and  $X_1, X_2$  are defined by (10.18).
- 10-3 Show that if  $h(x, y, z)$  satisfies (10.20), then, if  $\frac{\partial h}{\partial z} \neq 0$ , Equation (10.21) can be solved for  $z$  as  $z = \phi(x, y)$  where  $\phi$  satisfies (10.13)-(10.14). Also show that  $\frac{\partial h}{\partial z} = 0$  can occur only in the case of the trivial solution  $h = 0$  of (10.20).
- 10-4 Verify the expressions (10.53) and (10.54).
- 10-5 Show that the system below is locally feedback linearizable.
- $$\begin{aligned}\dot{x}_1 &= x_1^3 + x_2 \\ \dot{x}_2 &= x_2^3 + u\end{aligned}$$
- 10-6 Derive the equations of motion (10.59) for the single-link manipulator with joint elasticity of Figure 10.4 using Lagrange's equations.
- 10-7 Repeat Problem 6 where there is assumed to be viscous friction both on the link side and on the motor side of the spring in Figure 10.4.
- 10-8 Perform the calculations necessary to verify (10.65).
- 10-9 Derive the system of partial differential Equations (10.68) from the conditions (10.67). Also verify (10.69).
- 10-10 Compute the change of coordinates (10.71).
- 10-11 Verify Equations (10.73)-(10.74).
- 10-12 Verify Equations (10.78).
- 10-13 Design and simulate a linear outer loop control law  $v$  for the system (10.59) 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. (See reference [2] for some ideas.)
- 10-14 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?
- 10-15 What happens to the inverse coordinate transformation (10.78) as the joint stiffness  $k \rightarrow \infty$ ? Give a physical interpretation. Use this to show

that the system (10.59) reduces to the equation governing the rigid joint manipulator in the limit as  $k \rightarrow \infty$ .

- 10-16 Consider the single-link manipulator with elastic joint of Figure 10.4 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  $\phi$  is a diffeomorphism. Derive the equations of motion for the system and show that it is still feedback linearizable. Carry out the feedback linearizing transformation. Specialize the result to the case of a cubic characteristic, i.e.,  $\phi = k(q_1 - q_2)^3$ . The cubic spring characteristic is a more accurate description for many manipulators than is the linear spring, especially for elasticity arising from gear flexibility.
- 10-17 Consider again the single link flexible joint robot given by (10.59) and suppose that only the link angle,  $q_1$ , is measurable. Design an observer to estimate the full state vector,  $x = q_1, \dot{q}_1, q_2, \dot{q}_2)^T$ .

*Hint:* Set  $y = q_1 = Cx$  and show that the system can be written in state state as

$$\dot{x} = Ax + bu + \phi(y)$$

where  $\phi(y)$  is a nonlinear function depending only on the output  $y$ . Then a **linear observer with output injection** can be designed as

$$\dot{\hat{x}} = A\hat{x} + bu + \phi(y) + L(y - C\hat{x})$$

- 10-18 Fill in the details in Example 10.7 showing that the constraints are nonholonomic.
- 10-19 Fill in the details in Example 10.8 necessary to derive the vector fields  $g_1$  and  $g_2$  and show that the constraints are nonholonomic.
- 10-20 Carry out the calculations necessary to show that the equations of motion for the satellite with reaction wheels is given by Equation 10.124.
- 10-21 Show that the distribution  $\Delta = \text{span}(g_1, g_2, g_3)$  in for the satellite model (10.124) is involutive of rank 3.
- 10-22 Using Chow's Theorem, show that the satellite with reaction wheels (10.124) is controllable as long as any two of the three reaction wheels are functioning.



# *11*

---

## *COMPUTER VISION*

If a robot is to interact with its environment, then the robot must be able to sense its environment. Computer vision is one of the most powerful sensing modalities that currently exist. Therefore, in this chapter we present a number of basic concepts from the field of computer vision. It is not our intention here to cover the now vast field of computer vision. Rather, we aim to present a number of basic techniques that are applicable to the highly constrained problems that often present themselves in industrial applications. The material in this chapter, when combined with the material of previous chapters, should enable the reader to implement a rudimentary vision-based robotic manipulation system. For example, using techniques presented in this chapter, one could design a system that locates objects on a conveyor belt, and determines the positions and orientations of those objects. This information could then be used in conjunction with the inverse kinematic solution for the robot, along with various coordinate transformations, to command the robot to grasp these objects.

We begin by examining the geometry of the image formation process. This will provide us with the fundamental geometric relationships between objects in the world and their projections in an image. We then describe a calibration process that can be used to determine the values for the various camera parameters that appear in these relationships. We then consider image segmentation, the problem of dividing the image into distinct regions that correspond to the background and to objects in the scene. When there are multiple objects in the scene, it is often useful to deal with them individually; therefore, we next present an approach to component labelling. Finally, we describe how to compute the positions and orientations of objects in the image.

## 11.1 THE GEOMETRY OF IMAGE FORMATION

A digital image is a two-dimensional array of pixels that is formed by focusing light onto a two-dimensional array of sensing elements. A lens with focal length  $\lambda$  is used to focus the light onto the sensing array, which is often composed of CCD (charge-coupled device) sensors. The lens and sensing array are packaged together in a camera, which is connected to a digitizer or frame grabber. In the case of analog cameras, the digitizer converts the analog video signal that is output by the camera into discrete values that are then transferred to the pixel array by the frame grabber. In the case of digital cameras, a frame grabber merely transfers the digital data from the camera to the pixel array. Associated with each pixel in the digital image is a gray level value, typically between 0 and 255, which encodes the intensity of the light incident on the corresponding sensing element.

In robotics applications, it is often sufficient to consider only the geometric aspects of image formation. Therefore in this section we will describe only the geometry of the image formation process. We will not deal with the photometric aspects of image formation (e.g., we will not address issues related to depth of field, lens models, or radiometry).

We will begin the section by assigning a coordinate frame to the imaging system. We then discuss the popular pinhole model of image formation, and derive the corresponding equations that relate the coordinates of a point in the world to its image coordinates. Finally, we describe camera calibration, the process by which all of the relevant parameters associated with the imaging process can be determined.

### 11.1.1 The Camera Coordinate Frame

In order to simplify many of the equations of this chapter, it often will be useful to express the coordinates of objects relative to a camera centered coordinate frame. For this purpose, we define the camera coordinate frame as follows. Define the image plane,  $\pi$ , as the plane that contains the sensing array. The axes  $x_c$  and  $y_c$  form a basis for the image plane, and are typically taken to be parallel to the horizontal and vertical axes (respectively) of the image. The axis  $z_c$  is perpendicular to the image plane and aligned with the optic axis of the lens (i.e., it passes through the focal center of the lens). The origin of the camera frame is located at a distance  $\lambda$  behind the image plane. This point is also referred to as the *center of projection*. The point at which the optical axis intersects the image plane is known as the *principal point*. This coordinate frame is illustrated in Figure 11.1.

With this assignment of the camera frame, any point that is contained in the image plane will have coordinates  $(u, v, \lambda)$ . Thus, we can use  $(u, v)$  to parameterize the image plane, and we will refer to  $(u, v)$  as image plane coordinates.

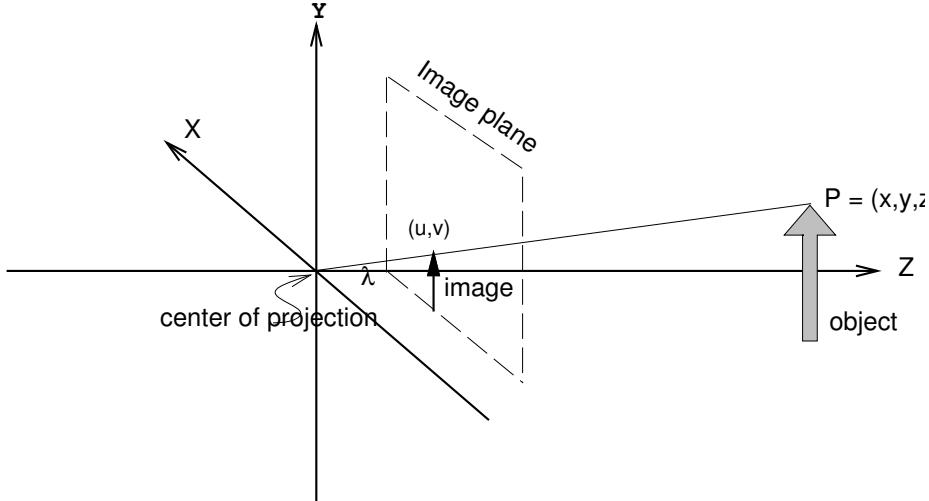


Fig. 11.1 Camera coordinate frame.

### 11.1.2 Perspective Projection

The image formation process is often modeled by the pinhole lens approximation. With this approximation, the lens is considered to be an ideal pinhole, and the pinhole is located at the focal center of the lens<sup>1</sup>. Light rays pass through this pinhole, and intersect the image plane.

Let  $P$  be a point in the world with coordinates  $x, y, z$  (relative to the camera frame). Let  $p$  denote the projection of  $P$  onto the image plane with coordinates  $(u, v, \lambda)$ . Under the pinhole assumption,  $P$ ,  $p$  and the origin of the camera frame will be collinear. This can be illustrated in Figure 11.1. Thus, for some unknown positive  $k$  we have

$$k \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u \\ v \\ \lambda \end{pmatrix} \quad (11.1)$$

which can be rewritten as the system of equations:

$$kx = u, \quad (11.2)$$

$$ky = v, \quad (11.3)$$

$$kz = \lambda. \quad (11.4)$$

<sup>1</sup>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.

This gives  $k = \lambda/z$ , which can be substituted into (11.2) and (11.3) to obtain

$$u = \lambda \frac{x}{z}, \quad v = \lambda \frac{y}{z}. \quad (11.5)$$

These are the well-known equations for **perspective projection**.

### 11.1.3 The Image Plane and the Sensor Array

As described above, the image is a discrete array of gray level values. We will denote the row and column indices for a pixel by the coordinates  $(r, c)$ . In order to relate digital images to the 3D world, we must determine the relationship between the image plane coordinates,  $(u, v)$ , and indices into the pixel array of pixels,  $(r, c)$ .

We typically define the origin of this pixel array to be located at a corner of the image (rather than 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 the  $s_x$  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 the camera coordinate frame<sup>2</sup>. Combining these, we obtain the following relationship between image plane coordinates and pixel array coordinates,

$$-\frac{u}{s_x} = (r - o_r), \quad -\frac{v}{s_y} = (c - o_c). \quad (11.6)$$

which gives,

$$u = -s_x(r - o_r), \quad v = -s_y(c - o_c). \quad (11.7)$$

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, it is not possible to obtain the exact image plane coordinates for a point from the  $(r, c)$  coordinates.

## 11.2 CAMERA CALIBRATION

The objective of camera calibration is to determine all of the parameters that are necessary to predict the image pixel coordinates  $(r, c)$  of the projection of a point in the camera's field of view, given that the coordinates of that point with respect to the world coordinate frame are known. In other words, given the coordinates of  $P$  relative to the world coordinate frame, after we have calibrated

<sup>2</sup>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 frame grabber.

the camera we will be able to predict  $(r, c)$ , the image pixel coordinates for the projection of this point.

### 11.2.1 Extrinsic Camera Parameters

To this point, in our derivations of the equations for perspective projection, we have dealt only with coordinates expressed relative to the camera frame. In typical robotics applications, tasks will be expressed in terms of the world coordinate frame, and it will therefore be necessary to perform coordinate transformations. If we know the position and orientation of the camera frame relative to the world coordinate frame we have

$$x^w = R_c^w x^c + O_c^w \quad (11.8)$$

or, if we know  $x^w$  and wish to solve for  $x^c$ ,

$$x^c = R_w^c (x^w - O_c^w) \quad (11.9)$$

In the remainder of this section, to simplify notation we will define

$$R = R_w^c, \quad T = -R_w^c O_c^w. \quad (11.10)$$

Thus,

$$x^c = Rx^w + T \quad (11.11)$$

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 or down, and can turn from side to side. In this case, the rotation matrix  $R$  is given by

$$R = R_{z,\theta} R_{x,\alpha}, \quad (11.12)$$

where  $\theta$  is the pan angle and  $\alpha$  is the tilt angle. More precisely,  $\theta$  is the angle between the world  $x$ -axis and the camera  $x$ -axis, about the world  $z$ -axis, while  $\alpha$  is the angle between the world  $z$ -axis and the camera  $z$ -axis, about the camera  $x$ -axis.

### 11.2.2 Intrinsic Camera Parameters

Using the pinhole model, we obtained the following equations that map the coordinates of a point expressed with respect to the camera frame to the corresponding pixel coordinates:

$$r = -\frac{u}{s_x} + o_r, \quad c = -\frac{v}{s_y} + o_c, \quad u = \lambda \frac{x}{z}, \quad v = \lambda \frac{y}{z}. \quad (11.13)$$

These equations can be combined to give

$$r = -\frac{\lambda}{s_x} \frac{x}{z} + o_r, \quad c = -\frac{\lambda}{s_y} \frac{y}{z} + o_c, \quad (11.14)$$

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  $\lambda, s_x, s_y$ ; it is sufficient to know the ratios

$$f_x = -\frac{\lambda}{s_x} \quad f_y = -\frac{\lambda}{s_y}. \quad (11.15)$$

These parameters,  $f_x, o_r, f_y, o_c$  are known as the intrinsic parameters of the camera. They are constant for a given camera, and do not change when the camera moves.

### 11.2.3 Determining the Camera Parameters

The task of camera calibration is to determine the intrinsic and extrinsic parameters of the camera. We will proceed by first determining the parameters associated with the image center, and then solving for the remaining 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. Although a full treatment of vanishing points is beyond the scope of this text, the idea is simple: a set of parallel lines in the world will project onto image lines that intersect at a single point, and this intersection point is known as a *vanishing point*. The vanishing points for three mutually orthogonal sets of lines in the world will define a triangle in the image. The orthocenter of this triangle (i.e., the point at which the three altitudes intersect) is the image principal point (a proof of this is beyond the scope of this text). 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, and determine the orthocenter for the resulting triangle.

Once we know the principal point, we proceed to determine the remaining camera parameters. This is done by constructing a linear 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. Thus, the first step in this stage of calibration is to acquire a data set of the form  $r_1, c_1, x_1, y_1, z_1, r_2, c_2, x_2, y_2, z_2, \dots, r_N, c_N, x_N, y_N, z_N$ , 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, e.g., by having a robot move a small bright light to 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 the linear system of equations. The extrinsic parameters of the camera are given by

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}. \quad (11.16)$$

With respect to the camera frame, the coordinates of a point in the world are thus given by

$$x^c = r_{11}x + r_{12}y + r_{13}z + T_x \quad (11.17)$$

$$y^c = r_{21}x + r_{22}y + r_{23}z + T_y \quad (11.18)$$

$$z^c = r_{31}x + r_{32}y + r_{33}z + T_z. \quad (11.19)$$

Combining this with (11.14) we obtain

$$r - o_r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \quad (11.20)$$

$$c - o_c = -f_y \frac{y^c}{z^c} = -f_y \frac{r_{21}x + r_{22}y + r_{23}z + T_y}{r_{31}x + r_{32}y + r_{33}z + T_z}. \quad (11.21)$$

Since we know the coordinates of the principal point, we can simplify these equations by using the simple coordinate transformation

$$r \leftarrow r - o_r, \quad c \leftarrow c - o_c. \quad (11.22)$$

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 each of these equations for  $z^c$ , and setting the resulting equations to be equal to one another. In particular, for the data points  $r_i, c_i, x_i, y_i, z_i$  we have

$$r_i f_y (r_{21}x_i + r_{22}y_i + r_{23}z_i + T_y) = c_i f_x (r_{11}x_i + r_{12}y_i + r_{13}z_i + T_x). \quad (11.23)$$

We define  $\alpha = f_x/f_y$  and rewrite this as:

$$r_i r_{21}x_i + r_i r_{22}y_i + r_i r_{23}z_i + r_i T_y - \alpha c_i r_{11}x_i - \alpha c_i r_{12}y_i - \alpha c_i r_{13}z_i - \alpha c_i T_x = 0. \quad (11.24)$$

We can combine the  $N$  such equations into the matrix equation

$$\begin{bmatrix} r_1 x_1 & r_1 y_1 & r_1 z_1 & r_1 & -c_1 x_1 & -c_1 y_1 & -c_1 z_1 & -c_1 \\ r_2 x_2 & r_2 y_2 & r_2 z_2 & r_2 & -c_2 x_2 & -c_2 y_2 & -c_2 z_2 & -c_2 \\ \vdots & \vdots \\ r_N x_N & r_N y_N & r_N z_N & r_N & -c_N x_N & -c_N y_N & -c_N z_N & -c_N \end{bmatrix} \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \\ T_y \\ \alpha r_{11} \\ \alpha r_{12} \\ \alpha r_{13} \\ \alpha T_x \end{bmatrix} = 0 \quad (11.25)$$

This is an equation of the form  $Ax = 0$ . As such, if  $\bar{x} = [\bar{x}_1, \dots, \bar{x}_8]^T$  is a solution, for (11.25) we only know that this solution is some scalar multiple of the desired solution,  $x$ , i.e.,

$$\bar{x} = k[r_{21}, r_{22}, r_{23}, T_y, \alpha r_{11}, \alpha r_{12}, \alpha r_{13}, \alpha T_x]^T, \quad (11.26)$$

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,

$$(\bar{x}_1^2 + \bar{x}_2^2 + \bar{x}_3^2)^{\frac{1}{2}} = (k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = |k|, \quad (11.27)$$

and likewise

$$(\bar{x}_5^2 + \bar{x}_6^2 + \bar{x}_7^2)^{\frac{1}{2}} = (\alpha^2 k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = \alpha|k| \quad (11.28)$$

(note that by definition,  $\alpha > 0$ ).

Our next task is to determine the sign of  $k$ . Using equations (11.14) we see that  $r \times x^c < 0$  (recall that we have used the coordinate transformation  $r \leftarrow r - o_r$ ). Therefore, to determine the sign of  $k$ , we first assume that  $k > 0$ . If  $r(r_{11}x + r_{12}y + r_{13}z + T_x) < 0$ , then we know that we have made the right choice and  $k > 0$ ; otherwise, we know that  $k < 0$ .

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  $\alpha = f_x/f_y$ , we need only determine  $T_z$  and  $f_x$ . Returning again to the projection equations, we can write

$$r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \quad (11.29)$$

Using an approach similar to that used above to solve for the first eight parameters, we can write this as the linear system

$$r(r_{31}x + r_{32}y + r_{33}z + T_z) = -f_x(r_{11}x + r_{12}y + r_{13}z + T_x) \quad (11.30)$$

which can easily be solved for  $T_z$  and  $f_x$ .

### 11.3 SEGMENTATION BY THRESHOLDING

Segmentation is the process by which an image is divided into meaningful components. Segmentation has been the topic of computer vision research since its earliest days, and the approaches to segmentation are far too numerous to survey here. These approaches are sometimes concerned with finding *features* in an image (e.g., edges), and sometimes concerned with partitioning the image into homogeneous regions (region-based segmentation). In many practical applications, the goal of segmentation is merely to divide the image into two regions: one region that corresponds to an object in the scene, and one region

```

For  $i = 0$  to  $N - 1$ 
     $H[i] \leftarrow 0$ 
For  $r = 0$  to  $N_{rows} - 1$ 
    For  $c = 0$  to  $N_{cols} - 1$ 
         $Index \leftarrow Image(r, c)$ 
         $H[Index] \leftarrow H[Index] + 1$ 

```

*Fig. 11.2* Pseudo-code to compute an image histogram.

that corresponds to the background. In many industrial applications, this segmentation can be accomplished by a straight-forward thresholding approach. Pixels whose gray level is greater than the threshold are considered to belong to the object, and pixels whose gray level is less than or equal to the threshold are considered to belong to the background.

In this section we will describe an algorithm that automatically selects a threshold. This basic idea behind the algorithm is that the pixels should be divided into two groups (background and object), and that the intensities of the pixels in a particular group should all be fairly similar. To quantify this idea, we will use some standard techniques from statistics. Thus, we begin the section with a quick review of the necessary concepts from statistics and then proceed to describe the threshold selection algorithm.

### 11.3.1 A Brief Statistics Review

Many approaches to segmentation exploit statistical information contained in the image. In this section, we briefly review some of the more useful statistical concepts that are used by segmentation algorithms.

The basic premise for most of these statistical concepts is that the gray level value associated with a pixel in an image is a random variable that takes on values in the set  $\{0, 1, \dots, N - 1\}$ . Let  $P(z)$  denote the probability that a pixel has gray level value  $z$ . In general, we will not know this probability, but we can estimate it with the use of a *histogram*. A histogram is an array,  $H$ , that encodes the number of occurrences of each gray level value. In particular, the entry  $H[z]$  is the number of times gray level value  $z$  occurs in the image. Thus,  $0 \leq H[z] \leq N_{rows} \times N_{cols}$  for all  $z$ . An algorithm to compute the histogram for an image is shown in figure 11.2.

Given the histogram for the image, we estimate the probability that a pixel will have gray level  $z$  by

$$P(z) = \frac{H[z]}{N_{rows} \times N_{cols}}. \quad (11.31)$$

Thus, the image histogram is a scaled version of our approximation of  $P$ .

Given  $P$ , we can compute the *average*, or *mean* value of the gray level values in the image. We denote the mean by  $\mu$ , and compute it by

$$\mu = \sum_{z=0}^{N-1} z P(z). \quad (11.32)$$

In many applications, the image will consist of one or more objects against some background. In such applications, it is often useful to compute the mean for each object in the image, and also for the background. This computation can be effected by constructing individual histogram arrays for each object, and for the background, in the image. If we denote by  $H_i$  the histogram for the  $i^{th}$  object in the image (where  $i = 0$  denotes the background), the mean for the  $i^{th}$  object is given by

$$\mu_i = \sum_{z=0}^{N-1} z \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}, \quad (11.33)$$

which is a straightforward generalization of (11.32). The term

$$\frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}$$

is in fact an estimate of the probability that a pixel will have gray level value  $z$  given that the pixel is a part of object  $i$  in the image. For this reason,  $\mu_i$  is sometimes called a *conditional mean*.

The mean conveys useful, but very limited information about the distribution of grey level values in an image. For example, if half or the pixels have gray value 127 and the remaining half have gray value 128, the mean will be  $\mu = 127.5$ . Likewise, if half or the pixels have gray value 255 and the remaining half have gray value 0, the mean will be  $\mu = 127.5$ . Clearly these two images are very different, but this difference is not reflected by the mean. One way to capture this difference is to compute the average deviation of gray values from the mean. This average would be small for the first example, and large for the second. We could, for example, use the average value of  $|z - \mu|$ ; however, it will be more convenient mathematically to use the square of this value instead. The resulting quantity is known as the *variance*, which is defined by

$$\sigma^2 = \sum_{z=0}^{N-1} (z - \mu)^2 P(z). \quad (11.34)$$

As with the mean, we can also compute the conditional variance,  $\sigma_i^2$  for each object in the image

$$\sigma_i^2 = \sum_{z=0}^{N-1} (z - \mu_i)^2 \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}. \quad (11.35)$$

### 11.3.2 Automatic Threshold Selection

We are now prepared to develop an automatic threshold selection algorithm. We will assume that the image consists of an object and a background, and that the background pixels have gray level values less than or equal to some threshold while the object pixels are above the threshold. Thus, for a given threshold value,  $z_t$ , we divide the image pixels into two groups: those pixels with gray level value  $z \leq z_t$ , and those pixels with gray level value  $z > z_t$ . We can compute the means and variance for each of these groups using the equations of Section 11.3.1. Clearly, the conditional means and variances depend on the choice of  $z_t$ , since it is the choice of  $z_t$  that determines which pixels will belong to each of the two groups. The approach that we take in this section is to determine the value for  $z_t$  that minimizes a function of the variances of these two groups of pixels.

In this section, it will be convenient to rewrite the conditional means and variances in terms of the pixels in the two groups. To do this, we define  $q_i(z_t)$  as the probability that a pixel in the image will belong to group  $i$  for a particular choice of threshold,  $z_t$ . Since all pixels in the background have gray value less than or equal to  $z_t$  and all pixels in the object have gray value greater than  $z_t$ , we can define  $q_i(z_t)$  for  $i = 0, 1$  by

$$q_0(z_t) = \frac{\sum_{z=0}^{z_t} H[z]}{(N_{rows} \times N_{cols})}, \quad q_1(z_t) = \frac{\sum_{z=z_t+1}^{N-1} H[z]}{(N_{rows} \times N_{cols})}. \quad (11.36)$$

We now rewrite (11.33) as

$$\mu_i = \sum_{z=0}^{N-1} z \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]} = \sum_{z=0}^{N-1} z \frac{H_i[z]/(N_{rows} \times N_{cols})}{\sum_{z=0}^{N-1} H_i[z]/(N_{rows} \times N_{cols})}$$

Using again the fact that the two pixel groups are defined by the threshold  $z_t$ , we have

$$\frac{H_0[z]}{(N_{rows} \times N_{cols})} = \frac{P(z)}{q_0(z_t)}, \quad z \leq z_t \quad \text{and} \quad \frac{H_1[z]}{(N_{rows} \times N_{cols})} = \frac{P(z)}{q_1(z_t)}, \quad z > z_t. \quad (11.37)$$

Thus, we can write the conditional means for the two groups as

$$\mu_0(z_t) = \sum_{z=0}^{z_t} z \frac{P(z)}{q_0(z_t)}, \quad \mu_1(z_t) = \sum_{z=z_t+1}^{N-1} z \frac{P(z)}{q_1(z_t)}. \quad (11.38)$$

Similarly, we can write the equations for the conditional variances by

$$\sigma_0^2(z_t) = \sum_{z=0}^{z_t} (z - \mu_0(z_t))^2 \frac{P(z)}{q_0(z_t)}, \quad \sigma_1^2(z_t) = \sum_{z=z_t+1}^{N-1} (z - \mu_1(z_t))^2 \frac{P(z)}{q_1(z_t)}. \quad (11.39)$$

We now turn to the selection of  $z_t$ . If nothing is known about the true values of  $\mu_i$  or  $\sigma_i^2$ , how can we determine the optimal value of  $z_t$ ? To answer this

question, recall that the variance is a measure of the average deviation of pixel intensities from the mean. Thus, if we make a good choice for  $z_t$ , we would expect that the variances  $\sigma_i^2(z_t)$  would be small. This reflects the assumption that pixels belonging to the object will be clustered closely about  $\mu_1$ , pixels belonging to the background will be clustered closely about  $\mu_0$ . We could, therefore, select the value of  $z_t$  that minimizes the sum of these two variances. However, it is unlikely that the object and background will occupy the same number of pixels in the image; merely adding the variances gives both regions equal importance. A more reasonable approach is to weight the two variances by the probability that a pixel will belong to the corresponding region,

$$\sigma_w^2(z_t) = q_0(z_t)\sigma_0^2(z_t) + q_1(z_t)\sigma_1^2(z_t). \quad (11.40)$$

The value  $\sigma_w^2$  is known as the *within-group variance*. The approach that we will take is to minimize this within-group variance.

At this point we could implement a threshold selection algorithm. The naive approach would be to simply iterate over all possible values of  $z_t$  and select the one for which  $\sigma_w^2(z_t)$  is smallest. Such an algorithm performs an enormous amount of calculation, much of which is identical for different candidate values of the threshold. For example, most of the calculations required to compute  $\sigma_w^2(z_t)$  are also required to compute  $\sigma_w^2(z_t + 1)$ ; the required summations change only slightly from one iteration to the next. Therefore, we now turn our attention to an efficient algorithm.

To develop an efficient algorithm, we take two steps. First, we will derive the between-group variance,  $\sigma_b^2$ , which depends on the within-group variance and the variance over the entire image. The between-group variance is a bit simpler to deal with than the within-group variance, and we will show that maximizing the between-group variance is equivalent to minimizing the within-group variance. Then, we will derive a recursive formulation for the between-group variance that lends itself to an efficient implementation.

To derive the between-group variance, we begin by expanding the equation for the total variance of the image, and then simplifying and grouping terms. The variance of the gray level values in the image is given by (11.34), which

can be rewritten as

$$\begin{aligned}
\sigma^2 &= \sum_{z=0}^{N-1} (z - \mu)^2 P(z) \\
&= \sum_{z=0}^{z_t} (z - \mu)^2 P(z) + \sum_{z=z_t+1}^{N-1} (z - \mu)^2 P(z) \\
&= \sum_{z=0}^{z_t} (z - \mu_0 + \mu_0 - \mu)^2 P(z) + \sum_{z=z_t+1}^{N-1} (z - \mu_1 + \mu_1 - \mu)^2 P(z) \\
&= \sum_{z=0}^{z_t} [(z - \mu_0)^2 + 2(z - \mu_0)(\mu_0 - \mu) + (\mu_0 - \mu)^2] P(z) \\
&\quad + \sum_{z=z_t+1}^{N-1} [(z - \mu_1)^2 + 2(z - \mu_1)(\mu_1 - \mu) + (\mu_1 - \mu)^2] P(z). \quad (11.41)
\end{aligned}$$

Note that we have not explicitly noted the dependence on  $z_t$  here. In the remainder of this section, to simplify notation, we will refer to the group probabilities and conditional means and variances as  $q_i$ ,  $\mu_i$ , and  $\sigma_i^2$ , without explicitly noting the dependence on  $z_t$ . This last expression (11.41) can be further simplified by examining the cross-terms

$$\begin{aligned}
\sum (z - \mu_i)(\mu_i - \mu) P(z) &= \sum z \mu_i P(z) - \sum z \mu P(z) - \sum \mu_i^2 P(z) + \sum \mu_i \mu P(z) \\
&= \mu_i \sum z P(z) - \mu \sum z P(z) - \mu_i^2 \sum P(z) + \mu_i \mu \sum P(z) \\
&= \mu_i(\mu_i q_i) - \mu(\mu_i q_i) - \mu_i^2 q_i + \mu_i \mu q_i \\
&= 0,
\end{aligned}$$

in which the summations are taken for  $z$  from 0 to  $z_t$  for the background pixels (i.e.,  $i = 0$ ) and  $z$  from  $z_t + 1$  to  $N - 1$  for the object pixels (i.e.,  $i = 1$ ). Therefore, we can simplify (11.41) to obtain

$$\begin{aligned}
\sigma^2 &= \sum_{z=0}^{z_t} [(z - \mu_0)^2 + (\mu_0 - \mu)^2] P(z) + \sum_{z=z_t+1}^{N-1} [(z - \mu_1)^2 + (\mu_1 - \mu)^2] P(z) \\
&= q_0 \sigma_0^2 + q_0(\mu_0 - \mu)^2 + q_1 \sigma_1^2 + q_1(\mu_1 - \mu)^2 \\
&= \{q_0 \sigma_0^2 + q_1 \sigma_1^2\} + \{q_0(\mu_0 - \mu)^2 + q_1(\mu_1 - \mu)^2\} \\
&= \sigma_w^2 + \sigma_b^2
\end{aligned} \quad (11.42)$$

in which

$$\sigma_b^2 = q_0(\mu_0 - \mu)^2 + q_1(\mu_1 - \mu)^2. \quad (11.43)$$

Since  $\sigma^2$  does not depend on the threshold value (i.e., it is constant for a specific image), minimizing  $\sigma_w^2$  is equivalent to maximizing  $\sigma_b^2$ . This is preferable because  $\sigma_b^2$  is a function only of the  $q_i$  and  $\mu_i$ , and is thus simpler to compute

than  $\sigma_w^2$ , which depends also on the  $\sigma_i^2$ . In fact, by expanding the squares in (11.43), using the facts that  $q_1 = 1 - q_0$  and  $\mu = q_1\mu_0 + q_1\mu_1$ , we obtain

$$\sigma_b^2 = q_0(1 - q_0)(\mu_0 - \mu_1)^2. \quad (11.44)$$

The simplest algorithm to maximize  $\sigma_b^2$  is to iterate over all possible threshold values, and select the one that maximizes  $\sigma_b^2$ . However, as discussed above, such an algorithm performs many redundant calculations, since most of the calculations required to compute  $\sigma_b^2(z_t)$  are also required to compute  $\sigma_b^2(z_t + 1)$ . Therefore, we now turn our attention to an efficient algorithm that maximizes  $\sigma_b^2(z_t)$ . The basic idea for the efficient algorithm is to re-use the computations needed for  $\sigma_b^2(z_t)$  when computing  $\sigma_b^2(z_t + 1)$ . In particular, we will derive expressions for the necessary terms at iteration  $z_t + 1$  in terms of expressions that were computed at iteration  $z_t$ . We begin with the group probabilities, and determine the recursive expression for  $q_0$  as

$$q_0(z_t + 1) = \sum_{z=0}^{z_t+1} P(z) = P(z_t + 1) + \sum_{z=0}^{z_t} P(z) = P(z_t + 1) + q_0(z_t). \quad (11.45)$$

In this expression,  $P(z_t + 1)$  can be obtained directly from the histogram array, and  $q_0(z_t)$  is directly available because it was computed on the previous iteration of the algorithm. Thus, given the results from iteration  $z_t$ , very little computation is required to compute the value for  $q_0$  at iteration  $z_t + 1$ .

For the conditional mean  $\mu_0(z_t)$  we have

$$\mu_0(z_t + 1) = \sum_{z=0}^{z_t+1} z \frac{P(z)}{q_0(z_t + 1)} \quad (11.46)$$

$$= \frac{(z_t + 1)P(z_t + 1)}{q_0(z_t + 1)} + \sum_{z=0}^{z_t} z \frac{P(z)}{q_0(z_t + 1)} \quad (11.47)$$

$$= \frac{(z_t + 1)P(z_t + 1)}{q_0(z_t + 1)} + \frac{q_0(z_t)}{q_0(z_t + 1)} \sum_{z=0}^{z_t} z \frac{P(z)}{q_0(z_t)} \quad (11.48)$$

$$= \frac{(z_t + 1)P(z_t + 1)}{q_0(z_t + 1)} + \frac{q_0(z_t)}{q_0(z_t + 1)} \mu_0(z_t) \quad (11.49)$$

Again, all of the quantities in this expression are available either from the histogram, or as the results of calculations performed at iteration  $z_t$  of the algorithm.

To compute  $\mu_1(z_t + 1)$ , we use the relationship  $\mu = q_0\mu_0 + q_1\mu_1$ , which can be easily obtained using (11.32) and (11.38). Thus, we have

$$\mu_1(z_t + 1) = \frac{\mu - q_0(z_t + 1)\mu_0(z_t + 1)}{q_1(z_t + 1)} = \frac{\mu - q_0(z_t + 1)\mu_0(z_t + 1)}{1 - q_0(z_t + 1)}. \quad (11.50)$$

We are now equipped to construct a highly efficient algorithm that automatically selects a threshold value that minimizes the within-group variance. This

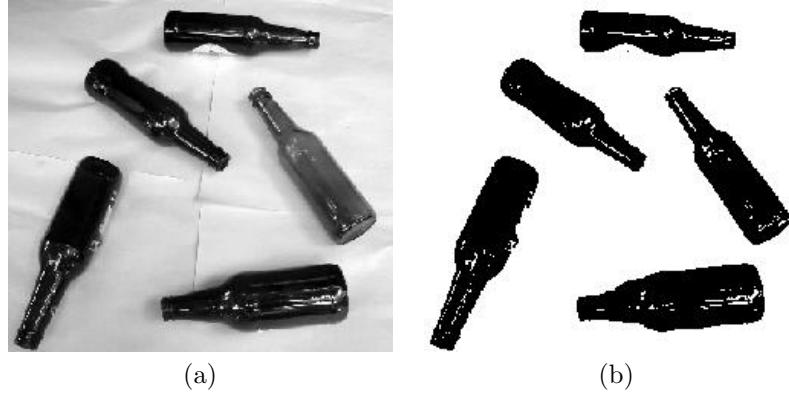


Fig. 11.3 (a) An image with 256 gray levels. (b) Thresholded version of the image in (a).

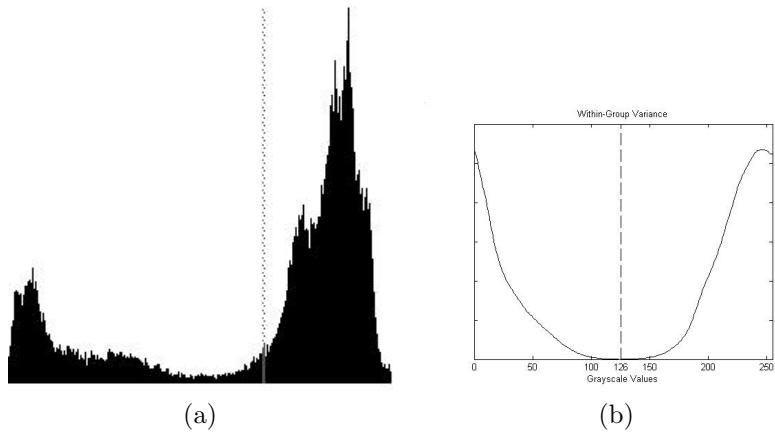


Fig. 11.4 (a) Histogram for the image shown in 11.3a (b) Within-group variance for the image shown in 11.3a

algorithm simply iterates from 0 to  $N - 1$  (where  $N$  is the total number of gray level values), computing  $q_0$ ,  $\mu_0$ ,  $\mu_1$  and  $\sigma_b^2$  at each iteration using the recursive formulations given in (11.45), (11.49), (11.50) and (11.44). The algorithm returns the value of  $z_t$  for which  $\sigma_b^2$  is largest. Figure 11.3 shows a grey level image and the binary, thresholded image that results from the application of this algorithm. Figure 11.4 shows the histogram and within-group variance for the grey level image.

## 11.4 CONNECTED COMPONENTS

It is often the case that multiple objects will be present in a single image. When this occurs, after thresholding there will be multiple connected components with gray level values that are above the threshold. In this section, we will first make precise the notion of a connected component, and then describe an algorithm that assigns a unique label to each connected component, i.e., all pixels within a single connected component have the same label, but pixels in different connected components have different labels.

In order to define what is meant by a connected component, it is first necessary to define what is meant by connectivity. For our purposes, it is sufficient to say that a pixel,  $A$ , with image pixel coordinates  $(r, c)$  is adjacent to four pixels, those with image pixel coordinates  $(r - 1, c)$ ,  $(r + 1, c)$ ,  $(r, c + 1)$ , and  $(r, c - 1)$ . In other words, each image pixel  $A$  (except those at the edges of the image) has four neighbors: the pixel directly above, directly below, directly to the right and directly to the left of pixel  $A$ . This relationship is sometimes referred to as *4-connectivity*, and we say that two pixels are 4-connected if they are adjacent by this definition. If we expand the definition of adjacency to include those pixels that are diagonally adjacent (i.e., the pixels with coordinates  $(r - 1, c - 1)$ ,  $(r - 1, c + 1)$ ,  $(r + 1, c - 1)$ , and  $(r + 1, c + 1)$ ), then we say that adjacent pixels are 8-connected. In this text, we will consider only the case of 4-connectivity.

A connected component is a collection of pixels,  $S$ , such that for any two pixels, say  $P$  and  $P'$ , in  $S$ , there is a 4-connected path between them and this path is contained in  $S$ . Intuitively, this definition means that it is possible to move from  $P$  to  $P'$  by “taking steps” only to adjacent pixels without ever leaving the region  $S$ . The purpose of a component labeling algorithm is to assign a unique label to each such  $S$ .

There are many component labeling algorithms that have been developed over the years. Here, we describe a simple algorithm that requires two passes over the image. This algorithm performs two raster scans of the image (note: a raster scan visits each pixel in the image by traversing from left to right, top to bottom, in the same way that one reads a page of text). On the first raster scan, when an object pixel  $P$ , (i.e., a pixel whose gray level is above the threshold value), is encountered, its previously visited neighbors (i.e., the pixel immediately above and the pixel immediately to the left of  $P$ ) are examined, and if they have gray value that is below the threshold (i.e., they are background pixels), a new label is given to  $P$ . This is done by using a global counter that is initialized to zero, and is incremented each time a new label is needed. If either of these two neighbors have already received labels, then  $P$  is given the smaller of these, and in the case when both of the neighbors have received labels, an equivalence is noted between those two labels. For example, in Figure 11.5, after the first raster scan labels (2,3,4) are noted as equivalent. In the second raster scan, each pixel’s label is replaced by the smallest label to which it is

|                     |                     |
|---------------------|---------------------|
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 |
| 0 X X X 0 0 0 0 0 0 | 0 1 1 1 0 0 0 0 0 0 |
| 0 X X X 0 0 0 0 0 0 | 0 1 1 1 0 0 0 0 0 0 |
| 0 X X X 0 0 0 0 0 0 | 0 1 1 1 0 0 0 0 0 0 |
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 X 0 0 X X 0 | 0 0 0 0 2 0 0 3 3 0 |
| 0 0 0 0 X 0 0 X X 0 | 0 0 0 0 2 0 0 3 3 0 |
| 0 0 0 0 X X X X X 0 | 0 0 0 0 2 2 2 2 2 0 |
| 0 X X X X X X X X 0 | 0 4 4 4 2 2 2 2 2 0 |
| 0 X X X X X X X X 0 | 0 4 4 4 2 2 2 2 2 0 |
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 |

|      |      |
|------|------|
| (a)  | (b)  |
| <br> | <br> |

|                     |                     |
|---------------------|---------------------|
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 |
| 0 1 1 1 0 0 0 0 0 0 | 0 1 1 1 0 0 0 0 0 0 |
| 0 1 1 1 0 0 0 0 0 0 | 0 1 1 1 0 0 0 0 0 0 |
| 0 1 1 1 0 0 0 0 0 0 | 0 1 1 1 0 0 0 0 0 0 |
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 2 0 0 3 3 0 | 0 0 0 0 2 0 0 2 2 0 |
| 0 0 0 0 2 0 0 3 3 0 | 0 0 0 0 2 0 0 2 2 0 |
| 0 0 0 0 2 2 2 X 2 0 | 0 0 0 0 2 2 2 2 2 0 |
| 0 4 4 4 X 2 2 2 2 0 | 0 2 2 2 2 2 2 2 2 0 |
| 0 4 4 4 2 2 2 2 2 0 | 0 2 2 2 2 2 2 2 2 0 |
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 |

|      |      |
|------|------|
| (c)  | (d)  |
| <br> | <br> |

Fig. 11.5 The image in (a) is a simple binary image. Background pixels are denoted by 0 and object pixels are denoted by X. Image (b) shows the assigned labels after the first raster scan. In image (c), an X denotes those pixels at which an equivalence is noted during the first raster scan. Image (d) shows the final component labelled image.

equivalent. Thus, in the example of Figure 11.5, at the end of the second raster scan labels 3 and 4 have been replaced by the label 2.

After this algorithm has assigned labels to the components in the image, it is not necessarily the case that the labels will be the consecutive integers  $(1, 2, \dots)$ . Therefore, a second stage of processing is sometimes used to relabel the components to achieve this. In other cases, it is desirable to give each component a label that is very different from the labels of the other components. For example, if the component labelled image is to be displayed, it is useful to



*Fig. 11.6* The image of Figure 11.3 after connected components have been labelled.

increase the contrast, so that distinct components will actually appear distinct in the image (a component with the label 2 will appear almost indistinguishable from a component with label 3 if the component labels are used as pixel gray values in the displayed component labelled image). The results of applying this process to the image in Figure 11.3 are shown in Figure 11.6.

When there are multiple connected object components, it is often useful to process each component individually. For example, we might like to compute the sizes of the various components. For this purpose, it is useful to introduce the *indicator function* for a component. The indicator function for component  $i$ , denoted by  $\mathcal{I}_i$ , is a function that takes on the value 1 for pixels that are contained in component  $i$ , and the value 0 for all other pixels:

$$\mathcal{I}_i(r, c) = \begin{cases} 1 & : \text{pixel } r, c \text{ is contained in component } i \\ 0 & : \text{otherwise} \end{cases}. \quad (11.51)$$

We will make use of the indicator function below, when we discuss computing statistics associated with the various objects in the image.

## 11.5 POSITION AND ORIENTATION

The ultimate goal of a robotic system is to manipulate objects in the world. In order to achieve this, it is necessary to know the positions and orientations of the objects that are to be manipulated. In this section, we address the problem of determining the position and orientation of objects *in the image*. If the camera has been calibrated, it is then possible to use these image position and orientations to infer the 3D positions and orientations of the objects. In general, this problem of inferring the 3D position and orientation from image measurements can be a difficult problem; however, for many cases that are faced by industrial robots we can obtain adequate solutions. For example, when

grasping parts from a conveyor belt, the depth,  $z$ , is fixed, and the perspective projection equations can be inverted if  $z$  is known.

We begin the section with a general discussion of moments, since moments will be used in the computation of both position and orientation of objects in the image.

### 11.5.1 Moments

*Moments* are functions defined on the image that can be used to summarize various aspects of the shape and size of objects in the image. The  $i, j$  moment for the  $k^{th}$  object, denoted by  $m_{ij}(k)$ , is defined by

$$m_{ij}(k) = \sum_{r,c} r^i c^j \mathcal{I}_k(r, c). \quad (11.52)$$

From this definition, it is evident that  $m_{00}$  is merely number of pixels in the object. The order of a moment is defined to be the sum  $i + j$ . The first order moments are of particular interest when computing the centroid of an object, and they are given by

$$m_{10}(k) = \sum_{r,c} r \mathcal{I}_k(r, c), \quad m_{01}(k) = \sum_{r,c} c \mathcal{I}_k(r, c). \quad (11.53)$$

It is often useful to compute moments with respect to the object center of mass. By doing so, we obtain characteristics that are invariant with respect to translation of the object. These moments are called *central moments*. The  $i, j$  central moment for the  $k^{th}$  object is defined by

$$C_{ij}(k) = \sum_{r,c} (r - \bar{r})^i (c - \bar{c})^j \mathcal{I}_k(r, c), \quad (11.54)$$

in which  $(\bar{r}, \bar{c})$  are the coordinates for the center of mass, or centroid, of the object.

### 11.5.2 The Centroid of an Object

It is convenient to define the position of an object to be the object's center of mass or centroid. By definition, the center of mass of an object is that point  $(\bar{r}, \bar{c})$  such that, if all of the object's mass were concentrated at  $(\bar{r}, \bar{c})$  the first moments would not change. Thus, we have

$$\sum_{r,c} \bar{r}_i \mathcal{I}_i(r, c) = \sum_{r,c} r \mathcal{I}_i(r, c) \Rightarrow \bar{r}_i = \frac{\sum_{r,c} r \mathcal{I}_i(r, c)}{\sum_{r,c} \mathcal{I}_i(r, c)} = \frac{m_{10}(i)}{m_{00}(i)} \quad (11.55)$$

$$\sum_{r,c} \bar{c}_i \mathcal{I}_i(r, c) = \sum_{r,c} c \mathcal{I}_i(r, c) \Rightarrow \bar{c}_i = \frac{\sum_{r,c} c \mathcal{I}_i(r, c)}{\sum_{r,c} \mathcal{I}_i(r, c)} = \frac{m_{01}(i)}{m_{00}(i)} \quad (11.56)$$

Figure 11.7 shows the centroids for the connected components of the image of Figure 11.3.



*Fig. 11.7* The segmented, component-labeled image of figure 11.3 showing the centroids and orientation of each component.

### 11.5.3 The Orientation of an Object

We will define the orientation of an object in the image to be the orientation of an axis that passes through the object such that the second moment of the object about that axis is minimal. This axis is merely the two-dimensional equivalent of the axis of least inertia.

For a given line in the image, the second moment of the object about that line is given by

$$\mathcal{L} = \sum_{r,c} d^2(r,c) \mathcal{I}(r,c) \quad (11.57)$$

in which  $d(r,c)$  is the minimum distance from the pixel with coordinates  $(r,c)$  to the line. Our task is to minimize  $\mathcal{L}$  with respect to all possible lines in the image plane. To do this, we will use the  $\rho, \theta$  parameterization of lines, and compute the partial derivatives of  $\mathcal{L}$  with respect to  $\rho$  and  $\theta$ . We find the minimum by setting these partial derivatives to zero.

With the  $\rho, \theta$  parameterization, a line consists of all those points  $x, y$  that satisfy

$$x \cos \theta + y \sin \theta - \rho = 0. \quad (11.58)$$

Thus,  $(\cos \theta, \sin \theta)$  gives the unit normal to the line, and  $\rho$  gives the perpendicular distance to the line from the origin. Under this parameterization, the distance from the line to the point with coordinates  $(r,c)$  is given by

$$d(r,c) = r \cos \theta + c \sin \theta - \rho. \quad (11.59)$$

Thus, our task is to find

$$\mathcal{L}^* = \min_{\rho, \theta} \sum_{r,c} (r \cos \theta + c \sin \theta - \rho)^2 \mathcal{I}(r,c) \quad (11.60)$$

We compute the partial derivative with respect to  $\rho$  as

$$\frac{d}{d\rho} \mathcal{L} = \frac{d}{d\rho} \sum_{r,c} (r \cos \theta + c \sin \theta - \rho)^2 \mathcal{I}(r, c) \quad (11.61)$$

$$= -2 \sum_{r,c} (r \cos \theta + c \sin \theta - \rho) \mathcal{I}(r, c) \quad (11.62)$$

$$= -2 \cos \theta \sum_{r,c} r \mathcal{I}(r, c) - 2 \sin \theta \sum_{r,c} c \mathcal{I}(r, c) + 2\rho \sum_{r,c} \mathcal{I}(r, c) \quad (11.63)$$

$$= -2(\cos \theta m_{10} + \sin \theta m_{01} - \rho m_{00}) \quad (11.64)$$

$$= -2(m_{00} \bar{r} \cos \theta + m_{00} \bar{c} \sin \theta - \rho m_{00}) \quad (11.65)$$

$$= -2m_{00}(\bar{r} \cos \theta + \bar{c} \sin \theta - \rho). \quad (11.66)$$

Now, setting this to zero we obtain

$$\bar{r} \cos \theta + \bar{c} \sin \theta - \rho = 0. \quad (11.67)$$

But this is just the equation of a line that passes through the point  $(\bar{r}, \bar{c})$ , and therefore we conclude that the inertia is minimized by a line that passes through the center of mass. We can use this knowledge to simplify the remaining computations. In particular, define the new coordinates  $(r', c')$  as

$$r' = r - \bar{r}, \quad c' = c - \bar{c}. \quad (11.68)$$

The line that minimizes  $\mathcal{L}$  passes through the point  $r' = 0, c' = 0$ , and therefore its equation can be written as

$$r' \cos \theta + c' \sin \theta = 0. \quad (11.69)$$

Before computing the partial derivative of  $\mathcal{L}$  (expressed in the new coordinate system) with respect to  $\theta$ , it is useful to perform some simplifications.

$$\mathcal{L} = \sum_{r,c} (r' \cos \theta + c' \sin \theta)^2 \mathcal{I}(r, c) \quad (11.70)$$

$$= \cos^2 \theta \sum_{r,c} (r')^2 \mathcal{I}(r, c) + 2 \cos \theta \sin \theta \sum_{r,c} (r' c') \mathcal{I}(r, c) + \sin^2 \theta \sum_{r,c} (c')^2 \mathcal{I}(r, c) \quad (11.71)$$

$$= C_{20} \cos^2 \theta + 2C_{11} \cos \theta \sin \theta + C_{02} \sin^2 \theta \quad (11.72)$$

in which the  $C_{ij}$  are the central moments given in (11.54). Note that the central moments depend on neither  $\rho$  nor  $\theta$ .

The final set of simplifications that we will make all rely on the double angle identities:

$$\cos^2 \theta = \frac{1}{2} + \frac{1}{2} \cos 2\theta \quad (11.73)$$

$$\sin^2 \theta = \frac{1}{2} - \frac{1}{2} \cos 2\theta \quad (11.74)$$

$$\cos \theta \sin \theta = \frac{1}{2} \sin 2\theta. \quad (11.75)$$

Substituting these into our expression for  $\mathcal{L}$  we obtain

$$\mathcal{L} = C_{20}\left(\frac{1}{2} + \frac{1}{2} \cos 2\theta\right) + 2C_{11}\left(\frac{1}{2} \sin 2\theta\right) + C_{02}\left(\frac{1}{2} - \frac{1}{2} \cos 2\theta\right) \quad (11.76)$$

$$= \frac{1}{2}(C_{20} + C_{02}) + \frac{1}{2}(C_{20} - C_{02}) \cos 2\theta + \frac{1}{2}C_{11} \sin 2\theta \quad (11.77)$$

It is now easy to compute the partial derivative with respect to  $\theta$ :

$$\frac{d}{d\theta}\mathcal{L} = \frac{d}{d\theta}\frac{1}{2}(C_{20} + C_{02}) + \frac{1}{2}(C_{20} - C_{02}) \cos 2\theta + \frac{1}{2}C_{11} \sin 2\theta \quad (11.78)$$

$$= -(C_{20} - C_{02}) \sin 2\theta + C_{11} \cos 2\theta, \quad (11.79)$$

and we setting this to zero we obtain

$$\tan 2\theta = \frac{C_{11}}{C_{20} - C_{02}}. \quad (11.80)$$

Figure 11.7 shows the orientations for the connected components of the image of Figure 11.3.

---

*Problems*

---

**TO BE WRITTEN**



# 12

---

## *VISION-BASED CONTROL*

In Chapter 9 we described how feedback from a force sensor can be used to control the forces and torques applied by the manipulator. In the case of force control, the quantities to be controlled (i.e., forces and torques) are measured directly by the sensor. Indeed, the output of a typical force sensor comprises six electric voltages that are proportional to the forces and torques experienced by the sensor. Force control is very similar to state-feedback control in this regard.

In this chapter, we consider the problem of vision-based control. Unlike force control, with vision-based control the quantities to be controlled cannot be measured directly from the sensor. For example, if the task is to grasp an object, the quantities to be controlled are pose variables, while the vision sensor, as we have seen in Chapter 11, provides a two-dimensional array of intensity values. There is, of course, a relationship between this array of intensity values and the geometry of the robot's workspace, but the task of inferring this geometry from an image is a difficult one that has been at the heart of computer vision research for many years. The problem faced in vision-based control is that of extracting a relevant and robust set of parameters from an image and using these to control the motion of the manipulator in real time.

Over the years, a variety of approaches have been taken to the problem of vision-based control. These vary based on how the image data is used, the relative configuration of camera and manipulator, choices of coordinate systems, etc. In this chapter, we focus primarily on one specific approach: image-based visual servo control for eye-in-hand camera configurations. We begin the chapter with a brief description of this approach, contrasting it with

other options. Following this, we develop the specific mathematical tools needed for this approach, both design and analysis.

## 12.1 APPROACHES TO VISION BASED-CONTROL

There are several distinct approaches to vision-based control. These vary based primarily on system configuration and how image data is used. In this section, we give a brief description of these considerations.

### 12.1.1 Where to put the camera

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 manipulator. These are often referred to as fixed camera vs. 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 off line. A 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.

With an eye-in-hand system, the camera is often attached to the manipulator above the wrist, i.e., 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 example, a camera attached to link three of an elbow manipulator (such as the one shown in Figure 3.1) will experience a significant change in field of view when joint 3 moves.

For either 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 for the two cases are similar, and in this text we will consider only the case of eye-in-hand systems.

### 12.1.2 How to use the image data

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 is 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 is used to build a partial 3D representation of the world. For example, if the task is to grasp an object, the perspective projection equations from Chapter 11 can be solved to determine the 3D coordinates of the grasp points relative to the camera coordinate frame. If these 3D coordinates can be obtained in real time, then they can be provided as set points to the robot controller, and control techniques described in Chapter 7 can be used. The main difficulties with position-based methods are related to the difficulty of building the 3D representation in real time. In particular, these methods tend to not be robust with respect to errors in calibration. Furthermore, with position-based methods, there is no direct control over the image itself. Therefore, a common problem with position-based methods is that camera motion can cause the object of interest to leave the camera field of view.

A second method known as image-based visual servo control directly uses the image data to control the robot motion. An error function is defined in terms of quantities that can be directly measured in an image (e.g., image coordinates of points, the orientation of lines in an image), and a control law is constructed that maps this error directly to robot motion. To date, the most common approach has been to use easily detected points on an object as feature points. The error function is then the vector difference between the desired and measured locations of these points in the image. 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.

Finally, these two approaches can be combined by using position-based methods to control certain degrees of freedom of the robot motion and using image-based methods to control the remaining degrees of freedom. Such methods essentially partition the set of degrees of freedom into disjoint sets, and are thus known as partitioned methods. We briefly describe a partitioned method in Section 12.6.

## 12.2 CAMERA MOTION AND INTERACTION MATRIX

As mentioned above, image-based methods map an image error function directly to robot motion without solving the 3D reconstruction problem. Recall the inverse velocity problem discussed in Chapter 4. Even though the inverse kinematics problem is difficult to solve and often ill posed (because the solution is not unique), 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 (e.g., even for the simple two-link planar arm the mapping is from  $\Re^2$  to the torus), the mapping of velocities is a linear map between linear subspaces (in the two-link example, a mapping from  $\Re^2$  to a plane that is tangent to the torus). Likewise, the relationship between vectors defined in terms of image features and camera velocities is a linear mapping between linear subspaces. We will now give a more rigorous explanation of this basic idea.

Let  $s(t)$  denote a vector of feature values that can be measured in an image. Its derivative,  $\dot{s}(t)$  is referred to as to as an **image feature velocity**. For example, if a single image point is used as a feature, we would have  $s(t) = (u(t), v(t))^T$ . 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  $\xi$  consist of linear velocity  $v$  and angular velocity  $\omega$ ,

$$\xi = \begin{bmatrix} v \\ \omega \end{bmatrix}, \quad (12.1)$$

i.e., the origin of the camera frame is moving with linear velocity  $v$ , and the camera frame is rotating about the axis  $\omega$  which is rooted at the origin of the camera frame as shown in Figure 12.1. The relationship between  $\dot{s}$  and  $\xi$  is given by

$$\dot{s} = L(s, q)\xi. \quad (12.2)$$

Here, the matrix  $L(s, q)$  is known as the **interaction matrix** or image Jacobian matrix. It was first introduced in [64], where it was referred to it as the *feature sensitivity matrix*. In [27] it was to as a Jacobian matrix (subsequently referred to in the literature as the image Jacobian), and in [24] it was given the name interaction matrix, the term that we will use. Note that the interaaction matrix is a function of both the configuration of the robot (as was also true for the manipulator Jacobian discribed in Chatper 4) and of the image feature values,  $s$ .

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.

### 12.2.1 Interaction matrix vs. Image Jacobian

The interaction matrix  $L$  is often referred to in the visual servo community by the name **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  $\xi$  is related to the variation in a set of parameters (either joint angles or image feature

**Figure showing the linear and angular velocity vectors for the camera frame.  
to be created**

*Fig. 12.1 Camera velocity*

velocities) by a linear transformation. Strictly speaking, the interaction matrix is not a Jacobian matrix, since  $\xi$  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 values).

### 12.3 THE INTERACTION MATRIX FOR POINTS

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 positioning a camera relative to some object that is to be manipulated. For example, a camera can be attached to a manipulator arm that is 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 12.3.4 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 at time  $t$  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  $s = (u, v)^T$  is the feature vector corresponding to the projection of this point in the image.

Our goal in this section is to derive the interaction matrix  $L$  that relates the velocity of the camera  $\xi$  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 relative

velocity of the point  $p$  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 manipulations we arrive to the interaction matrix that satisfies  $\dot{s} = L\xi$ .

### 12.3.1 Velocity of a fixed point relative to a moving camera

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  $p(t)$  the coordinates of  $p$  relative to the moving camera frame at time  $t$ , we have

$$p^0 = R(t)p(t) + O(t). \quad (12.3)$$

Thus, at time  $t$  we can solve for the coordinates of  $p$  relative to the camera frame by

$$p(t) = R^T(t)p^0 - R^T(t)O(t), \quad (12.4)$$

which is merely the time varying version of Equation (2.55). Now, to find the velocity of the point  $p$  relative to the moving camera frame, we merely differentiate this equation (as in Chapter 4). We will drop the explicit reference to time in these equations to simplify notation, but the reader is advised to bear in mind that both the rotation matrix  $R$  and the location of the origin of the camera frame  $O$  are time varying quantities. The derivative is computed as follows

$$\begin{aligned} \frac{d}{dt}p(t) &= \frac{d}{dt}[R^T p^0] - \frac{d}{dt}[R^T O] \\ &= \left[ \frac{d}{dt}R \right]^T p^0 - \left[ \frac{d}{dt}R \right]^T O - R^T \frac{d}{dt}O \\ &= \left[ \frac{d}{dt}R \right]^T (p^0 - O) - R^T \dot{O} \end{aligned} \quad (12.5)$$

In this equation, the quantity  $p^0 - O$  is merely the vector from the origin of the moving frame to the fixed point  $p$ , expressed in coordinates relative to the fixed frame, and thus  $R^T(p^0 - O) = p$  is the vector from the origin of the moving frame to the point  $p$  expressed relative to the moving frame. Using Equation (4.19), we can write the derivative of  $R$  as  $\dot{R} = S(\omega)R$ , which allows us to write Equation (12.5) as

$$\begin{aligned} &= [S(\omega)R]^T (p^0 - O) - R^T \dot{O} \\ &= R^T S^T(\omega) (p^0 - O) - R^T \dot{O} \\ &= R^T [S(-\omega) (p^0 - O)] - R^T \dot{O} \\ &= -R^T \omega \times R^T (p^0 - O) - R^T \dot{O} \end{aligned}$$

The vector  $\omega$  gives the angular velocity vector for the moving frame expressed relative to the fixed frame, i.e.,  $\omega = \omega^0$ . Therefore,  $R^T \omega = R_0^c \omega^0 = \omega^c$  gives

the angular velocity vector for the moving frame relative to the moving frame. Similarly, note that  $R^T\dot{\Omega} = \dot{\Omega}^c$ . Using these conventions, we can immediately write the equation for the velocity of  $p$  relative to the moving camera frame

$$\dot{p} = -\omega^c \times p - \dot{\Omega}^c \quad (12.6)$$

It is interesting to note that this velocity of a fixed point relative to a moving frame is merely  $-1$  times the velocity of a moving point (i.e., a point attached to a moving frame) relative to a fixed frame.

### Example 12.1 Camera motion in the plane

*TO BE WRITTEN: camera motion is in the plane parallel to the image plane (i.e., rotation about optic axis, translation parallel to camera x-y axes). Compute the velocity of a fixed point relative to the camera frame*

◊

#### 12.3.2 Constructing the Interaction Matrix

To simplify notation, we define the coordinates for  $p$  relative to the camera frame as  $p = (x, y, z)^T$ . By this convention, the velocity of  $p$  relative to the moving frame is merely the vector  $\dot{p} = (\dot{x}, \dot{y}, \dot{z})^T$ . We will denote the coordinates for the angular velocity vector by  $\omega^c = (\omega_x, \omega_y, \omega_z)^T = R^T\omega$ . To further simplify notation, we assign coordinates  $R^T\dot{\Omega} = (v_x, v_y, v_z)^T = \dot{\Omega}^c$ . Using these conventions, we can write Equation (12.6) as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = - \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

which can be written as the system of three equations

$$\dot{x} = y\omega_z - z\omega_y - v_x \quad (12.7)$$

$$\dot{y} = z\omega_x - x\omega_z - v_y \quad (12.8)$$

$$\dot{z} = x\omega_y - y\omega_x - v_z \quad (12.9)$$

Assuming that the imaging geometry can be modeled by perspective projection as given by Equation (11.5), we can express  $x$  and  $y$  in terms of image coordinates  $u, v$  of the projection of  $p$  in the image and the depth  $z$  as

$$x = \frac{uz}{\lambda}, \quad y = \frac{vz}{\lambda}$$

Substituting these into Equations (12.7)-(12.9) we obtain

$$\dot{x} = \frac{vz}{\lambda}\omega_z - z\omega_y - v_x \quad (12.10)$$

$$\dot{y} = z\omega_x - \frac{uz}{\lambda}\omega_z + -v_y \quad (12.11)$$

$$\dot{z} = \frac{uz}{\lambda}\omega_y - \frac{vz}{\lambda}\omega_x - v_z \quad (12.12)$$

These equations express the velocity  $\dot{p}$  in terms of the image coordinates  $u, v$  the depth of the point  $p$ , and the angular and linear velocity of the camera. Our goal is to derive an expression that relates the image velocity  $(\dot{u}, \dot{v})^T$  to the angular and linear velocity of the camera frame. Therefore, we will now find expressions for  $(\dot{u}, \dot{v})^T$  and then combine these with Equations (12.10)-(12.12).

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 (12.10) and (12.12) into this expression gives

$$\begin{aligned}\dot{u} &= \frac{\lambda}{z^2} \left( z \left[ \frac{vz}{\lambda} \omega_z - z\omega_y - v_x \right] - \frac{uz}{\lambda} \left[ \frac{uz}{\lambda} \omega_y - \frac{vz}{\lambda} \omega_x - v_z \right] \right) \\ &= -\frac{\lambda}{z} v_x + \frac{u}{z} v_z + \frac{uv}{\lambda} \omega_x - \frac{\lambda^2 + u^2}{\lambda} \omega_y + v \omega_z\end{aligned}\quad (12.13)$$

We can apply the same technique for  $\dot{v}$

$$\dot{v} = \frac{d}{dt} \frac{\lambda y}{z} = \lambda \frac{z\dot{y} - y\dot{z}}{z^2}$$

and substituting Equations (12.11) and (12.12) into this expression gives

$$\begin{aligned}\dot{v} &= \frac{\lambda}{z^2} \left( z \left[ -\frac{uz}{\lambda} \omega_z + z\omega_x - v_y \right] - \frac{vz}{\lambda} \left[ \frac{uz}{\lambda} \omega_y - \frac{vz}{\lambda} \omega_x - v_z \right] \right) \\ &= -\frac{\lambda}{z} v_y + \frac{v}{z} v_z + \frac{\lambda^2 + v^2}{\lambda} \omega_x - \frac{uv}{\lambda} \omega_y - u \omega_z\end{aligned}\quad (12.14)$$

Equations (12.13) and (12.14) can be combined and written in matrix form as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{\lambda} & -\frac{\lambda^2 + u^2}{\lambda} & v \\ 0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2 + v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (12.15)$$

The matrix in this equation is the interaction matrix for a point. It relates the velocity of the camera to the velocity of the image projection of the point. Note that this interaction matrix is a function of the image coordinates of the point,  $u$  and  $v$ , and of the depth of the point with respect to the camera frame,  $z$ . Therefore, this equation is typically written as

$$\dot{s} = L_p(u, v, z)\xi \quad (12.16)$$

### Example 12.2 Camera motion in the plane (cont.)

*TO BE WRITTEN: Give specific image coordinates to continue the previous example and compute the image motion as a function of camera velocity.*

◇

### 12.3.3 Properties of the Interaction Matrix for Points

Equation (12.16) can be decomposed and written as

$$\dot{s} = L_v(u, v, z)v + L_\omega(u, v)\omega \quad (12.17)$$

in which  $L_v(u, v, z)$  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 (i.e., 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  $L_v(u, v, z)$ , 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 methods that we discuss in Section 12.6.

The camera velocity  $\xi$  has six degrees-of-freedom, while only two values,  $u$  and  $v$ , are observed in the image. Thus, one would expect that not all camera motions case observable changes in the image. More precisely,  $L \in \mathbb{R}^{2 \times 6}$  and therefore has a null space of dimension 4, i.e., the system

$$0 = L(s, q)\xi$$

has solution vectors  $\xi$  that lie in a four-dimensional subspace of  $\mathbb{R}^6$ . For the case of a single point, it can be shown that the null space of the interaction matrix given in (12.15) is spanned by the four vectors

$$\begin{bmatrix} u \\ v \\ \lambda \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ u \\ v \\ \lambda \end{bmatrix} \begin{bmatrix} uvz \\ -(u^2 + \lambda^2)z \\ \lambda vz \\ -\lambda^2 \\ 0 \\ u\lambda \end{bmatrix} \begin{bmatrix} \lambda(u^2 + v^2 + \lambda^2)z \\ 0 \\ -u(u^2 + v^2 + \lambda^2)z \\ uv\lambda \\ -(u^2 + \lambda^2)z \\ u\lambda^2 \end{bmatrix}$$

The first two of these vectors have particularly intuitive interpretations. The first corresponds to motion of the camera frame along the projection ray that contains the point  $p$ , and the second corresponds to rotation of the camera frame about a projection ray that contains  $p$ .

### 12.3.4 The Interaction Matrix for Multiple Points

It is straightforward to generalize the development above the case in which several points are used to define the image feature vector. Consider the case for which the feature vector 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

$$s = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix} \quad \text{and} \quad z = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

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,

$$\dot{s} = L_c(s, z)\xi$$

This interaction matrix is thus obtained by stacking the  $n$  interaction matrices for the individual feature points,

$$\begin{aligned} L_c(s, z) &= \begin{bmatrix} L_1(u_1, v_1, z_1) \\ \vdots \\ L_n(u_n, v_n, z_n) \end{bmatrix} \\ &= \begin{bmatrix} -\frac{\lambda}{z_1} & 0 & \frac{u_1}{z_1} & \frac{u_1 v_1}{\lambda} & -\frac{\lambda^2 + u_1^2}{\lambda} & v_1 \\ 0 & -\frac{\lambda}{z_1} & \frac{v_1}{z_1} & \frac{\lambda^2 + v_1^2}{\lambda} & -\frac{u_1 v_1}{\lambda} & -u_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{\lambda}{z_n} & 0 & \frac{u_n}{z_n} & \frac{u_n v_n}{\lambda} & -\frac{\lambda^2 + u_n^2}{\lambda} & v_n \\ 0 & -\frac{\lambda}{z_n} & \frac{v_n}{z_n} & \frac{\lambda^2 + v_n^2}{\lambda} & -\frac{u_n v_n}{\lambda} & -u_n \end{bmatrix} \end{aligned}$$

Thus, we have  $L_c \in \mathbb{R}^{2n \times 6}$  and therefore three points are sufficient to solve for  $\xi$  given the image measurements  $\dot{s}$ .

## 12.4 IMAGE-BASED CONTROL LAWS

With image-based control, the goal configuration is defined by a desired configuration of image features, denoted by  $s^d$ . 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,  $u(t)$ . 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, i.e., 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.

The most common approach to image-based control is to compute a desired camera velocity and use this as the control  $u(t) = \xi$ . Relating image feature velocities to the camera velocity  $\xi$  is typically done by solving Equation (12.2). Solving this equation will give a desired camera velocity. In some cases, this can be done simply by inverting the interaction matrix, but in other cases the pseudoinverse must be used, as described below.

### 12.4.1 Computing Camera Motion

For the case of  $k$  feature values and  $m$  components of the camera body velocity  $\xi$ , 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.,  $\text{rank}(L) = \min(k, m)$ ), it can be used to compute  $\xi$  from  $\dot{s}$ . There are three cases that must be considered:  $k = m$ ,  $k > m$ , and  $k < m$ . We now discuss each of these.

When  $k = m$  and  $L$  is full rank,  $L$  is nonsingular, and  $L^{-1}$  exists. Therefore, in this case,  $\xi = L^{-1}\dot{s}$ .

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  $\xi$ , i.e., there are certain components of the camera motion that can not be observed. In this case we can compute a solution given by

$$\xi = L^+ \dot{s} + (I - L^+ L)b$$

where  $L^+$  is the pseudoinverse for  $L$  given by

$$L^+ = L^T (L L^T)^{-1}$$

and  $b \in \mathbb{R}^k$  is an arbitrary vector. Note the similarity between this equation and Equation (4.128) which gives the solution for the inverse velocity problem (i.e., solving for joint velocities to achieve a desired end-effector velocity) for redundant manipulators.

In general, for  $k < m$ ,  $(I - LL^+) \neq 0$ , and 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  $\xi$  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  $\dot{s}$  is 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  $\xi$ . 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

$$\xi = L^+ \dot{s} \quad (12.18)$$

in which the pseudoinverse is given by

$$L^+ = (L^T L)^{-1} L^T \quad (12.19)$$

#### 12.4.2 Proportional Control Schemes

Many modern robots are equipped with controllers that accept as input a command velocity  $\xi$  for the end effector. Thus, we define our control input as  $u(t) = \xi$ . Using the results above, we can define a proportional control law as

$$u(t) = -KL^+e(t). \quad (12.20)$$

in which  $K$  is an  $m \times m$  diagonal, positive definite gain matrix.

The derivative of the error function is given by

$$\dot{e}(t) = \frac{d}{dt}(s(t) - s^d) = \dot{s}(t) = L\xi$$

and substituting Equation (12.20) for  $\xi$  we obtain

$$\dot{e}(t) = -KLL^+e(t) \quad (12.21)$$

If  $k = m$  and  $L$  has full rank, then  $L^+ = L^{-1}$ , and we have

$$\dot{e}(t) = -Ke(t)$$

From linear system theory, we know that this system is stable when the eigen values of  $K$  are positive, which motivates the selection of  $K$  as a positive definite matrix.

When  $k > m$ , as is typically the case for visual servo systems, a sufficient condition for stability is that the matrix product  $KLL^+$  is positive definite. This is easily demonstrated using the Lyapunov function

$$V(t) = \frac{1}{2}\|e(t)\|^2 = \frac{1}{2}e^T e$$

Using Equation (12.21) the derivative  $\dot{V}$  is given by

$$\begin{aligned}\dot{V} &= \frac{d}{dt} \frac{1}{2} e^T e \\ &= e^T \dot{e} \\ &= -e^T (KLL^+) e\end{aligned}$$

and we see that  $\dot{V} < 0$  when  $KLL^+$  is positive definite.

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  $\hat{L}^+$  and we can use the control  $u(t) = -K\hat{L}^+e(t)$ . It is easy to show, by a proof analogous to the one above, that the resulting visual servo system will be stable when  $KL\hat{L}^+$  is positive definite. This helps to explain the robustness of image-based control methods to calibration errors in the computer vision system.

## 12.5 THE RELATIONSHIP BETWEEN END EFFECTOR AND CAMERA MOTIONS

The output of a visual servo controller is a camera velocity  $\xi_c$ , 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 4.10. In most applications, the camera frame is not coincident with the end effector frame, but is rigidly attached to it. In this case, the two frames are related by the constant homogeneous transformation

$$T_c^6 = \begin{bmatrix} R_c^6 & d_c^6 \\ 0 & 1 \end{bmatrix} \quad (12.22)$$

Our goal is to find the relationship between the body velocity of the camera frame  $\xi_c = (v_c, \omega_c)^T$  and the body velocity of the end effector frame  $\xi_6 = (v_6, \omega_6)^T$ . Furthermore, we will assume that the camera velocity is given with respect to the camera frame (i.e., we are given the coordinates  $\xi_c^c$ ), and that we wish to determine the end effector velocity relative to the end effector frame (i.e., we wish to find the coordinates  $\xi_6^6$ ). Once we obtain  $\xi_6^6$ , it is a simple matter to express  $\xi = (v_6, \omega_6)^T$  with respect to the base frame, as we will see below.

Since the two frames are rigidly attached, the angular velocity of the end effector frame is the same as the angular velocity for the camera frame. An easy way to show this is by computing the angular velocities of each frame by taking the derivatives of the appropriate rotation matrices (such as was done

in Chapter 4). The derivation is as follows,

$$\begin{aligned} R_c^0 &= R_6^0 R_c^6 \\ \frac{d}{dt} R_c^0 &= \frac{d}{dt} R_6^0 R_c^6 \\ \dot{R}_c^0 &= \dot{R}_6^0 R_c^6 \\ S(\omega_c^0) R_c^0 &= S(\omega_6^0) R_6^0 R_c^6 \\ S(\omega_c^0) &= S(\omega_6^0) \end{aligned}$$

Thus, we have  $\omega_c^0 = \omega_6^0$ , and it is clear that the angular velocity of the end effector is identical to the angular velocity of the camera frame,  $\omega_6 = \omega_c$ . If the coordinates of this angular velocity are given with respect to the camera frame and we wish to express the angular velocity with respect to the end effector frame, we merely use the rotational coordinate transformation

$$\omega_6^6 = R_c^6 \omega_c^c \quad (12.23)$$

If the camera is moving with body velocity  $\xi = (v_c, \omega_c)^T$ , then the linear velocity of the origin of the end effector frame (which is rigidly attached to the camera frame) is given by  $v_c + \omega_c \times r$ , with  $r$  the vector from the origin of the camera frame to the origin of the end effector frame. From Equation (12.22),  $d_c^6$  gives the coordinates of the origin of the camera frame with respect to the end effector frame, and therefore we can express  $r$  in coordinates relative to the camera frame as  $r^c = -R_6^c d_c^6$ . Thus, we write  $\omega_c \times r$  in the coordinates with respect to the camera frame as

$$\omega_c^c \times (-R_6^c d_c^6) = R_6^c d_c^6 \times \omega_c^c$$

Now to express this free vector with respect to the end effector frame, we merely apply a rotation transformation,

$$\begin{aligned} R_c^6 (R_6^c d_c^6 \times \omega_c^c) &= d_c^6 \times R_c^6 \omega_c^c \\ &= S(d_c^6) R_c^6 \omega_c^c \end{aligned} \quad (12.24)$$

Expressing  $v_c$  relative to the end effector frame is also accomplished by a simple rotational transformation,

$$v_c^6 = R_c^6 v_c^c \quad (12.25)$$

Combining Equations (12.23), (12.24), and (12.25) into a single matrix equation, we obtain

$$\xi_6^6 = \begin{bmatrix} R_c^6 & S(d_c^6) R_c^6 \\ 0_{3 \times 3} & R_c^6 \end{bmatrix} \xi_c^c$$

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  $\omega_6$ , 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 12.3 Eye-in-hand system with SCARA arm

*TO BE WRITTEN:* This example will show the required motion of the SCARA arm to cause a pure rotation about the optic axis of the camera, for the camera attached to the end effector, optic axis of the camera parallel to world z-axis.

◊

## 12.6 PARTITIONED APPROACHES

### TO BE REVISED

Although image-based methods are versatile and robust to calibration and sensing errors, they sometimes fail when the required camera motion is large. Consider, for example, the case when the required camera motion is a large rotation about the optic axis. If point features are used, a pure rotation of the camera about the optic axis would cause each feature point to trace a trajectory in the image that lies on a circle. Image-based methods, in contrast, would cause each feature point to move in a straight line from its current image position to its desired position. The induced camera motion would be a retreat along the optic axis, and for a required rotation of  $\pi$ , the camera would retreat to  $z = -\infty$ , at which point  $\det L = 0$ , and the controller would fail. 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.

To combat this problem, a number of **partitioned methods** have been introduced. These methods use the interaction matrix to control only a subset of the camera degrees of freedom, using other methods to control the remaining degrees of freedom. Consider Equation (12.15). We can write this equation as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{uv}{\lambda} & -\frac{\lambda^2 + u^2}{\lambda} \\ 0 & -\frac{\lambda}{z} & \frac{\lambda^2 + v^2}{\lambda} & -\frac{uv}{\lambda} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_x \\ \omega_y \end{bmatrix} + \begin{bmatrix} \frac{u}{z} & v \\ \frac{v}{z} & -u \end{bmatrix} \begin{bmatrix} v_z \\ \omega_z \end{bmatrix}$$

If we generalize this to  $k$  feature points, we stack the resulting interaction matrices as in Section 12.3.4, and the resulting relationship is given by

$$\dot{s} = L_{xy}\xi_{xy} + L_z\xi_z \quad (12.26)$$

$$= \dot{s}_{xy} + \dot{s}_z \quad (12.27)$$

In Equation (12.27),  $\dot{s}_z = L_z\xi_z$  gives the component of  $\dot{s}$  due to the camera motion along and rotation about the optic 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.

## TO APPEAR

*Fig. 12.2* Feature used to determine  $\omega_z$ .

Equation (12.26) allows us to partition the control  $u$  into two components,  $u_{xy} = \xi_{xy}$  and  $u_z = \xi_z$ . Suppose that we have established a control scheme to determine the value  $\xi_z = u_z$ . Using an image-based method to find  $u_{xy}$ , we would solve Equation (12.26) for  $\xi_{xy}$ ,

$$\xi_{xy} = L_{xy}^+ \{ \dot{s} - L_z \xi_z \} \quad (12.28)$$

$$= L_{xy}^+ \{ \dot{s} - \dot{s}_z \} \quad (12.29)$$

This equation has an intuitive explanation.  $-L_{xy}^+ L_z \xi_z$  is the required value of  $\xi_{xy}$  to cancel the feature motion  $\dot{s}_z$ . The control  $u_{xy} = \xi_{xy} = L_{xy}^+ \dot{s}$  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  $\xi_z$  has been accounted for.

In [15],  $\xi_z$ , is computed using two image features that are simple and computationally inexpensive to compute. The image feature used to determine  $\omega_z$  is  $\theta_{ij}$ , with  $0 \leq \theta_{ij} < 2\pi$  the angle between the horizontal axis of the image plane and the directed line segment joining feature points two feature point. This is illustrated in Figure 12.2. For numerical conditioning it is advantageous to select the longest line segment that can be constructed from the feature points, and allowing that this may change during the motion as the feature point configuration changes. The value for  $\omega_z$  is given by

$$\omega_z = \gamma_{\omega_z} (\theta_{ij}^d - \theta_{ij})$$

in which  $\theta_{ij}^d$  is the desired value, and  $\gamma_{\omega_z}$  is a scalar gain coefficient. This form allows explicit control over the direction of rotation, which may be important to avoid mechanical motion limits. For example if a hard stop exists at  $\theta_s$  then

$$\omega_z = \gamma_{\omega_z} \operatorname{sgn}(\theta_{ij}^d - \theta_s) \operatorname{sgn}(\theta_{ij} - \theta_s) [\theta_{ij}^d - \theta_{ij}]$$

will avoid motion through that stop.

The image feature used to determine  $v_z$  is the square root of the area of the regular polygon enclosed by the feature points. If we denote the area of the polygon by  $\sigma^2$ , we determine  $v_z$  as

$$v_z = \gamma_{v_z} \ln\left(\frac{\sigma^d}{\sigma}\right) \quad (12.30)$$

## TO APPEAR

*Fig. 12.3* Feature used to determine  $v_z$ .

## TO APPEAR

*Fig. 12.4* Proposed partitioned IBVS for pure target rotation ( $\pi$  rad). (a) Image-plane feature motion (initial location is  $\circ$ , desired location is  $\bullet$ ), (b) Feature error trajectory, (c) Cartesian translation trajectory.

The advantages of this approach are that (1) it is a scalar; (2) it is rotation invariant thus decoupling camera rotation from Z-axis translation. (3) it can be cheaply computed.

Figure 12.4 shows the performance of the proposed partitioned controller for the case of desired rotation by  $\pi$  about the optic axis. The important features are that the camera does not retreat since  $\sigma$  is constant at  $\sigma = 0$ . The rotation  $\theta$  monotonically decreases and the feature points move in a circle. The feature coordinate error is initially increasing, unlike the classical IBVS case in which feature error is monotonically decreasing.

An example that involves more complex translational and rotational motion is shown in Figure 12.5. The new features decrease monotonically, but the error in  $s$  does not decrease monotonically and the points follow complex curves on the image plane. Figure 12.6 compares the Cartesian camera motion for the two IBVS methods. The proposed partitioned method has eliminated the camera retreat and also exhibits better behavior for the X- and Y-axis motion. However the consequence is much more complex image plane feature motion that admits the possibility of the points leaving the field of view.

Other partitioned methods have been proposed in [48, 19, 53], but these rely on advanced concepts from projective geometry, and are beyond the scope of this text.

## TO APPEAR

*Fig. 12.5* Proposed partitioned IBVS for general target motion. (a) Image-plane feature motion (dashed line shows straight line motion for classical IBVS), (b) Feature error trajectory.

## TO APPEAR

*Fig. 12.6* Comparison of Cartesian camera motion for classic and new partitioned IBVS for general target motion.

### 12.7 MOTION PERCEPTIBILITY

Recall the that notion of manipulability described in Section 4.11 gave a quantitative measure of the scaling from joint velocities to end-effector velocities. **Motion perceptibility** [69, 68] is an analogous concept that relates camera velocity to the velocity of features in the image. The notion of **resolvability** introduced in [55, 56] is similar. Intuitively, motion perceptibility quantifies the magnitude of changes to image features that result from motion of the camera.

Consider the set of all robot tool velocities  $\xi$  such that

$$\|\xi\| = (\xi_1^2 + \xi_2^2 + \dots \xi_m^2)^{1/2} \leq 1. \quad (12.31)$$

As above, there are three cases to consider. Suppose that  $k > m$  (i.e., there are redundant image features). We may use Equation (12.18) to obtain

$$\begin{aligned} \|\xi\| &= \xi^T \cdot \xi \\ &= (L^+ \dot{s})^T (L^+ \dot{s}) \\ &= \dot{s}^T (L^{+T} L^+) \dot{s} \leq 1 \end{aligned} \quad (12.32)$$

Now, consider the singular value decomposition of  $L$ , given by

$$L = U \Sigma V^T. \quad (12.33)$$

in which

$$U = [u_1 u_2 \dots u_k], \quad V = [v_1 v_2 \dots v_m] \quad (12.34)$$

are orthogonal matrices, and  $\Sigma \in \Re^{k \times m}$  with

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \\ & 0 & & \end{bmatrix} \quad (12.35)$$

and the  $\sigma_i$  are the singular values of  $L$ , and  $\sigma_1 \geq \sigma_2 \dots \geq \sigma_m$ .

For this case, the pseudoinverse of the image Jacobian,  $L^+$ , is given by Equation (12.19). Using this with Equations (12.32) and (12.33) we obtain

$$\dot{s}^T U \begin{bmatrix} \sigma_1^{-2} & & & \\ & \sigma_2^{-2} & & \\ & & \ddots & \\ & & & \sigma_m^{-2} \\ & 0 & & \end{bmatrix} U^T \dot{s} \leq 1 \quad (12.36)$$

Consider the orthogonal transformation of  $\dot{s}$  given by

$$\tilde{\dot{s}} = U^T \dot{s} \quad (12.37)$$

Substituting this into Equation (12.36) we obtain

$$\sum_{i=1}^m \frac{1}{\sigma_i^2} \tilde{\dot{s}}_i \leq 1 \quad (12.38)$$

Equation (12.38) 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 (12.38) as a quantitative measure of the perceptibility of motion. The volume of the motion perceptibility ellipsoid is given by

$$K \sqrt{\det(L^T L)}, \quad (12.39)$$

in which  $K$  is a scaling constant that depends on the dimension of the ellipsoid,  $m$ . 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 be  $w_v$ , as

$$w_v = \sqrt{\det(L^T L)} = \sigma_1 \sigma_2 \dots \sigma_m. \quad (12.40)$$

The motion perceptibility measure,  $w_v$ , has the following properties, which are direct analogs of properties derived by Yoshikawa for manipulability [78].

- In general,  $w_v = 0$  holds if and only if  $\text{rank}(L) < \min(k, m)$ , (i.e., 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

$$(\sigma_1)^{-1} \leq \frac{\|\Delta\xi\|}{\|\Delta\dot{s}\|} \leq (\sigma_m)^{-1}. \quad (12.41)$$

There are other quantitative methods that could be used to evaluate the perceptibility of motion. For example, in the context of feature selection, Feddema [26] has used the condition number for the image Jacobian, given by  $\|L\|\|L^{-1}\|$ .

## 12.8 CHAPTER SUMMARY

### TO APPEAR

---

*Problems*

---

**TO APPEAR**



# *Appendix A*

## *Geometry and*

## *Trigonometry*

### A.1 TRIGONOMETRY

#### A.1.1 Atan2

The function  $\theta = A \tan(x, y)$  computes the arc tangent function, where  $x$  and  $y$  are the cosine and sine, respectively, of the angle  $\theta$ . This function uses the signs of  $x$  and  $y$  to select the appropriate quadrant for the angle  $\theta$ . Specifically,  $A \tan(x, y)$  is defined for all  $(x, y) \neq (0, 0)$  and equals the unique angle  $\theta$  such that

$$\cos \theta = \frac{x}{(x^2 + y^2)^{\frac{1}{2}}}, \quad \sin \theta = \frac{y}{(x^2 + y^2)^{\frac{1}{2}}}. \quad (\text{A.1})$$

For example,  $A \tan(1, -1) = -\frac{\pi}{4}$ , while  $A \tan(-1, 1) = +\frac{3\pi}{4}$ . Note that if both  $x$  and  $y$  are zero,  $A \tan$  is undefined.

**A.1.2 Reduction formulas**

$$\begin{aligned}\sin(-\theta) &= -\sin \theta & \sin\left(\frac{\pi}{2} + \theta\right) &= \cos \theta \\ \cos(-\theta) &= \cos \theta & \tan\left(\frac{\pi}{2} + \theta\right) &= -\cot \theta \\ \tan(-\theta) &= -\tan \theta & \tan(\theta - \pi) &= \tan \theta\end{aligned}$$

**A.1.3 Double angle identities**

$$\begin{aligned}\sin(x \pm y) &= \sin x \cos y \pm \cos x \sin y \\ \cos(x \pm y) &= \cos x \cos y \mp \sin x \sin y \\ \tan(x \pm y) &= \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y}\end{aligned}$$

**A.1.4 Law of cosines**

If a triangle has sides of length  $a$ ,  $b$  and  $c$ , and  $\theta$  is the angle opposite the side of length  $a$ , then

$$a^2 = b^2 + c^2 - 2bc \cos \theta \quad (\text{A.2})$$

# Appendix B

## Linear Algebra

In this book we assume that the reader has some familiarity with basic properties of vectors and matrices, such as matrix addition, subtraction, multiplication, matrix transpose, and determinants. These concepts will not be defined here. For additional background see [4].

The symbol  $\mathbb{R}$  will denote the set of real numbers, and  $\mathbb{R}^n$  will denote the usual vector space on  $n$ -tuples over  $\mathbb{R}$ . 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  $A, B, C, R$ , etc., denote matrices. Unless otherwise stated, vectors will be defined as column vectors. Thus, the statement  $x \in \mathbb{R}^n$  means that

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, x_i \in \mathbb{R}. \quad (\text{B.1})$$

The vector  $x$  is thus an  $n$ -tuple, arranged in a column with components  $x_1, \dots, x_n$ . We will frequently denote this as

$$x = [x_1, \dots, x_n]^T \quad (\text{B.2})$$

where the superscript  $T$  denotes transpose. The length or **norm** of a vector  $x \in \mathbb{R}^n$  is

$$\|x\| = (x_1^2 + \dots + x_n^2)^{\frac{1}{2}}. \quad (\text{B.3})$$

The **scalar product**, denoted  $\langle x, y \rangle$ , or  $x^T y$  of two vectors  $x$  and  $y$  belonging to  $\mathbb{R}^n$  is a real number defined by

$$\langle x, y \rangle = x^T y = x_1 y_1 + \cdots + x_n y_n. \quad (\text{B.4})$$

Thus,

$$\|x\| = \langle x, x \rangle^{\frac{1}{2}} \quad (\text{B.5})$$

The scalar product of vectors is commutative, that is,

$$\langle x, y \rangle = \langle y, x \rangle. \quad (\text{B.6})$$

We also have the useful inequalities,

$$|\langle x, y \rangle| \leq \|x\| \|y\| \quad (\text{Cauchy-Schwartz}) \quad (\text{B.7})$$

$$\|x + y\| \leq \|x\| + \|y\| \quad (\text{Triangle Inequality}) \quad (\text{B.8})$$

For vectors in  $\mathbb{R}^3$  the scalar product can be expressed as

$$|\langle x, y \rangle| = \|x\| \|y\| \cos(\theta) \quad (\text{B.9})$$

where  $\theta$  is the angle between the vectors  $x$  and  $y$ .

The **outer product** of two vectors  $x$  and  $y$  belonging to  $\mathbb{R}^n$  is an  $n \times n$  matrix defined by

$$xy^T = \begin{bmatrix} x_1 y_1 & \cdot & \cdot & x_1 y_n \\ x_2 y_1 & \cdot & \cdot & x_2 y_n \\ \cdot & \cdot & \cdot & \cdot \\ x_n y_1 & \cdot & \cdot & x_n y_n \end{bmatrix}. \quad (\text{B.10})$$

From (B.10) we can see that the scalar product and the outer product are related by

$$\langle x, y \rangle = x^T y = \text{Tr}(xy^T) \quad (\text{B.11})$$

where the function  $\text{Tr}(\cdot)$  denotes the trace of a matrix, that is, the sum of the diagonal elements of the matrix.

We will use  $i, j$  and  $k$  to denote the standard unit vectors in  $\mathbb{R}^3$

$$i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad j = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \quad (\text{B.12})$$

Using this notation a vector  $x = [x_1, x_2, x_3]^T$  may be written as

$$x = x_1 i + x_2 j + x_3 k. \quad (\text{B.13})$$

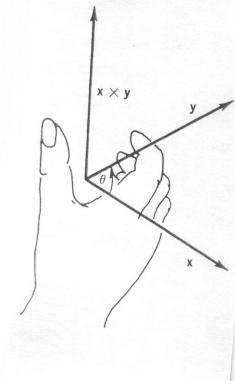


Fig. B.1 The right hand rule.

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

$$c = x \times y = \det \begin{bmatrix} i & j & k \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \quad (\text{B.14})$$

$$= (x_2y_3 - x_3y_2)i + (x_3y_1 - x_1y_3)j + (x_1y_2 - x_2y_1)k. \quad (\text{B.15})$$

The cross product is a vector whose magnitude is

$$\|c\| = \|x\| \|y\| \sin(\theta) \quad (\text{B.16})$$

where  $\theta$  is the angle between  $x$  and  $y$  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 as shown in Figure B.2. We can remember the right hand rule as being the direction of advancement of a right-handed screw rotated from the positive  $x$  axis is rotated into the positive  $y$  axis through the smallest angle between the axes. The cross product has the properties

$$x \times y = -y \times x$$

$$x \times (y + z) = x \times y + x \times z \quad (\text{B.17})$$

$$\alpha(x \times y) = (\alpha x) \times y = x \times (\alpha y) \quad (\text{B.18})$$

## B.1 DIFFERENTIATION OF VECTORS

Suppose that the vector  $x(t) = (x_1(t), \dots, x_n(t))^T$  is a function of time. Then the time derivative  $\dot{x}$  of  $x$  is just the vector

$$\dot{x} = (\dot{x}_1(t), \dots, \dot{x}_n(t))^T \quad (\text{B.19})$$

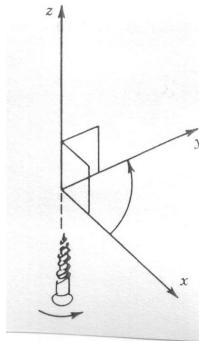


Fig. B.2 The right-handed coordinate frame.

that is, the vector can be differentiated coordinate wise. Likewise, the derivative  $dA/dt$  of a matrix  $A = (a_{ij})$  is just the matrix  $(\dot{a}_{ij})$ . 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.

$$\frac{d}{dt} \langle x, y \rangle = \langle \frac{dx}{dt}, y \rangle + \langle x, \frac{dy}{dt} \rangle \quad (\text{B.20})$$

$$\frac{d}{dt} (x \times y) = \frac{dx}{dt} \times y + x \times \frac{dy}{dt}. \quad (\text{B.21})$$

## B.2 LINEAR INDEPENDENCE

A set of vectors  $\{x_1, \dots, x_n\}$  is said to **linearly independent** if and only if

$$\sum_{i=1}^n \alpha_i x_i = 0 \quad (\text{B.22})$$

implies

$$x_j = 0 \text{ for all } i. \quad (\text{B.23})$$

The **rank** of a matrix  $A$  is the largest number of linearly independent rows (or columns) of  $A$ . Thus the rank of an  $n \times m$  matrix can be no greater than the minimum of  $n$  and  $m$ .

### B.3 CHANGE OF COORDINATES

A matrix can be thought of as representing a linear transformation from  $\mathbb{R}^n$  to  $\mathbb{R}^n$  in the sense that  $A$  takes a vector  $x$  to a new vector  $y$  according to

$$y = Ax \quad (\text{B.24})$$

$y$  is called the **image** of  $x$  under the transformation  $A$ . If the vectors  $x$  and  $y$  are represented in terms of the standard unit vectors  $i$ ,  $j$ , and  $k$ , then the columns of  $A$  are themselves vectors which represent the images of the basis vectors  $i$ ,  $j$ ,  $k$ . Often it is desired to represent vectors with respect to a second coordinate frame with basis vectors  $e$ ,  $f$ , and  $g$ . In this case the matrix representing the same linear transformation as  $A$ , but relative to this new basis, is given by

$$A' = T^{-1}AT \quad (\text{B.25})$$

where  $T$  is a non-singular matrix with column vectors  $e$ ,  $f$ ,  $g$ . The transformation  $T^{-1}AT$  is called a **similarity transformation** of the matrix  $A$ .

### B.4 EIGENVALUES AND EIGENVECTORS

The **eigenvalues** of a matrix  $A$  are the solutions in  $s$  of the equation

$$\det(sI - A) = O. \quad (\text{B.26})$$

The function,  $\det(sI - A)$  is a polynomial in  $s$  called the **characteristic polynomial** of  $A$ . If  $s_e$  is an eigenvalue of  $A$ , an eigenvector of  $A$  corresponding to  $s_e$  is a nonzero vector  $x$  satisfying the system of linear equations

$$(s_e I - A) = 0. \quad (\text{B.27})$$

or, equivalently,

$$Ax = s_e x. \quad (\text{B.28})$$

If the eigenvalues  $s_1, \dots, s_n$  of  $A$  are distinct, then there exists a similarity transformation  $A' = T^{-1}AT$ , such that  $A'$  is a diagonal matrix with the eigenvalues  $s_1, \dots, s_n$  on the main diagonal, that is,

$$A' = \text{diag}[s_1, \dots, s_n]. \quad (\text{B.29})$$

### B.5 SINGULAR VALUE DECOMPOSITION (SVD)

For a 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) of a matrix, which we now introduce.

As we described above, for  $J \in \mathbb{R}^{m \times n}$ , we have  $JJ^T \in \mathbb{R}^{m \times m}$ . This square matrix has eigenvalues and eigenvectors that satisfy

$$JJ^T u_i = \lambda_i u_i \quad (\text{B.30})$$

in which  $\lambda_i$  and  $u_i$  are corresponding eigenvalue and eigenvector pairs for  $JJ^T$ . We can rewrite this equation to obtain

$$\begin{aligned} JJ^T u_i - \lambda_i u_i &= 0 \\ (JJ^T - \lambda_i I) u_i &= 0. \end{aligned} \quad (\text{B.31})$$

The latter equation implies that the matrix  $(JJ^T - \lambda_i I)$  is singular, and we can express this in terms of its determinant as

$$\det(JJ^T - \lambda_i I) = 0. \quad (\text{B.32})$$

We can use Equation (B.32) to find the eigenvalues  $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_m \geq 0$  for  $JJ^T$ . The *singular values* for the Jacobian matrix  $J$  are given by the square roots of the eigenvalues of  $JJ^T$ ,

$$\sigma_i = \sqrt{\lambda_i}. \quad (\text{B.33})$$

The singular value decomposition of the matrix  $J$  is then given by

$$J = U\Sigma V^T, \quad (\text{B.34})$$

in which

$$U = [u_1 u_2 \dots u_m], \quad V = [v_1 v_2 \dots v_n] \quad (\text{B.35})$$

are orthogonal matrices, and  $\Sigma \in \mathbb{R}^{m \times n}$ .

$$\Sigma = \left[ \begin{array}{cccc|c} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_m & & \\ \end{array} \right]. \quad (\text{B.36})$$

We can compute the SVD of  $J$  as follows. We begin by finding the singular values,  $\sigma_i$ , of  $J$  using Equations (B.32) and (B.33). These singular values can then be used to find the eigenvectors  $u_1, \dots, u_m$  that satisfy

$$JJ^T u_i = \sigma_i^2 u_i. \quad (\text{B.37})$$

These eigenvectors comprise the matrix  $U = [u_1 u_2 \dots u_m]$ . The system of equations (B.37) can be written as

$$JJ^T U = U \Sigma_m^2 \quad (\text{B.38})$$

if we define the matrix  $\Sigma_m$  as

$$\Sigma_m = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \end{bmatrix}.$$

Now, define

$$V_m = J^T U \Sigma_m^{-1} \quad (\text{B.39})$$

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.34):

$$U \Sigma V^T = U [\Sigma_m \mid 0] \begin{bmatrix} V_m^T \\ V_{n-m}^T \end{bmatrix} \quad (\text{B.40})$$

$$= U \Sigma_m V_m^T \quad (\text{B.41})$$

$$= U \Sigma_m (J^T U \Sigma_m^{-1})^T \quad (\text{B.42})$$

$$= U \Sigma_m (\Sigma_m^{-1})^T U^T J \quad (\text{B.43})$$

$$= U \Sigma_m \Sigma_m^{-1} U^T J \quad (\text{B.44})$$

$$= U U^T J \quad (\text{B.45})$$

$$= J. \quad (\text{B.46})$$

Here, Equation (B.40) follows immediately from our construction of the matrices  $U$ ,  $V$  and  $\Sigma_m$ . Equation (B.42) is obtained by substituting Equation (B.39) into Equation (B.41). Equation (B.44) follows because  $\Sigma_m^{-1}$  is a diagonal matrix, and thus symmetric. Finally, Equation (B.46) is obtained using the fact that  $U^T = U^{-1}$ , since  $U$  is orthogonal.



# Appendix C

## Lyapunov Stability

We give here some basic definitions of stability and Lyapunov functions and present a sufficient condition for showing stability of a class of nonlinear systems. For simplicity we treat only time-invariant systems. For a more general treatment of the subject the reader is referred to [?].

**Definition C.1** Consider a nonlinear system on  $\mathbb{R}^n$

$$\dot{x} = f(x) \quad (\text{C.1})$$

where  $f(x)$  is a vector field on  $\mathbb{R}^n$  and suppose that  $f(0) = 0$ . Then the origin in  $\mathbb{R}^n$  is said to be an **equilibrium point** for (C.1).

If initially the system (C.1) satisfies  $x(t_0) = 0$  then the function  $x(t_0) \equiv 0$  for  $t > t_0$  can be seen to be a solution of (C.1) called the **null** or **equilibrium** solution. In other words, if the system represented by (C.1) starts initially at the equilibrium, then it remains at the equilibrium thereafter. The question of stability deals with the solutions of (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 in some sense. We can formalize this notion into the following.

**Definition C.2** *The null solution  $x(t) = 0$  is **stable** if and only if, for any  $\epsilon > 0$  there exist  $\delta(\epsilon) > 0$  such that*

$$\|x(t_0)\| < \delta \text{ implies } \|x(t)\| < \epsilon \text{ for all } t > t_0. \quad (\text{C.2})$$

This situation is illustrated by Figure C.1 and says that the system is stable

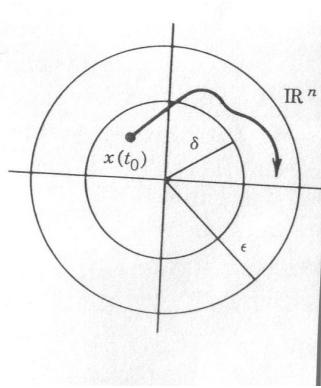


Fig. C.1 Illustrating the definition of stability.

if the solution remains within a ball of radius  $\epsilon$  around the equilibrium, so long as the initial condition lies in a ball of radius  $\delta$  around the equilibrium. Notice that the required  $\delta$  will depend on the given  $\epsilon$ . To put it another way, a system is stable if “small” perturbations in the initial conditions, results in “small” perturbations from the null solution.

**Definition C.3** *The null solution  $x(t) = 0$  is **asymptotically stable** if and only if there exists  $\delta > 0$  such that*

$$\|x(t_0)\| < \delta \text{ implies } \|x(t)\| \rightarrow 0 \text{ as } t \rightarrow \infty. \quad (\text{C.3})$$

In other words, asymptotic stability means that if the system is perturbed away from the equilibrium it will return asymptotically to the equilibrium. 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 arbitrary initial conditions.

We know that a linear system

$$\dot{x} = Ax \quad (\text{C.4})$$

will be globally asymptotically stable provided that all eigenvalues of the matrix  $A$  lie in the open left half of the complex plane. For nonlinear systems stability cannot be so easily determined.

Another important notion related to stability is the notion of **uniform ultimate boundedness** of solutions.

**Definition C.4** A solution  $x(t) : [t_0, \infty] \rightarrow \mathbb{R}^n$  (C.1) with initial condition  $x(t_0) = x_0$  is said to **uniformly ultimately bounded** (u.u.b.) with respect to a set  $S$  if there is a nonnegative constant  $T(x_0, S)$  such that

$$x(t) \in S \text{ for all } t \geq t_0 + T.$$

Uniform ultimate boundedness says that the solution trajectory of (C.1)) beginning at  $x_0$  at time  $t_0$  will ultimately enter and remain within the set  $S$ . If the set  $S$  is a small region about the equilibrium, then uniform ultimate boundedness is a practical notion of stability, which is useful in control system design.

### C.0.1 Quadratic Forms and Lyapunov Functions

**Definition C.5** Given a symmetric matrix  $P = (p_{ij})$  the scalar function

$$V(x) = x^T P x = \sum_{i,j=1}^n p_{ij} x_i x_j \quad (\text{C.5})$$

is said to be a **quadratic form**.  $V(x)$ , equivalently the quadratic form, is said to be **positive definite** if and only if

$$V(x) > 0 \quad (\text{C.6})$$

for  $x \neq 0$ .

Note that  $V(0) = 0$ .  $V(x)$  will be positive definite if and only if the matrix  $P$  is a positive definite matrix, that is, has all eigenvalues positive.

The level surfaces of  $V$ , given as solutions of  $V(x) = \text{constant}$  are ellipsoids in  $\mathbb{R}^n$ . A positive definite quadratic form is like a norm. In fact, given the usual norm  $\|x\|$  on  $\mathbb{R}^n$ , the function  $V$  given as

$$V(x) = x^T x = \|x\|^2 \quad (\text{C.7})$$

is a positive definite quadratic form.

**Definition C.6** Let  $V(x) : \mathbb{R}^n \rightarrow \mathbb{R}$  be a continuous function with continuous first partial derivatives in a neighborhood of the origin in  $\mathbb{R}^n$ . Further suppose that  $V$  is positive definite, that is,  $V(0) = 0$  and  $V > 0$  for  $x \neq 0$ . Then  $V$  is called a **Lyapunov Function Candidate** (for the system (C.1)).

The positive definite function  $V$  is also like a norm. For the most part we will be utilizing Lyapunov function candidates that are quadratic forms, but 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.

By the derivative of  $V$  along trajectories of (C.1), or the derivative of  $V$  in the direction of the vector field defining (C.1), we mean

$$\dot{V}(t) = \langle dV, f \rangle = \frac{\partial V}{\partial x_1} f_1(x) + \cdots + \frac{\partial V}{\partial x_n} f_n(x). \quad (\text{C.8})$$

Suppose that we evaluate the Lyapunov function candidate  $V$  at points along a solution trajectory  $x(t)$  of (C.1) and find that  $\dot{V}(t)$  is decreasing for increasing  $t$ . Intuitively, since  $V$  acts like a norm, this must mean that the given solution trajectory must be converging toward the origin. This is the idea of Lyapunov stability theory.

### C.0.2 Lyapunov Stability

**Theorem 6** *The null solution of (C.1) is stable if there exists a Lyapunov function candidate  $V$  such that  $\dot{V}$  is negative semi-definite along solution trajectories of (C.1), that is, if*

$$\dot{V} = \langle dV, f(x) \rangle = dV^T f(x) \leq 0. \quad (\text{C.9})$$

Equation (C.9) says that the derivative of  $V$  computed along solutions of (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.9) says that the solution must remain near the origin. If a Lyapunov function candidate  $V$  can be found satisfying (C.9) then  $V$  is called a **Lyapunov Function** for the system (C.1). Note that Theorem 6 gives only a sufficient condition for stability of (C.1). If one is unable to find a Lyapunov function satisfying (C.9) it does not mean that the system is unstable. However, an easy sufficient condition for instability of (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 7** *The null solution of (C.1) is asymptotically stable if there exists a Lyapunov function candidate  $V$  such that  $\dot{V}$  is strictly negative definite along solutions of (C.1), that is,*

$$\dot{V}(x) < 0. \quad (\text{C.10})$$

The strict inequality in (C.10) means that  $V$  is actually decreasing along solution trajectories of (C.1) and hence the trajectories must be converging to the equilibrium point.

**Corollary C.1** *Let  $V$  be a Lyapunov function candidate and let  $S$  be any level surface of  $V$ , that is,*

$$S(c_0) = \{x \in \mathbb{R}^n | V(x) = c_0\} \quad (\text{C.11})$$

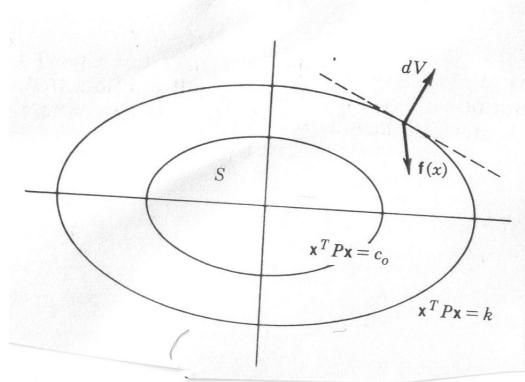


Fig. C.2 Illustrating ultimate boundedness.

for some constant  $c_0 > 0$ . Then a solution  $x(t)$  of (C.1) is uniformly ultimately bounded with respect to  $S$  if

$$\dot{V} = \langle dV, f(x) \rangle < 0 \quad (\text{C.12})$$

for  $x$  outside of  $S$ .

If  $\dot{V}$  is negative outside of  $S$  then the solution trajectory outside of  $S$  must be pointing toward  $S$  as shown in Figure C.2. Once the trajectory reaches  $S$  we may or may not be able to draw further conclusions about the system, except that the trajectory is trapped inside  $S$ .

### C.0.3 Lyapunov Stability for Linear Systems

Consider the linear system (C.4) and let

$$V(x) = x^T Px \quad (\text{C.13})$$

be a Lyapunov function candidate, where  $P$  is symmetric and positive definite. Computing  $\dot{V}$  along solutions of (C.4) yields

$$\begin{aligned} \dot{V} &= \dot{x}^T Px + x^T P \dot{x} \\ &= x^T (A^T P + PA)x \\ &= -x^T Qx \end{aligned} \quad (\text{C.14})$$

where we have defined  $Q$  as

$$A^T P + PA = -Q. \quad (\text{C.15})$$

Theorem C.8 now says that if  $Q$  given by (C.15) is positive definite (it is automatically symmetric since  $P$  is) then the linear system (C.4) is stable. One

approach that we can now take is to first fix  $Q$  to be symmetric, positive definite and solve (C.15), which is now called the **matrix Lyapunov equation**, for  $P$ . If a symmetric positive definite solution  $P$  can be found to this equation, then (C.4) is stable and  $x^T P x$  is a Lyapunov function for the linear system (C.4). The converse to this statement also holds. In fact, we can summarize these statements as

**Theorem 8** *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 Lyapunov equation (C.11) 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, namely, (C.11), which is certainly easier than finding all the roots of the characteristic polynomial and, for large systems, is more efficient than, say, the Routh test.

The strict inequality in (C.7) may be difficult to obtain for a given system and Lyapunov function candidate. We therefore discuss LaSalle's Theorem which can be used to prove asymptotic stability even when  $V$  is only negative semi-definite.

#### C.0.4 LaSalle's Theorem

Given the system (C.1) suppose a Lyapunov function candidate  $V$  is found such that, along solution trajectories

$$\dot{V} \leq 0. \quad (\text{C.16})$$

Then (C.1) is asymptotically stable if  $V$  does not vanish identically along any solution of (C.1) other than the null solution, that is, (C.1) is asymptotically stable if the only solution of (C.1) satisfying

$$\dot{V} \equiv 0 \quad (\text{C.17})$$

is the null solution.

# Appendix D

## State Space Theory of Dynamical Systems

Here we give a brief introduction to some concepts in the state space theory of linear and nonlinear systems.

**Definition D.1** A vector field  $f$  is a continuous function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ .

We can think of a differential equation

$$\dot{x}(t) = f(x(t)) \quad (\text{D.1})$$

as being defined by a vector field  $f$  on  $\mathbb{R}^n$ . A solution  $t \rightarrow x(t)$  of (D.1) with  $x(t_0) = x_0$  is then a curve  $C$  in  $\mathbb{R}^n$ , beginning at  $x_0$  parametrized 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 (D.1). For two dimensional systems, we can represent

$$t \rightarrow \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \quad (\text{D.2})$$

by a curve  $C$  in the plane.

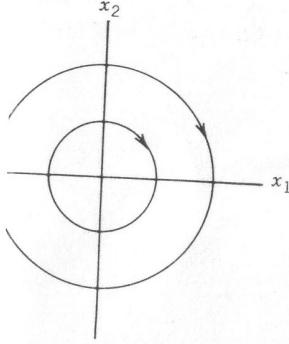


Fig. D.1 Phase portrait for Example B.1

**Example D.1** Consider the two-dimensional system

$$\dot{x}_1 = x_2 \quad x_1(0) = x_{10} \quad (\text{D.3})$$

$$\dot{x}_2 = -x_1 \quad x_2(0) = x_{20} \quad (\text{D.4})$$

In the phase plane the solutions of this equation are circles of radius

$$r = \sqrt{x_{10}^2 + x_{20}^2} \quad (\text{D.5})$$

To see this consider the equation

$$x_1^2(t) + x_2^2(t) = r^2 \quad (\text{D.6})$$

Clearly the initial conditions satisfy this equation. If we differentiate (D.6) in the direction of the vector field  $f = (x_2, -x_1)^T$  that defines (D.3)-(D.4) we obtain

$$2x_1\dot{x}_1 + 2x_2\dot{x}_2 = 2x_1x_2 - 2x_2x_1 = 0. \quad (\text{D.7})$$

Thus  $f$  is tangent to the circle. The graph of such curves  $C$  in the  $x_1 - x_2$  plane for different initial conditions are shown in Figure D.1.  $\diamond$

The  $x_1 - x_2$  plane is called the **phase plane** and the trajectories of the system (D.3)-(D.4) form what is called the **phase portrait**. For linear systems of the form

$$\dot{x} = Ax \quad (\text{D.8})$$

in  $\mathbb{R}^2$  the phase portrait is determined by the eigenvalues and eigenvectors of  $A$ . For example, consider the system

$$\dot{x}_1 = x_2 \quad (\text{D.9})$$

$$\dot{x}_2 = x_1. \quad (\text{D.10})$$

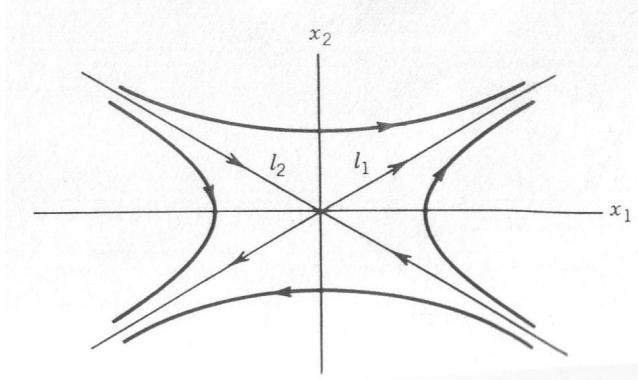


Fig. D.2 Phase portrait for Example B.2.

In this case

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (\text{D.11})$$

The phase portrait is shown in Figure D.2. The lines  $\ell_1$  and  $\ell_2$  are in the direction of the eigenvectors of  $A$  and are called **eigen-subspaces** of  $A$ .

#### D.0.5 State Space Representation of Linear Systems

Consider a single-input/single-output linear control system with input  $u$  and output  $y$  of the form

$$a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = u. \quad (\text{D.12})$$

The characteristic polynomial, whose roots are the open loop poles, is given as

$$p(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_0. \quad (\text{D.13})$$

For simplicity we suppose that  $p(x)$  is monic, that is,  $a_n = 1$ . The standard way of representing (D.12) in state space is to define  $n$  state variables  $x_1, x_2, \dots, x_n$  as

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} = \dot{x}_1 \\ x_3 &= \ddot{y} = \dot{x}_2 \\ &\vdots \\ x_n &= \frac{d^{n-1} y}{dt^{n-1}} = \dot{x}_{n-1} \end{aligned} \quad (\text{D.14})$$

and express (D.12) as the system of first order differential equations

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_{n-1} &= x_n \\ \dot{x}_n &= \frac{d^n y}{dt^n} = -a_0 y - a_1 \frac{dy}{dt} - \cdots - a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + u \\ &= -a_0 x_1 - a_1 x_2 - \cdots - a_{n-1} x_n + u.\end{aligned}\tag{D.15}$$

In matrix form this system of equations is written as

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdot & \cdot & 0 \\ 0 & 0 & 1 & \cdot & 0 \\ & & \ddots & \ddots & \\ & & & & 1 \\ -a_0 & \cdot & \cdot & \cdot & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{D.16}$$

or

$$\dot{x} = Ax + bu \quad x \in \mathbb{R}^n.$$

The output  $y$  can be expressed as

$$\begin{aligned}y &= [1, 0, \dots, 0]x \\ &= c^T x.\end{aligned}\tag{D.17}$$

It is easy to show that

$$\det(sI - A) = s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0 \tag{D.18}$$

and so the last row of the matrix  $A$  consists of precisely the coefficients of the characteristic polynomial of the system, and furthermore the eigenvalues of  $A$  are the open loop poles of the system.

In the Laplace domain, the transfer function  $\frac{Y(s)}{U(s)}$  is equivalent to

$$\frac{Y(s)}{U(s)} = c^T (sI - A)^{-1} b. \tag{D.19}$$

# *References*

1. M. Arbib. *Computers and the Cybernetic Society*. Academic Press, New York, 1977.
2. 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, 1985.
3. H. Asada and J-J. Slotine. *Robot Analysis and Control*. Wiley, New York, 1986.
4. S. Barnett. *Matrix Methods for Engineers and Scientists*. McGraw-Hill, London, 1979.
5. Jerome Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, December 1991.
6. G. Beni and S. Hackwood, editors. *Recent Advances in Robotics*. Wiley, New York, 1985.
7. D. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, second edition, 1999.
8. William M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press, 1986.

9. O. Botema and B. Roth. *Theoretical Kinematics*. North Holland, Amsterdam, 1979.
10. M Brady and et al., editors. *Robot Motion: Planning and Control*. MIT Press, Cambridge, MA, 1983.
11. 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.
12. J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.
13. H. Choset, K. M. Lynch, S. A. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, first edition, 2005.
14. J. C. Colson and N. D. Perreira. Kinematic arrangements used in industrial robots. In *Proc. 13th International Symposium on Industrial Robots*, 1983.
15. 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.
16. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, Reading, MA, 1986.
17. A. J. Critchlow. *Introduction to Robotics*. Macmillan, New York, 1985.
18. M. L. Curtiss. *Matrix Groups*. Springer-Verlag, New York, second edition, 1984.
19. 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.
20. J. Denavit and R. S. Hartenberg. A kinematic notation for lower pair mechanisms. *Applied Mechanics*, 22:215–221, 1955.
21. R. Dorf. *Robotics and Automated Manufacturing*. Reston, VA, 1983.
22. J. Duffy. *Analysis of Mechanisms and Robot Manipulators*. Wiley, New York, 1980.
23. J. Engleberger. *Robotics in Practice*. Kogan Page, London, 1980.
24. 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.

25. S. E. Fahlman. A planning system for robot construction tasks. *Artificial Intelligence*, 5:1–49, 1974.
26. 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.
27. 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.
28. R. Fikes and N. Nilsson. STRIPS: A new approach to th eapplicatin of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.
29. S. H. Friedberg, A. J. Insel, and L. E. Spence. *Linear Algebra*. Prentice-Hall, Englewood Cliffs, NJ, 1979.
30. K.S. Fu, R. C. Gonzalez, and C.S.G. Lee. *Robotics: Control Sensing, Vision, and Intelligence*. McGraw-Hill, St Louis, 1987.
31. K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson, editors. *The Algorithmic Foundations of Robotics*. A. K. Peters, Boston, 1995.
32. 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.
33. M. Groover and et al. *Industrial Robotics: Technology, Programming, and Applications*. McGraw-Hill, St. Louis, 1986.
34. J.M. Hollerbach and S. Gideon. Wrist-partitioned inverse kinematic accelerations and manipulator dynamics. *International Journal of Robotics Research*, 4:61–76, 1983.
35. 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.
36. Subbarao Kambhampati and Larry S. Davis. Multiresolution path planning for mobile robots. *IEEE Journal of Robotics and Automation*, 2(3):135–145, September 1986.
37. Lydia E. Kavraki. *Random Networks in Configuration Space for Fast Path Planning*. PhD thesis, Stanford University, Stanford, CA, 1994.
38. 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.

39. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.
40. D.E. Koditschek. Robot planning and control via potential functions. In *The Robotics Review 1*, pages 349–367. MIT Press, 1989.
41. Y. Koren. *Robotics for Engineers*. McGraw-Hill, St. Louis, 1985.
42. J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
43. C.S.G. Lee. Robot arm kinematics, dynamics, and control. *Computer*, 15(12):62–80, 1982.
44. C.S.G. Lee, R. C. Gonzales, and K. S. Fu. *Tutorial on Robotics*. IEEE Computer Society Press, Silver Spring, MD, 1983.
45. 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.
46. Ming Lin and Dinesh Manocha. Efficient contact determination in dynamic environments. *International Journal of Computational Geometry and Applications*, 7(1):123–151, 1997.
47. T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, February 1983.
48. 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.
49. D. McCloy and M. Harris. *Robotics: An Introduction*. Halstead Press, New York, 1986.
50. P. McCorduck. *Machines Who Think*. W.H. Freeman, San Francisco, 1979.
51. M. Minsky, editor. *Robotics*. Omni Publications International, Ltd., New York, 1985.
52. 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.
53. 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.
54. I. S. Sokolnikoff nad R. M. Redheffer. *Mathematical Methods of Physics and Modern Engineering*. McGraw-Hill, New York, 1958.

55. 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.
56. 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.
57. N. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *Proc. Int. Joint Conf. on Art. Intell.*, 1969.
58. 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.
59. R. Paul. *Robot Manipulators: Mathematics, Programming and Control*. MIT Press, Cambridge, MA, 1982.
60. R. P. Paul, B. E. Shimano, and G. Mayer. Kinematic control equations for simple manipulators. *IEEE Trans. Systems, Man., and Cybernetics*, SMC-11(6):339–455, 1981.
61. D. L. Pieper. *The Kinematics of Manipulators under Computer Control*. PhD thesis, Stanford University, 1968.
62. J. N. Reddy and M.L. Rasmussen. *Advanced Engineering Analysis*. Wiley, New York, 1982.
63. Reid. *Robotics: A Systems Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
64. 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.
65. J. T. Schwartz, M. Sharir, and J. Hopcroft, editors. *Planning, Geometry, and Complexity of Robot Motion*. Ablex, Norwood, NJ, 1987.
66. M. Shahinpoor. The exact inverse kinematics solutions for the rhino xr-2 robot. *Robotics Age*, 7(8):6–14, 1985.
67. M. Shahinpoor. *Robot Engineering Textbook*. Harper and Row, New York, 1987.
68. 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.
69. 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.

70. W. Snyder. *Industrial Robots: Computer Interfacing and Control*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
71. C.H. Su and C. W. Radcliffe. *Kinematics and Mechanisms Design*. Wiley, New York, 1978.
72. 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.
73. 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.
74. Gino van den Bergen. *User's Guide to the SOLID Interference Detection Library*. Eindhoven University of Technology, Eindhoven, The Netherlands, 1999.
75. E. T. Whirraker. *Dynamics of Particles and Rigid Bodies*. Cambridge University Press, London, 1904.
76. D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *J. Dyn. Sys., Meas. Cont.*, December 1972.
77. W. Wolovich. *Robotics: Basic Analysis and Design*. Holt, Rinehart, and Winston, New York, 1985.
78. T. Yoshikawa. Manipulability of robotic mechanisms. *International Journal of Robotics Research*, 4(2):3–9, 1985.

# *Index*

- Accuracy, 7  
Across Variable, 287  
Actuator Dynamics, 231  
Angle, 69  
Angular momentum, 320, 323  
  conservation of, 299  
Angular velocity, 119  
Anthropomorphic, 10  
Anti-windup, 260  
Apparent Damping, 293  
Apparent Inertia, 293  
Apparent Stiffness, 293  
Application area, 6  
Approach, 74  
Arbitrary, 255  
Arm, 7  
Armature, 231  
Arm singularity, 133  
Articulated (RRR), 6  
Artificial Constraint, 284–285  
Assembly, 6  
Atan2, 48  
Atan, 377  
Average, 237  
Axis/Angle, 47  
Axis/angle representation, 52  
Back emf, 231  
Back emf Constant, 233  
Bang-Bang, 182  
Base, 18  
Basic homogeneous transformation, 55  
Basic rotation matrix, 36  
Basis, 283  
Bilinear Form, 283  
Blend time, 179  
Capacitive, 288  
Capacitive Environment, 288  
Cartesian (PPP), 6  
Centrifugal, 202  
Characteristic polynomial, 307  
Chow’s theorem, 299, 325  
Christoffel symbol, 201  
Closed-form equations, 216  
Closing, 9  
Codistribution, 301  
Compensator, 230  
Completely integrable, 305  
Completely observable, 257  
Completely state-controllable, 254  
Compliance Frame, 284  
Computed torque, 23, 267  
Computer interface, 7  
Configuration, 4  
Configuration kinematic equation, 68  
Configuration space, 4  
Conservation of angular momentum, 323  
Constraint, 282  
Constraint Frame, 284  
Constraint  
  Holonomic, 187, 191

- Constraint
  - holonomic, 319
- Constraint
  - Nonholonomic, 191
- Constraint
  - nonholonomic, 299
  - Pfaffian, 319
  - rolling, 320
- Constraints
  - Holonomic, 188
- Continous Path Tracking, 229
- Continuous path, 6
- Control, 1
- Control computer, 7
- Control
  - Independent Joint, 230
- Control
  - inner loop/outer loop, 307
- Control
  - inverse dynamic, 267
- Controllability, 254
- Controllability, 325
- Controllability matrix, 254
- Controllability rank condition, 325
- Controllable, 254
- Controller resolution, 8
- Control
  - outer loop, 307
- Coordinates
  - Generalized, 190
  - Generalized, 192
- Coriolis, 202
- Cotangent space, 300
- Covector field, 300
- Cross-product form, 22
- Current
  - Armature, 233
- Current frame, 43, 46
- Cylindrical (RPP), 6
- D'Alembert's Principle, 194
- Damping
  - Apparent, 293
- DC-gain, 288
- DC-Motor, 189
- Decoupled, 211
- Degrees-of-freedom, 4
- Denavit-Hartenberg, 19
- Dexterous workspace, 5
- Diffeomorphism, 300, 309, 317
- Direct Drive Robot, 230
- Directional derivative, 303
- Displacement
  - Virtual, 192
- Distribution, 301
  - Involutive, 306
- Disturbance, 230
- Disturbance Rejection, 230
- Double integrator system, 267
- Driftless system, 320, 325
- Driftless systems, 299
- Dual vector space, 300
- Dynamics, 1, 187
  - In task space, 291
  - Newton-Euler Formulation, 188
- Effective inertia, 237
- Effort, 287
- End-effector, 9, 74
- End-of-arm tooling, 7
- Energy, 287
  - Kinetic, 187–188
  - Potential, 187, 189
- Environment, 281
  - Capacitive, 288
  - Classification of, 293
  - Inertial, 288
  - Resistive, 288
- Environment Stiffness, 286
- Equation
  - Euler-Lagrange, 187
  - Euler-Lagrange, 188
- Estimation error, 257
- Euler Angle, 47
- Euler Angles, 47, 192
- Euler-Lagrange equation, 188
- External and internal sensor, 7
- External power source, 7
- Eye-in-hand configuration, 356
- Feedback linearizable, 308
- Feedback linearization, 299
  - global, 314
- Feedforward control, 23, 244
- Five-bar linkage, 209
- Fixed-camera configuration, 356
- Fixed frame, 44, 46
- Flow, 287
- Force, 281
- Force control, 24
- Force Control, 281
- Force
  - Generalized, 190
  - Generalized, 195
  - Gravitational, 189
- Forward, 68
- Forward kinematic equation, 19
- Forward kinematics problem, 18
- Frame
  - compliance, 284
  - constraint, 284
- Frame
  - current, 46
  - fixed, 46
- Frobenius theorem, 304, 311
- Gear Train, 189
- Generalized Coordinates, 190, 192

- Generalized Force, 190, 195
- Geometric nonlinear control, 299
- Global feedback linearization, 309
- Gradient, 303
- Group, 59
- Guarded motion, 171
- Gyroscopic term, 217
- Hand, 9
- Hardware/Software trade-off, 229
- Holonomic constraint, 319
- Home, 17
- Homogeneous coordinate, 19
- Homogeneous representation, 55
- Homogeneous transformation, 19, 55
- Hybrid control, 24
- Hybrid Impedance Control, 293
- Image-based visual servo control, 357
- Image feature velocity, 358
- Image Jacobian, 358
- Impedance, 286
- Impedance, 288
- Impedance control, 24
- Impedance Control, 292
  - Hybrid, 293
- Impedance
  - Dual, 294
- Impedance Operator, 288
- Implicit function theorem, 305
- Independent Joint Control, 230
- Inductance
  - Armature, 233
- Inertia
  - Apparent, 293
  - Inertial, 288
- Inertial Environment, 288
- Inertia matrix, 200
- Inertia Tensor, 197
- Inner-loop, 269
- Inner loop control, 299
- Inner product, 303
- Integrability
  - complete, 305
- Integral manifold, 304–305, 324
- Integrator windup, 260
- Interaction matrix, 358–359
- Invariant, 283
- Inverse dynamics, 23
- Inverse Dynamics, 291
- Inverse dynamics, 299
- Inverse dynamics control, 267
- Inverse Kinematics, 20
- Inverse kinematics, 85
- Inverse orientation kinematic, 87
- Inverse position kinematic, 87
- Involutive, 306
- Involutive closure, 324
- Jacobian, 22, 113, 123, 290
  - Jacobian matrix, 302
  - Joint flexibility, 299, 311
  - Joints, 3
  - Joint torque sensor, 282
  - Joint variable, 4
  - Killing Form, 283
  - Kinematically redundant, 4
  - Kinematic chain, 3
  - Kinematics, 1
  - Kinetic Energy, 187–188
  - Klein Form, 283
  - Lagrangian, 187, 189, 196
  - Laplace Domain, 288
  - Laplace Transform, 288
  - Law of Cosines, 20
  - Left arm, 91
  - Length, 69
  - Lie bracket, 302, 306
  - Lie derivative, 303
  - Linear Quadratic (LQ) Optimal Control, 256
  - Linear Segments with Parabolic Blends, 177
  - Linear state feedback control law, 254
  - Links, 3
  - LSPB, 177
  - Magnetic Flux, 231
  - Manifold, 300, 302
  - Manipulation, 320
  - Manipulator Jacobian, 123
  - Manipulator
    - spherical, 11
  - Matrix Algebraic Riccati equation, 256
  - Matrix
    - inertia, 200
    - transformation, 67
  - Mechanical Impedance, 286
  - Method of computed torque, 247
  - Method of control, 6
  - Minimum phase, 245
  - Mobile robot, 320
  - Mobile robots, 299
  - Mobility Tensor, 292
  - Motion
    - guarded, 171
  - Motion pereptibility, 372
  - Motor
    - AC, 231
    - Brushless DC, 231
    - DC, 230–231
    - Rotor, 231
    - Stator, 231
  - Natural Constraint, 284–285
  - Network Model, 287
  - Newton-Euler formulation, 215
  - Newton's Second Law, 188
  - Non-assembly, 6
  - Nonholonomic constraint, 299, 319
  - Non-servo, 6

- Normal, 74
- Norton Equivalent, 289
- Norton Network, 289
- Numerically controlled milling machines, 2
- Observability, 257
- Observability matrix, 258
- Observable, 257
- Observer, 256–257
- Offset, 69
- One-Port, 287
- One-Port Network, 287
- Opening, 9
- Operator
  - Impedance, 288
- Orientation, 4
- Orientation matrix, 19
- Orientation of the tool frame, 19
- Orthogonal, 34, 284
- Orthonormal Basis, 284
- Outer-loop, 269
- Outer Loop, 291
- Outer loop control, 299, 307
- Parallelogram linkage, 10
- Partitioned methods, 369
- Permanent magnet, 232
- Permanent Magnet DC Motor, 230
- Perspective projection, 334
- Pfaffian constraint, 319
- Pitch, 49
- Planning, 1
- Point-to-point, 6
- Point to point, 171
- Point-to-Point Control, 229
- Port Variable, 287
- Position, 281
- Position-based visual servo control, 357
- Positioning, 4
- Post-multiply, 46
- Potential Energy, 187, 189
- Power, 287
- Power source, 6
- Premultiply, 46
- Principle
  - D'Alembert, 194
- Principle of Virtual Work, 188, 193
- Prismatic, 3
- Quaternion, 61
- Reachable workspace, 5
- Reciprocal Basis, 283
- Reciprocity, 284
- Reciprocity Condition, 285
- Rejecting, 23
- Repeatability, 7
- Representation
  - axis/angle, 52
  - homogeneous, 55
- Resistance
  - Armature, 233
  - Resistive, 288
  - Resistive Environment, 288
  - Resolvability, 372
  - Reverse order, 44
  - Revolute, 10, 3
  - Right arm, 91
  - Robot, 1
  - Robota, 1
  - Robot
    - Direct Drive, 230
    - Robot
      - flexible joint, 311
    - Robotic System, 7
  - Robot Institute of America, 2
  - Robot
    - mobile, 299
    - mobile, 320
  - Roll, 49
  - Rolling constraint, 320
  - Rolling contact, 299
  - Roll-pitch-yaw, 47
  - Rotation matrix, 33
  - Rotor, 231
  - Satellite, 326
  - Saturation, 242
  - SCARA, 12
  - SCARA (RRP), 6
  - Second Method of Lyapunov, 23
  - Separation Principle, 258
  - Servo, 6
  - Set-point tracking problem, 238
  - Singular configuration, 22, 132
  - Singular configurations, 114
  - Singularity, 132
  - Skew symmetric, 115
  - Sliding, 74
  - Spherical manipulator, 11
  - Spherical (RRP), 6
  - Spherical wrists, 8
  - State, 5
  - State space, 5
  - Stator, 231
  - Stiffness
    - Apparent, 293
  - Strain gauge, 282
  - Switching time, 182
  - Symbol
    - Christoffel, 201
  - System
    - double integrator, 267
  - System
    - driftless, 320
    - driftless, 325
  - Tactile sensor, 282
  - Tangent plane, 305
  - Tangent space, 300, 324

- Task Space, 290
- Task Space Dynamics, 291
- Teach and playback mode, 171
- Teleoperators, 2
- Theorem
  - Frobenius, 304
- Thévenin Equivalent, 289
- Thévenin Network, 289
- Through Variable, 287
- Tool frame, 74
- Torque
  - computed, 267
- Torque Constant, 233
- Track, 23
- Tracking, 230
- Tracking and Disturbance Rejection Problem, 23
- Trajectory, 171
- Trajectory Control, 281
- Transformation
  - basic homogeneous, 55
  - homogeneous, 55
- Transformation matrix, 67
- Transpose, 290
- Twist, 69
- Two-argument arctangent function, 48
- Two link manipulator, 290
- Unicycle, 320
- Vector field
  - complete, 324
  - smooth, 300
- Velocity
  - angular, 119
- Via points, 171
- Via points, 182
- Virtual displacement, 192
- Virtual Displacement, 290
- Virtual Displacements, 188
- Virtual Work, 187, 290
- Vision, 1
- Voltage
  - Armature, 233
- Workspace, 5
  - dexterous, 5
  - reachable, 5
- Work
  - Virtual, 187
  - Virtual, 290
- World, 18
- Wrist, 8
- Wrist center, 87
- Wrist force sensor, 282
- Wrist singularity, 133
- Yaw, 49