INTRODUCTION TO ROBOTICS

MECHANICS AND CONTROL

THIRD EDITION

JOHN J. CRAIG

Introduction to Robotics

Mechanics and Control

Third Edition

John J. Craig

PEARSON
Prentice Hall

Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data
Craig, John J., 1955–
Introduction to robotics: mechanics and control / John J. Craig.—3rd ed.
p. cm.
Bibliography: p.
Includes index.
ISBN 0-201-54361-3
1. Robotics. I. Title

TJ211.C67 1989 88-37607
629.8'92—dc19

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Associate Editor: *Alice Dworkin*
Editorial Assistant: *Carole Snyder*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *James Buckley*
Director of Creative Services: *Paul Belfanti*
Art Director: *Jayne Conte*
Cover Designer: *Bruce Kenselaar*
Art Editor: *Greg Dulles*
Manufacturing Manager: *Trudy Pisciotti*
Manufacturing Buyer: *Lisa McDowell*
Senior Marketing Manager: *Holly Stark*

PEARSON Prentice Hall

Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, NJ 07458

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-201-54361-3

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Ltd., *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Ltd., *Toronto*
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., *Upper Saddle River, New Jersey*

Contents

Preface

Scientists often have the feeling that, through their work, they are learning about some aspect of themselves. Physicists see this connection in their work; so do, for example, psychologists and chemists. In the study of robotics, the connection between the field of study and ourselves is unusually obvious. And, unlike a science that seeks only to analyze, robotics as currently pursued takes the engineering bent toward synthesis. Perhaps it is for these reasons that the field fascinates so many of us.

The study of robotics concerns itself with the desire to synthesize some aspects of human function by the use of mechanisms, sensors, actuators, and computers. Obviously, this is a huge undertaking, which seems certain to require a multitude of ideas from various "classical" fields.

Currently, different aspects of robotics research are carried out by experts in various fields. It is usually not the case that any single individual has the entire area of robotics in his or her grasp. A partitioning of the field is natural to expect. At a relatively high level of abstraction, splitting robotics into four major areas seems reasonable: mechanical manipulation, locomotion, computer vision, and artificial intelligence.

This book introduces the science and engineering of mechanical manipulation. This subdiscipline of robotics has its foundations in several classical fields. The major relevant fields are mechanics, control theory, and computer science. In this book, Chapters 1 through 8 cover topics from mechanical engineering and mathematics, Chapters 9 through 11 cover control-theoretical material, and Chapters 12 and 13 might be classed as computer-science material. Additionally, the book emphasizes computational aspects of the problems throughout; for example, each chapter that is concerned predominantly with mechanics has a brief section devoted to computational considerations.

This book evolved from class notes used to teach "Introduction to Robotics" at Stanford University during the autumns of 1983 through 1985. The first and second editions have been used at many institutions from 1986 through 2002. The third edition has benefited from this use and incorporates corrections and improvements due to feedback from many sources. Thanks to all those who sent corrections to the author.

This book is appropriate for a senior undergraduate- or first-year graduate-level course. It is helpful if the student has had one basic course in statics and dynamics and a course in linear algebra and can program in a high-level language. Additionally, it is helpful, though not absolutely necessary, that the student have completed an introductory course in control theory. One aim of the book is to present material in a simple, intuitive way. Specifically, the audience need not be strictly mechanical engineers, though much of the material is taken from that field. At Stanford, many electrical engineers, computer scientists, and mathematicians found the book quite readable.

Directly, this book is of use to those engineers developing robotic systems, but the material should be viewed as important background material for anyone who will be involved with robotics. In much the same way that software developers have usually studied at least some hardware, people not directly involved with the mechanics and control of robots should have some such background as that offered by this text.

Like the second edition, the third edition is organized into 13 chapters. The material will fit comfortably into an academic semester; teaching the material within an academic quarter will probably require the instructor to choose a couple of chapters to omit. Even at that pace, all of the topics cannot be covered in great depth. In some ways, the book is organized with this in mind; for example, most chapters present only one approach to solving the problem at hand. One of the challenges of writing this book has been in trying to do justice to the topics covered within the time constraints of usual teaching situations. One method employed to this end was to consider only material that directly affects the study of mechanical manipulation.

At the end of each chapter is a set of exercises. Each exercise has been assigned a difficulty factor, indicated in square brackets following the exercise's number. Difficulties vary between [00] and [50], where [00] is trivial and [50] is an unsolved research problem.[1] Of course, what one person finds difficult, another might find easy, so some readers will find the factors misleading in some cases. Nevertheless, an effort has been made to appraise the difficulty of the exercises.

At the end of each chapter there is a programming assignment in which the student applies the subject matter of the corresponding chapter to a simple three-jointed planar manipulator. This simple manipulator is complex enough to demonstrate nearly all the principles of general manipulators without bogging the student down in too much complexity. Each programming assignment builds upon the previous ones, until, at the end of the course, the student has an entire library of manipulator software.

Additionally, with the third edition we have added MATLAB exercises to the book. There are a total of 12 MATLAB exercises associated with Chapters 1 through 9. These exercises were developed by Prof. Robert L. Williams II of Ohio University, and we are greatly indebted to him for this contribution. These exercises can be used with the MATLAB Robotics Toolbox[2] created by Peter Corke, Principal Research Scientist with CSIRO in Australia.

Chapter 1 is an introduction to the field of robotics. It introduces some background material, a few fundamental ideas, and the adopted notation of the book, and it previews the material in the later chapters.

Chapter 2 covers the mathematics used to describe positions and orientations in 3-space. This is extremely important material: By definition, mechanical manipulation concerns itself with moving objects (parts, tools, the robot itself) around in space. We need ways to describe these actions in a way that is easily understood and is as intuitive as possible.

[1] I have adopted the same scale as in *The Art of Computer Programming* by D. Knuth (Addison-Wesley).

[2] For the MATLAB Robotics Toolbox, go to http://www.cat.csiro.au/cmst/staff/pic/robotics/Toolbox1.htm.

Chapters 3 and 4 deal with the geometry of mechanical manipulators. They introduce the branch of mechanical engineering known as kinematics, the study of motion without regard to the forces that cause it. In these chapters, we deal with the kinematics of manipulators, but restrict ourselves to static positioning problems.

Chapter 5 expands our investigation of kinematics to velocities and static forces.

In Chapter 6, we deal for the first time with the forces and moments required to cause motion of a manipulator. This is the problem of manipulator dynamics.

Chapter 7 is concerned with describing motions of the manipulator in terms of trajectories through space.

Chapter 8 many topics related to the mechanical design of a manipulator. For example, how many joints are appropriate, of what type should they be, and how should they be arranged?

In Chapters 9 and 10, we study methods of controlling a manipulator (usually with a digital computer) so that it will faithfully track a desired position trajectory through space. Chapter 9 restricts attention to linear control methods; Chapter 10 extends these considerations to the nonlinear realm.

Chapter 11 covers the field of active force control with a manipulator. That is, we discuss how to control the application of forces by the manipulator. This mode of control is important when the manipulator comes into contact with the environment around it, such as during the washing of a window with a sponge.

Chapter 12 overviews methods of programming robots, specifically the elements needed in a robot programming system, and the particular problems associated with programming industrial robots.

Chapter 13 introduces off-line simulation and programming systems, which represent the latest extension to the man–robot interface.

I would like to thank the many people who have contributed their time to helping me with this book. First, my thanks to the students of Stanford's ME219 in the autumn of 1983 through 1985, who suffered through the first drafts, found many errors, and provided many suggestions. Professor Bernard Roth has contributed in many ways, both through constructive criticism of the manuscript and by providing me with an environment in which to complete the first edition. At SILMA Inc., I enjoyed a stimulating environment, plus resources that aided in completing the second edition. Dr. Jeff Kerr wrote the first draft of Chapter 8, Prof. Robert L. Williams II contributed the MATLAB exercises found at the end of each chapter, and Peter Corke expanded his Robotics Toolbox to support this book's style of the Denavit–Hartenberg notation. I owe a debt to my previous mentors in robotics: Marc Raibert, Carl Ruoff, Tom Binford, and Bernard Roth.

Many others around Stanford, SILMA, Adept, and elsewhere have helped in various ways—my thanks to John Mark Agosta, Mike Ali, Lynn Balling, Al Barr, Stephen Boyd, Chuck Buckley, Joel Burdick, Jim Callan, Brian Carlisle, Monique Craig, Subas Desa, Tri Dai Do, Karl Garcia, Ashitava Ghosal, Chris Goad, Ron Goldman, Bill Hamilton, Steve Holland, Peter Jackson, Eric Jacobs, Johann Jäger, Paul James, Jeff Kerr, Oussama Khatib, Jim Kramer, Dave Lowe, Jim Maples, Dave Marimont, Dave Meer, Kent Ohlund, Madhusudan Raghavan, Richard Roy, Ken Salisbury, Bruce Shimano, Donalda Speight, Bob Tilove, Sandy Wells, and Dave Williams.

The students of Prof. Roth's Robotics Class of 2002 at Stanford used the second edition and forwarded many reminders of the mistakes that needed to get fixed for the third edition.

Finally I wish to thank Tom Robbins at Prentice Hall for his guidance with the first edition and now again with the present edition.

J.J.C.

CHAPTER 1

Introduction

1.1 BACKGROUND

The history of industrial automation is characterized by periods of rapid change in popular methods. Either as a cause or, perhaps, an effect, such periods of change in automation techniques seem closely tied to world economics. Use of the **industrial robot**, which became identifiable as a unique device in the 1960s [1], along with computer-aided design (CAD) systems and computer-aided manufacturing (CAM) systems, characterizes the latest trends in the automation of the manufacturing process. These technologies are leading industrial automation through another transition, the scope of which is still unknown [2].

In North America, there was much adoption of robotic equipment in the early 1980s, followed by a brief pull-back in the late 1980s. Since that time, the market has been growing (Fig. 1.1), although it is subject to economic swings, as are all markets.

Figure 1.2 shows the number of robots being installed per year in the major industrial regions of the world. Note that Japan reports numbers somewhat differently from the way that other regions do: they count some machines as robots that in other parts of the world are not considered robots (rather, they would be simply considered "factory machines"). Hence, the numbers reported for Japan are somewhat inflated.

A major reason for the growth in the use of industrial robots is their declining cost. Figure 1.3 indicates that, through the decade of the 1990s, robot prices dropped while human labor costs increased. Also, robots are not just getting cheaper, they are becoming more effective—faster, more accurate, more flexible. If we factor these *quality adjustments* into the numbers, the cost of using robots is dropping even faster than their price tag is. As robots become more cost effective at their jobs, and as human labor continues to become more expensive, more and more industrial jobs become candidates for robotic automation. This is the single most important trend propelling growth of the industrial robot market. A secondary trend is that, economics aside, as robots become more capable they become *able* to do more and more tasks that might be dangerous or impossible for human workers to perform.

The applications that industrial robots perform are gradually getting more sophisticated, but it is still the case that, in the year 2000, approximately 78% of the robots installed in the US were welding or material-handling robots [3].

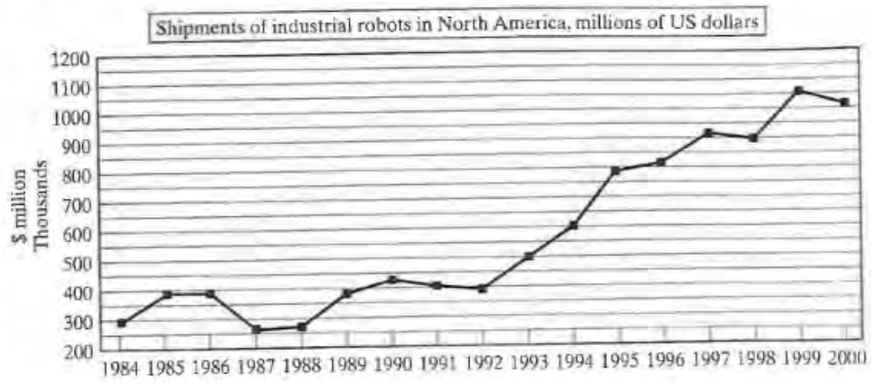


FIGURE 1.1: Shipments of industrial robots in North America in millions of US dollars [3].

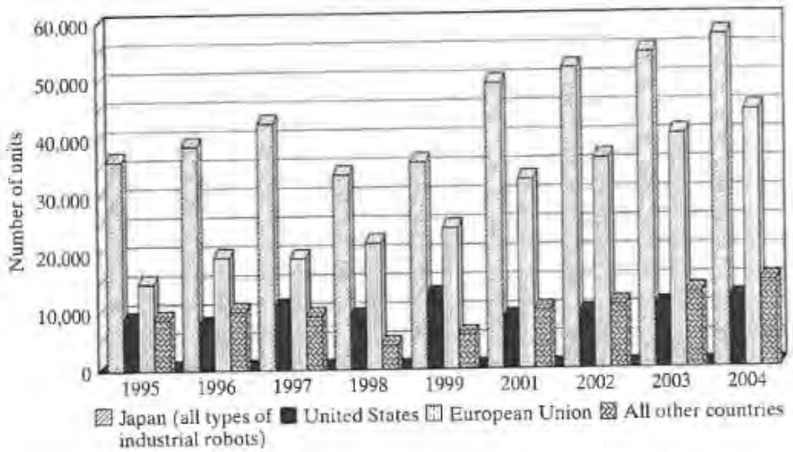


FIGURE 1.2: Yearly installations of multipurpose industrial robots for 1995–2000 and forecasts for 2001–2004 [3].

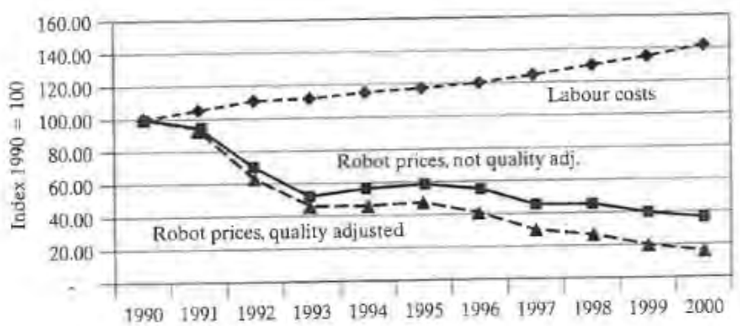


FIGURE 1.3: Robot prices compared with human labor costs in the 1990s [3].

FIGURE 1.4: The Adept 6 manipulator has six rotational joints and is popular in many applications. Courtesy of Adept Technology, Inc.

A more challenging domain, **assembly** by industrial robot, accounted for 10% of installations.

This book focuses on the mechanics and control of the most important form of the industrial robot, the **mechanical manipulator**. Exactly what constitutes an industrial robot is sometimes debated. Devices such as that shown in Fig. 1.4 are always included, while numerically controlled (NC) milling machines are usually not. The distinction lies somewhere in the sophistication of the programmability of the device—if a mechanical device can be programmed to perform a wide variety of applications, it is probably an industrial robot. Machines which are for the most part limited to one class of task are considered **fixed automation**. For the purposes of this text, the distinctions need not be debated; most material is of a basic nature that applies to a wide variety of programmable machines.

By and large, the study of the mechanics and control of manipulators is not a new science, but merely a collection of topics taken from "classical" fields. Mechanical engineering contributes methodologies for the study of machines in static and dynamic situations. Mathematics supplies tools for describing spatial motions and other attributes of manipulators. Control theory provides tools for designing and evaluating algorithms to realize desired motions or force applications. Electrical-engineering techniques are brought to bear in the design of sensors and interfaces for industrial robots, and computer science contributes a basis for programming these devices to perform a desired task.

1.2 THE MECHANICS AND CONTROL OF MECHANICAL MANIPULATORS

The following sections introduce some terminology and briefly preview each of the topics that will be covered in the text.

Description of position and orientation

In the study of robotics, we are constantly concerned with the location of objects in three-dimensional space. These objects are the links of the manipulator, the parts and tools with which it deals, and other objects in the manipulator's environment. At a crude but important level, these objects are described by just two attributes: position and orientation. Naturally, one topic of immediate interest is the manner in which we represent these quantities and manipulate them mathematically.

In order to describe the position and orientation of a body in space, we will always attach a coordinate system, or **frame**, rigidly to the object. We then proceed to describe the position and orientation of this frame with respect to some reference coordinate system. (See Fig. 1.5.)

Any frame can serve as a reference system within which to express the position and orientation of a body, so we often think of *transforming* or *changing the description of* these attributes of a body from one frame to another. Chapter 2 discusses conventions and methodologies for dealing with the description of position and orientation and the mathematics of manipulating these quantities with respect to various coordinate systems.

Developing good skills concerning the description of position and rotation of rigid bodies is highly useful even in fields outside of robotics.

Forward kinematics of manipulators

Kinematics is the science of motion that treats motion without regard to the forces which cause it. Within the science of kinematics, one studies position, velocity,

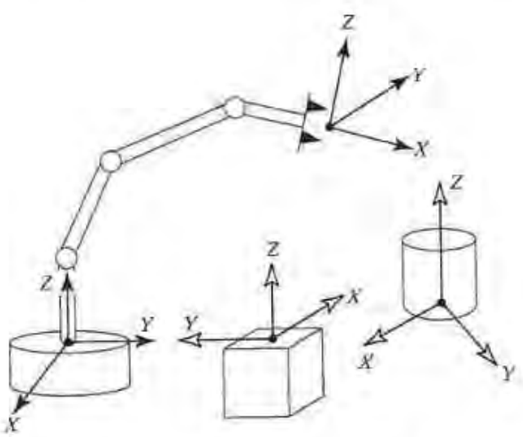


FIGURE 1.5: Coordinate systems or "frames" are attached to the manipulator and to objects in the environment.

acceleration, and all higher order derivatives of the position variables (with respect to time or any other variable(s)). Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion.

Manipulators consist of nearly rigid **links**, which are connected by **joints** that allow relative motion of neighboring links. These joints are usually instrumented with position sensors, which allow the relative position of neighboring links to be measured. In the case of rotary or **revolute** joints, these displacements are called **joint angles**. Some manipulators contain sliding (or **prismatic**) joints, in which the relative displacement between links is a translation, sometimes called the **joint offset**.

The number of **degrees of freedom** that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. This is a general term used for any mechanism. For example, a four-bar linkage has only one degree of freedom (even though there are three moving members). In the case of typical industrial robots, because a manipulator is usually an open kinematic chain, and because each joint position is usually defined with a single variable, the number of joints equals the number of degrees of freedom.

At the free end of the chain of links that make up the manipulator is the **end-effector**. Depending on the intended application of the robot, the end-effector could be a gripper, a welding torch, an electromagnet, or another device. We generally describe the position of the manipulator by giving a description of the **tool frame**, which is attached to the end-effector, relative to the **base frame**, which is attached to the nonmoving base of the manipulator. (See Fig. 1.6.)

A very basic problem in the study of mechanical manipulation is called **forward kinematics**. This is the static geometrical problem of computing the position and orientation of the end-effector of the manipulator. Specifically, given a set of joint

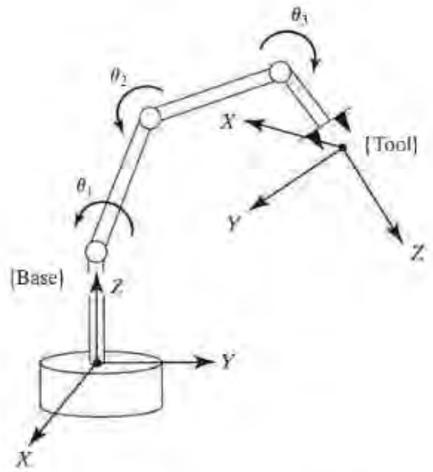


FIGURE 1.6: Kinematic equations describe the tool frame relative to the base frame as a function of the joint variables.

angles, the forward kinematic problem is to compute the position and orientation of the tool frame relative to the base frame. Sometimes, we think of this as changing the representation of manipulator position from a **joint space** description into a **Cartesian space** description.[1] This problem will be explored in Chapter 3.

Inverse kinematics of manipulators

In Chapter 4, we will consider the problem of **inverse kinematics.** This problem is posed as follows: Given the position and orientation of the end-effector of the manipulator, calculate all possible sets of joint angles that could be used to attain this given position and orientation. (See Fig. 1.7.) This is a fundamental problem in the practical use of manipulators.

This is a rather complicated geometrical problem that is routinely solved thousands of times daily in human and other biological systems. In the case of an artificial system like a robot, we will need to create an algorithm in the control computer that can make this calculation. In some ways, solution of this problem is the most important element in a manipulator system.

We can think of this problem as a *mapping* of "locations" in 3-D Cartesian space to "locations" in the robot's internal joint space. This need naturally arises anytime a goal is specified in external 3-D space coordinates. Some early robots lacked this algorithm—they were simply moved (sometimes by hand) to desired locations, which were then recorded as a set of joint values (i.e., as a location in joint space) for later playback. Obviously, if the robot is used purely in the mode of recording and playback of joint locations and motions, no algorithm relating

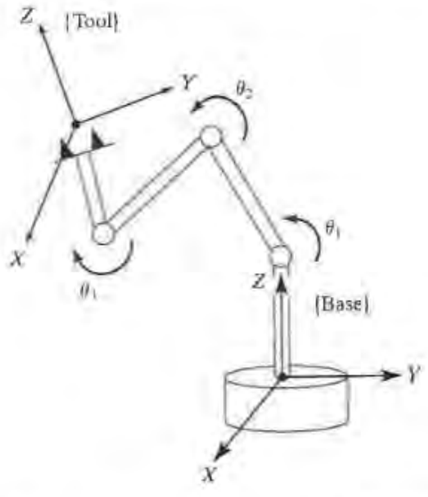


FIGURE 1.7: For a given position and orientation of the tool frame, values for the joint variables can be calculated via the inverse kinematics.

[1] By *Cartesian space* we mean the space in which the position of a point is given with three numbers, and in which the orientation of a body is given with three numbers. It is sometimes called *task space* or *operational space*.

joint space to Cartesian space is needed. These days, however, it is rare to find an industrial robot that lacks this basic inverse kinematic algorithm.

The inverse kinematics problem is not as simple as the forward kinematics one. Because the kinematic equations are nonlinear, their solution is not always easy (or even possible) in a closed form. Also, questions about the existence of a solution and about multiple solutions arise.

Study of these issues gives one an appreciation for what the human mind and nervous system are accomplishing when we, seemingly without conscious thought, move and manipulate objects with our arms and hands.

The existence or nonexistence of a kinematic solution defines the **workspace** of a given manipulator. The lack of a solution means that the manipulator cannot attain the desired position and orientation because it lies outside of the manipulator's workspace.

Velocities, static forces, singularities

In addition to dealing with static positioning problems, we may wish to analyze manipulators in motion. Often, in performing velocity analysis of a mechanism, it is convenient to define a matrix quantity called the **Jacobian** of the manipulator. The Jacobian specifies a **mapping** from velocities in joint space to velocities in Cartesian space. (See Fig. 1.8.) The nature of this mapping changes as the configuration of the manipulator varies. At certain points, called **singularities**, this mapping is not invertible. An understanding of the phenomenon is important to designers and users of manipulators.

Consider the rear gunner in a World War I–vintage biplane fighter plane (illustrated in Fig. 1.9). While the pilot flies the plane from the front cockpit, the rear gunner's job is to shoot at enemy aircraft. To perform this task, his gun is mounted in a mechanism that rotates about two axes, the motions being called azimuth and elevation. Using these two motions (two degrees of freedom), the gunner can direct his stream of bullets in any direction he desires in the upper hemisphere.

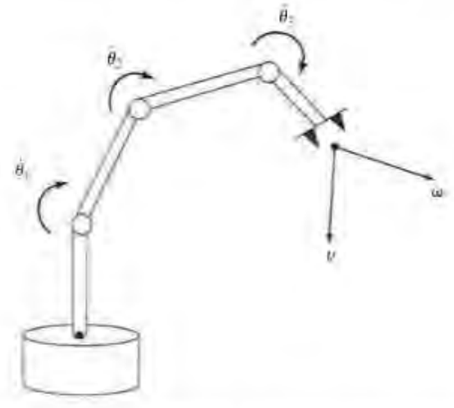


FIGURE 1.8: The geometrical relationship between joint rates and velocity of the end-effector can be described in a matrix called the Jacobian.

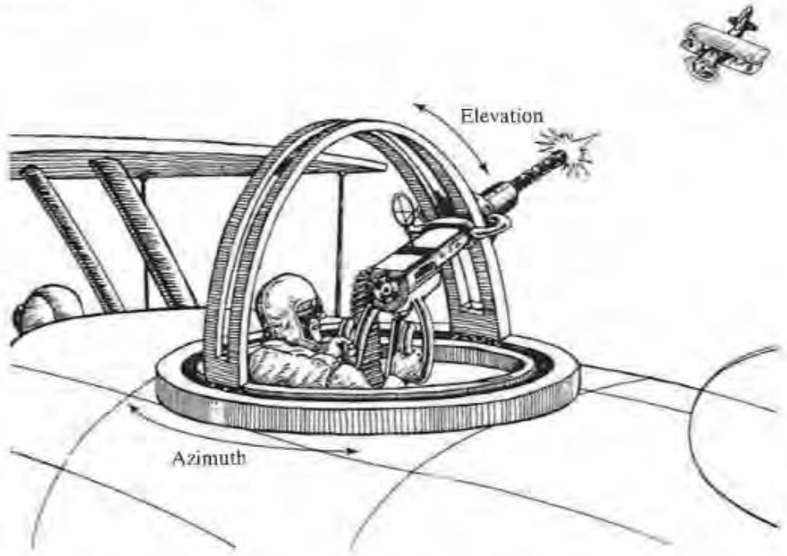


FIGURE 1.9: A World War I biplane with a pilot and a rear gunner. The rear-gunner mechanism is subject to the problem of singular positions.

An enemy plane is spotted at azimuth one o'clock and elevation 25 degrees! The gunner trains his stream of bullets on the enemy plane and tracks its motion so as to hit it with a continuous stream of bullets for as long as possible. He succeeds and thereby downs the enemy aircraft.

A second enemy plane is seen at azimuth one o'clock and elevation 70 degrees! The gunner orients his gun and begins firing. The enemy plane is moving so as to obtain a higher and higher elevation relative to the gunner's plane. Soon the enemy plane is passing nearly overhead. What's this? The gunner is no longer able to keep his stream of bullets trained on the enemy plane! He found that, as the enemy plane flew overhead, he was required to change his azimuth at a very high rate. He was not able to swing his gun in azimuth quickly enough, and the enemy plane escaped!

In the latter scenario, the lucky enemy pilot was saved by a *singularity*! The gun's orienting mechanism, while working well over most of its operating range, becomes less than ideal when the gun is directed straight upwards or nearly so. To track targets that pass through the position directly overhead, a very fast motion around the azimuth axis is required. The closer the target passes to the point directly overhead, the faster the gunner must turn the azimuth axis to track the target. If the target flies directly over the gunner's head, he would have to spin the gun on its azimuth axis at infinite speed!

Should the gunner complain to the mechanism designer about this problem? Could a better mechanism be designed to avoid this problem? It turns out that you really can't avoid the problem very easily. In fact, any two-degree-of-freedom orienting mechanism that has exactly two rotational joints cannot avoid having this problem. In the case of this mechanism, with the stream of bullets directed

straight up, their direction aligns with the axis of rotation of the azimuth rotation. This means that, at exactly this point, the azimuth rotation does not cause a change in the direction of the stream of bullets. We know we need two degrees of freedom to orient the stream of bullets, but, at this point, we have lost the effective use of one of the joints. Our mechanism has become **locally degenerate** at this location and behaves as if it only has one degree of freedom (the elevation direction).

This kind of phenomenon is caused by what is called a **singularity of the mechanism**. All mechanisms are prone to these difficulties, including robots. Just as with the rear gunner's mechanism, these singularity conditions do not prevent a robot arm from positioning anywhere within its workspace. However, they can cause problems with *motions* of the arm in their neighborhood.

Manipulators do not always move through space; sometimes they are also required to touch a workpiece or work surface and apply a static force. In this case the problem arises: Given a desired contact force and moment, what set of **joint torques** is required to generate them? Once again, the Jacobian matrix of the manipulator arises quite naturally in the solution of this problem.

Dynamics

Dynamics is a huge field of study devoted to studying the forces required to cause motion. In order to accelerate a manipulator from rest, glide at a constant end-effector velocity, and finally decelerate to a stop, a complex set of torque functions must be applied by the joint actuators.[2] The exact form of the required functions of actuator torque depend on the spatial and temporal attributes of the path taken by the end-effector and on the mass properties of the links and payload, friction in the joints, and so on. One method of controlling a manipulator to follow a desired path involves calculating these actuator torque functions by using the dynamic equations of motion of the manipulator.

Many of us have experienced lifting an object that is actually much lighter than we expected (e.g., getting a container of milk from the refrigerator which we thought was full, but was nearly empty). Such a misjudgment of payload can cause an unusual lifting motion. This kind of observation indicates that the human control system is more sophisticated than a purely kinematic scheme. Rather, our manipulation control system makes use of knowledge of mass and other dynamic effects. Likewise, algorithms that we construct to control the motions of a robot manipulator should take dynamics into account.

A second use of the dynamic equations of motion is in **simulation**. By reformulating the dynamic equations so that acceleration is computed as a function of actuator torque, it is possible to simulate how a manipulator would move under application of a set of actuator torques. (See Fig. 1.10.) As computing power becomes more and more cost effective, the use of simulations is growing in use and importance in many fields.

In Chapter 6, we develop dynamic equations of motion, which may be used to control or simulate the motion of manipulators.

[2]We use *joint actuators* as the generic term for devices that power a manipulator—for example, electric motors, hydraulic and pneumatic actuators, and muscles.

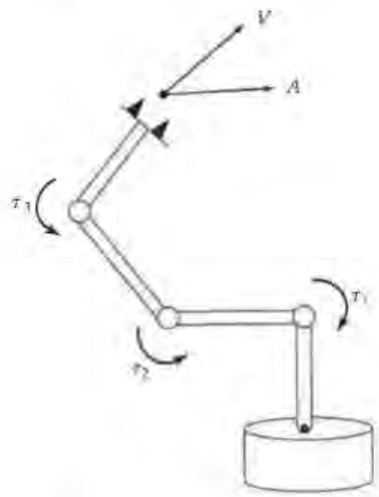


FIGURE 1.10: The relationship between the torques applied by the actuators and the resulting motion of the manipulator is embodied in the dynamic equations of motion.

Trajectory generation

A common way of causing a manipulator to move from here to there in a smooth, controlled fashion is to cause each joint to move as specified by a smooth function of time. Commonly, each joint starts and ends its motion at the same time, so that the manipulator motion appears coordinated. Exactly how to compute these motion functions is the problem of **trajectory generation**. (See Fig. 1.11.)

Often, a path is described not only by a desired destination but also by some intermediate locations, or **via points**, through which the manipulator must pass en route to the destination. In such instances the term **spline** is sometimes used to refer to a smooth function that passes through a set of via points.

In order to force the end-effector to follow a straight line (or other geometric shape) through space, the desired motion must be converted to an equivalent set of joint motions. This **Cartesian trajectory generation** will also be considered in Chapter 7.

Manipulator design and sensors

Although manipulators are, in theory, universal devices applicable to many situations, economics generally dictates that the intended task domain influence the mechanical design of the manipulator. Along with issues such as size, speed, and load capability, the designer must also consider the number of joints and their geometric arrangement. These considerations affect the manipulator's workspace size and quality, the stiffness of the manipulator structure, and other attributes.

The more joints a robot arm contains, the more dextrous and capable it will be. Of course, it will also be harder to build and more expensive. In order to build

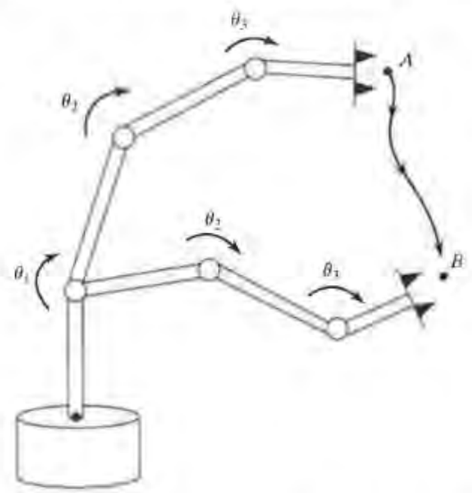


FIGURE 1.11: In order to move the end-effector through space from point A to point B, we must compute a trajectory for each joint to follow.

a useful robot, that can take two approaches: build a **specialized robot** for a specific task, or build a **universal robot** that would able to perform a wide variety of tasks. In the case of a specialized robot, some careful thinking will yield a solution for how many joints are needed. For example, a specialized robot designed solely to place electronic components on a flat circuit board does not need to have more than four joints. Three joints allow the position of the hand to attain any position in three-dimensional space, with a fourth joint added to allow the hand to rotate the grasped component about a vertical axis. In the case of a universal robot, it is interesting that fundamental properties of the physical world we live in dictate the "correct" minimum number of joints—that minimum number is six.

Integral to the design of the manipulator are issues involving the choice and location of actuators, transmission systems, and internal-position (and sometimes force) sensors. (See Fig. 1.12.) These and other design issues will be discussed in Chapter 8.

Linear position control

Some manipulators are equipped with stepper motors or other actuators that can execute a desired trajectory directly. However, the vast majority of manipulators are driven by actuators that supply a force or a torque to cause motion of the links. In this case, an algorithm is needed to compute torques that will cause the desired motion. The problem of dynamics is central to the design of such algorithms, but does not in itself constitute a solution. A primary concern of a **position control system** is to compensate automatically for errors in knowledge of the parameters of a system and to suppress disturbances that tend to perturb the system from the desired trajectory. To accomplish this, position and velocity **sensors** are monitored by the **control algorithm**, which computes torque commands for the actuators. (See

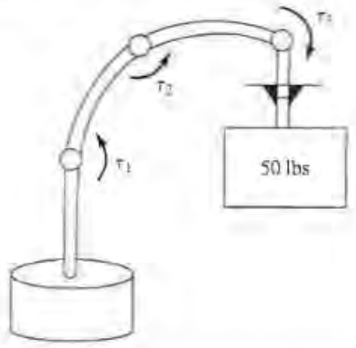


FIGURE 1.12: The design of a mechanical manipulator must address issues of actuator choice, location, transmission system, structural stiffness, sensor location, and more.

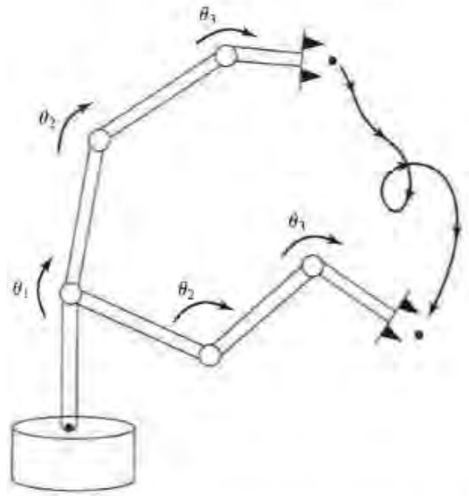


FIGURE 1.13: In order to cause the manipulator to follow the desired trajectory, a position-control system must be implemented. Such a system uses feedback from joint sensors to keep the manipulator on course.

Fig. 1.13.) In Chapter 9, we will consider control algorithms whose synthesis is based on linear approximations to the dynamics of a manipulator. These linear methods are prevalent in current industrial practice.

Nonlinear position control

Although control systems based on approximate linear models are popular in current industrial robots, it is important to consider the complete nonlinear dynamics of the manipulator when synthesizing control algorithms. Some industrial robots are now being introduced which make use of **nonlinear control** algorithms in their

controllers. These nonlinear techniques of controlling a manipulator promise better performance than do simpler linear schemes. Chapter 10 will introduce nonlinear control systems for mechanical manipulators.

Force control

The ability of a manipulator to control forces of contact when it touches parts, tools, or work surfaces seems to be of great importance in applying manipulators to many real-world tasks. **Force control** is complementary to position control, in that we usually think of only one or the other as applicable in a certain situation. When a manipulator is moving in free space, only position control makes sense, because there is no surface to react against. When a manipulator is touching a rigid surface, however, position-control schemes can cause excessive forces to build up at the contact or cause contact to be lost with the surface when it was desired for some application. Manipulators are rarely constrained by reaction surfaces in all directions simultaneously, so a mixed or **hybrid** control is required, with some directions controlled by a **position-control law** and remaining directions controlled by a **force-control law**. (See Fig. 1.14.) Chapter 11 introduces a methodology for implementing such a force-control scheme.

A robot should be instructed to wash a window by maintaining a certain force in the direction perpendicular to the plane of the glass, while following a motion trajectory in directions tangent to the plane. Such split or **hybrid** control specifications are natural for such tasks.

Programming robots

A robot programming language serves as the interface between the human user and the industrial robot. Central questions arise: How are motions through space described easily by the programmer? How are multiple manipulators programmed

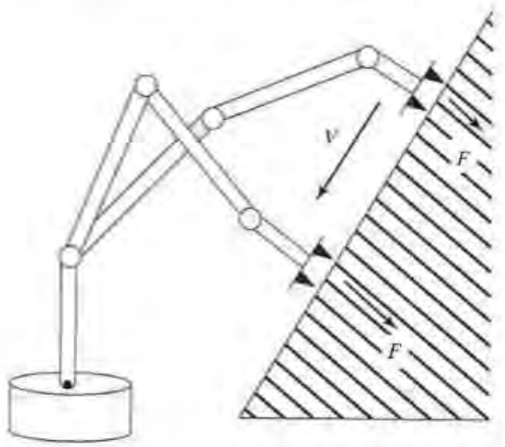


FIGURE 1.14: In order for a manipulator to slide across a surface while applying a constant force, a hybrid position–force control system must be used.

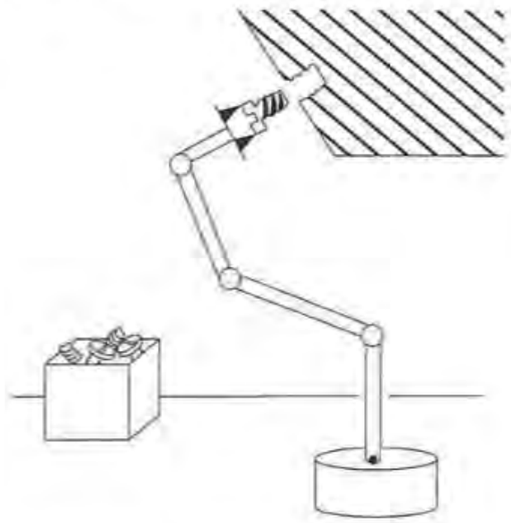


FIGURE 1.15: Desired motions of the manipulator and end-effector, desired contact forces, and complex manipulation strategies can be described in a *robot programming language*.

so that they can work in parallel? How are sensor-based actions described in a language?

Robot manipulators differentiate themselves from **fixed automation** by being "flexible," which means programmable. Not only are the movements of manipulators programmable, but, through the use of sensors and communications with other factory automation, manipulators can *adapt* to variations as the task proceeds. (See Fig. 1.15.)

In typical robot systems, there is a shorthand way for a human user to instruct the robot which path it is to follow. First of all, a special point on the hand (or perhaps on a grasped tool) is specified by the user as the **operational point**, sometimes also called the **TCP** (for Tool Center Point). Motions of the robot will be described by the user in terms of desired locations of the operational point relative to a user-specified coordinate system. Generally, the user will define this reference coordinate system relative to the robot's base coordinate system in some task-relevant location.

Most often, paths are constructed by specifying a sequence of **via points**. Via points are specified relative to the reference coordinate system and denote locations along the path through which the TCP should pass. Along with specifying the via points, the user may also indicate that certain speeds of the TCP be used over various portions of the path. Sometimes, other modifiers can also be specified to affect the motion of the robot (e.g., different smoothness criteria, etc.). From these inputs, the trajectory-generation algorithm must plan all the details of the motion: velocity profiles for the joints, time duration of the move, and so on. Hence, input

to the trajectory-generation problem is generally given by constructs in the robot programming language.

The sophistication of the user interface is becoming extremely important as manipulators and other programmable automation are applied to more and more demanding industrial applications. The problem of programming manipulators encompasses all the issues of "traditional" computer programming and so is an extensive subject in itself. Additionally, some particular attributes of the manipulator-programming problem cause additional issues to arise. Some of these topics will be discussed in Chapter 12.

Off-line programming and simulation

An **off-line programming system** is a robot programming environment that has been sufficiently extended, generally by means of computer graphics, that the development of robot programs can take place without access to the robot itself. A common argument raised in their favor is that an off-line programming system will not cause production equipment (i.e., the robot) to be tied up when it needs to be reprogrammed; hence, automated factories can stay in production mode a greater percentage of the time. (See Fig. 1.16.)

They also serve as a natural vehicle to tie computer-aided design (CAD) data bases used in the design phase of a product to the actual manufacturing of the product. In some cases, this direct use of CAD data can dramatically reduce the programming time required for the manufacturing process. Chapter 13 discusses the elements of industrial robot off-line programming systems.

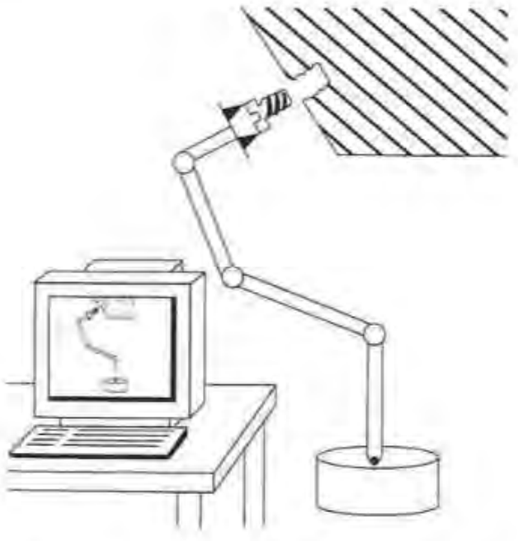


FIGURE 1.16: Off-line programming systems, generally providing a computer graphics interface, allow robots to be programmed without access to the robot itself during programming.

1.3 NOTATION

Notation is always an issue in science and engineering. In this book, we use the following conventions:

1. Usually, variables written in uppercase represent vectors or matrices. Lowercase variables are scalars.
2. Leading subscripts and superscripts identify which coordinate system a quantity is written in. For example, ${}^{A}P$ represents a position vector written in coordinate system $\{A\}$, and ${}^{A}_{B}R$ is a rotation matrix[3] that specifies the relationship between coordinate systems $\{A\}$ and $\{B\}$.
3. Trailing superscripts are used (as widely accepted) for indicating the inverse or transpose of a matrix (e.g., R^{-1}, R^{T}).
4. Trailing subscripts are not subject to any strict convention but may indicate a vector component (e.g., x, y, or z) or may be used as a description—as in P_{bolt}, the position of a bolt.
5. We will use many trigonometric functions. Our notation for the cosine of an angle θ_1 may take any of the following forms: $\cos\theta_1 = c\theta_1 = c_1$.

Vectors are taken to be column vectors; hence, row vectors will have the transpose indicated explicitly.

A note on vector notation in general: Many mechanics texts treat vector quantities at a very abstract level and routinely use vectors defined relative to different coordinate systems in expressions. The clearest example is that of addition of vectors which are given or known relative to differing reference systems. This is often very convenient and leads to compact and somewhat elegant formulas. For example, consider the angular velocity, ${}^{0}\omega_4$, of the last body in a series connection of four rigid bodies (as in the links of a manipulator) relative to the fixed base of the chain. Because angular velocities sum vectorially, we may write a very simple vector equation for the angular velocity of the final link:

$${}^{0}\omega_4 = {}^{0}\omega_1 + {}^{1}\omega_2 + {}^{2}\omega_3 + {}^{3}\omega_4. \tag{1.1}$$

However, unless these quantities are expressed with respect to a common coordinate system, they cannot be summed, and so, though elegant, equation (1.1) has hidden much of the "work" of the computation. For the particular case of the study of mechanical manipulators, statements like that of (1.1) hide the chore of bookkeeping of coordinate systems, which is often the very idea that we need to deal with in practice.

Therefore, in this book, we carry frame-of-reference information in the notation for vectors, and we do not sum vectors unless they are in the same coordinate system. In this way, we derive expressions that solve the "bookkeeping" problem and can be applied directly to actual numerical computation.

BIBLIOGRAPHY

[1] B. Roth, "Principles of Automation," Future Directions in Manufacturing Technology, Based on the Unilever Research and Engineering Division Symposium held at Port Sunlight, April 1983, Published by Unilever Research, UK.

[3] This term will be introduced in Chapter 2.

[2] R. Brooks, "Flesh and Machines," Pantheon Books, New York, 2002.

[3] The International Federation of Robotics, and the United Nations, "World Robotics 2001," Statistics, Market Analysis, Forecasts, Case Studies and Profitability of Robot Investment, United Nations Publication, New York and Geneva, 2001.

General-reference books

[4] R. Paul, *Robot Manipulators*, MIT Press, Cambridge, MA, 1981.

[5] M. Brady et al., *Robot Motion*, MIT Press, Cambridge, MA, 1983.

[6] W. Synder, *Industrial Robots: Computer Interfacing and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[7] Y. Koren, *Robotics for Engineers*, McGraw-Hill, New York, 1985.

[8] H. Asada and J.J. Slotine, *Robot Analysis and Control*, Wiley, New York, 1986.

[9] K. Fu, R. Gonzalez, and C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, New York, 1987.

[10] E. Riven, *Mechanical Design of Robots*, McGraw-Hill, New York, 1988.

[11] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[12] M. Spong, *Robot Control: Dynamics, Motion Planning, and Analysis*, IEEE Press, New York, 1992.

[13] S.Y. Nof, *Handbook of Industrial Robotics*, 2nd Edition, Wiley, New York, 1999.

[14] L.W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*, Wiley, New York, 1999.

[15] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd Edition, Springer-Verlag, London, 2000.

[16] G. Schmierer and R. Schraft, *Service Robots*, A.K. Peters, Natick, MA, 2000.

General-reference journals and magazines

[17] *Robotics World*.

[18] *IEEE Transactions on Robotics and Automation*.

[19] *International Journal of Robotics Research (MIT Press)*.

[20] *ASME Journal of Dynamic Systems, Measurement, and Control*.

[21] *International Journal of Robotics & Automation (IASTED)*.

EXERCISES

1.1 [20] Make a chronology of major events in the development of industrial robots over the past 40 years. See Bibliography and general references.

1.2 [20] Make a chart showing the major applications of industrial robots (e.g., spot welding, assembly, etc.) and the percentage of installed robots in use in each application area. Base your chart on the most recent data you can find. See Bibliography and general references.

1.3 [40] Figure 1.3 shows how the cost of industrial robots has declined over the years. Find data on the cost of human labor in various specific industries (e.g., labor in the auto industry, labor in the electronics assembly industry, labor in agriculture, etc.) and create a graph showing how these costs compare to the use of robotics. You should see that the robot cost curve "crosses" various the human cost curves

of different industries at different times. From this, derive approximate dates when robotics first became cost effective for use in various industries.

1.4 [10] In a sentence or two, define kinematics, workspace, and trajectory.

1.5 [10] In a sentence or two, define frame, degree of freedom, and position control.

1.6 [10] In a sentence or two, define force control, and robot programming language.

1.7 [10] In a sentence or two, define nonlinear control, and off-line programming.

1.8 [20] Make a chart indicating how labor costs have risen over the past 20 years.

1.9 [20] Make a chart indicating how the computer performance–price ratio has increased over the past 20 years.

1.10 [20] Make a chart showing the major users of industrial robots (e.g., aerospace, automotive, etc.) and the percentage of installed robots in use in each industry. Base your chart on the most recent data you can find. (See reference section.)

PROGRAMMING EXERCISE (PART 1)

Familiarize yourself with the computer you will use to do the programming exercises at the end of each chapter. Make sure you can create and edit files and can compile and execute programs.

MATLAB EXERCISE 1

At the end of most chapters in this textbook, a MATLAB exercise is given. Generally, these exercises ask the student to program the pertinent robotics mathematics in MATLAB and then check the results of the MATLAB Robotics Toolbox. The textbook assumes familiarity with MATLAB and linear algebra (matrix theory). Also, the student must become familiar with the MATLAB Robotics Toolbox. For MATLAB Exercise 1,

a) Familiarize yourself with the MATLAB programming environment if necessary. At the MATLAB software prompt, try typing *demo* and *help*. Using the color-coded MATLAB editor, learn how to create, edit, save, run, and debug m-files (ASCII files with series of MATLAB statements). Learn how to create arrays (matrices and vectors), and explore the built-in MATLAB linear-algebra functions for matrix and vector multiplication, dot and cross products, transposes, determinants, and inverses, and for the solution of linear equations. MATLAB is based on the language C, but is generally much easier to use. Learn how to program logical constructs and loops in MATLAB. Learn how to use subprograms and functions. Learn how to use comments (%) for explaining your programs and tabs for easy readability. Check out www.mathworks.com for more information and tutorials. Advanced MATLAB users should become familiar with Simulink, the graphical interface of MATLAB, and with the MATLAB Symbolic Toolbox.

b) Familiarize yourself with the MATLAB Robotics Toolbox, a third-party toolbox developed by Peter I. Corke of CSIRO, Pinjarra Hills, Australia. This product can be downloaded for free from www.cat.csiro.au/cmst/staff/pic/robot. The source code is readable and changeable, and there is an international community of users, at robot-toolbox@lists.msa.cmst.csiro.au. Download the MATLAB Robotics Toolbox, and install it on your computer by using the *.zip* file and following the instructions. Read the *README* file, and familiarize yourself with the various functions available to the user. Find the *robot.pdf* file—this is the user manual giving background information and detailed usage of all of the Toolbox functions. Don't worry if you can't understand the purpose of these functions yet; they deal with robotics mathematics concepts covered in Chapters 2 through 7 of this book.

CHAPTER 2

Spatial descriptions and transformations

2.1 INTRODUCTION

Robotic manipulation, by definition, implies that parts and tools will be moved around in space by some sort of mechanism. This naturally leads to a need for representing positions and orientations of parts, of tools, and of the mechanism itself. To define and manipulate mathematical quantities that represent position and orientation, we must define coordinate systems and develop conventions for representation. Many of the ideas developed here in the context of position and orientation will form a basis for our later consideration of linear and rotational velocities, forces, and torques.

We adopt the philosophy that somewhere there is a **universe coordinate system** to which everything we discuss can be referenced. We will describe all positions and orientations with respect to the universe coordinate system or with respect to other Cartesian coordinate systems that are (or could be) defined relative to the universe system.

2.2 DESCRIPTIONS: POSITIONS, ORIENTATIONS, AND FRAMES

A **description** is used to specify attributes of various objects with which a manipulation system deals. These objects are parts, tools, and the manipulator itself. In this section, we discuss the description of positions, of orientations, and of an entity that contains both of these descriptions: the frame.

2. Write a routine that calculates where the tool is, relative to the station frame. The input to the routine is a vector of joint angles:

```
Procedure WHERE(VAR theta: vec3; VAR trels: frame);
```

Obviously, WHERE must make use of descriptions of the tool frame and the robot base frame in order to compute the location of the tool relative to the station frame. The values of ${}^{W}_{T}T$ and ${}^{S}_{B}T$ should be stored in global memory (or, as a second choice, you may pass them as arguments in WHERE).

3. A tool frame and a station frame for a certain task are defined as follows by the user:

$$ {}^{W}_{T}T = [x\ y\ \theta] = [0.1\ 0.2\ 30.0]. $$

$$ {}^{B}_{S}T = [x\ y\ \theta] = [-0.1\ 0.3\ 0.0]. $$

Calculate the position and orientation of the tool relative to the station frame for the following three configurations (in units of degrees) of the arm:

$$ [\theta_1\ \theta_2\ \theta_3] = [0.0\ 90.0\ -90.0], $$

$$ [\theta_1\ \theta_2\ \theta_3] = [-23.6\ -30.3\ 48.0], $$

$$ [\theta_1\ \theta_2\ \theta_3] = [130.0\ 40.0\ 12.0]. $$

MATLAB EXERCISE 3

This exercise focuses on DH parameters and on the forward-pose (position and orientation) kinematics transformation for the planar 3-DOF, 3R robot (of Figures 3.6 and 3.7). The following fixed-length parameters are given: $L_1 = 4$, $L_2 = 3$, and $L_3 = 2$ (m).

a) Derive the DH parameters. You can check your results against Figure 3.8.

b) Derive the neighboring homogeneous transformation matrices ${}^{i-1}_{i}T$, $i = 1, 2, 3$. These are functions of the joint-angle variables θ_i, $i = 1, 2, 3$. Also, derive the constant ${}^{3}_{H}T$ by inspection: The origin of $\{H\}$ is in the center of the gripper fingers, and the orientation of $\{H\}$ is always the same as the orientation of $\{3\}$.

c) Use Symbolic MATLAB to derive the forward-pose kinematics solution ${}^{0}_{3}T$ and ${}^{0}_{H}T$ symbolically (as a function of θ_i). Abbreviate your answer, using $s_i = \sin(\theta_i)$, $c_i = \cos(\theta_i)$, and so on. Also, there is a $(\theta_1 + \theta_2 + \theta_3)$ simplification, by using sum-of-angle formulas, that is due to the parallel Z_i axes. Calculate the forward-pose kinematics results (both ${}^{0}_{3}T$ and ${}^{0}_{H}T$) via MATLAB for the following input cases:

i) $\Theta = \{\theta_1\ \ \theta_2\ \ \theta_3\}^T = \{0\ \ 0\ \ 0\}^T$.

ii) $\Theta = \{10^\circ\ \ 20^\circ\ \ 30^\circ\}^T$.

iii) $\Theta = \{90^\circ\ \ 90^\circ\ \ 90^\circ\}^T$.

For all three cases, check your results by sketching the manipulator configuration and deriving the forward-pose kinematics transformation by inspection. (Think of the definition of ${}^{0}_{H}T$ in terms of a rotation matrix and a position vector.) Include frames $\{H\}$, $\{3\}$, and $\{0\}$ in your sketches.

d) Check all your results by means of the Corke MATLAB Robotics Toolbox. Try functions *link*(), *robot*(), and *fkine*().

CHAPTER 4

Inverse manipulator kinematics

4.1 INTRODUCTION

In the last chapter, we considered the problem of computing the position and orientation of the tool relative to the user's workstation when given the joint angles of the manipulator. In this chapter, we investigate the more difficult converse problem: Given the desired position and orientation of the tool relative to the station, how do we compute the set of joint angles which will achieve this desired result? Whereas Chapter 3 focused on the **direct kinematics** of manipulators, here the focus is the **inverse kinematics** of manipulators.

Solving the problem of finding the required joint angles to place the tool frame, $\{T\}$, relative to the station frame, $\{S\}$, is split into two parts. First, frame transformations are performed to find the wrist frame, $\{W\}$, relative to the base frame, $\{B\}$, and then the inverse kinematics are used to solve for the joint angles.

4.2 SOLVABILITY

The problem of solving the kinematic equations of a manipulator is a nonlinear one. Given the numerical value of ${}^{0}_{N}T$, we attempt to find values of $\theta_1, \theta_2, \ldots, \theta_n$. Consider the equations given in (3.14). In the case of the PUMA 560 manipulator, the precise statement of our current problem is as follows: Given ${}^{0}_{6}T$ as sixteen numeric values (four of which are trivial), solve (3.14) for the six joint angles θ_1 through θ_6.

For the case of an arm with six degrees of freedom (like the one corresponding to the equations in (3.14)), we have 12 equations and six unknowns. However, among the 9 equations arising from the rotation-matrix portion of ${}^{0}_{6}T$, only 3 are independent. These, added to the 3 equations from the position-vector portion of ${}^{0}_{6}T$,

give 6 equations with six unknowns. These equations are nonlinear, transcendental equations, which can be quite difficult to solve. The equations of (3.14) are those of a robot that had very simple link parameters—many of the α_i were 0 or ± 90 degrees. Many link offsets and lengths were zero. It is easy to imagine that, for the case of a general mechanism with six degrees of freedom (with all link parameters nonzero) the kinematic equations would be much more complex than those of (3.14). As with any nonlinear set of equations, we must concern ourselves with the existence of solutions, with multiple solutions, and with the method of solution.

Existence of solutions

The question of whether any solution exists at all raises the question of the manipulator's **workspace**. Roughly speaking, workspace is that volume of space that the end-effector of the manipulator can reach. For a solution to exist, the specified goal point must lie within the workspace. Sometimes, it is useful to consider two definitions of workspace: **Dextrous workspace** is that volume of space that the robot end-effector can reach with all orientations. That is, at each point in the dextrous workspace, the end-effector can be arbitrarily oriented. The **reachable workspace** is that volume of space that the robot can reach in at least one orientation. Clearly, the dextrous workspace is a subset of the reachable workspace.

Consider the workspace of the two-link manipulator in Fig. 4.1. If $l_1 = l_2$, then the reachable workspace consists of a disc of radius $2l_1$. The dextrous workspace consists of only a single point, the origin. If $l_1 \neq l_2$, then there is no dextrous workspace, and the reachable workspace becomes a ring of outer radius $l_1 + l_2$ and inner radius $|l_1 - l_2|$. Inside the reachable workspace there are two possible orientations of the end-effector. On the boundaries of the workspace there is only one possible orientation.

These considerations of workspace for the two-link manipulator have assumed that all the joints can rotate 360 degrees. This is rarely true for actual mechanisms. When joint limits are a subset of the full 360 degrees, then the workspace is obviously correspondingly reduced, either in extent, or in the number of possible orientations

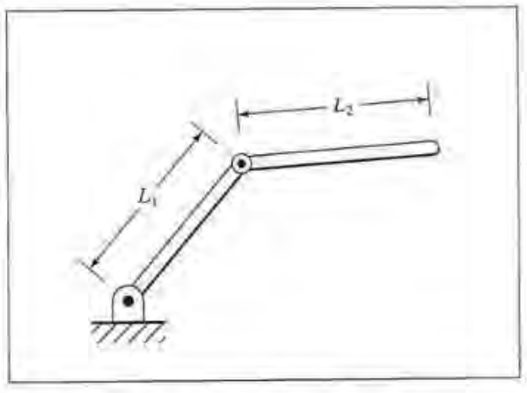


FIGURE 4.1: Two-link manipulator with link lengths l_1 and l_2.

attainable. For example, if the arm in Fig. 4.1 has full 360-degree motion for θ_1, but only $0 \leq \theta_2 \leq 180°$, then the reachable workspace has the same extent, but only one orientation is attainable at each point.

When a manipulator has fewer than six degrees of freedom, it cannot attain general goal positions and orientations in 3-space. Clearly, the planar manipulator in Fig. 4.1 cannot reach out of the plane, so any goal point with a nonzero Z-coordinate value can be quickly rejected as unreachable. In many realistic situations, manipulators with four or five degrees of freedom are employed that operate out of a plane, but that clearly cannot reach general goals. Each such manipulator must be studied to understand its workspace. In general, the workspace of such a robot is a subset of a subspace that can be associated with any particular robot. Given a general goal-frame specification, an interesting problem arises in connection with manipulators having fewer than six degrees of freedom: What is the nearest attainable goal frame?

Workspace also depends on the tool-frame transformation, because it is usually the tool-tip that is discussed when we speak of reachable points in space. Generally, the tool transformation is performed independently of the manipulator kinematics and inverse kinematics, so we are often led to consider the workspace of the wrist frame, $\{W\}$. For a given end-effector, a tool frame, $\{T\}$, is defined; given a goal frame, $\{G\}$, the corresponding $\{W\}$ frame is calculated, and then we ask: Does this desired position and orientation of $\{W\}$ lie in the workspace? In this way, the workspace that we must concern ourselves with (in a computational sense) is different from the one imagined by the user, who is concerned with the workspace of the end-effector (the $\{T\}$ frame).

If the desired position and orientation of the wrist frame is in the workspace, then at least one solution exists.

Multiple solutions

Another possible problem encountered in solving kinematic equations is that of multiple solutions. A planar arm with three revolute joints has a large dextrous workspace in the plane (given "good" link lengths and large joint ranges), because any position in the interior of its workspace can be reached with any orientation. Figure 4.2 shows a three-link planar arm with its end-effector at a certain position

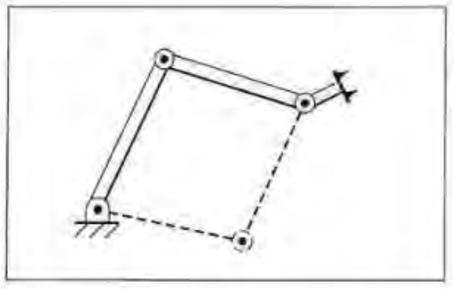


FIGURE 4.2: Three-link manipulator. Dashed lines indicate a second solution.

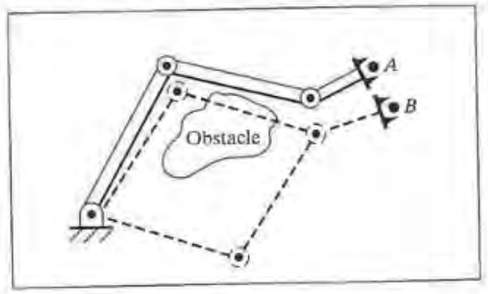


FIGURE 4.3: One of the two possible solutions to reach point B causes a collision.

and orientation. The dashed lines indicate a second possible configuration in which the same end-effector position and orientation are achieved.

The fact that a manipulator has multiple solutions can cause problems, because the system has to be able to choose one. The criteria upon which to base a decision vary, but a very reasonable choice would be the *closest* solution. For example, if the manipulator is at point A, as in Fig. 4.3, and we wish to move it to point B, a good choice would be the solution that minimizes the amount that each joint is required to move. Hence, in the absence of the obstacle, the upper dashed configuration in Fig. 4.3 would be chosen. This suggests that one input argument to our kinematic inverse procedure might be the present position of the manipulator. In this way, if there is a choice, our algorithm can choose the solution closest in joint-space. However, the notion of "close" might be defined in several ways. For example, typical robots could have three large links followed by three smaller, orienting links near the end-effector. In this case, weights might be applied in the calculation of which solution is "closer" so that the selection favors moving smaller joints rather than moving the large joints, when a choice exists. The presence of obstacles might force a "farther" solution to be chosen in cases where the "closer" solution would cause a collision—in general, then, we need to be able to calculate all the possible solutions. Thus, in Fig. 4.3, the presence of the obstacle implies that the lower dashed configuration is to be used to reach point B.

The number of solutions depends upon the number of joints in the manipulator but is also a function of the link parameters (α_i, a_i, and d_i for a rotary joint manipulator) and the allowable ranges of motion of the joints. For example, the PUMA 560 can reach certain goals with eight different solutions. Figure 4.4 shows four solutions; all place the hand with the same position and orientation. For each solution pictured, there is another solution in which the last three joints "flip" to an alternate configuration according to the following formulas:

$$\begin{aligned} \theta_4' &= \theta_4 + 180^\circ, \\ \theta_5' &= -\theta_5, \\ \theta_6' &= \theta_6 + 180^\circ. \end{aligned} \tag{4.1}$$

So, in total, there can be eight solutions for a single goal. Because of limits on joint ranges, some of these eight could be inaccessible.

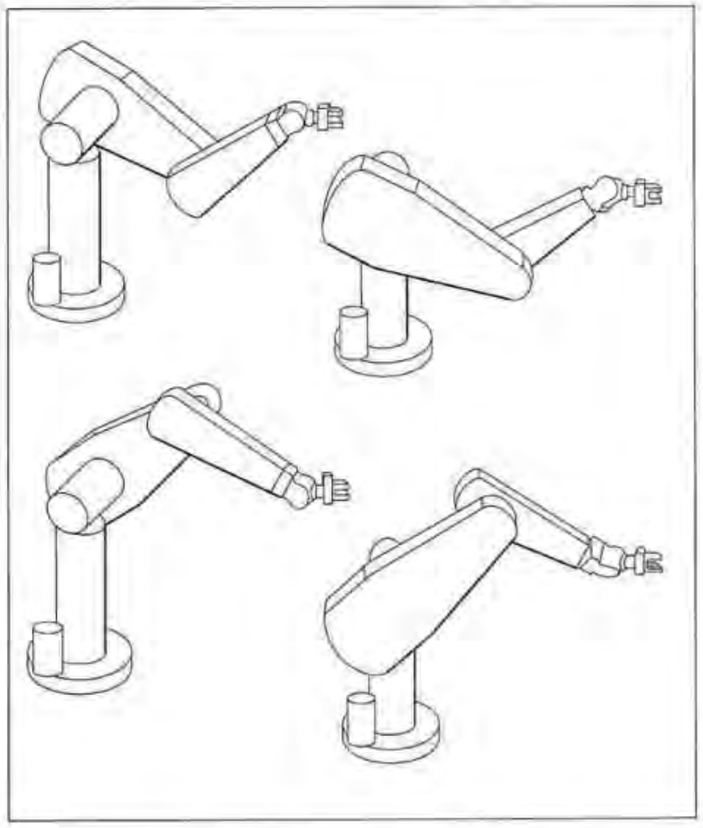


FIGURE 4.4: Four solutions of the PUMA 560.

In general, the more nonzero link parameters there are, the more ways there will be to reach a certain goal. For example, consider a manipulator with six rotational joints. Figure 4.5 shows how the maximum number of solutions is related to how many of the link length parameters (the a_i) are zero. The more that are nonzero, the bigger is the maximum number of solutions. For a completely general rotary-jointed manipulator with six degrees of freedom, there are up to sixteen solutions possible [1, 6].

Method of solution

Unlike linear equations, there are no general algorithms that may be employed to solve a set of nonlinear equations. In considering methods of solution, it will be wise to define what constitutes the "solution" of a given manipulator.

A manipulator will be considered solvable if the joint variables can be determined by an algorithm that allows one to determine *all* the sets of joint variables associated with a given position and orientation [2].

a_i	Number of solutions
$a_1 = a_3 = a_5 = 0$	≤ 4
$a_3 = a_5 = 0$	≤ 8
$a_5 = 0$	≤ 16
All $a_i \neq 0$	≤ 16

FIGURE 4.5: Number of solutions vs. nonzero a_i.

The main point of this definition is that we require, in the case of multiple solutions, that it be possible to calculate all solutions. Hence, we do not consider some numerical iterative procedures as solving the manipulator—namely, those methods not guaranteed to find all the solutions.

We will split all proposed manipulator solution strategies into two broad classes: **closed-form solutions** and **numerical solutions**. Because of their iterative nature, numerical solutions generally are much slower than the corresponding closed-form solution; so much so, in fact, that, for most uses, we are not interested in the numerical approach to solution of kinematics. Iterative numerical solution to kinematic equations is a whole field of study in itself (see [6,11,12]) and is beyond the scope of this text.

We will restrict our attention to closed-form solution methods. In this context, "closed form" means a solution method based on analytic expressions or on the solution of a polynomial of degree 4 or less, such that noniterative calculations suffice to arrive at a solution. Within the class of closed-form solutions, we distinguish two methods of obtaining the solution: **algebraic** and **geometric**. These distinctions are somewhat hazy: Any geometric methods brought to bear are applied by means of algebraic expressions, so the two methods are similar. The methods differ perhaps in approach only.

A major recent result in kinematics is that, according to our definition of solvability, *all systems with revolute and prismatic joints having a total of six degrees of freedom in a single series chain are solvable*. However, this general solution is a numerical one. Only in special cases can robots with six degrees of freedom be solved analytically. These robots for which an analytic (or closed-form) solution exists are characterized either by having several intersecting joint axes or by having many α_i equal to 0 or ± 90 degrees. Calculating numerical solutions is generally time consuming relative to evaluating analytic expressions; hence, it is considered very important to design a manipulator so that a closed-form solution exists. Manipulator designers discovered this very soon, and now virtually all industrial manipulators are designed sufficiently simply that a closed-form solution can be developed.

A sufficient condition that a manipulator with six revolute joints have a closed-form solution is that three neighboring joint axes intersect at a point. Section 4.6 discusses this condition. Almost every manipulator with six degrees of freedom built today has three axes intersecting. For example, axes 4, 5, and 6 of the PUMA 560 intersect.

4.3 THE NOTION OF MANIPULATOR SUBSPACE WHEN $n < 6$

The set of reachable goal frames for a given manipulator constitutes its reachable workspace. For a manipulator with n degrees of freedom (where $n < 6$), this reachable workspace can be thought of as a portion of an n-degree-of-freedom **subspace**. In the same manner in which the workspace of a six-degree-of-freedom manipulator is a subset of space, the workspace of a simpler manipulator is a subset of its subspace. For example, the subspace of the two-link robot of Fig. 4.1 is a plane, but the workspace is a subset of this plane, namely a circle of radius $l_1 + l_2$ for the case that $l_1 = l_2$.

One way to specify the subspace of an n-degree-of-freedom manipulator is to give an expression for its wrist or tool frame as a function of n variables that locate it. If we consider these n variables to be free, then, as they take on all possible values, the subspace is generated.

EXAMPLE 4.1

Give a description of the subspace of ${}^B_W T$ for the three-link manipulator from Chapter 3, Fig. 3.6.

The subspace of ${}^B_W T$ is given by

$$ {}^B_W T = \begin{bmatrix} c_\phi & -s_\phi & 0.0 & x \\ s_\phi & c_\phi & 0.0 & y \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.2} $$

where x and y give the position of the wrist and ϕ describes the orientation of the terminal link. As x, y, and ϕ are allowed to take on arbitrary values, the subspace is generated. Any wrist frame that does not have the structure of (4.2) lies outside the subspace (and therefore lies outside the workspace) of this manipulator. Link lengths and joint limits restrict the workspace of the manipulator to be a subset of this subspace.

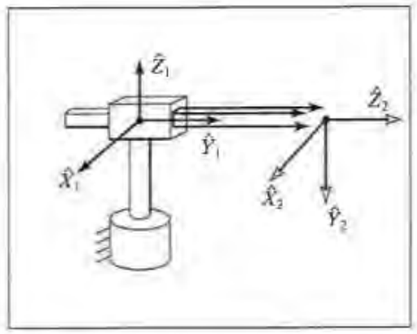


FIGURE 4.6: A polar two-link manipulator.

EXAMPLE 4.2

Give a description of the subspace of ${}^0_2T$ for the polar manipulator with two degrees of freedom shown in Fig. 4.6. We have

$$ {}^0P_{2ORG} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}, \tag{4.3} $$

where x and y can take any values. The orientation is restricted because the ${}^0\hat{Z}_2$ axis must point in a direction that depends on x and y. The ${}^0\hat{Y}_2$ axis always points down, and the ${}^0\hat{X}_2$ axis can be computed as the cross product ${}^0\hat{Y}_2 \times {}^0\hat{Z}_2$. In terms of x and y, we have

$$ {}^0\hat{Z}_2 = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} \\ \frac{y}{\sqrt{x^2+y^2}} \\ 0 \end{bmatrix}. \tag{4.4} $$

The subspace can therefore be given as

$$ {}^0_2T = \begin{bmatrix} \frac{y}{\sqrt{x^2+y^2}} & 0 & \frac{x}{\sqrt{x^2+y^2}} & x \\ \frac{-x}{\sqrt{x^2+y^2}} & 0 & \frac{y}{\sqrt{x^2+y^2}} & y \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.5} $$

Usually, in defining a goal for a manipulator with n degrees of freedom, we use n parameters to specify the goal. If, on the other hand, we give a specification of a full six degrees of freedom, we will not in general be able to reach the goal with an $n < 6$ manipulator. In this case, we might be interested instead in reaching a goal that lies in the manipulator's subspace and is as "near" as possible to the original desired goal.

Hence, when specifying *general* goals for a manipulator with fewer than six degrees of freedom, one solution strategy is the following:

1. Given a general goal frame, ${}^S_GT$, compute a modified goal frame, ${}^S_{G'}T$, such that ${}^S_{G'}T$ lies in the manipulator's subspace and is as "near" to ${}^S_GT$ as possible. A definition of "near" must be chosen.
2. Compute the inverse kinematics to find joint angles using ${}^S_{G'}T$ as the desired goal. Note that a solution still might not be possible if the goal point is not in the manipulator's workspace.

It generally makes sense to position the tool-frame origin to the desired location and then choose an attainable orientation that is near the desired orientation. As we saw in Examples 4.1 and 4.2, computation of the subspace is dependent on manipulator geometry. Each manipulator must be individually considered to arrive at a method of making this computation.

Section 4.7 gives an example of *projecting* a general goal into the subspace of a manipulator with five degrees of freedom in order to compute joint angles that will result in the manipulator's reaching the attainable frame nearest to the desired one.

4.4 ALGEBRAIC VS. GEOMETRIC

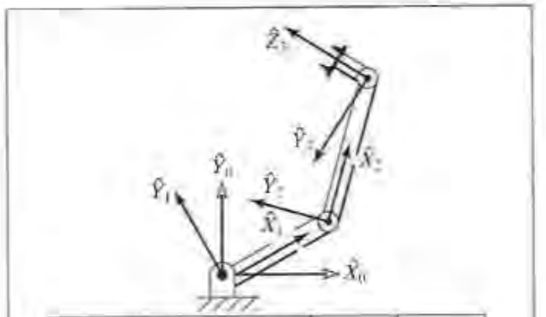
As an introduction to solving kinematic equations, we will consider two different approaches to the solution of a simple planar three-link manipulator.

Algebraic solution

Consider the three-link planar manipulator introduced in Chapter 3. It is shown with its link parameters in Fig. 4.7.

Following the method of Chapter 3, we can use the link parameters easily to find the kinematic equations of this arm:

$$ {}^B_WT = {}^0_3T = \begin{bmatrix} c_{123} & -s_{123} & 0.0 & l_1c_1 + l_2c_{12} \\ s_{123} & c_{123} & 0.0 & l_1s_1 + l_2s_{12} \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.6} $$



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3

FIGURE 4.7: Three-link planar manipulator and its link parameters.

To focus our discussion on inverse kinematics, we will assume that the necessary transformations have been performed so that the goal point is a specification of the wrist frame relative to the base frame, that is, ${}^{B}_{W}T$. Because we are working with a planar manipulator, specification of these goal points can be accomplished most easily by specifying three numbers: x, y, and ϕ, where ϕ is the orientation of link 3 in the plane (relative to the $+\hat{X}$ axis). Hence, rather than giving a general ${}^{B}_{W}T$ as a goal specification, we will assume a transformation with the structure

$$ {}^{B}_{W}T = \begin{bmatrix} c_\phi & -s_\phi & 0.0 & x \\ s_\phi & c_\phi & 0.0 & y \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.7} $$

All attainable goals must lie in the subspace implied by the structure of equation (4.7). By equating (4.6) and (4.7), we arrive at a set of four nonlinear equations that must be solved for θ_1, θ_2, and θ_3:

$$ c_\phi = c_{123}, \tag{4.8} $$

$$ s_\phi = s_{123}, \tag{4.9} $$

$$ x = l_1 c_1 + l_2 c_{12}, \tag{4.10} $$

$$ y = l_1 s_1 + l_2 s_{12}. \tag{4.11} $$

We now begin our algebraic solution of equations (4.8) through (4.11). If we square both (4.10) and (4.11) and add them, we obtain

$$ x^2 + y^2 = l_1^2 + l_2^2 + 2 l_1 l_2 c_2, \tag{4.12} $$

where we have made use of

$$ \begin{aligned} c_{12} &= c_1 c_2 - s_1 s_2, \\ s_{12} &= c_1 s_2 + s_1 c_2. \end{aligned} \tag{4.13} $$

Solving (4.12) for c_2, we obtain

$$ c_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}. \tag{4.14} $$

In order for a solution to exist, the right-hand side of (4.14) must have a value between -1 and 1. In the solution algorithm, this constraint would be checked at this time to find out whether a solution exists. Physically, if this constraint is not satisfied, then the goal point is too far away for the manipulator to reach.

Assuming the goal is in the workspace, we write an expression for s_2 as

$$ s_2 = \pm\sqrt{1 - c_2^2}. \tag{4.15} $$

Finally, we compute θ_2, using the two-argument arctangent routine[1]:

$$ \theta_2 = \text{Atan2}(s_2, c_2). \tag{4.16} $$

[1] See Section 2.8.

The choice of signs in (4.15) corresponds to the multiple solution in which we can choose the "elbow-up" or the "elbow-down" solution. In determining θ_2, we have used one of the recurring methods for solving the type of kinematic relationships that often arise, namely, to determine both the sine and cosine of the desired joint angle and then apply the two-argument arctangent. This ensures that we have found all solutions and that the solved angle is in the proper quadrant.

Having found θ_2, we can solve (4.10) and (4.11) for θ_1. We write (4.10) and (4.11) in the form

$$ x = k_1 c_1 - k_2 s_1, \tag{4.17} $$

$$ y = k_1 s_1 + k_2 c_1, \tag{4.18} $$

where

$$ \begin{aligned} k_1 &= l_1 + l_2 c_2, \\ k_2 &= l_2 s_2. \end{aligned} \tag{4.19} $$

In order to solve an equation of this form, we perform a change of variables. Actually, we are changing the way in which we write the constants k_1 and k_2.

If

$$ r = +\sqrt{k_1^2 + k_2^2} \tag{4.20} $$

and

$$ \gamma = \text{Atan2}(k_2, k_1), $$

then

$$ \begin{aligned} k_1 &= r\cos\gamma, \\ k_2 &= r\sin\gamma. \end{aligned} \tag{4.21} $$

Equations (4.17) and (4.18) can now be written as

$$ \frac{x}{r} = \cos\gamma\cos\theta_1 - \sin\gamma\sin\theta_1, \tag{4.22} $$

$$ \frac{y}{r} = \cos\gamma\sin\theta_1 + \sin\gamma\cos\theta_1. \tag{4.23} $$

so

$$ \cos(\gamma + \theta_1) = \frac{x}{r}, \tag{4.24} $$

$$ \sin(\gamma + \theta_1) = \frac{y}{r}. \tag{4.25} $$

Using the two-argument arctangent, we get

$$ \gamma + \theta_1 = \text{Atan2}\left(\frac{y}{r}, \frac{x}{r}\right) = \text{Atan2}(y, x), \tag{4.26} $$

and so

$$ \theta_1 = \text{Atan2}(y, x) - \text{Atan2}(k_2, k_1). \tag{4.27} $$

Note that, when a choice of sign is made in the solution of θ_2 above, it will cause a sign change in k_2, thus affecting θ_1. The substitutions used, (4.20) and (4.21), constitute a method of solution of a form appearing frequently in kinematics—namely, that of (4.10) or (4.11). Note also that, if $x = y = 0$, then (4.27) becomes undefined—in this case, θ_1 is arbitrary.

Finally, from (4.8) and (4.9), we can solve for the sum of θ_1 through θ_3:

$$\theta_1 + \theta_2 + \theta_3 = \text{Atan2}(s_\phi, c_\phi) = \phi. \tag{4.28}$$

From this, we can solve for θ_3, because we know the first two angles. It is typical with manipulators that have two or more links moving in a plane that, in the course of solution, expressions for sums of joint angles arise.

In summary, an algebraic approach to solving kinematic equations is basically one of manipulating the given equations into a form for which a solution is known. It turns out that, for many common geometries, several forms of transcendental equations commonly arise. We have encountered a couple of them in this preceding section. In Appendix C, more are listed.

Geometric solution

In a geometric approach to finding a manipulator's solution, we try to decompose the spatial geometry of the arm into several plane-geometry problems. For many manipulators (particularly when the $\alpha_i = 0$ or ± 90) this can be done quite easily. Joint angles can then be solved for by using the tools of plane geometry [7]. For the arm with three degrees of freedom shown in Fig. 4.7, because the arm is planar, we can apply plane geometry directly to find a solution.

Figure 4.8 shows the triangle formed by l_1, l_2, and the line joining the origin of frame {0} with the origin of frame {3}. The dashed lines represent the other possible configuration of the triangle, which would lead to the same position of the frame {3}. Considering the solid triangle, we can apply the "law of cosines" to solve for θ_2:

$$x^2 + y^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(180 + \theta_2). \tag{4.29}$$

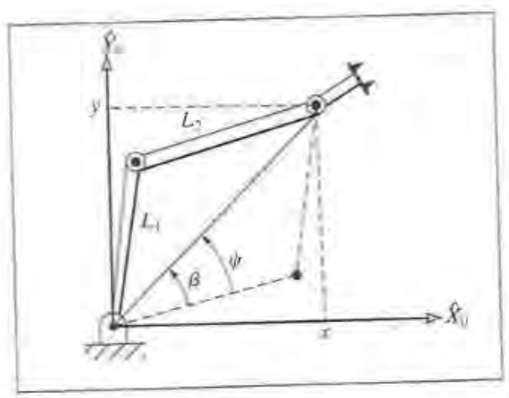


FIGURE 4.8: Plane geometry associated with a three-link planar robot.

Now, $\cos(180 + \theta_2) = -\cos(\theta_2)$, so we have

$$c_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}. \tag{4.30}$$

In order for this triangle to exist, the distance to the goal point $\sqrt{x^2 + y^2}$ must be less than or equal to the sum of the link lengths, $l_1 + l_2$. This condition would be checked at this point in a computational algorithm to verify existence of solutions. This condition is not satisfied when the goal point is out of reach of the manipulator. Assuming a solution exists, this equation is solved for that value of θ_2 that lies between 0 and -180 degrees, because only for these values does the triangle in Fig. 4.8 exist. The other possible solution (the one indicated by the dashed-line triangle) is found by symmetry to be $\theta_2' = -\theta_2$.

To solve for θ_1, we find expressions for angles ψ and β as indicated in Fig. 4.8. First, β may be in any quadrant, depending on the signs of x and y. So we must use a two-argument arctangent:

$$\beta = \text{Atan2}(y, x). \tag{4.31}$$

We again apply the law of cosines to find ψ:

$$\cos\psi = \frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}. \tag{4.32}$$

Here, the arccosine must be solved so that $0 \le \psi \le 180°$, in order that the geometry which leads to (4.32) will be preserved. These considerations are typical when using a geometric approach—we must apply the formulas we derive only over a range of variables such that the geometry is preserved. Then we have

$$\theta_1 = \beta \pm \psi. \tag{4.33}$$

where the plus sign is used if $\theta_2 < 0$ and the minus sign if $\theta_2 > 0$.

We know that angles in a plane add, so the sum of the three joint angles must be the orientation of the last link:

$$\theta_1 + \theta_2 + \theta_3 = \phi. \tag{4.34}$$

This equation is solved for θ_3 to complete our solution.

4.5 ALGEBRAIC SOLUTION BY REDUCTION TO POLYNOMIAL

Transcendental equations are often difficult to solve because, even when there is only one variable (say, θ), it generally appears as $\sin\theta$ and $\cos\theta$. Making the following substitutions, however, yields an expression in terms of a single variable, u:

$$u = \tan\frac{\theta}{2},$$

$$\cos\theta = \frac{1 - u^2}{1 + u^2}, \tag{4.35}$$

$$\sin\theta = \frac{2u}{1 + u^2}.$$

This is a very important geometric substitution used often in solving kinematic equations. These substitutions convert transcendental equations into polynomial equations in u. Appendix A lists these and other trigonometric identities.

EXAMPLE 4.3

Convert the transcendental equation

$$a\cos\theta + b\sin\theta = c \tag{4.36}$$

into a polynomial in the tangent of the half angle, and solve for θ.

Substituting from (4.35) and multiplying through by $1+u^2$, we have

$$a(1-u^2) + 2bu = c(1+u^2). \tag{4.37}$$

Collecting powers of u yields

$$(a+c)u^2 - 2bu + (c-a) = 0, \tag{4.38}$$

which is solved by the quadratic formula:

$$u = \frac{b \pm \sqrt{b^2+a^2-c^2}}{a+c}. \tag{4.39}$$

Hence,

$$\theta = 2\tan^{-1}\left(\frac{b \pm \sqrt{b^2+a^2-c^2}}{a+c}\right). \tag{4.40}$$

Should the solution for u from (4.39) be complex, there is no real solution to the original transcendental equation. Note that, if $a+c=0$, the argument of the arctangent becomes infinity and hence $\theta = 180°$. In a computer implementation, this potential division by zero should be checked for ahead of time. This situation results when the quadratic term of (4.38) vanishes, so that the quadratic degenerates into a linear equation.

Polynomials up to degree four possess closed-form solutions [8, 9], so manipulators sufficiently simple that they can be solved by algebraic equations of this degree (or lower) are called **closed-form-solvable** manipulators.

4.6 PIEPER'S SOLUTION WHEN THREE AXES INTERSECT

As mentioned earlier, although a completely general robot with six degrees of freedom does not have a closed-form solution, certain important special cases can be solved. Pieper [3, 4] studied manipulators with six degrees of freedom in which three consecutive axes intersect at a point.[2] In this section, we outline the method he developed for the case of all six joints revolute, with the last three axes intersecting. His method applies to other configurations, which include prismatic

[2]Included in this family of manipulators are those with three consecutive parallel axes, because they meet at the point at infinity.

joints, and the interested reader should see [4]. Pieper's work applies to the majority of commercially available industrial robots.

When the last three axes intersect, the origins of link frames {4}, {5}, and {6} are all located at this point of intersection. This point is given in base coordinates as

$${}^0P_{4ORG} = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3P_{4ORG} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{4.41}$$

or, using the fourth column of (3.6) for $i=4$, as

$${}^0P_{4ORG} = {}^0_1T\,{}^1_2T\,{}^2_3T \begin{bmatrix} a_3 \\ -d_4 s\alpha_3 \\ d_4 c\alpha_3 \\ 1 \end{bmatrix}. \tag{4.42}$$

or as

$${}^0P_{4ORG} = {}^0_1T\,{}^1_2T \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_3(\theta_3) \\ 1 \end{bmatrix}, \tag{4.43}$$

where

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ 1 \end{bmatrix} = {}^2_3T \begin{bmatrix} a_3 \\ -d_4 s\alpha_3 \\ d_4 c\alpha_3 \\ 1 \end{bmatrix}. \tag{4.44}$$

Using (3.6) for ${}^2_3T$ in (4.44) yields the following expressions for f_1:

$$\begin{aligned} f_1 &= a_3c_3 + d_4s\alpha_3s_3 + a_2, \\ f_2 &= a_3c\alpha_2s_3 - d_4s\alpha_3c\alpha_2c_3 - d_4s\alpha_2c\alpha_3 - d_3s\alpha_2, \\ f_3 &= a_3s\alpha_2s_3 - d_4s\alpha_3s\alpha_2c_3 + d_4c\alpha_2c\alpha_3 + d_3c\alpha_2. \end{aligned} \tag{4.45}$$

Using (3.6) for ${}^0_1T$ and ${}^1_2T$ in (4.43), we obtain

$${}^0P_{4ORG} = \begin{bmatrix} c_1g_1 - s_1g_2 \\ s_1g_1 + c_1g_2 \\ g_3 \\ 1 \end{bmatrix}. \tag{4.46}$$

where

$$\begin{aligned} g_1 &= c_2f_1 - s_2f_2 + a_1, \\ g_2 &= s_2c\alpha_1f_1 + c_2c\alpha_1f_2 - s\alpha_1f_3 - d_2s\alpha_1, \\ g_3 &= s_2s\alpha_1f_1 + c_2s\alpha_1f_2 + c\alpha_1f_3 + d_2c\alpha_1. \end{aligned} \tag{4.47}$$

We now write an expression for the squared magnitude of ${}^0P_{4ORG}$, which we will denote as $r = x^2+y^2+z^2$, and which is seen from (4.46) to be

$$r = g_1^2 + g_2^2 + g_3^2; \tag{4.48}$$

so, using (4.47) for the g_i, we have

$$r = f_1^2 + f_2^2 + f_3^2 + a_1^2 + d_2^2 + 2d_2 f_3 + 2a_1(c_2 f_1 - s_2 f_2). \tag{4.49}$$

We now write this equation, along with the Z-component equation from (4.46), as a system of two equations in the form

$$\begin{aligned} r &= (k_1 c_2 + k_2 s_2)2a_1 + k_3, \\ z &= (k_1 s_2 - k_2 c_2)s\alpha_1 + k_4, \end{aligned} \tag{4.50}$$

where

$$\begin{aligned} k_1 &= f_1, \\ k_2 &= -f_2, \\ k_3 &= f_1^2 + f_2^2 + f_3^2 + a_1^2 + d_2^2 + 2d_2 f_3, \\ k_4 &= f_3 c\alpha_1 + d_2 c\alpha_1. \end{aligned} \tag{4.51}$$

Equation (4.50) is useful because dependence on θ_1 has been eliminated and because dependence on θ_2 takes a simple form.

Now let us consider the solution of (4.50) for θ_3. We distinguish three cases:

1. If $a_1 = 0$, then we have $r = k_3$, where r is known. The right-hand side (k_3) is a function of θ_3 only. After the substitution (4.35), a quadratic equation in $\tan\frac{\theta_3}{2}$ may be solved for θ_3.
2. If $s\alpha_1 = 0$, then we have $z = k_4$, where z is known. Again, after substituting via (4.35), a quadratic equation arises that can be solved for θ_3.
3. Otherwise, eliminate s_2 and c_2 from (4.50) to obtain

$$\frac{(r - k_3)^2}{4a_1^2} + \frac{(z - k_4)^2}{s^2\alpha_1} = k_1^2 + k_2^2. \tag{4.52}$$

This equation, after the (4.35) substitution for θ_3, results in an equation of degree 4, which can be solved for θ_3.[3]

Having solved for θ_3, we can solve (4.50) for θ_2 and (4.46) for θ_1.

To complete our solution, we need to solve for θ_4, θ_5, and θ_6. These axes intersect, so these joint angles affect the orientation of only the last link. We can compute them from nothing more than the rotation portion of the specified goal, ${}^0_6R$. Having obtained θ_1, θ_2, and θ_3, we can compute ${}^0_4R|_{\theta_4=0}$, by which notation we mean the orientation of link frame {4} relative to the base frame when $\theta_4 = 0$. The desired orientation of {6} differs from this orientation only by the action of the last three joints. Because the problem was specified as given ${}^0_6R$, we can compute

$${}^4_6R|_{\theta_4=0} = {}^0_4R^{-1}|_{\theta_4=0}\,{}^0_6R. \tag{4.53}$$

[3] It is helpful to note that $f_1^2 + f_2^2 + f_3^2 = a_3^2 + d_4^2 + d_3^2 + a_2^2 + 2d_4 d_3 c\alpha_3 + 2a_2 a_3 c_3 + 2a_2 d_4 s\alpha_3 s_3$.

For many manipulators, these last three angles can be solved for by using exactly the Z–Y–Z Euler angle solution given in Chapter 2, applied to ${}^4_0R|_{\theta_4=0}$. For any manipulator (with intersecting axes 4, 5, and 6), the last three joint angles can be solved for as a set of appropriately defined Euler angles. There are always two solutions for these last three joints, so the total number of solutions for the manipulator will be twice the number found for the first three joints.

4.7 EXAMPLES OF INVERSE MANIPULATOR KINEMATICS

In this section, we work out the inverse kinematics of two industrial robots. One manipulator solution is done purely algebraically; the second solution is partially algebraic and partially geometric. The following solutions do not constitute a cookbook method of solving manipulator kinematics, but they do show many of the common manipulations likely to appear in most kinematic solutions. Note that Pieper's method of solution (covered in the preceding section) can be used for these manipulators, but here we choose to approach the solution a different way, to give insight into various available methods.

The Unimation PUMA 560

As an example of the algebraic solution technique applied to a manipulator with six degrees of freedom, we will solve the kinematic equations of the PUMA 560, which were developed in Chapter 3. This solution is in the style of [5].

We wish to solve

$$\begin{aligned} {}^0_6T &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= {}^0_1T(\theta_1)\,{}^1_2T(\theta_2)\,{}^2_3T(\theta_3)\,{}^3_4T(\theta_4)\,{}^4_5T(\theta_5)\,{}^5_6T(\theta_6) \end{aligned} \tag{4.54}$$

for θ_i when ${}^0_6T$ is given as numeric values.

A restatement of (4.54) that puts the dependence on θ_1 on the left-hand side of the equation is

$$[{}^0_1T(\theta_1)]^{-1}\,{}^0_6T = {}^1_2T(\theta_2)\,{}^2_3T(\theta_3)\,{}^3_4T(\theta_4)\,{}^4_5T(\theta_5)\,{}^5_6T(\theta_6). \tag{4.55}$$

Inverting ${}^0_1T$, we write (4.55) as

$$\begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1_6T, \tag{4.56}$$

where ${}^1_6T$ is given by equation (3.13) developed in Chapter 3. This simple technique of multiplying each side of a transform equation by an inverse is often used to advantage in separating out variables in the search for a solvable equation.

Equating the (2, 4) elements from both sides of (4.56), we have

$$-s_1 p_x + c_1 p_y = d_3. \tag{4.57}$$

To solve an equation of this form, we make the trigonometric substitutions

$$p_x = \rho \cos\phi,$$
$$p_y = \rho \sin\phi, \tag{4.58}$$

where

$$\rho = \sqrt{p_x^2 + p_y^2},$$
$$\phi = \text{Atan2}(p_y, p_x). \tag{4.59}$$

Substituting (4.58) into (4.57), we obtain

$$c_1 s_\phi - s_1 c_\phi = \frac{d_3}{\rho}. \tag{4.60}$$

From the difference-of-angles formula,

$$\sin(\phi - \theta_1) = \frac{d_3}{\rho}. \tag{4.61}$$

Hence,

$$\cos(\phi - \theta_1) = \pm\sqrt{1 - \frac{d_3^2}{\rho^2}}, \tag{4.62}$$

and so

$$\phi - \theta_1 = \text{Atan2}\left(\frac{d_3}{\rho}, \pm\sqrt{1 - \frac{d_3^2}{\rho^2}}\right). \tag{4.63}$$

Finally, the solution for θ_1 may be written as

$$\theta_1 = \text{Atan2}(p_y, p_x) - \text{Atan2}\left(d_3, \pm\sqrt{p_x^2 + p_y^2 - d_3^2}\right). \tag{4.64}$$

Note that we have found two possible solutions for θ_1, corresponding to the plus-or-minus sign in (4.64). Now that θ_1 is known, the left-hand side of (4.56) is known. If we equate both the (1,4) elements and the (3,4) elements from the two sides of (4.56), we obtain

$$c_1 p_x + s_1 p_y = a_3 c_{23} - d_4 s_{23} + a_2 c_2,$$
$$-p_z = a_3 s_{23} + d_4 c_{23} + a_2 s_2. \tag{4.65}$$

If we square equations (4.65) and (4.57) and add the resulting equations, we obtain

$$a_3 c_3 - d_4 s_3 = K, \tag{4.66}$$

where

$$K = \frac{p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2 - d_3^2 - d_4^2}{2a_2}. \tag{4.67}$$

Note that dependence on θ_1 has been removed from (4.66). Equation (4.66) is of the same form as (4.57) and so can be solved by the same kind of trigonometric substitution to yield a solution for θ_3:

$$\theta_3 = \text{Atan2}(a_3, d_4) - \text{Atan2}(K, \pm\sqrt{a_3^2 + d_4^2 - K^2}). \tag{4.68}$$

The plus-or-minus sign in (4.68) leads to two different solutions for θ_3. If we consider (4.54) again, we can now rewrite it so that all the left-hand side is a function of only knowns and θ_2:

$$[{}^0_3T(\theta_2)]^{-1}\,{}^0_6T = {}^3_4T(\theta_4)\,{}^4_5T(\theta_5)\,{}^5_6T(\theta_6), \tag{4.69}$$

or

$$\begin{bmatrix} c_1c_{23} & s_1c_{23} & -s_{23} & -a_2c_3 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 \\ -s_1 & c_1 & 0 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^3_6T, \tag{4.70}$$

where ${}^3_6T$ is given by equation (3.11) developed in Chapter 3. Equating both the (1,4) elements and the (2,4) elements from the two sides of (4.70), we get

$$c_1c_{23}p_x + s_1c_{23}p_y - s_{23}p_z - a_2c_3 = a_3,$$
$$-c_1s_{23}p_x - s_1s_{23}p_y - c_{23}p_z + a_2s_3 = d_4. \tag{4.71}$$

These equations can be solved simultaneously for s_{23} and c_{23}, resulting in

$$s_{23} = \frac{(-a_3 - a_2c_3)p_z + (c_1p_x + s_1p_y)(a_2s_3 - d_4)}{p_z^2 + (c_1p_x + s_1p_y)^2},$$
$$c_{23} = \frac{(a_2s_3 - d_4)p_z - (a_3 + a_2c_3)(c_1p_x + s_1p_y)}{p_z^2 + (c_1p_x + s_1p_y)^2}. \tag{4.72}$$

The denominators are equal and positive, so we solve for the sum of θ_2 and θ_3 as

$$\theta_{23} = \text{Atan2}[(-a_3 - a_2c_3)p_z - (c_1p_x + s_1p_y)(d_4 - a_2s_3),$$
$$(a_2s_3 - d_4)p_z - (a_3 + a_2c_3)(c_1p_x + s_1p_y)]. \tag{4.73}$$

Equation (4.73) computes four values of θ_{23}, according to the four possible combinations of solutions for θ_1 and θ_3; then, four possible solutions for θ_2 are computed as

$$\theta_2 = \theta_{23} - \theta_3, \tag{4.74}$$

where the appropriate solution for θ_3 is used when forming the difference.

Now the entire left side of (4.70) is known. Equating both the (1,3) elements and the (3,3) elements from the two sides of (4.70), we get

$$r_{13}c_1c_{23} + r_{23}s_1c_{23} - r_{33}s_{23} = -c_4s_5,$$
$$-r_{13}s_1 + r_{23}c_1 = s_4s_5. \tag{4.75}$$

As long as $s_5 \neq 0$, we can solve for θ_4 as

$$\theta_4 = \text{Atan2}(-r_{13}s_1 + r_{23}c_1, -r_{13}c_1c_{23} - r_{23}s_1c_{23} + r_{33}s_{23}). \tag{4.76}$$

When $\theta_5 = 0$, the manipulator is in a singular configuration in which joint axes 4 and 6 line up and cause the same motion of the last link of the robot. In this case, all that matters (and all that can be solved for) is the sum or difference of θ_4 and θ_6. This situation is detected by checking whether both arguments of the Atan2 in (4.76) are near zero. If so, θ_4 is chosen arbitrarily,[4] and when θ_6 is computed later, it will be computed accordingly.

If we consider (4.54) again, we can now rewrite it so that all the left-hand side is a function of only knowns and θ_4, by rewriting it as

$$[{}^0_4T(\theta_4)]^{-1}\,{}^0_6T = {}^4_5T(\theta_5){}^5_6T(\theta_6). \tag{4.77}$$

where $[{}^0_4T(\theta 4)]^{-1}$ is given by

$$\begin{bmatrix} c_1c_{23}c_4 + s_1s_4 & s_1c_{23}c_4 - c_1s_4 & -s_{23}c_4 & -a_2c_3c_3 + d_3s_4 - a_3c_4 \\ -c_1c_{23}s_4 + s_1c_4 & -s_1c_{23}s_4 - c_1c_4 & s_{23}s_4 & a_2c_3s_4 + d_3c_4 + a_3s_4 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.78}$$

and ${}^4_6T$ is given by equation (3.10) developed in Chapter 3. Equating both the (1,3) elements and the (3,3) elements from the two sides of (4.77), we get

$$\begin{aligned} r_{13}(c_1c_{23}c_4 + s_1s_4) + r_{23}(s_1c_{23}c_4 - c_1s_4) - r_{33}(s_{23}c_4) &= -s_5, \\ r_{13}(-c_1s_{23}) + r_{23}(-s_1s_{23}) + r_{33}(-c_{23}) &= c_5. \end{aligned} \tag{4.79}$$

Hence, we can solve for θ_5 as

$$\theta_5 = \text{Atan2}(s_5, c_5). \tag{4.80}$$

where s_5 and c_5 are given by (4.79).

Applying the same method one more time, we compute $({}^0_5T)^{-1}$ and write (4.54) in the form

$$({}^0_5T)^{-1}\,{}^0_6T = {}^5_6T(\theta_6). \tag{4.81}$$

Equating both the (3,1) elements and the (1,1) elements from the two sides of (4.77) as we have done before, we get

$$\theta_6 = \text{Atan2}(s_6, c_6). \tag{4.82}$$

where

$$\begin{aligned} s_6 &= -r_{11}(c_1c_{23}s_4 - s_1c_4) - r_{21}(s_1c_{23}s_4 + c_1c_4) + r_{31}(s_{23}s_4), \\ c_6 &= r_{11}[(c_1c_{23}c_4 + s_1s_4)c_5 - c_1s_{23}s_5] + r_{21}[(s_1c_{23}c_4 - c_1s_4)c_5 - s_1s_{23}s_5] \\ &\quad - r_{31}(s_{23}c_4c_5 + c_{23}s_5). \end{aligned}$$

Because of the plus-or-minus signs appearing in (4.64) and (4.68), these equations compute four solutions. Additionally, there are four more solutions obtained by

[4] It is usually chosen to be equal to the present value of joint 4.

"flipping" the wrist of the manipulator. For each of the four solutions computed above, we obtain the flipped solution by

$$\begin{aligned} \theta'_4 &= \theta_4 + 180^\circ, \\ \theta'_5 &= -\theta_5, \\ \theta'_6 &= \theta_6 + 180^\circ. \end{aligned} \tag{4.83}$$

After all eight solutions have been computed, some (or even all) of them might have to be discarded because of joint-limit violations. Of any remaining valid solutions, usually the one closest to the present manipulator configuration is chosen.

The Yasukawa Motoman L-3

As the second example, we will solve the kinematic equations of the Yasukawa Motoman L-3, which were developed in Chapter 3. This solution will be partially algebraic and partially geometric. The Motoman L-3 has three features that make the inverse kinematic problem quite different from that of the PUMA. First, the manipulator has only five joints, so it is not able to position and orient its end-effector in order to attain *general* goal frames. Second, the four-bar type of linkages and chain-drive scheme cause one actuator to move two or more joints. Third, the actuator position limits are not constants, but depend on the positions of the other actuators, so finding out whether a computed set of actuator values is in range is not trivial.

If we consider the nature of the subspace of the Motoman manipulator (and the same applies to many manipulators with five degrees of freedom), we quickly realize that this subspace can be described by giving one constraint on the attainable orientation: The pointing direction of the tool, that is, the $\hat{Z}_T$ axis, must lie in the "plane of the arm." This plane is the vertical plane that contains the axis of joint 1 and the point where axes 4 and 5 intersect. The orientation nearest to a general orientation is the one obtained by rotating the tool's pointing direction so that it lies in the plane, using a minimum amount of rotation. Without developing an explicit expression for this subspace, we will construct a method for projecting a general goal frame into it. Note that this entire discussion is for the case that the wrist frame and tool frame differ only by a translation along $\hat{Z}_w$.

In Fig. 4.9, we indicate the plane of the arm by its normal, $\hat{M}$, and the desired pointing direction of the tool by $\hat{Z}_T$. This pointing direction must be rotated by angle θ about some vector $\hat{K}$ in order to cause the new pointing direction, $\hat{Z}'_T$, to lie in the plane. It is clear that the $\hat{K}$ that minimizes θ lies in the plane and is orthogonal to both $\hat{Z}_T$ and $\hat{Z}'_T$.

For any given goal frame, $\hat{M}$ is defined as

$$\hat{M} = \frac{1}{\sqrt{p_x^2 + p_y^2}} \begin{bmatrix} -p_y \\ p_x \\ 0 \end{bmatrix}, \tag{4.84}$$

where p_x and p_y are the X and Y coordinates of the desired tool position. Then K is given by

$$K = \hat{M} \times \hat{Z}_T. \tag{4.85}$$

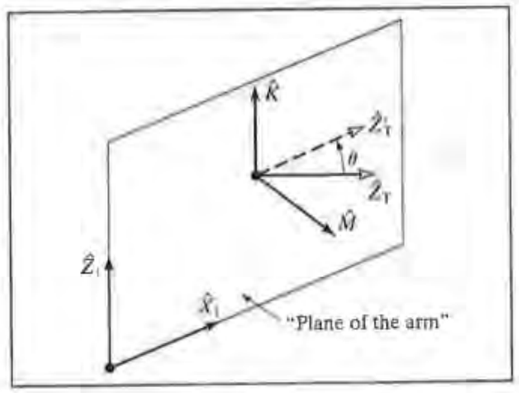


FIGURE 4.9: Rotating a goal frame into the Motoman's subspace.

The new $\hat{Z}_T'$ is

$$\hat{Z}_T' = \hat{K} \times \hat{M}. \tag{4.86}$$

The amount of rotation, θ, is given by

$$\begin{aligned} \cos\theta &= \hat{Z}_T \cdot \hat{Z}_T', \\ \sin\theta &= (\hat{Z}_T \times \hat{Z}_T') \cdot \hat{K}. \end{aligned} \tag{4.87}$$

Using Rodriques's formula (see Exercise 2.20), we have

$$\hat{Y}_T' = c\theta \hat{Y}_T + s\theta(\hat{K} \times \hat{Y}_T) + (1 - c\theta)(\hat{K} \cdot \hat{Y}_T)\hat{K}. \tag{4.88}$$

Finally, we compute the remaining unknown column of the new rotation matrix of the tool as

$$\hat{X}_T' = \hat{Y}_T' \times \hat{Z}_T'. \tag{4.89}$$

Equations (4.84) through (4.89) describe a method of projecting a given general goal orientation into the subspace of the Motoman robot.

Assuming that the given wrist frame, ${}^B_W T$, lies in the manipulator's subspace, we solve the kinematic equations as follows. In deriving the kinematic equations for the Motoman L-3, we formed the product of link transformations:

$${}^0_5 T = {}^0_1 T \, {}^1_2 T \, {}^2_3 T \, {}^3_4 T \, {}^4_5 T. \tag{4.90}$$

If we let

$${}^0_5 T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.91}$$

and premultiply both sides by ${}^0_1 T^{-1}$, we have

$${}^0_1 T^{-1} \, {}^0_5 T = {}^1_2 T \, {}^2_3 T \, {}^3_4 T \, {}^4_5 T. \tag{4.92}$$

where the left-hand side is

$$\begin{bmatrix} c_1 r_{11} + s_1 r_{21} & c_1 r_{12} + s_1 r_{22} & c_1 r_{13} + s_1 r_{23} & c_1 p_x + s_1 p_y \\ -r_{31} & -r_{32} & -r_{33} & -p_z \\ -s_1 r_{11} + c_1 r_{21} & -s_1 r_{12} + c_1 r_{22} & -s_1 r_{13} + c_1 r_{23} & -s_1 p_x + c_1 p_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.93}$$

and the right-hand side is

$$\begin{bmatrix} * & * & s_{234} & * \\ * & * & -c_{234} & * \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \tag{4.94}$$

in the latter, several of the elements have not been shown. Equating the (3,4) elements, we get

$$-s_1 p_x + c_1 p_y = 0, \tag{4.95}$$

which gives us[5]

$$\theta_1 = \text{Atan2}(p_y, p_x). \tag{4.96}$$

Equating the (3,1) and (3,2) elements, we get

$$\begin{aligned} s_5 &= -s_1 r_{11} + c_1 r_{21}, \\ c_5 &= -s_1 r_{12} + c_1 r_{22}. \end{aligned} \tag{4.97}$$

from which we calculate θ_5 as

$$\theta_5 = \text{Atan2}(r_{21} c_1 - r_{11} s_1, r_{22} c_1 - r_{12} s_1). \tag{4.98}$$

Equating the (2,3) and (1,3) elements, we get

$$\begin{aligned} c_{234} &= r_{33}, \\ s_{234} &= c_1 r_{13} + s_1 r_{23}, \end{aligned} \tag{4.99}$$

which leads to

$$\theta_{234} = \text{Atan2}(r_{13} c_1 + r_{23} s_1, r_{33}). \tag{4.100}$$

To solve for the individual angles θ_2, θ_3, and θ_4, we will take a geometric approach. Figure 4.10 shows the plane of the arm with point A at joint axis 2, point B at joint axis 3, and point C at joint axis 4.

From the law of cosines applied to triangle ABC, we have

$$\cos\theta_3 = \frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2 l_2 l_3}. \tag{4.101}$$

Next, we have[6]

$$\theta_3 = \text{Atan2}\left(\sqrt{1 - \cos^2\theta_3}, \cos\theta_3\right). \tag{4.102}$$

[5] For this manipulator, a second solution would violate joint limits and so is not calculated.

[6] For this manipulator, a second solution would violate joint limits and so is not calculated.

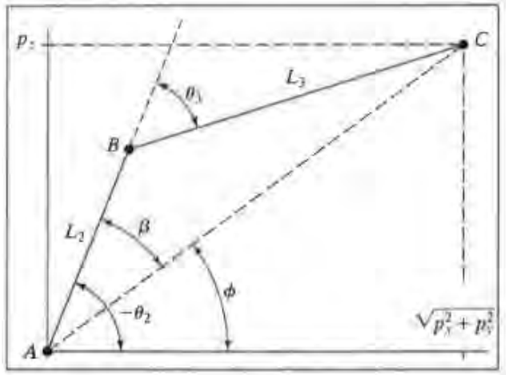


FIGURE 4.10: The plane of the Motoman manipulator.

From Fig. 4.10, we see that $\theta_2 = -\phi - \beta$, or

$$\theta_2 = -\text{Atan2}\left(p_z, \sqrt{p_x^2 + p_y^2}\right) - \text{Atan2}(l_3 \sin\theta_3, l_2 + l_3 \cos\theta_3). \tag{4.103}$$

Finally, we have

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3. \tag{4.104}$$

Having solved for joint angles, we must perform the further computation to obtain the actuator values. Referring to Section 3.7, we solve equation (3.16) for the A_i:

$$\begin{aligned}
A_1 &= \frac{1}{k_1}(\theta_1 - \lambda_1),\\
A_2 &= \frac{1}{k_2}\left(\sqrt{-2\alpha_2\beta_2 \cos\left(\theta_2 - \Omega_2 - \tan^{-1}\left(\frac{\phi_2}{\gamma_2}\right) + 270^\circ\right) + \alpha_2^2 + \beta_2^2} - \lambda_2\right),\\
A_3 &= \frac{1}{k_3}\left(\sqrt{-2\alpha_3\beta_3 \cos\left(\theta_2 + \theta_3 - \tan^{-1}\left(\frac{\phi_3}{\gamma_3}\right) + 90^\circ\right) + \alpha_3^2 + \beta_3^2} - \lambda_3\right),\\
A_4 &= \frac{1}{k_4}(180^\circ + \lambda_4 - \theta_2 - \theta_3 - \theta_4),\\
A_5 &= \frac{1}{k_5}(\lambda_5 - \theta_5).
\end{aligned} \tag{4.105}$$

The actuators have limited ranges of motion, so we must check that our computed solution is in range. This "in range" check is complicated by the fact that the mechanical arrangement makes actuators interact and affect each other's allowed range of motion. For the Motoman robot, actuators 2 and 3 interact in such a way that the following relationship must always be obeyed:

$$A_2 - 10,000 > A_3 > A_2 + 3000. \tag{4.106}$$

That is, the limits of actuator 3 are a function of the position of actuator 2. Similarly,

$$32,000 - A_4 < A_5 < 55,000. \tag{4.107}$$

Now, one revolution of joint 5 corresponds to 25,600 actuator counts, so, when $A_4 > 2600$, there are two possible solutions for A_5. This is the only situation in which the Yasukawa Motoman L-3 has more than one solution.

4.8 THE STANDARD FRAMES

The ability to solve for joint angles is really the central element in many robot control systems. Again, consider the paradigm indicated in Fig. 4.11, which shows the standard frames.

The way these frames are used in a general robot system is as follows:

1. The user specifies to the system where the station frame is to be located. This might be at the corner of a work surface, as in Fig. 4.12, or even affixed to a moving conveyor belt. The station frame, $\{S\}$, is defined relative to the base frame, $\{B\}$.
2. The user specifies the description of the tool being used by the robot by giving the $\{T\}$-frame specification. Each tool the robot picks up could have a different $\{T\}$ frame associated with it. Note that the same tool grasped in different ways requires different $\{T\}$-frame definitions. $\{T\}$ is specified relative to $\{W\}$—that is, ${}^W_T T$.
3. The user specifies the goal point for a robot motion by giving the description of the goal frame, $\{G\}$, relative to the station frame. Often, the definitions of $\{T\}$ and $\{S\}$ remain fixed for several motions of the robot. In this case, once they are defined, the user simply gives a series of $\{G\}$ specifications.

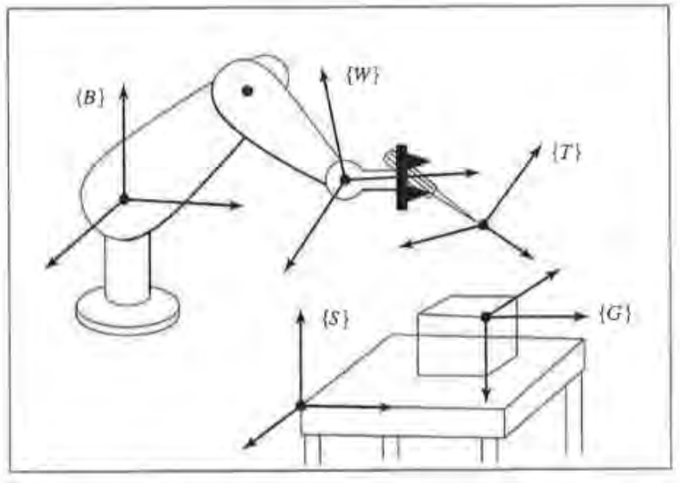


FIGURE 4.11: Location of the "standard" frames.

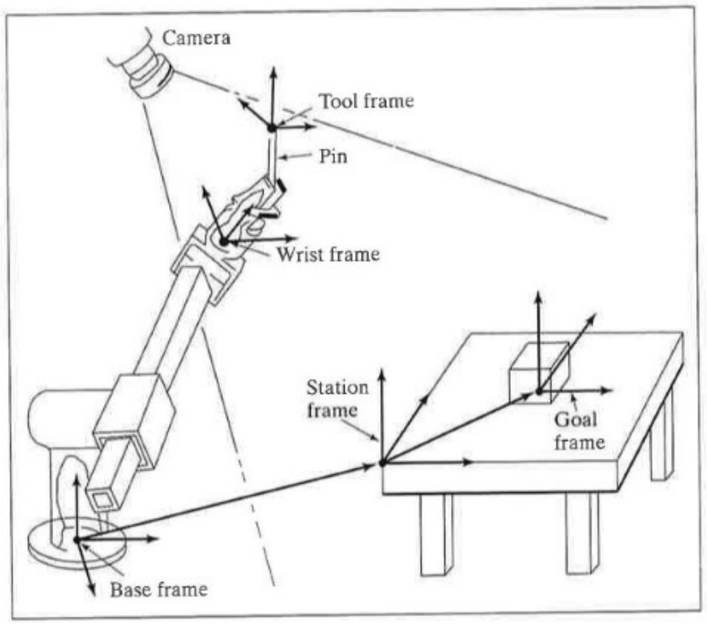


FIGURE 4.12: Example workstation.

In many systems, the tool frame definition (${}^{W}_{T}T$) is constant (for example, it is defined with its origin at the center of the fingertips). Also, the station frame might be fixed or might easily be taught by the user with the robot itself. In such systems, the user need not be aware of the five standard frames—he or she simply thinks in terms of moving the tool to locations (goals) with respect to the work area specified by station frame.

4. The robot system calculates a series of joint angles to move the joints through in order that the tool frame will move from its initial location in a smooth manner until $\{T\} = \{G\}$ at the end of motion.

4.9 SOLVE-ING A MANIPULATOR

The SOLVE function implements Cartesian transformations and calls the inverse kinematics function. Thus, the inverse kinematics are generalized so that arbitrary tool-frame and station-frame definitions may be used with our basic inverse kinematics, which solves for the wrist frame relative to the base frame.

Given the goal-frame specification, ${}^{S}_{T}T$, SOLVE uses the tool and station definitions to calculate the location of $\{W\}$ relative to $\{B\}$, ${}^{B}_{W}T$:

$$ {}^{B}_{W}T = {}^{B}_{S}T\,{}^{S}_{T}T\,{}^{W}_{T}T^{-1}. \tag{4.108} $$

Then, the inverse kinematics take ${}^{B}_{W}T$ as an input and calculate θ_1 through θ_n.

4.10 REPEATABILITY AND ACCURACY

Many industrial robots today move to goal points that have been taught. A **taught point** is one that the manipulator is moved to physically, and then the joint position sensors are read and the joint angles stored. When the robot is commanded to return to that point in space, each joint is moved to the stored value. In simple "teach and playback" manipulators such as these, the inverse kinematic problem never arises, because goal points are never specified in Cartesian coordinates. When a manufacturer specifies how precisely a manipulator can return to a taught point, he is specifying the **repeatability** of the manipulator.

Any time a goal position and orientation are specified in Cartesian terms, the inverse kinematics of the device must be computed in order to solve for the required joint angles. Systems that allow goals to be described in Cartesian terms are capable of moving the manipulator to points that were never taught—points in its workspace to which it has perhaps never gone before. We will call such points **computed points**. Such a capability is necessary for many manipulation tasks. For example, if a computer vision system is used to locate a part that the robot must grasp, the robot must be able to move to the Cartesian coordinates supplied by the vision sensor. The precision with which a computed point can be attained is called the **accuracy** of the manipulator.

The accuracy of a manipulator is bounded by the repeatability. Clearly, accuracy is affected by the precision of parameters appearing in the kinematic equations of the robot. Errors in knowledge of the Denavit–Hartenberg parameters will cause the inverse kinematic equations to calculate joint angle values that are in error. Hence, although the repeatability of most industrial manipulators is quite good, the accuracy is usually much worse and varies quite a bit from manipulator to manipulator. Calibration techniques can be devised that allow the accuracy of a manipulator to be improved through estimation of that particular manipulator's kinematic parameters [10].

4.11 COMPUTATIONAL CONSIDERATIONS

In many path-control schemes, which we will consider in Chapter 7, it is necessary to calculate the inverse kinematics of a manipulator at fairly high rates, for example, 30 Hz or faster. Therefore, computational efficiency is an issue. These speed requirements rule out the use of numerical-solution techniques that are iterative in nature; for this reason, we have not considered them.

Most of the general comments of Section 3.10, made for forward kinematics, also hold for the problem of inverse kinematics. For the inverse-kinematic case, a table-lookup Atan2 routine is often used to attain higher speeds.

Structure of the computation of multiple solutions is also important. It is generally fairly efficient to generate all of them in parallel, rather than pursuing one after another serially. Of course, in some applications, when all solutions are not required, substantial time is saved by computing only one.

When a geometric approach is used to develop an inverse-kinematic solution, it is sometimes possible to calculate multiple solutions by simple operations on the various angles solved for in obtaining the first solution. That is, the first solution

is moderately expensive computationally, but the other solutions are found very quickly by summing and differencing angles, subtracting π, and so on.

BIBLIOGRAPHY

[1] B. Roth, J. Rastegar, and V. Scheinman, "On the Design of Computer Controlled Manipulators," *On the Theory and Practice of Robots and Manipulators*, Vol. 1, First CISM-IFToMM Symposium, September 1973, pp. 93–113.

[2] B. Roth, "Performance Evaluation of Manipulators from a Kinematic Viewpoint," *Performance Evaluation of Manipulators*, National Bureau of Standards, special publication, 1975.

[3] D. Pieper and B. Roth, "The Kinematics of Manipulators Under Computer Control," *Proceedings of the Second International Congress on Theory of Machines and Mechanisms*, Vol. 2, Zakopane, Poland, 1969, pp. 159–169.

[4] D. Pieper, "The Kinematics of Manipulators Under Computer Control," Unpublished Ph.D. Thesis, Stanford University, 1968.

[5] R.P. Paul, B. Shimano, and G. Mayer, "Kinematic Control Equations for Simple Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 6, 1981.

[6] L. Tsai and A. Morgan, "Solving the Kinematics of the Most General Six- and Five-degree-of-freedom Manipulators by Continuation Methods," Paper 84-DET-20, ASME Mechanisms Conference, Boston, October 7–10, 1984.

[7] C.S.G. Lee and M. Ziegler, "Geometric Approach in Solving Inverse Kinematics of PUMA Robots," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-20, No. 6, November 1984.

[8] W. Beyer, *CRC Standard Mathematical Tables*, 25th edition, CRC Press, Inc., Boca Raton, FL, 1980.

[9] R. Burington, *Handbook of Mathematical Tables and Formulas*, 5th edition, McGraw-Hill, New York, 1973.

[10] J. Hollerbach, "A Survey of Kinematic Calibration," in *The Robotics Review*, O. Khatib, J. Craig, and T. Lozano-Perez, Editors, MIT Press, Cambridge, MA, 1989.

[11] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 108, 1986.

[12] D. Baker and C. Wampler, "On the Inverse Kinematics of Redundant Manipulators," *International Journal of Robotics Research*, Vol. 7, No. 2, 1988.

[13] L.W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*, Wiley, New York, 1999.

EXERCISES

4.1 [15] Sketch the fingertip workspace of the three-link manipulator of Chapter 3, Exercise 3.3 for the case $l_1 = 15.0$, $l_2 = 10.0$, and $l_3 = 3.0$.

4.2 [26] Derive the inverse kinematics of the three-link manipulator of Chapter 3, Exercise 3.3.

4.3 [12] Sketch the fingertip workspace of the 3-DOF manipulator of Chapter 3, Example 3.4.

4.4 [24] Derive the inverse kinematics of the 3-DOF manipulator of Chapter 3, Example 3.4.

4.5 [38] Write a Pascal (or C) subroutine that computes all possible solutions for the PUMA 560 manipulator that lie within the following joint limits:

$$-170.0 < \theta_1 < 170.0,$$
$$-225.0 < \theta_2 < 45.0,$$
$$-250.0 < \theta_3 < 75.0,$$
$$-135.0 < \theta_4 < 135.0,$$
$$-100.0 < \theta_5 < 100.0,$$
$$-180.0 < \theta_6 < 180.0.$$

Use the equations derived in Section 4.7 with these numerical values (in inches):

$$a_2 = 17.0,$$
$$a_3 = 0.8,$$
$$d_3 = 4.9,$$
$$d_4 = 17.0.$$

4.6 [15] Describe a simple algorithm for choosing the nearest solution from a set of possible solutions.

4.7 [10] Make a list of factors that might affect the repeatability of a manipulator. Make a second list of additional factors that affect the accuracy of a manipulator.

4.8 [12] Given a desired position and orientation of the hand of a three-link planar rotary-jointed manipulator, there are two possible solutions. If we add one more rotational joint (in such a way that the arm is still planar), how many solutions are there?

4.9 [26] Figure 4.13 shows a two-link planar arm with rotary joints. For this arm, the second link is half as long as the first—that is, $l_1 = 2l_2$. The joint range limits in

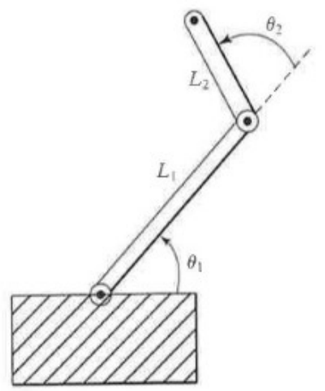


FIGURE 4.13: Two-link planar manipulator.

degrees are

$$0 < \theta_1 < 180.$$

$$-90 < \theta_2 < 180.$$

Sketch the approximate reachable workspace (an area) of the tip of link 2.

4.10 [23] Give an expression for the subspace of the manipulator of Chapter 3, Example 3.4.

4.11 [24] A 2-DOF positioning table is used to orient parts for arc-welding. The forward kinematics that locate the bed of the table (link 2) with respect to the base (link 0) are

$${}^0_2T = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & l_2s_1 + l_1 \\ s_2 & c_2 & 0 & 0 \\ -s_1c_2 & s_1s_2 & c_1 & l_2c_1 + h_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Given any unit direction fixed in the frame of the bed (link 2), ${}^2\hat{V}$, give the inverse-kinematic solution for θ_1, θ_2 such that this vector is aligned with ${}^0\hat{Z}$ (i.e., upward). Are there multiple solutions? Is there a singular condition for which a unique solution cannot be obtained?

4.12 [22] In Fig. 4.14, two $3R$ mechanisms are pictured. In both cases, the three axes intersect at a point (and, over all configurations, this point remains fixed in space). The mechanism in Fig. 4.14(a) has link twists (α_i) of magnitude 90 degrees. The mechanism in Fig. 4.14(b) has one twist of ϕ in magnitude and the other of $180-\phi$ in magnitude.
The mechanism in Fig. 4.14(a) can be seen to be in correspondence with Z–Y–Z Euler angles, and therefore we know that it suffices to orient link 3 (with arrow in figure) arbitrarily with respect to the fixed link 0. Because ϕ is not equal to 90 degrees, it turns out that the other mechanism cannot orient link 3 arbitrarily.

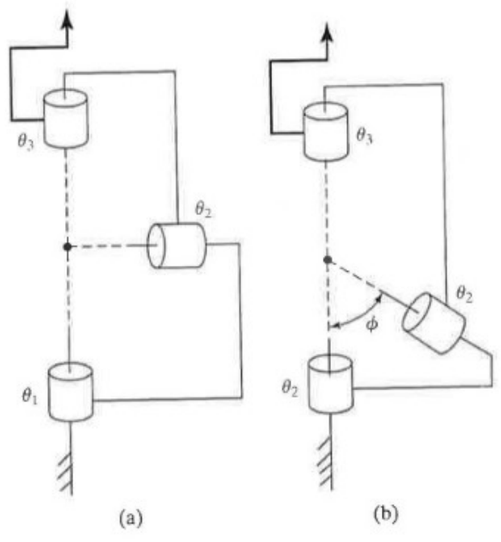


FIGURE 4.14: Two $3R$ mechanisms (Exercise 4.12).

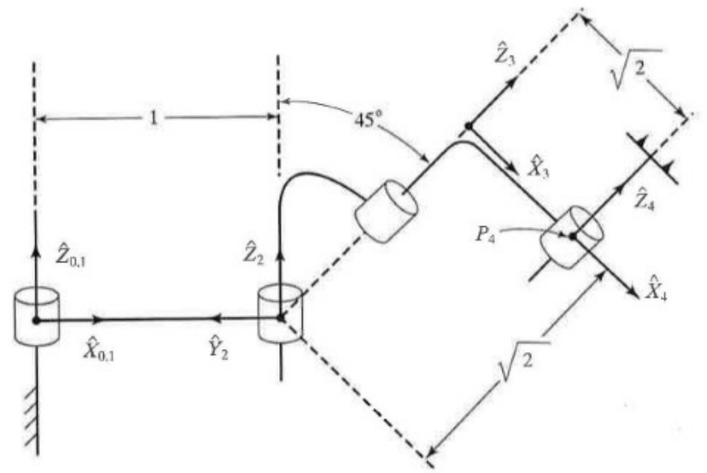


FIGURE 4.15: A $4R$ manipulator shown in the position $\Theta = [0, 90°, -90°, 0]^T$ (Exercise 4.16).

Describe the set of orientations that are *unattainable* with the second mechanism. Note that we assume that all joints can turn 360 degrees (i.e. no limits) and we assume that the links may pass through each other if need be (i.e., workspace not limited by self-collisions).

4.13 [13] Name two reasons for which closed-form analytic kinematic solutions are preferred over iterative solutions.

4.14 [14] There exist 6-DOF robots for which the kinematics are NOT closed-form solvable. Does there exist any 3-DOF robot for which the (position) kinematics are NOT closed-form solvable?

4.15 [38] Write a subroutine that solves quartic equations in closed form. (See [8, 9].)

4.16 [25] A $4R$ manipulator is shown schematically in Fig. 4.15. The nonzero link parameters are $a_1 = 1$, $\alpha_2 = 45°$, $d_3 = \sqrt{2}$, and $a_3 = \sqrt{2}$, and the mechanism is pictured in the configuration corresponding to $\Theta = [0, 90°, -90°, 0]^T$. Each joint has $\pm 180°$ as limits. Find all values of θ_3 such that

$${}^0P_{4ORG} = [1.1, 1.5, 1.707]^T.$$

4.17 [25] A $4R$ manipulator is shown schematically in Fig. 4.16. The nonzero link parameters are $\alpha_1 = -90°$, $d_2 = 1$, $\alpha_2 = 45°$, $d_3 = 1$, and $a_3 = 1$, and the mechanism is pictured in the configuration corresponding to $\Theta = [0, 0, 90°, 0]^T$. Each joint has $\pm 180°$ as limits. Find all values of θ_3 such that

$${}^0P_{4ORG} = [0.0, 1.0, 1.414]^T.$$

4.18 [15] Consider the RRP manipulator shown in Fig. 3.37. How many solutions do the (position) kinematic equations possess?

4.19 [15] Consider the RRR manipulator shown in Fig. 3.38. How many solutions do the (position) kinematic equations possess?

4.20 [15] Consider the RPP manipulator shown in Fig. 3.39. How many solutions do the (position) kinematic equations possess?

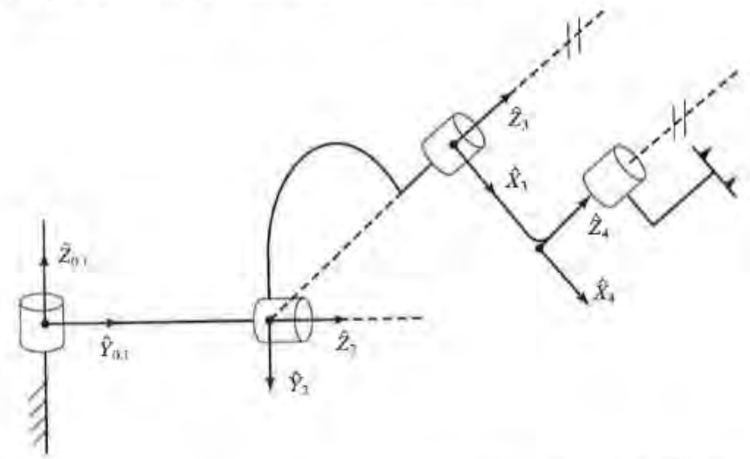


FIGURE 4.16: A $4R$ manipulator shown in the position $\Theta = [0, 0, 90°, 0]^T$ (Exercise 4.17).

4.21 [15] Consider the PRR manipulator shown in Fig. 3.40. How many solutions do the (position) kinematic equations possess?

4.22 [15] Consider the PPP manipulator shown in Fig. 3.41. How many solutions do the (position) kinematic equations possess?

4.23 [38] The following kinematic equations arise in a certain problem:

$$\sin\xi = a\sin\theta + b,$$
$$\sin\phi = c\cos\theta + d,$$
$$\psi = \xi + \phi.$$

Given a, b, c, d, and ψ, show that, in the general case, there are four solutions for θ. Give a special condition under which there are just two solutions for θ.

4.24 [20] Given the description of link frame $\{i\}$ in terms of link frame $\{i-1\}$, find the four Denavit–Hartenberg parameters as functions of the elements of ${}^{i-1}_{i}T$.

PROGRAMMING EXERCISE (PART 4)

1. Write a subroutine to calculate the inverse kinematics for the three-link manipulator of Section 4.4. The routine should pass arguments in the form

```
Procedure INVKIN(VAR wrelb: frame; VAR current, near, far: vec3;
VAR sol: boolean);
```

where "wrelb," an input, is the wrist frame specified relative to the base frame; "current," an input, is the current position of the robot (given as a vector of joint angles); "near" is the nearest solution; "far" is the second solution; and "sol" is a flag that indicates whether solutions were found. (sol = FALSE if no solutions were found). The link lengths (meters) are

$$l_1 = l_2 = 0.5.$$

The joint ranges of motion are

$$-170° \le \theta_i \le 170°.$$

Test your routine by calling it back-to-back with KIN to demonstrate that they are indeed inverses of one another.

2. A tool is attached to link 3 of the manipulator. This tool is described by ${}^{W}_{T}T$, the tool frame relative to the wrist frame. Also, a user has described his work area, the station frame relative to the base of the robot, as ${}^{B}_{S}T$. Write the subroutine

```
Procedure SOLVE(VAR trels: frame; VAR current, near, far: vec3;
VAR sol: boolean);
```

where "trels" is the $\{T\}$ frame specified relative to the $\{S\}$ frame. Other parameters are exactly as in the `INVKIN` subroutine. The definitions of $\{T\}$ and $\{S\}$ should be globally defined variables or constants. SOLVE should use calls to `TMULT`, `TINVERT`, and `INVKIN`.

3. Write a main program that accepts a goal frame specified in terms of x, y, and ϕ. This goal specification is $\{T\}$ relative to $\{S\}$, which is the way the user wants to specify goals.

The robot is using the same tool in the same working area as in Programming Exercise (Part 2), so $\{T\}$ and $\{S\}$ are defined as

$${}^{W}_{T}T = [x\ y\ \theta] = [0.1\ 0.2\ 30.0],$$
$${}^{B}_{S}T = [x\ y\ \theta] = [-0.1\ 0.3\ 0.0].$$

Calculate the joint angles for each of the following three goal frames:

$$[x_1\ y_1\ \phi_1] = [0.0\ 0.0\ -90.0],$$
$$[x_2\ y_2\ \phi_2] = [0.6\ -0.3\ 45.0],$$
$$[x_3\ y_3\ \phi_3] = [-0.4\ 0.3\ 120.0],$$
$$[x_4\ y_4\ \phi_4] = [0.8\ 1.4\ 30.0].$$

Assume that the robot will start with all angles equal to 0.0 and move to these three goals in sequence. The program should find the nearest solution with respect to the previous goal point. You should call `SOLVE` and `WHERE` back-to-back to make sure they are truly inverse functions.

MATLAB EXERCISE 4

This exercise focuses on the inverse-pose kinematics solution for the planar 3-DOF, 3R robot. (See Figures 3.6 and 3.7; the DH parameters are given in Figure 3.8.) The following fixed-length parameters are given: $L_1 = 4$, $L_2 = 3$, and $L_3 = 2$(m).

a) Analytically derive, by hand, the inverse-pose solution for this robot: Given ${}^{0}_{H}T$, calculate all possible multiple solutions for $(\theta_1\ \theta_2\ \theta_3)$. (Three methods are presented in the text—choose one of these.) Hint: To simplify the equations, first calculate ${}^{0}_{3}T$ from ${}^{0}_{H}T$ and L_3.

b) Develop a MATLAB program to solve this planar 3R robot inverse-pose kinematics problem completely (i.e., to give all multiple solutions). Test your program, using the following input cases:

i) $${}^0_H T = \begin{bmatrix} 1 & 0 & 0 & 9 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

ii) $${}^0_H T = \begin{bmatrix} 0.5 & -0.866 & 0 & 7.5373 \\ 0.866 & 0.6 & 0 & 3.9266 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

iii) $${}^0_H T = \begin{bmatrix} 0 & 1 & 0 & -3 \\ -1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

iv) $${}^0_H T = \begin{bmatrix} 0.866 & 0.5 & 0 & -3.1245 \\ -0.5 & 0.866 & 0 & 9.1674 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For all cases, employ a circular check to validate your results: Plug each resulting set of joint angles (for each of the multiple solutions) back into the forward-pose kinematics MATLAB program to demonstrate that you get the originally commanded ${}^0_H T$.

c) Check all results by means of the Corke MATLAB Robotics Toolbox. Try function *ikine()*.

CHAPTER 5

Jacobians: velocities and static forces

5.1 INTRODUCTION

In this chapter, we expand our consideration of robot manipulators beyond static-positioning problems. We examine the notions of linear and angular velocity of a rigid body and use these concepts to analyze the motion of a manipulator. We also will consider forces acting on a rigid body, and then use these ideas to study the application of static forces with manipulators.

It turns out that the study of both velocities and static forces leads to a matrix entity called the **Jacobian**[1] of the manipulator, which will be introduced in this chapter.

The field of kinematics of mechanisms is not treated in great depth here. For the most part, the presentation is restricted to only those ideas which are fundamental to the particular problem of robotics. The interested reader is urged to study further from any of several texts on mechanics [1–3].

5.2 NOTATION FOR TIME-VARYING POSITION AND ORIENTATION

Before investigating the description of the motion of a rigid body, we briefly discuss some basics: the differentiation of vectors, the representation of angular velocity, and notation.

[1] Mathematicians call it the "Jacobian matrix," but roboticists usually shorten it to simply "Jacobian."

Jacobian matrix with the new configuration Θ_{new} before completing the resolved-rate calculations for the next time step.

Develop a MATLAB program to calculate the Jacobian matrix and to simulate resolved-rate control for the planar 3R robot. Given the robot lengths $L_1 = 4$, $L_2 = 3$, and $L_3 = 2$ (m); the initial joint angles $\Theta = \{\theta_1 \ \theta_2 \ \theta_3\}^T = \{10^\circ \ 20^\circ \ 30^\circ\}^T$, and the constant commanded Cartesian rates ${}^0\{\dot{X}\} = \{\dot{x} \ \dot{y} \ \omega_z\}^T = \{0.2 \ -0.3 \ -0.2\}^T$ (m/s, m/s, rad/s), simulate for exactly 5 sec, using time steps of exactly $dt = 0.1$ sec. In the same program loop, calculate the inverse-statics problem—that is, calculate the joint torques $\mathrm{T} = \{\tau_1 \ \tau_2 \ \tau_3\}^T$ (Nm), given the constant commanded Cartesian wrench ${}^0\{W\} = \{f_x \ f_y \ m_z\}^T = \{1 \ 2 \ 3\}^T$ (N, N, Nm). Also, in the same loop, animate the robot to the screen during each time step, so that you can watch the simulated motion to verify that it is correct.

a) For the specific numbers assigned, present five plots (each set on a separate graph, please):

1. the three active joint rates $\dot{\Theta} = \{\dot{\theta}_1 \ \dot{\theta}_2 \ \dot{\theta}_3\}^T$ vs. time;
2. the three active joint angles $\Theta = \{\theta_1 \ \theta_2 \ \theta_3\}^T$ vs. time;
3. the three Cartesian components of ${}^0_H T$, $X = \{x \ y \ \phi\}^T$ (rad is fine for ϕ so that it will fit) vs. time;
4. the Jacobian matrix determinant $|J|$ vs. time—comment on nearness to singularities during the simulated resolved-rate motion;
5. the three active joint torques $\mathrm{T} = \{\tau_1 \ \tau_2 \ \tau_3\}^T$ vs. time.

Carefully label (by hand is fine!) each component on each plot; also, label the axes with names and units.

b) Check your Jacobian matrix results for the initial and final joint-angle sets by means of the Corke MATLAB Robotics Toolbox. Try function *jacob0()*. **Caution**: The toolbox Jacobian functions are for motion of {3} with respect to {0}, not for {H} with respect to {0} as in the problem assignment. The preceding function gives the Jacobian result in {0} coordinates; *jacobn()* would give results in {3} coordinates.

CHAPTER 6

Manipulator dynamics

6.1 INTRODUCTION

Our study of manipulators so far has focused on kinematic considerations only. We have studied static positions, static forces, and velocities; but we have never considered *the forces required to cause motion*. In this chapter, we consider the equations of motion for a manipulator—the way in which motion of the manipulator arises from torques applied by the actuators or from external forces applied to the manipulator.

Dynamics of mechanisms is a field in which many books have been written. Indeed, one can spend years studying the field. Obviously, we cannot cover the material in the completeness it deserves. However, certain formulations of the dynamics problem seem particularly well suited to application to manipulators. In particular, methods which make use of the serial-chain nature of manipulators are natural candidates for our study.

There are two problems related to the dynamics of a manipulator that we wish to solve. In the first problem, we are given a trajectory point, Θ, $\dot{\Theta}$, and $\ddot{\Theta}$, and we wish to find the required vector of joint torques, τ. This formulation of dynamics is useful for the problem of controlling the manipulator (Chapter 10). The second problem is to calculate how the mechanism will move under application of a set of joint torques. That is, given a torque vector, τ, calculate the resulting motion of the manipulator, Θ, $\dot{\Theta}$, and $\ddot{\Theta}$. This is useful for simulating the manipulator.

6.2 ACCELERATION OF A RIGID BODY

We now extend our analysis of rigid-body motion to the case of accelerations. At any instant, the linear and angular velocity vectors have derivatives that are called the linear and angular accelerations, respectively. That is,

$$ {}^B\dot{V}_Q = \frac{d}{dt}\,{}^BV_Q = \lim_{\Delta t \to o} \frac{{}^BV_Q(t+\Delta t) - {}^BV_Q(t)}{\Delta t}. \tag{6.1} $$

and

$$ {}^A\dot{\Omega}_B = \frac{d}{dt}\,{}^A\Omega_B = \lim_{\Delta t \to o} \frac{{}^A\Omega_B(t+\Delta t) - {}^A\Omega_B(t)}{\Delta t}. \tag{6.2} $$

As with velocities, when the reference frame of the differentiation is understood to be some universal reference frame, $\{U\}$, we will use the notation

$$ \dot{v}_A = {}^U\dot{V}_{AORG} \tag{6.3} $$

and

$$ \dot{\omega}_A = {}^U\dot{\Omega}_A. \tag{6.4} $$

Linear acceleration

We start by restating (5.12), an important result from Chapter 5, which describes the velocity of a vector ${}^BQ$ as seen from frame $\{A\}$ when the origins are coincident:

$$ {}^AV_Q = {}^A_BR\,{}^BV_Q + {}^A\Omega_B \times {}^A_BR\,{}^BQ. \tag{6.5} $$

The left-hand side of this equation describes how ${}^AQ$ is changing in time. So, because origins are coincident, we could rewrite (6.5) as

$$ \frac{d}{dt}({}^A_BR\,{}^BQ) = {}^A_BR\,{}^BV_Q + {}^A\Omega_B \times {}^A_BR\,{}^BQ. \tag{6.6} $$

This form of the equation will be useful when deriving the corresponding acceleration equation.

By differentiating (6.5), we can derive expressions for the acceleration of ${}^BQ$ as viewed from $\{A\}$ when the origins of $\{A\}$ and $\{B\}$ coincide:

$$ {}^A\dot{V}_Q = \frac{d}{dt}({}^A_BR\,{}^BV_Q) + {}^A\dot{\Omega}_B \times {}^A_BR\,{}^BQ + {}^A\Omega_B \times \frac{d}{dt}({}^A_BR\,{}^BQ). \tag{6.7} $$

Now we apply (6.6) twice—once to the first term, and once to the last term. The right-hand side of equation (6.7) becomes

$$ \begin{aligned} {}^A_BR\,{}^B\dot{V}_Q &+ {}^A\Omega_B \times {}^A_BR\,{}^BV_Q + {}^A\dot{\Omega}_B \times {}^A_BR\,{}^BQ \\ &+ {}^A\Omega_B \times ({}^A_BR\,{}^BV_Q + {}^A\Omega_B \times {}^A_BR\,{}^BQ). \end{aligned} \tag{6.8} $$

Combining two terms, we get

$$ {}^A_BR\,{}^B\dot{V}_Q + 2{}^A\Omega_B \times {}^A_BR\,{}^BV_Q + {}^A\dot{\Omega}_B \times {}^A_BR\,{}^BQ + {}^A\Omega_B \times ({}^A\Omega_B \times {}^A_BR\,{}^BQ). \tag{6.9} $$

Finally, to generalize to the case in which the origins are not coincident, we add one term which gives the linear acceleration of the origin of $\{B\}$, resulting in the final general formula:

$$ \begin{aligned} {}^A\dot{V}_{BORG} + {}^A_BR\,{}^B\dot{V}_Q + 2{}^A\Omega_B \times {}^A_BR\,{}^BV_Q + {}^A\dot{\Omega}_B \times {}^A_BR\,{}^BQ \\ + {}^A\Omega_B \times ({}^A\Omega_B \times {}^A_BR\,{}^BQ). \end{aligned} \tag{6.10} $$

A particular case that is worth pointing out is when ${}^BQ$ is constant, or

$$ {}^BV_Q = {}^B\dot{V}_Q = 0. \tag{6.11} $$

In this case, (6.10) simplifies to

$$ {}^A\dot{V}_Q = {}^A\dot{V}_{BORG} + {}^A\Omega_B \times ({}^A\Omega_B \times {}^A_BR\,{}^BQ) + {}^A\dot{\Omega}_B \times {}^A_BR\,{}^BQ. \tag{6.12} $$

We will use this result in calculating the linear acceleration of the links of a manipulator with rotational joints. When a prismatic joint is present, the more general form of (6.10) will be used.

Angular acceleration

Consider the case in which $\{B\}$ is rotating relative to $\{A\}$ with ${}^A\Omega_B$ and $\{C\}$ is rotating relative to $\{B\}$ with ${}^B\Omega_C$. To calculate ${}^A\Omega_C$, we sum the vectors in frame $\{A\}$:

$$ {}^A\Omega_C = {}^A\Omega_B + {}^A_BR\,{}^B\Omega_C. \tag{6.13} $$

By differentiating, we obtain

$$ {}^A\dot{\Omega}_C = {}^A\dot{\Omega}_B + \frac{d}{dt}({}^A_BR\,{}^B\Omega_C). \tag{6.14} $$

Now, applying (6.6) to the last term of (6.14), we get

$$ {}^A\dot{\Omega}_C = {}^A\dot{\Omega}_B + {}^A_BR\,{}^B\dot{\Omega}_C + {}^A\Omega_B \times {}^A_BR\,{}^B\Omega_C. \tag{6.15} $$

We will use this result to calculate the angular acceleration of the links of a manipulator.

6.3 MASS DISTRIBUTION

In systems with a single degree of freedom, we often talk about the mass of a rigid body. In the case of rotational motion about a single axis, the notion of the *moment of inertia* is a familiar one. For a rigid body that is free to move in three dimensions, there are infinitely many possible rotation axes. In the case of rotation about an arbitrary axis, we need a complete way of characterizing the mass distribution of a rigid body. Here, we introduce the **inertia tensor**, which, for our purposes, can be thought of as a generalization of the scalar moment of inertia of an object.

We shall now define a set of quantities that give information about the distribution of mass of a rigid body relative to a reference frame. Figure 6.1 shows a rigid body with an attached frame. Inertia tensors can be defined relative to any frame, but we will always consider the case of an inertia tensor defined for a frame

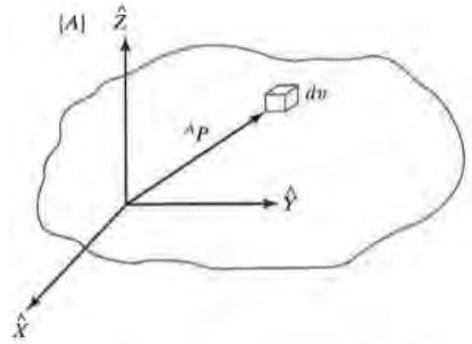


FIGURE 6.1: The inertia tensor of an object describes the object's mass distribution. Here, the vector ^AP locates the differential volume element, dv.

attached to the rigid body. Where it is important, we will indicate, with a leading superscript, the frame of reference of a given inertia tensor. The inertia tensor relative to frame $\{A\}$ is expressed in the matrix form as the 3×3 matrix

$$^AI = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}. \tag{6.16}$$

where the scalar elements are given by

$$\begin{aligned} I_{xx} &= \iiint_V (y^2 + z^2)\rho dv, \\ I_{yy} &= \iiint_V (x^2 + z^2)\rho dv, \\ I_{zz} &= \iiint_V (x^2 + y^2)\rho dv, \\ I_{xy} &= \iiint_V xy\rho dv, \\ I_{xz} &= \iiint_V xz\rho dv, \\ I_{yz} &= \iiint_V yz\rho dv, \end{aligned} \tag{6.17}$$

in which the rigid body is composed of differential volume elements, dv, containing material of density ρ. Each volume element is located with a vector, $^AP = [xyz]^T$, as shown in Fig. 6.1.

The elements I_{xx}, I_{yy}, and I_{zz} are called the **mass moments of inertia**. Note that, in each case, we are integrating the mass elements, ρdv, times the squares of the perpendicular distances from the corresponding axis. The elements with mixed indices are called the **mass products of inertia**. This set of six independent quantities

will, for a given body, depend on the position and orientation of the frame in which they are defined. If we are free to choose the orientation of the reference frame, it is possible to cause the products of inertia to be zero. The axes of the reference frame when so aligned are called the **principal axes** and the corresponding mass moments are the **principal moments** of inertia.

EXAMPLE 6.1

Find the inertia tensor for the rectangular body of uniform density ρ with respect to the coordinate system shown in Fig. 6.2.

First, we compute I_{xx}. Using volume element $dv = dx\,dy\,dz$, we get

$$\begin{aligned} I_{xx} &= \int_0^h \int_0^l \int_0^w (y^2 + z^2)\rho\,dx\,dy\,dz \\ &= \int_0^h \int_0^l (y^2 + z^2)\omega\rho dy\,dz \\ &= \int_0^h \left(\frac{l^3}{3} + z^2 l\right)\omega\rho dz \\ &= \left(\frac{hl^3\omega}{3} + \frac{h^3 l\omega}{3}\right)\rho \\ &= \frac{m}{3}(l^2 + h^2), \end{aligned} \tag{6.18}$$

where m is the total mass of the body. Permuting the terms, we can get I_{yy} and I_{zz} by inspection:

$$I_{yy} = \frac{m}{3}(w^2 + h^2) \tag{6.19}$$

and

$$I_{zz} = \frac{m}{3}(l^2 + w^2). \tag{6.20}$$

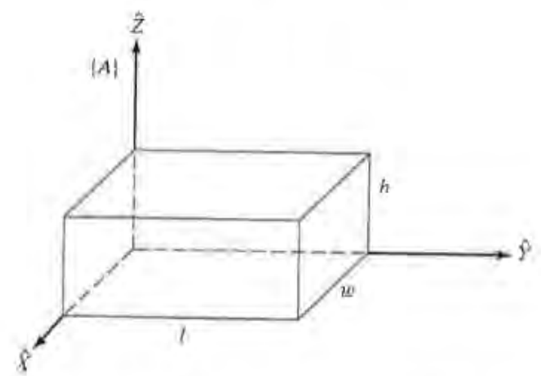


FIGURE 6.2: A body of uniform density.

We next compute I_{xy}:

$$\begin{aligned} I_{xy} &= \int_0^h \int_0^l \int_0^w xy\rho \, dx \, dy \, dz \\ &= \int_0^h \int_0^l \frac{\omega^2}{2} y\rho \, dy \, dz \\ &= \int_0^h \frac{\omega^2 l^2}{4} \rho dz \\ &= \frac{m}{4}\omega l. \end{aligned} \tag{6.21}$$

Permuting the terms, we get

$$I_{xz} = \frac{m}{4}h\omega \tag{6.22}$$

and

$$I_{yz} = \frac{m}{4}hl. \tag{6.23}$$

Hence, the inertia tensor for this object is

$${}^A I = \begin{bmatrix} \frac{m}{3}(l^2+h^2) & -\frac{m}{4}\omega l & -\frac{m}{4}h\omega \\ -\frac{m}{4}\omega l & \frac{m}{3}(\omega^2+h^2) & -\frac{m}{4}hl \\ -\frac{m}{4}h\omega & -\frac{m}{4}hl & \frac{m}{3}(l^2+\omega^2) \end{bmatrix}. \tag{6.24}$$

As noted, the inertia tensor is a function of the location and orientation of the reference frame. A well-known result, the **parallel-axis theorem**, is one way of computing how the inertia tensor changes under *translations* of the reference coordinate system. The parallel-axis theorem relates the inertia tensor in a frame with origin at the center of mass to the inertia tensor with respect to another reference frame. Where $\{C\}$ is located at the center of mass of the body, and $\{A\}$ is an arbitrarily translated frame, the theorem can be stated [1] as

$$\begin{aligned} {}^A I_{zz} &= {}^C I_{zz} + m(x_c^2 + y_c^2), \\ {}^A I_{xy} &= {}^C I_{xy} - m x_c y_c, \end{aligned} \tag{6.25}$$

where $P_c = [x_c, y_c, z_c]^T$ locates the center of mass relative to $\{A\}$. The remaining moments and products of inertia are computed from permutations of x, y, and z in (6.25). The theorem may be stated in vector–matrix form as

$${}^A I = {}^C I + m[P_c^T P_c I_3 - P_c P_c^T], \tag{6.26}$$

where I_3 is the 3×3 identity matrix.

EXAMPLE 6.2

Find the inertia tensor for the same solid body described for Example 6.1 when it is described in a coordinate system with origin at the body's center of mass.

We can apply the parallel-axis theorem, (6.25), where

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \omega \\ l \\ h \end{bmatrix}.$$

Next, we find

$$\begin{aligned} {}^C I_{zz} &= \frac{m}{12}(\omega^2 + l^2), \\ {}^C I_{xy} &= 0. \end{aligned} \tag{6.27}$$

The other elements are found by symmetry. The resulting inertia tensor written in the frame at the center of mass is

$${}^C I = \begin{bmatrix} \frac{m}{12}(h^2+l^2) & 0 & 0 \\ 0 & \frac{m}{12}(\omega^2+h^2) & 0 \\ 0 & 0 & \frac{m}{12}(l^2+\omega^2) \end{bmatrix}. \tag{6.28}$$

The result is diagonal, so frame $\{C\}$ must represent the principal axes of this body.

Some additional facts about inertia tensors are as follows:

1. If two axes of the reference frame form a plane of symmetry for the mass distribution of the body, the products of inertia having as an index the coordinate that is normal to the plane of symmetry will be zero.
2. Moments of inertia must always be positive. Products of inertia may have either sign.
3. The sum of the three moments of inertia is invariant under orientation changes in the reference frame.
4. The eigenvalues of an inertia tensor are the principal moments for the body. The associated eigenvectors are the principal axes.

Most manipulators have links whose geometry and composition are somewhat complex, so that the application of (6.17) is difficult in practice. A pragmatic option is actually to measure rather than to calculate the moment of inertia of each link by using a measuring device (e.g., an *inertia pendulum*).

6.4 NEWTON'S EQUATION, EULER'S EQUATION

We will consider each link of a manipulator as a rigid body. If we know the location of the center of mass and the inertia tensor of the link, then its mass distribution is completely characterized. In order to move the links, we must accelerate and decelerate them. The forces required for such motion are a function of the acceleration desired and of the mass distribution of the links. Newton's equation, along with its rotational analog, Euler's equation, describes how forces, inertias, and accelerations relate.

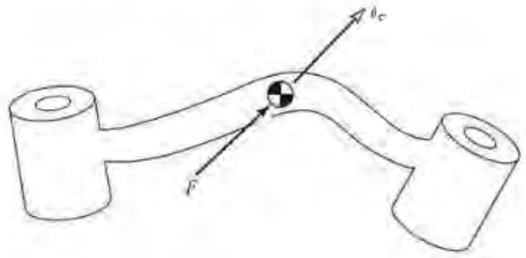


FIGURE 6.3: A force F acting at the center of mass of a body causes the body to accelerate at $\dot{v}_C$.

Newton's equation

Figure 6.3 shows a rigid body whose center of mass is accelerating with acceleration $\dot{v}_C$. In such a situation, the force, F, acting at the center of mass and causing this acceleration is given by Newton's equation

$$F = m\dot{v}_C. \tag{6.29}$$

where m is the total mass of the body.

Euler's equation

Figure 6.4 shows a rigid body rotating with angular velocity ω and with angular acceleration $\dot{\omega}$. In such a situation, the moment N, which must be acting on the body to cause this motion, is given by Euler's equation

$$N = {}^C I\dot{\omega} + \omega \times {}^C I\omega, \tag{6.30}$$

where ${}^C I$ is the inertia tensor of the body written in a frame, $\{C\}$, whose origin is located at the center of mass.

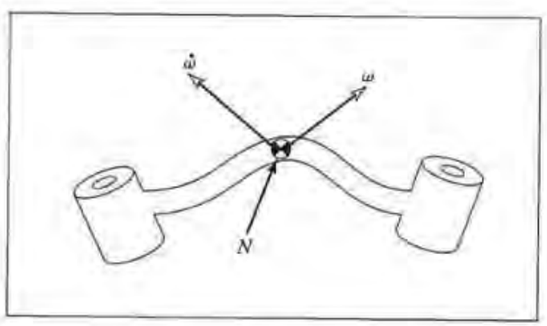


FIGURE 6.4: A moment N is acting on a body, and the body is rotating with velocity ω and accelerating at $\dot{\omega}$.

6.5 ITERATIVE NEWTON–EULER DYNAMIC FORMULATION

We now consider the problem of computing the torques that correspond to a given trajectory of a manipulator. We assume we know the position, velocity, and acceleration of the joints, $(\Theta, \dot{\Theta}, \ddot{\Theta})$. With this knowledge, and with knowledge of the kinematics and the mass-distribution information of the robot, we can calculate the joint torques required to cause this motion. The algorithm presented is based upon the method published by Luh, Walker, and Paul in [2].

Outward iterations to compute velocities and accelerations

In order to compute inertial forces acting on the links, it is necessary to compute the rotational velocity and linear and rotational acceleration of the center of mass of each link of the manipulator at any given instant. These computations will be done in an iterative way, starting with link 1 and moving successively, link by link, *outward* to link n.

The "propagation" of rotational velocity from link to link was discussed in Chapter 5 and is given (for joint $i+1$ rotational) by

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}. \tag{6.31}$$

From (6.15), we obtain the equation for transforming angular acceleration from one link to the next:

$${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}. \tag{6.32}$$

When joint $i+1$ is prismatic, this simplifies to

$${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i. \tag{6.33}$$

The linear acceleration of each link-frame origin is obtained by the application of (6.12):

$${}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R[{}^{i}\omega_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i]. \tag{6.34}$$

For prismatic joint $i+1$, (6.34) becomes (from (6.10))

$$\begin{aligned}{}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i) \\ + 2{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}.\end{aligned} \tag{6.35}$$

We also will need the linear acceleration of the center of mass of each link, which also can be found by applying (6.12):

$${}^{i}\dot{v}_{C_i} = {}^{i}\dot{\omega}_i \times {}^{i}P_{C_i} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{C_i}) + {}^{i}\dot{v}_i. \tag{6.36}$$

Here, we imagine a frame, $\{C_i\}$, attached to each link, having its origin located at the center of mass of the link and having the same orientation as the link frame, $\{i\}$. Equation (6.36) doesn't involve joint motion at all and so is valid for joint $i+1$, regardless of whether it is revolute or prismatic.

Note that the application of the equations to link 1 is especially simple, because ${}^{0}\omega_0 = {}^{0}\dot{\omega}_0 = 0$.

The force and torque acting on a link

Having computed the linear and angular accelerations of the mass center of each link, we can apply the Newton–Euler equations (Section 6.4) to compute the inertial force and torque acting at the center of mass of each link. Thus we have

$$F_i = m\dot{v}_{C_i},$$
$$N_i = {}^{C_i}I\dot{\omega}_i + \omega_i \times {}^{C_i}I\omega_i, \tag{6.37}$$

where $\{C_i\}$ has its origin at the center of mass of the link and has the same orientation as the link frame, $\{i\}$.

Inward iterations to compute forces and torques

Having computed the forces and torques acting on each link, we now need to calculate the joint torques that will result in these net forces and torques being applied to each link.

We can do this by writing a force-balance and moment-balance equation based on a free-body diagram of a typical link. (See Fig. 6.5.) Each link has forces and torques exerted on it by its neighbors and in addition experiences an inertial force and torque. In Chapter 5, we defined special symbols for the force and torque exerted by a neighbor link, which we repeat here:

f_i = force exerted on link i by link $i-1$,
n_i = torque exerted on link i by link $i-1$.

By summing the forces acting on link i, we arrive at the force-balance relationship:

$${}^iF_i = {}^if_i - {}^i_{i+1}R\,{}^{i+1}f_{i+1}. \tag{6.38}$$

By summing torques about the center of mass and setting them equal to zero, we arrive at the torque-balance equation:

$${}^iN_i = {}^in_i - {}^in_{i+1} + (-{}^iP_{C_i}) \times {}^if_i - ({}^iP_{i+1} - {}^iP_{C_i}) \times {}^if_{i+1}. \tag{6.39}$$

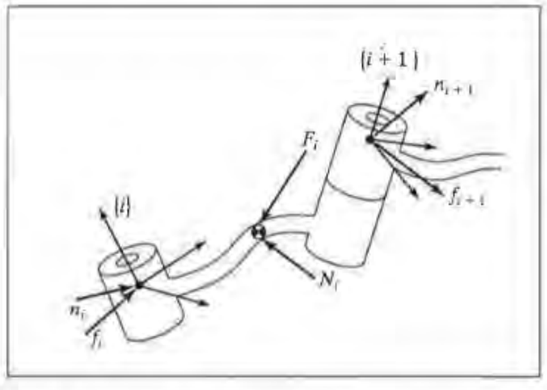


FIGURE 6.5: The force balance, including inertial forces, for a single manipulator link.

Using the result from the force-balance relation (6.38) and adding a few rotation matrices, we can write (6.39) as

$${}^iN_i = {}^in_i - {}^i_{i+1}R\,{}^{i+1}n_{i+1} - {}^iP_{C_i} \times {}^iF_i - {}^iP_{i+1} \times {}^i_{i+1}R\,{}^{i+1}f_{i+1}. \tag{6.40}$$

Finally, we can rearrange the force and torque equations so that they appear as iterative relationships from higher numbered neighbor to lower numbered neighbor:

$${}^if_i = {}^i_{i+1}R\,{}^{i+1}f_{i+1} + {}^iF_i, \tag{6.41}$$
$${}^in_i = {}^iN_i + {}^i_{i+1}R\,{}^{i+1}n_{i+1} + {}^iP_{C_i} \times {}^iF_i + {}^iP_{i+1} \times {}^i_{i+1}R\,{}^{i+1}f_{i+1}. \tag{6.42}$$

These equations are evaluated link by link, starting from link n and working inward toward the base of the robot. These *inward force iterations* are analogous to the static force iterations introduced in Chapter 5, except that inertial forces and torques are now considered at each link.

As in the static case, the required joint torques are found by taking the $\hat{Z}$ component of the torque applied by one link on its neighbor:

$$\tau_i = {}^in_i^T\,{}^i\hat{Z}_i. \tag{6.43}$$

For joint i prismatic, we use

$$\tau_i = {}^if_i^T\,{}^i\hat{Z}_i, \tag{6.44}$$

where we have used the symbol τ for a linear actuator force.

Note that, for a robot moving in free space, ${}^{N+1}f_{N+1}$ and ${}^{N+1}n_{N+1}$ are set equal to zero, and so the first application of the equations for link n is very simple. If the robot is in contact with the environment, the forces and torques due to this contact can be included in the force balance by having nonzero ${}^{N+1}f_{N+1}$ and ${}^{N+1}n_{N+1}$.

The iterative Newton–Euler dynamics algorithm

The complete algorithm for computing joint torques from the motion of the joints is composed of two parts. First, link velocities and accelerations are iteratively computed from link 1 out to link n and the Newton–Euler equations are applied to each link. Second, forces and torques of interaction and joint actuator torques are computed recursively from link n back to link 1. The equations are summarized next for the case of all joints rotational:

Outward iterations: $i : 0 \rightarrow 5$

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}, \tag{6.45} $$

$$ {}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}, \tag{6.46} $$

$$ {}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i), \tag{6.47} $$

$$ {}^{i+1}\dot{v}_{C_{i+1}} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1}, \tag{6.48} $$

$$ {}^{i+1}F_{i+1} = m_{i+1}\,{}^{i+1}\dot{v}_{C_{i+1}}, \tag{6.49} $$

$$ {}^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1}. \tag{6.50} $$

Inward iterations: $i : 6 \rightarrow 1$

$$ {}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i, \tag{6.51} $$

$$ {}^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\,{}^{i+1}f_{i+1}, \tag{6.52} $$

$$ \tau_i = {}^{i}n_i^T\,{}^{i}\hat{Z}_i. \tag{6.53} $$

Inclusion of gravity forces in the dynamics algorithm

The effect of gravity loading on the links can be included quite simply by setting ${}^{0}\dot{v}_0 = G$, where G has the magnitude of the gravity vector but points in the opposite direction. This is equivalent to saying that the base of the robot is accelerating upward with 1 g acceleration. This fictitious upward acceleration causes exactly the same effect on the links as gravity would. So, with no extra computational expense, the gravity effect is calculated.

6.6 ITERATIVE VS. CLOSED FORM

Equations (6.46) through (6.53) give a computational scheme whereby, given the joint positions, velocities, and accelerations, we can compute the required joint torques. As with our development of equations to compute the Jacobian in Chapter 5, these relations can be used in two ways: as a numerical computational algorithm, or as an algorithm used analytically to develop symbolic equations.

Use of the equations as a numerical computational algorithm is attractive because the equations apply to any robot. Once the inertia tensors, link masses, P_{C_i} vectors, and ${}^{i+1}_{i}R$ matrices are specified for a particular manipulator, the equations can be applied directly to compute the joint torques corresponding to any motion.

However, we often are interested in obtaining better insight into the structure of the equations. For example, what is the form of the gravity terms? How does the magnitude of the gravity effects compare with the magnitude of the inertial effects? To investigate these and other questions, it is often useful to write closed-form dynamic equations. These equations can be derived by applying the recursive

Newton–Euler equations symbolically to Θ, $\dot{\Theta}$, and $\ddot{\Theta}$. This is analogous to what we did in Chapter 5 to derive the symbolic form of the Jacobian.

6.7 AN EXAMPLE OF CLOSED-FORM DYNAMIC EQUATIONS

Here we compute the closed-form dynamic equations for the two-link planar manipulator shown in Fig. 6.6. For simplicity, we assume that the mass distribution is extremely simple: All mass exists as a point mass at the distal end of each link. These masses are m_1 and m_2.

First, we determine the values of the various quantities that will appear in the recursive Newton–Euler equations. The vectors that locate the center of mass for each link are

$$ {}^{1}P_{C_1} = l_1\hat{X}_1, $$

$$ {}^{2}P_{C_2} = l_2\hat{X}_2. $$

Because of the point-mass assumption, the inertia tensor written at the center of mass for each link is the zero matrix:

$$ {}^{C_1}I_1 = 0, $$

$$ {}^{C_2}I_2 = 0. $$

There are no forces acting on the end-effector, so we have

$$ f_3 = 0, $$

$$ n_3 = 0. $$

The base of the robot is not rotating; hence, we have

$$ \omega_0 = 0, $$

$$ \dot{\omega}_0 = 0. $$

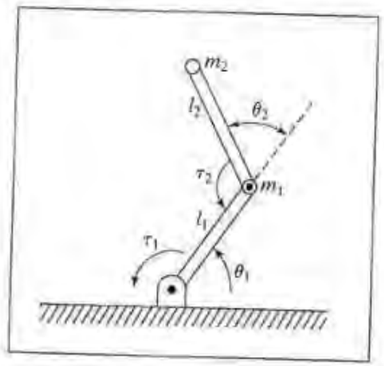


FIGURE 6.6: Two-link planar manipulator with point masses at distal ends of links.

To include gravity forces, we will use

$$^0\dot{v}_0 = g\hat{Y}_0.$$

The rotation between successive link frames is given by

$$^{i}_{i+1}R = \begin{bmatrix} c_{i+1} & -s_{i+1} & 0.0 \\ s_{i+1} & c_{i+1} & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix},$$

$$^{i+1}_{i}R = \begin{bmatrix} c_{i+1} & s_{i+1} & 0.0 \\ -s_{i+1} & c_{i+1} & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}.$$

We now apply equations (6.46) through (6.53).

The outward iterations for link 1 are as follows:

$$^1\omega_1 = \dot{\theta}_1\,{}^1\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix},$$

$$^1\dot{\omega}_1 = \ddot{\theta}_1\,{}^1\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix},$$

$$^1\dot{v}_1 = \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} = \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix},$$

$$^1\dot{v}_{C_1} = \begin{bmatrix} 0 \\ l_1\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} -l_1\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix} = \begin{bmatrix} -l_1\dot{\theta}_1^2 + gs_1 \\ l_1\ddot{\theta}_1 + gc_1 \\ 0 \end{bmatrix},$$

$$^1F_1 = \begin{bmatrix} -m_1l_1\dot{\theta}_1^2 + m_1gs_1 \\ m_1l_1\ddot{\theta}_1 + m_1gc_1 \\ 0 \end{bmatrix},$$

$$^1N_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{6.54}$$

The outward iterations for link 2 are as follows:

$$^2\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix},$$

$$^2\dot{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 + \ddot{\theta}_2 \end{bmatrix},$$

$$^2\dot{v}_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -l_1\dot{\theta}_1^2 + gs_1 \\ l_1\ddot{\theta}_1 + gc_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1\ddot{\theta}_1s_2 - l_1\dot{\theta}_1^2c_2 + gs_{12} \\ l_1\ddot{\theta}_1c_2 + l_1\dot{\theta}_1^2s_2 + gc_{12} \\ 0 \end{bmatrix},$$

$$^2\dot{v}_{C_2} = \begin{bmatrix} 0 \\ l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix} + \begin{bmatrix} -l_2(\dot{\theta}_1 + \dot{\theta}_2)^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l_1\ddot{\theta}_1s_2 - l_1\dot{\theta}_1^2c_2 + gs_{12} \\ l_1\ddot{\theta}_1c_2 + l_1\dot{\theta}_1^2s_2 + gc_{12} \\ 0 \end{bmatrix}, \tag{6.55}$$

$$^2F_2 = \begin{bmatrix} m_2l_1\ddot{\theta}_1s_2 - m_2l_1\dot{\theta}_1^2c_2 + m_2gs_{12} - m_2l_2(\dot{\theta}_1 + \dot{\theta}_2)^2 \\ m_2l_1\ddot{\theta}_1c_2 + m_2l_1\dot{\theta}_1^2s_2 + m_2gc_{12} + m_2l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix},$$

$$^2N_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

The inward iterations for link 2 are as follows:

$$^2f_2 = {}^2F_2,$$

$$^2n_2 = \begin{bmatrix} 0 \\ 0 \\ m_2l_1l_2c_2\ddot{\theta}_1 + m_2l_1l_2s_2\dot{\theta}_1^2 + m_2l_2gc_{12} + m_2l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) \end{bmatrix}. \tag{6.56}$$

The inward iterations for link 1 are as follows:

$$^1f_1 = \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} m_2l_1s_2\ddot{\theta}_1 - m_2l_1c_2\dot{\theta}_1^2 + m_2gs_{12} - m_2l_2(\dot{\theta}_1 + \dot{\theta}_2)^2 \\ m_2l_1c_2\ddot{\theta}_1 + m_2l_1s_2\dot{\theta}_1^2 + m_2gc_{12} + m_2l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix} + \begin{bmatrix} -m_1l_1\dot{\theta}_1^2 + m_1gs_1 \\ m_1l_1\ddot{\theta}_1 + m_1gc_1 \\ 0 \end{bmatrix},$$

$$^1n_1 = \begin{bmatrix} 0 \\ 0 \\ m_2l_1l_2c_2\ddot{\theta}_1 + m_2l_1l_2s_2\dot{\theta}_1^2 + m_2l_2gc_{12} + m_2l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_1l_1^2\ddot{\theta}_1 + m_1l_1gc_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_2l_1^2\ddot{\theta}_1 - m_2l_1l_2s_2(\dot{\theta}_1 + \dot{\theta}_2)^2 + m_2l_1gs_2s_{12} \\ +m_2l_1l_2c_2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2l_1gc_2c_{12} \end{bmatrix}. \tag{6.57}$$

Extracting the $\hat{Z}$ components of the ${}^{i}n_{i}$, we find the joint torques:

$$\tau_1 = m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 c_2(2\ddot{\theta}_1 + \ddot{\theta}_2) + (m_1 + m_2) l_1^2 \ddot{\theta}_1 - m_2 l_1 l_2 s_2 \dot{\theta}_2^2$$
$$-2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1,$$
$$\tau_2 = m_2 l_1 l_2 c_2 \ddot{\theta}_1 + m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2 g c_{12} + m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2). \quad (6.58)$$

Equations (6.58) give expressions for the torque at the actuators as a function of joint position, velocity, and acceleration. Note that these rather complex functions arose from one of the simplest manipulators imaginable. Obviously, the closed-form equations for a manipulator with six degrees of freedom will be quite complex.

6.8 THE STRUCTURE OF A MANIPULATOR'S DYNAMIC EQUATIONS

It is often convenient to express the dynamic equations of a manipulator in a single equation that hides some of the details, but shows some of the structure of the equations.

The state-space equation

When the Newton–Euler equations are evaluated symbolically for any manipulator, they yield a dynamic equation that can be written in the form

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta), \quad (6.59)$$

where $M(\Theta)$ is the $n \times n$ **mass matrix** of the manipulator, $V(\Theta, \dot{\Theta})$ is an $n \times 1$ vector of centrifugal and Coriolis terms, and $G(\Theta)$ is an $n \times 1$ vector of gravity terms. We use the term **state-space equation** because the term $V(\Theta, \dot{\Theta})$, appearing in (6.59), has both position and velocity dependence [3].

Each element of $M(\Theta)$ and $G(\Theta)$ is a complex function that depends on Θ, the position of all the joints of the manipulator. Each element of $V(\Theta, \dot{\Theta})$ is a complex function of both Θ and $\dot{\Theta}$.

We may separate the various types of terms appearing in the dynamic equations and form the mass matrix of the manipulator, the centrifugal and Coriolis vector, and the gravity vector.

EXAMPLE 6.3

Give $M(\Theta)$, $V(\Theta, \dot{\Theta})$, and $G(\Theta)$ for the manipulator of Section 6.7.

Equation (6.59) defines the manipulator mass matrix, $M(\Theta)$; it is composed of all those terms which multiply $\ddot{\Theta}$ and is a function of Θ. Therefore, we have

$$M(\Theta) = \begin{bmatrix} l_2^2 m_2 + 2l_1 l_2 m_2 c_2 + l_1^2(m_1 + m_2) & l_2^2 m_2 + l_1 l_2 m_2 c_2 \\ l_2^2 m_2 + l_1 l_2 m_2 c_2 & l_2^2 m_2 \end{bmatrix}. \quad (6.60)$$

Any manipulator mass matrix is symmetric and positive definite, and is, therefore, always invertible.

The velocity term, $V(\Theta, \dot{\Theta})$, contains all those terms that have any dependence on joint velocity. Thus, we obtain

$$V(\Theta, \dot{\Theta}) = \begin{bmatrix} -m_2 l_1 l_2 s_2 \dot{\theta}_2^2 - 2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 \\ m_2 l_1 l_2 s_2 \dot{\theta}_1^2 \end{bmatrix}. \quad (6.61)$$

A term like $-m_2 l_1 l_2 s_2 \dot{\theta}_2^2$ is caused by a **centrifugal force**, and is recognized as such because it depends on the square of a joint velocity. A term such as $-2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2$ is caused by a **Coriolis force** and will always contain the product of two different joint velocities.

The gravity term, $G(\Theta)$, contains all those terms in which the gravitational constant, g, appears. Therefore, we have

$$G(\Theta) = \begin{bmatrix} m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1 \\ m_2 l_2 g c_{12} \end{bmatrix}. \quad (6.62)$$

Note that the gravity term depends only on Θ and not on its derivatives.

The configuration-space equation

By writing the velocity-dependent term, $V(\Theta, \dot{\Theta})$, in a different form, we can write the dynamic equations as

$$\tau = M(\Theta)\ddot{\Theta} + B(\Theta)[\dot{\Theta}\dot{\Theta}] + C(\Theta)[\dot{\Theta}^2] + G(\Theta), \quad (6.63)$$

where $B(\Theta)$ is a matrix of dimensions $n \times n(n-1)/2$ of Coriolis coefficients, $[\dot{\Theta}\dot{\Theta}]$ is an $n(n-1)/2 \times 1$ vector of joint velocity products given by

$$[\dot{\Theta}\dot{\Theta}] = [\dot{\theta}_1\dot{\theta}_2 \ \dot{\theta}_1\dot{\theta}_3 \ \cdots \ \dot{\theta}_{n-1}\dot{\theta}_n]^T, \quad (6.64)$$

$C(\Theta)$ is an $n \times n$ matrix of centrifugal coefficients, and $[\dot{\Theta}^2]$ is an $n \times 1$ vector given by

$$[\dot{\theta}_1^2 \ \dot{\theta}_2^2 \ \cdots \ \dot{\theta}_n^2]^T. \quad (6.65)$$

We will call (6.63) the **configuration-space equation**, because the matrices are functions only of manipulator position [3].

In this form of the dynamic equations, the complexity of the computation is seen to be in the form of computing various parameters which are a function of only the manipulator position, Θ. This is important in applications (such as computer control of a manipulator) in which the dynamic equations must be updated as the manipulator moves. (Equation (6.63) gives a form in which parameters are a function of joint position only and can be updated at a rate related to how fast the manipulator is changing configuration.) We will consider this form again with regard to the problem of manipulator control in Chapter 10.

EXAMPLE 6.4

Give $B(\Theta)$ and $C(\Theta)$ (from (6.63)) for the manipulator of Section 6.7.

For this simple two-link manipulator, we have

$$[\dot{\Theta}\Theta] = [\dot{\theta}_1\dot{\theta}_2],$$
$$[\dot{\Theta}^2] = \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix}. \quad (6.66)$$

So we see that

$$B(\Theta) = \begin{bmatrix} -2m_2 l_1 l_2 s_2 \\ 0 \end{bmatrix} \tag{6.67}$$

and

$$C(\Theta) = \begin{bmatrix} 0 & -m_2 l_1 l_2 s_2 \\ m_2 l_1 l_2 s_2 & 0 \end{bmatrix}. \tag{6.68}$$

6.9 LAGRANGIAN FORMULATION OF MANIPULATOR DYNAMICS

The Newton–Euler approach is based on the elementary dynamic formulas (6.29) and (6.30) and on an analysis of forces and moments of constraint acting between the links. As an alternative to the Newton–Euler method, in this section we briefly introduce the **Lagrangian dynamic formulation**. Whereas the Newton–Euler formulation might be said to be a "force balance" approach to dynamics, the Lagrangian formulation is an "energy-based" approach to dynamics. Of course, for the same manipulator, both will give the same equations of motion. Our statement of Lagrangian dynamics will be brief and somewhat specialized to the case of a serial-chain mechanical manipulator with rigid links. For a more complete and general reference, see [4].

We start by developing an expression for the kinetic energy of a manipulator. The kinetic energy of the ith link, k_i, can be expressed as

$$k_i = \tfrac{1}{2} m_i v_{C_i}^T v_{C_i} + \tfrac{1}{2} {}^i\omega_i^T {}^{C_i}I_i {}^i\omega_i, \tag{6.69}$$

where the first term is kinetic energy due to linear velocity of the link's center of mass and the second term is kinetic energy due to angular velocity of the link. The total kinetic energy of the manipulator is the sum of the kinetic energy in the individual links—that is,

$$k = \sum_{i=1}^{n} k_i. \tag{6.70}$$

The v_{C_i} and ${}^i\omega_i$ in (6.69) are functions of Θ and $\dot{\Theta}$, so we see that the kinetic energy of a manipulator can be described by a scalar formula as a function of joint position and velocity, $k(\Theta, \dot{\Theta})$. In fact, the kinetic energy of a manipulator is given by

$$k(\Theta, \dot{\Theta}) = \tfrac{1}{2}\dot{\Theta}^T M(\Theta)\dot{\Theta}, \tag{6.71}$$

where $M(\Theta)$ is the $n \times n$ manipulator mass matrix already introduced in Section 6.8. An expression of the form of (6.71) is known as a **quadratic form** [5], since when expanded out, the resulting scalar equation is composed solely of terms whose dependence on the $\dot{\theta}_i$ is quadratic. Further, because the total kinetic energy must always be positive, the manipulator mass matrix must be a so-called **positive definite** matrix. Positive definite matrices are those having the property that their quadratic form is always a positive scalar. Equation (6.71) can be seen to be analogous to the familiar expression for the kinetic energy of a point mass:

$$k = \tfrac{1}{2} m v^2. \tag{6.72}$$

The fact that a manipulator mass matrix must be positive definite is analogous to the fact that a scalar mass is always a positive number.

The potential energy of the ith link, u_i, can be expressed as

$$u_i = -m_i {}^0g^T {}^0P_{C_i} + u_{ref_i}, \tag{6.73}$$

where ${}^0g$ is the 3×1 gravity vector, ${}^0P_{C_i}$ is the vector locating the center of mass of the ith link, and u_{ref_i} is a constant chosen so that the minimum value of u_i is zero.[1] The total potential energy stored in the manipulator is the sum of the potential energy in the individual links—that is,

$$u = \sum_{i=1}^{n} u_i. \tag{6.74}$$

Because the ${}^0P_{C_i}$ in (6.73) are functions of Θ, we see that the potential energy of a manipulator can be described by a scalar formula as a function of joint position, $u(\Theta)$.

The Lagrangian dynamic formulation provides a means of deriving the equations of motion from a scalar function called the **Lagrangian**, which is defined as the difference between the kinetic and potential energy of a mechanical system. In our notation, the Lagrangian of a manipulator is

$$\mathcal{L}(\Theta, \dot{\Theta}) = k(\Theta, \dot{\Theta}) - u(\Theta). \tag{6.75}$$

The equations of motion for the manipulator are then given by

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\Theta}} - \frac{\partial \mathcal{L}}{\partial \Theta} = \tau, \tag{6.76}$$

where τ is the $n \times 1$ vector of actuator torques. In the case of a manipulator, this equation becomes

$$\frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}} - \frac{\partial k}{\partial \Theta} + \frac{\partial u}{\partial \Theta} = \tau, \tag{6.77}$$

where the arguments of $k(\cdot)$ and $u(\cdot)$ have been dropped for brevity.

EXAMPLE 6.5

The links of an RP manipulator, shown in Fig. 6.7, have inertia tensors

$$ {}^{C_1}I_1 = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix},$$

$$ {}^{C_2}I_2 = \begin{bmatrix} I_{xx2} & 0 & 0 \\ 0 & I_{yy2} & 0 \\ 0 & 0 & I_{zz2} \end{bmatrix}, \tag{6.78}$$

[1] Actually, only the partial derivative of the potential energy with respect to Θ will appear in the dynamics, so this constant is arbitrary. This corresponds to defining the potential energy relative to an arbitrary zero reference height.

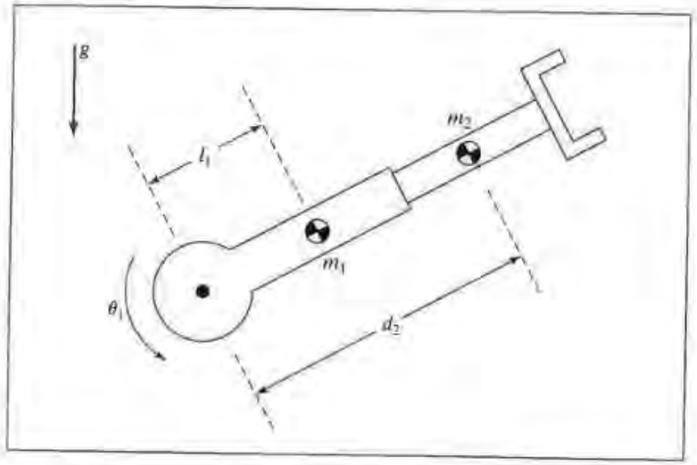


FIGURE 6.7: The RP manipulator of Example 6.5.

and total mass m_1 and m_2. As shown in Fig. 6.7, the center of mass of link 1 is located at a distance l_1 from the joint-1 axis, and the center of mass of link 2 is at the variable distance d_2 from the joint-1 axis. Use Lagrangian dynamics to determine the equation of motion for this manipulator.

Using (6.69), we write the kinetic energy of link 1 as

$$k_1 = \tfrac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 + \tfrac{1}{2} I_{zz1} \dot{\theta}_1^2 \tag{6.79}$$

and the kinetic energy of link 2 as

$$k_2 = \tfrac{1}{2} m_2 (d_2^2 \dot{\theta}_1^2 + \dot{d}_2^2) + \tfrac{1}{2} I_{zz2} \dot{\theta}_1^2. \tag{6.80}$$

Hence, the total kinetic energy is given by

$$k(\Theta, \dot{\Theta}) = \tfrac{1}{2}(m_1 l_1^2 + I_{zz1} + I_{zz2} + m_2 d_2^2)\dot{\theta}_1^2 + \tfrac{1}{2} m_2 \dot{d}_2^2. \tag{6.81}$$

Using (6.73), we write the potential energy of link 1 as

$$u_1 = m_1 l_1 g \sin(\theta_1) + m_1 l_1 g \tag{6.82}$$

and the potential energy of link 2 as

$$u_2 = m_2 g d_2 \sin(\theta_1) + m_2 g d_{2max}, \tag{6.83}$$

where d_{2max} is the maximum extension of joint 2. Hence, the total potential energy is given by

$$u(\Theta) = g(m_1 l_1 + m_2 d_2)\sin(\theta_1) + m_1 l_1 g + m_2 g d_{2max}. \tag{6.84}$$

Next, we take partial derivatives as needed for (6.77):

$$\frac{\partial k}{\partial \dot{\Theta}} = \begin{bmatrix} (m_1 l_1^2 + I_{zz1} + I_{zz2} + m_2 d_2^2)\dot{\theta}_1 \\ m_2 \dot{d}_2 \end{bmatrix}, \tag{6.85}$$

$$\frac{\partial k}{\partial \Theta} = \begin{bmatrix} 0 \\ m_2 d_2 \dot{\theta}_1^2 \end{bmatrix}, \tag{6.86}$$

$$\frac{\partial u}{\partial \Theta} = \begin{bmatrix} g(m_1 l_1 + m_2 d_2)\cos(\theta_1) \\ g m_2 \sin(\theta_1) \end{bmatrix}. \tag{6.87}$$

Finally, substituting into (6.77), we have

$$\begin{aligned} \tau_1 &= (m_1 l_1^2 + I_{zz1} + I_{zz2} + m_2 d_2^2)\ddot{\theta}_1 + 2 m_2 d_2 \dot{\theta}_1 \dot{d}_2 \\ &\quad + (m_1 l_1 + m_2 d_2) g \cos(\theta_1), \\ \tau_2 &= m_2 \ddot{d}_2 - m_2 d_2 \dot{\theta}_1^2 + m_2 g \sin(\theta_1). \end{aligned} \tag{6.88}$$

From (6.89), we can see that

$$\begin{aligned} M(\Theta) &= \begin{bmatrix} (m_1 l_1^2 + I_{zz1} + I_{zz2} + m_2 d_2^2) & 0 \\ 0 & m_2 \end{bmatrix}, \\ V(\Theta, \dot{\Theta}) &= \begin{bmatrix} 2 m_2 d_2 \dot{\theta}_1 \dot{d}_2 \\ -m_2 d_2 \dot{\theta}_1^2 \end{bmatrix}, \\ G(\Theta) &= \begin{bmatrix} (m_1 l_1 + m_2 d_2) g \cos(\theta_1) \\ m_2 g \sin(\theta_1) \end{bmatrix}. \end{aligned} \tag{6.89}$$

6.10 FORMULATING MANIPULATOR DYNAMICS IN CARTESIAN SPACE

Our dynamic equations have been developed in terms of the position and time derivatives of the manipulator joint angles, or in **joint space,** with the general form

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta). \tag{6.90}$$

We developed this equation in joint space because we could use the serial-link nature of the mechanism to advantage in deriving the equations. In this section, we discuss the formulation of the dynamic equations that relate acceleration of the end-effector expressed in Cartesian space to Cartesian forces and moments acting at the end-effector.

The Cartesian state-space equation

As explained in Chapters 10 and 11, it might be desirable to express the dynamics of a manipulator with respect to Cartesian variables in the general form [6]

$$\mathcal{F} = M_x(\Theta)\ddot{\mathcal{X}} + V_x(\Theta, \dot{\Theta}) + G_x(\Theta), \tag{6.91}$$

where $\mathcal{F}$ is a force–torque vector acting on the end-effector of the robot, and χ is an appropriate Cartesian vector representing position and orientation of the end-effector [7]. Analogous to the joint-space quantities, $M_x(\Theta)$ is the **Cartesian mass matrix**, $V_x(\Theta, \dot{\Theta})$ is a vector of velocity terms in Cartesian space, and $G_x(\Theta)$ is a vector of gravity terms in Cartesian space. Note that the fictitious forces acting on the end-effector, $\mathcal{F}$, could in fact be applied by the actuators at the joints by using the relationship

$$\tau = J^T(\Theta)\mathcal{F}, \tag{6.92}$$

where the Jacobian, $J(\Theta)$, is written in the same frame as $\mathcal{F}$ and $\dot{\chi}$, usually the tool frame, $\{T\}$.

We can derive the relationship between the terms of (6.90) and those of (6.91) in the following way. First, we premultiply (6.90) by the inverse of the Jacobian transpose to obtain

$$J^{-T}\tau = J^{-T}M(\Theta)\ddot{\Theta} + J^{-T}V(\Theta, \dot{\Theta}) + J^{-T}G(\Theta). \tag{6.93}$$

or

$$\mathcal{F} = J^{-T}M(\Theta)\ddot{\Theta} + J^{-T}V(\Theta, \dot{\Theta}) + J^{-T}G(\Theta). \tag{6.94}$$

Next, we develop a relationship between joint space and Cartesian acceleration, starting with the definition of the Jacobian.

$$\dot{\chi} = J\dot{\Theta}. \tag{6.95}$$

and differentiating to obtain

$$\ddot{\chi} = \dot{J}\dot{\Theta} + J\ddot{\Theta}. \tag{6.96}$$

Solving (6.96) for joint-space acceleration leads to

$$\ddot{\Theta} = J^{-1}\ddot{\chi} - J^{-1}\dot{J}\dot{\Theta}, \tag{6.97}$$

Substituting (6.97) into (6.94), we have

$$\mathcal{F} = J^{-T}M(\Theta)J^{-1}\ddot{\chi} - J^{-T}M(\Theta)J^{-1}\dot{J}\dot{\Theta} + J^{-T}V(\Theta, \dot{\Theta}) + J^{-T}G(\Theta), \tag{6.98}$$

from which we derive the expressions for the terms in the Cartesian dynamics as

$$\begin{aligned} M_x(\Theta) &= J^{-T}(\Theta)M(\Theta)J^{-1}(\Theta), \\ V_x(\Theta, \dot{\Theta}) &= J^{-T}(\Theta)(V(\Theta, \dot{\Theta}) - M(\Theta)J^{-1}(\Theta)\dot{J}(\Theta)\dot{\Theta}), \\ G_x(\Theta) &= J^{-T}(\Theta)G(\Theta). \end{aligned} \tag{6.99}$$

Note that the Jacobian appearing in equations (6.100) is written in the same frames as $\mathcal{F}$ and χ in (6.91); the choice of this frame is arbitrary.[2] Note that, when the manipulator approaches a singularity, certain quantities in the Cartesian dynamics become infinite.

[2] Certain choices could facilitate computation.

EXAMPLE 6.6

Derive the Cartesian-space form of the dynamics for the two-link planar arm of Section 6.7. Write the dynamics in terms of a frame attached to the end of the second link.

For this manipulator, we have already obtained the dynamics (in Section 6.7) and the Jacobian (equation (5.66)), which we restate here:

$$J(\Theta) = \begin{bmatrix} l_1 s_2 & 0 \\ l_1 c_2 + l_2 & l_2 \end{bmatrix}. \tag{6.100}$$

First, compute the inverse Jacobian:

$$J^{-1}(\Theta) = \frac{1}{l_1 l_2 s_2}\begin{bmatrix} l_2 & 0 \\ -l_1 c_2 - l_2 & l_1 s_2 \end{bmatrix}. \tag{6.101}$$

Next, obtain the time derivative of the Jacobian:

$$\dot{J}(\Theta) = \begin{bmatrix} l_1 c_2 \dot{\theta}_2 & 0 \\ -l_1 s_2 \theta_2 & 0 \end{bmatrix}. \tag{6.102}$$

Using (6.100) and the results of Section 6.7, we get

$$\begin{aligned} M_x(\Theta) &= \begin{bmatrix} m_2 + \frac{m_1}{s_2^2} & 0 \\ 0 & m_2 \end{bmatrix}, \\ V_x(\Theta, \dot{\Theta}) &= \begin{bmatrix} -(m_2 l_1 c_2 + m_2 l_2)\dot{\theta}_1^2 - m_2 l_2 \dot{\theta}_2^2 - (2m_2 l_2 + m_2 l_1 c_2 + m_1 l_1 \frac{c_2}{s_2^2})\dot{\theta}_1\dot{\theta}_2 \\ m_2 l_1 s_2 \dot{\theta}_1^2 + l_1 m_2 s_2 \dot{\theta}_1 \dot{\theta}_2 \end{bmatrix}, \\ G_x(\Theta) &= \begin{bmatrix} m_1 g \frac{c_1}{s_2} + m_2 g s_{12} \\ m_2 g c_{12} \end{bmatrix}. \end{aligned} \tag{6.103}$$

When $s_2 = 0$, the manipulator is in a singular position, and some of the dynamic terms go to infinity. For example, when $\theta_2 = 0$ (arm stretched straight out), the effective Cartesian mass of the end-effector becomes infinite in the $\hat{X}_2$ direction of the link-2 tip frame, as expected. In general, at a singular configuration there is a certain direction, the *singular direction* in which motion is impossible, but general motion in the subspace "orthogonal" to this direction is possible [8].

The Cartesian configuration space torque equation

Combining (6.91) and (6.92), we can write equivalent joint torques with the dynamics expressed in Cartesian space:

$$\tau = J^T(\Theta)(M_x(\Theta)\ddot{\chi} + V_x(\Theta, \dot{\Theta}) + G_x(\Theta)). \tag{6.104}$$

We will find it useful to write this equation in the form

$$\tau = J^T(\Theta)M_x(\Theta)\ddot{\chi} + B_x(\Theta)[\dot{\Theta}\dot{\Theta}] + C_x(\Theta)[\dot{\Theta}^2] + G(\Theta), \tag{6.105}$$

where $B_x(\Theta)$ is a matrix of dimension $n \times n(n-1)/2$ of Coriolis coefficients, $[\dot{\Theta}\dot{\Theta}]$ is an $n(n-1)/2 \times 1$ vector of joint velocity products given by

$$[\dot{\Theta}\dot{\Theta}] = [\dot{\theta}_1\dot{\theta}_2 \;\; \dot{\theta}_1\dot{\theta}_3 \;\; \cdots \dot{\theta}_{n-1}\dot{\theta}_n]^T. \tag{6.106}$$

$C_x(\Theta)$ is an $n \times n$ matrix of centrifugal coefficients, and $[\dot{\Theta}^2]$ is an $n \times 1$ vector given by

$$[\dot{\theta}_1^2 \;\; \dot{\theta}_2^2 \;\; \ldots \dot{\theta}_n^2]^T. \tag{6.107}$$

Note that, in (6.105), $G(\Theta)$ is the same as in the joint-space equation, but in general, $B_x(\Theta) \neq B(\Theta)$ and $C_x(\Theta) \neq C(\Theta)$.

EXAMPLE 6.7

Find $B_x(\Theta)$ and $C_x(\Theta)$ (from (6.105)) for the manipulator of Section 6.7.

If we form the product $J^T(\Theta)V_x(\Theta, \dot{\Theta})$, we find that

$$B_x(\Theta) = \begin{bmatrix} m_1 l_1^2 \frac{c_2}{s_2} - m_2 l_1 l_2 s_2 \\ m_2 l_1 l_2 s_2 \end{bmatrix} \tag{6.108}$$

and

$$C_x(\Theta) = \begin{bmatrix} 0 & -m_2 l_1 l_2 s_2 \\ m_2 l_1 l_2 s_2 & 0 \end{bmatrix}. \tag{6.109}$$

6.11 INCLUSION OF NONRIGID BODY EFFECTS

It is important to realize that the dynamic equations we have derived *do not encompass all the effects acting on a manipulator*. They include only those forces which arise from rigid body mechanics. The most important source of forces that are *not* included is friction. All mechanisms are, of course, affected by frictional forces. In present-day manipulators, in which significant gearing is typical, the forces due to friction can actually be quite large—perhaps equaling 25% of the torque required to move the manipulator in typical situations.

In order to make dynamic equations reflect the reality of the physical device, it is important to model (at least approximately) these forces of friction. A very simple model for friction is **viscous friction**, in which the torque due to friction is proportional to the velocity of joint motion. Thus, we have

$$\tau_{friction} = v\dot{\theta}, \tag{6.110}$$

where v is a viscous-friction constant. Another possible simple model for friction, **Coulomb friction**, is sometimes used. Coulomb friction is constant except for a sign dependence on the joint velocity and is given by

$$\tau_{friction} = c\, sgn(\dot{\theta}), \tag{6.111}$$

where c is a Coulomb-friction constant. The value of c is often taken at one value when $\dot{\theta} = 0$, the static coefficient, but at a lower value, the dynamic coefficient, when $\dot{\theta} \neq 0$. Whether a joint of a particular manipulator exhibits viscous or Coulomb friction is a complicated issue of lubrication and other effects. A reasonable model is to include both, because both effects are likely:

$$\tau_{friction} = c\, sgn(\dot{\theta}) + v\dot{\theta}. \tag{6.112}$$

It turns out that, in many manipulator joints, friction also displays a dependence on the joint position. A major cause of this effect might be gears that are not perfectly round—their eccentricity would cause friction to change according to joint position. So a fairly complex friction model would have the form

$$\tau_{friction} = f(\theta, \dot{\theta}). \tag{6.113}$$

These friction models are then added to the other dynamic terms derived from the rigid-body model, yielding the more complete model

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}). \tag{6.114}$$

There are also other effects, which are neglected in this model. For example, the assumption of rigid body links means that we have failed to include bending effects (which give rise to resonances) in our equations of motion. However, these effects are extremely difficult to model and are beyond the scope of this book. (See [9, 10].)

6.12 DYNAMIC SIMULATION

To simulate the motion of a manipulator, we must make use of a model of the dynamics such as the one we have just developed. Given the dynamics written in closed form as in (6.59), simulation requires solving the dynamic equation for acceleration:

$$\ddot{\Theta} = M^{-1}(\Theta)[\tau - V(\Theta, \dot{\Theta}) - G(\Theta) - F(\Theta, \dot{\Theta})]. \tag{6.115}$$

We can then apply any of several known **numerical integration** techniques to integrate the acceleration to compute future positions and velocities.

Given initial conditions on the motion of the manipulator, usually in the form

$$\begin{aligned} \Theta(0) &= \Theta_0, \\ \dot{\Theta}(0) &= 0, \end{aligned} \tag{6.116}$$

we integrate (6.115) forward in time numerically by steps of size Δt. There are many methods of performing numerical integration [11]. Here, we introduce the simplest integration scheme, called *Euler integration*: Starting with $t = 0$, iteratively compute

$$\begin{aligned} \dot{\Theta}(t + \Delta t) &= \dot{\Theta}(t) + \ddot{\Theta}(t)\Delta t, \\ \Theta(t + \Delta t) &= \Theta(t) + \dot{\Theta}(t)\Delta t + \tfrac{1}{2}\ddot{\Theta}(t)\Delta t^2, \end{aligned} \tag{6.117}$$

where, for each iteration, (6.115) is computed to calculate $\ddot{\Theta}$. In this way, the position, velocity, and acceleration of the manipulator caused by a certain input torque function can be computed numerically.

Euler integration is conceptually simple, but other, more sophisticated integration techniques are recommended for accurate and efficient simulation [11]. How

to select the size of Δt is an issue that is often discussed. It should be sufficiently small that breaking continuous time into these small increments is a reasonable approximation. It should be sufficiently large that an excessive amount of computer time is not required to compute a simulation.

6.13 COMPUTATIONAL CONSIDERATIONS

Because the dynamic equations of motion for typical manipulators are so complex, it is important to consider computational issues. In this section, we restrict our attention to joint-space dynamics. Some issues of computational efficiency of Cartesian dynamics are discussed in [7, 8].

A historical note concerning efficiency

Counting the number of multiplications and additions for the equations (6.46)–(6.53) when taking into consideration the simple first outward computation and simple last inward computation, we get

$$126n - 99 \text{ multiplications,}$$

$$106n - 92 \text{ additions,}$$

where n is the number of links (here, at least two). Although still somewhat complex, the formulation is tremendously efficient in comparison with some previously suggested formulations of manipulator dynamics. The first formulation of the dynamics for a manipulator [12, 13] was done via a fairly straightforward Lagrangian approach whose required computations came out to be approximately [14]

$$32n^4 + 86n^3 + 171n^2 + 53n - 128 \text{ multiplications,}$$

$$25n^4 + 66n^3 + 129n^2 + 42n - 96 \text{ additions.}$$

For a typical case, $n = 6$, the iterative Newton–Euler scheme is about 100 times more efficient! The two approaches must of course yield equivalent equations, and numeric calculations would yield exactly the same results, but the structure of the equations is quite different. This is not to say that a Lagrangian approach cannot be made to produce efficient equations. Rather, this comparison indicates that, in formulating a computational scheme for this problem, care must be taken as regards efficiency. The relative efficiency of the method we have presented stems from posing the computations as iterations from link to link and in the particulars of how the various quantities are represented [15].

Renaud [16] and Liegois et al. [17] made early contributions concerning the formulation of the mass-distribution descriptions of the links. While studying the modeling of human limbs, Stepanenko and Vukobratovic [18] began investigating a "Newton–Euler" approach to dynamics instead of the somewhat more traditional Lagrangian approach. This work was revised for efficiency by Orin et al. [19] in an application to the legs of walking robots. Orin's group improved the efficiency somewhat by writing the forces and moments in the local link-reference frames instead of in the inertial frame. They also noticed the sequential nature of calculations from one link to the next and speculated that an efficient recursive formulation might exist. Armstrong [20] and Luh, Walker, and Paul

[2] paid close attention to details of efficiency and published an algorithm that is $O(n)$ in complexity. This was accomplished by setting up the calculations in an iterative (or recursive) nature and by expressing the velocities and accelerations of the links in the local link frames. Hollerbach [14] and Silver [15] further explored various computational algorithms. Hollerbach and Sahar [21] showed that, for certain specialized geometries, the complexity of the algorithm would reduce further.

Efficiency of closed form vs. that of iterative form

The iterative scheme introduced in this chapter is quite efficient as a general means of computing the dynamics of any manipulator, but closed-form equations derived for a particular manipulator will usually be even more efficient. Consider the two-link planar manipulator of Section 6.7. Plugging $n = 2$ into the formulas given in Section 6.13, we find that our iterative scheme would require 153 multiplications and 120 additions to compute the dynamics of a general two-link. However, our particular two-link arm happens to be quite simple: It is planar, and the masses are treated as point masses. So, if we consider the closed-form equations that we worked out in Section 6.7, we see that computation of the dynamics in this form requires about 30 multiplications and 13 additions. This is an extreme case, because the particular manipulator is very simple, but it illustrates the point that symbolic closed-form equations are likely to be the most efficient formulation of dynamics. Several authors have published articles showing that, for any given manipulator, customized closed-form dynamics are more efficient than even the best of the general schemes [22–27].

Hence, if manipulators are designed to be *simple* in the kinematic and dynamic sense, they will have dynamic equations that are simple. We might define a **kinematically simple** manipulator to be a manipulator that has many (or all) of its link twists equal to 0°, 90°, or −90° and many of its link lengths and offsets equal to zero. We might define a **dynamically simple** manipulator as one for which each link-inertia tensor is diagonal in frame $\{C_i\}$.

The drawback of formulating closed-form equations is simply that it currently requires a fair amount of human effort. However, symbolic manipulation programs that can derive the closed-form equations of motion of a device and automatically factor out common terms and perform trigonometric substitutions have been developed [25, 28–30].

Efficient dynamics for simulation

When dynamics are to be computed for the purpose of performing a numerical simulation of a manipulator, we are interested in solving for the joint accelerations, given the manipulator's current position and velocity and the input torques. An efficient computational scheme must therefore address both the computation of the dynamic equations studied in this chapter and efficient schemes for solving equations (for joint accelerations) and performing numerical integration. Several efficient methods for dynamic simulation of manipulators are reported in [31].

Memorization schemes

In any computational scheme, a trade-off can be made between computations and memory usage. In the problem of computing the dynamic equation of a manipulator (6.59), we have implicitly assumed that, when a value of τ is needed, it is computed as quickly as possible from Θ, $\dot{\Theta}$, and $\ddot{\Theta}$ at run time. If we wish, we can trade off this computational burden at the cost of a tremendously large memory by precomputing (6.59) for all possible Θ, $\dot{\Theta}$, and $\ddot{\Theta}$ values (suitably quantized). Then, when dynamic information is needed, the answer is found by table lookup.

The size of the memory required is large. Assume that each joint angle range is quantized to ten discrete values; likewise, assume that velocities and accelerations are quantized to ten ranges each. For a six-jointed manipulator, the number of cells in the $(\Theta, \dot{\Theta}, \ddot{\Theta})$ quantized space is $(10 \times 10 \times 10)^6$. In each of these cells, there are six torque values. Assuming each torque value requires one computer word, this memory size is 6×10^{18} words! Also, note that the table needs to be recomputed for a change in the mass of the load—or else another dimension needs to be added to account for all possible loads.

There are many intermediate solutions that trade off memory for computation in various ways. For example, if the matrices appearing in equation (6.63) were precomputed, the table would have only one dimension (in Θ) rather than three. After the functions of Θ are looked up, a modest amount of computation (given by (6.63)) is done. For more details and for other possible parameterizations of this problem, see [3] and [6].

BIBLIOGRAPHY

[1] I. Shames, *Engineering Mechanics*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[2] J.Y.S. Luh, M.W. Walker, and R.P. Paul, "On-Line Computational Scheme for Mechanical Manipulators," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 1980.

[3] M. Raibert, "Mechanical Arm Control Using a State Space Memory," SME paper MS77-750, 1977.

[4] K.R. Symon, *Mechanics*, 3rd edition, Addison-Wesley, Reading, MA, 1971.

[5] B. Noble, *Applied Linear Algebra*, Prentice-Hall, Englewood Cliffs, NJ, 1969.

[6] O. Khatib, "Commande Dynamique dans L'Espace Operationnel des Robots Manipulateurs en Presence d'Obstacles," These de Docteur-Ingenieur, Ecole Nationale Superieure de l'Aeronautique et de L'Espace (ENSAE), Toulouse.

[7] O. Khatib, "Dynamic Control of Manipulators in Operational Space," Sixth IFTOMM Congress on Theory of Machines and Mechanisms, New Delhi, December 15–20, 1983.

[8] O. Khatib, "The Operational Space Formulation in Robot Manipulator Control," 15th ISIR, Tokyo, September 11–13, 1985.

[9] E Schmitz, "Experiments on the End-Point Position Control of a Very Flexible One-Link Manipulator," Unpublished Ph.D. Thesis, Department of Aeronautics and Astronautics, Stanford University, SUDAAR No. 547, June 1985.

[10] W. Book, "Recursive Lagrangian Dynamics of Flexible Manipulator Arms," *International Journal of Robotics Research*, Vol. 3, No. 3, 1984.

[11] S. Conte and C. DeBoor, *Elementary Numerical Analysis: An Algorithmic Approach*, 2nd edition, McGraw-Hill, New York, 1972.

[12] J. Uicker, "On the Dynamic Analysis of Spatial Linkages Using 4 × 4 Matrices," Unpublished Ph.D dissertation, Northwestern University, Evanston, IL, 1965.

[13] J. Uicker, "Dynamic Behaviour of Spatial Linkages," *ASME Mechanisms*, Vol. 5, No. 68, pp. 1–15.

[14] J.M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," in *Robot Motion*, M. Brady et al., Editors, MIT Press, Cambridge, MA, 1983.

[15] W. Silver, "On the Equivalence of Lagrangian and Newton–Euler Dynamics for Manipulators," *International Journal of Robotics Research*, Vol. 1, No. 2, pp. 60–70.

[16] M. Renaud, "Contribution à l'Etude de la Modélisation et de la Commande des Systèmes Mécaniques Articulés," Thèse de Docteur-Ingénieur, Université Paul Sabatier, Toulouse, December 1975.

[17] A. Liegois, W. Khalil, J.M. Dumas, and M. Renaud, "Mathematical Models of Interconnected Mechanical Systems," Symposium on the Theory and Practice of Robots and Manipulators, Poland, 1976.

[18] Y. Stepanenko and M. Vukobratovic, "Dynamics of Articulated Open-Chain Active Mechanisms," *Math-Biosciences* Vol. 28, 1976, pp. 137–170.

[19] D.E. Orin et al, "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton–Euler Methods," *Math-Biosciences* Vol. 43, 1979, pp. 107–130.

[20] W.W. Armstrong, "Recursive Solution to the Equations of Motion of an N-Link Manipulator," *Proceedings of the 5th World Congress on the Theory of Machines and Mechanisms*, Montreal, July 1979.

[21] J.M. Hollerbach and G. Sahar, "Wrist-Partitioned Inverse Accelerations and Manipulator Dynamics," MIT AI Memo No. 717, April 1983.

[22] T.K. Kanade, P.K. Khosla, and N. Tanaka, "Real-Time Control of the CMU Direct Drive Arm II Using Customized Inverse Dynamics," *Proceedings of the 23rd IEEE Conference on Decision and Control*, Las Vegas, NV, December 1984.

[23] A. Izaguirre and R.P. Paul, "Computation of the Inertial and Gravitational Coefficients of the Dynamic Equations for a Robot Manipulator with a Load," *Proceedings of the 1985 International Conference on Robotics and Automation*, pp. 1024–1032, St. Louis, March 1985.

[24] B. Armstrong, O. Khatib, and J. Burdick, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, April 1986, pp. 510–518.

[25] J.W. Burdick, "An Algorithm for Generation of Efficient Manipulator Dynamic Equations," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, April 7–11, 1986, pp. 212–218.

[26] T.R. Kane and D.A. Levinson, "The Use of Kane's Dynamical Equations in Robotics," *The International Journal of Robotics Research*, Vol. 2, No. 3, Fall 1983, pp. 3–20.

[27] M. Renaud, "An Efficient Iterative Analytical Procedure for Obtaining a Robot Manipulator Dynamic Model," First International Symposium of Robotics Research, NH, August 1983.

[28] W. Schiehlen, "Computer Generation of Equations of Motion," in *Computer Aided Analysis and Optimization of Mechanical System Dynamics*, E.J. Haug, Editor, Springer-Verlag, Berlin & New York, 1984.

[29] G. Cesareo, F. Nicolo, and S. Nicosia, "DYMIR: A Code for Generating Dynamic Model of Robots," in *Advanced Software in Robotics*, Elsevier Science Publishers. North-Holland, 1984.

[30] J. Murray, and C. Neuman, "ARM: An Algebraic Robot Dynamic Modelling Program," IEEE International Conference on Robotics, Atlanta, March 1984.

[31] M. Walker and D. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 104, 1982.

EXERCISES

6.1 [12] Find the inertia tensor of a right cylinder of homogeneous density with respect to a frame with origin at the center of mass of the body.

6.2 [32] Construct the dynamic equations for the two-link manipulator in Section 6.7 when each link is modeled as a rectangular solid of homogeneous density. Each link has dimensions l_i, ω_i, and h_i and total mass m_i.

6.3 [43] Construct the dynamic equations for the three-link manipulator of Chapter 3, Exercise 3.3. Consider each link to be a rectangular solid of homogeneous density with dimensions l_i, ω_i, and h_i and total mass m_i.

6.4 [13] Write the set of equations that correspond to (6.46)–(6.53) for the case where the mechanism could have sliding joints.

6.5 [30] Construct the dynamic equations for the two-link nonplanar manipulator shown in Fig. 6.8. Assume that all the mass of the links can be considered as a point mass located at the distal (outermost) end of the link. The mass values are m_1 and m_2, and the link lengths are l_1 and l_2. This manipulator is like the first two links of the arm in Exercise 3.3. Assume further that viscous friction is acting at each joint, with coefficients v_1 and v_2.

6.6 [32] Derive the Cartesian space form of the dynamics for the two-link planar manipulator of Section 6.7 in terms of the base frame. *Hint*: See Example 6.5, but use the Jacobian written in the base frame.

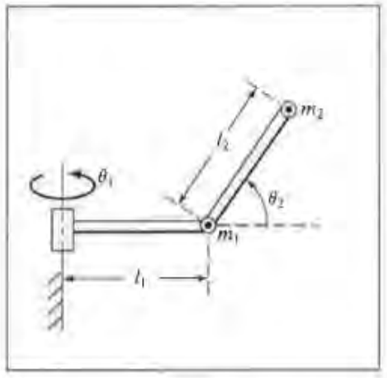


FIGURE 6.8: Two-link nonplanar manipulator with point masses at distal ends of links.

6.7 [18] How many memory locations would be required to store the dynamic equations of a general three-link manipulator in a table? Quantize each joint's position, velocity, and acceleration into 16 ranges. Make any assumptions needed.

6.8 [32] Derive the dynamic equations for the two-link manipulator shown in Fig. 4.6. Link 1 has an inertia tensor given by

$$C_1 I = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix}.$$

Assume that link 2 has all its mass, m_2, located at a point at the end-effector. Assume that gravity is directed downward (opposite $\hat{Z}_1$).

6.9 [37] Derive the dynamic equations for the three-link manipulator with one prismatic joint shown in Fig. 3.9. Link 1 has an inertia tensor given by

$$C_1 I = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix}.$$

Link 2 has point mass m_2 located at the origin of its link frame. Link 3 has an inertia tensor given by

$$C_3 I = \begin{bmatrix} I_{xx3} & 0 & 0 \\ 0 & I_{yy3} & 0 \\ 0 & 0 & I_{zz3} \end{bmatrix}.$$

Assume that gravity is directed opposite $\hat{Z}_1$ and that viscous friction of magnitude v_i is active at each joint.

6.10 [35] Derive the dynamic equations in Cartesian space for the manipulator of Exercise 6.8. Write the equations in frame {2}.

6.11 [20] A certain one-link manipulator has

$$C_1 I = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix}.$$

Assume that this is just the inertia of the link itself. If the motor armature has a moment of inertia I_m and the gear ratio is 100, what is the total inertia as seen from the motor shaft {1}?

6.12 [20] The single-degree-of-freedom "manipulator" in Fig. 6.9 has total mass $m = 1$, with the center of mass at

$${}^1P_C = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix},$$

and has inertia tensor

$${}^C I_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

From rest at $t = 0$, the joint angle θ_1 moves in accordance with the time function

$$\theta_1(t) = bt + ct^2$$

in radians. Give the angular acceleration of the link and the linear acceleration of the center of mass in terms of frame {1} as a function of t.

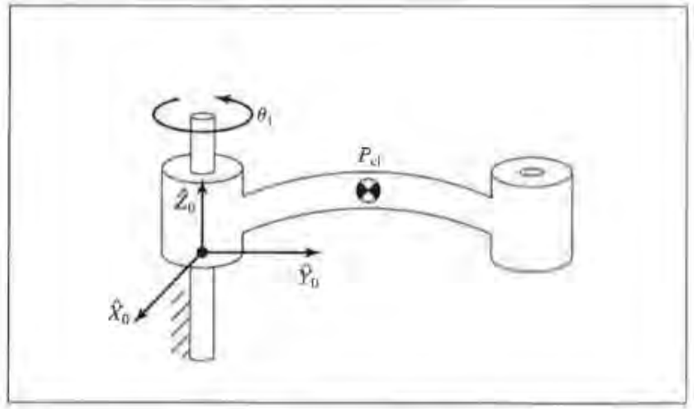


FIGURE 6.9: One-link "manipulator" of Exercise 6.12.

6.13 [40] Construct the Cartesian dynamic equations for the two-link nonplanar manipulator shown in Fig. 6.8. Assume that all the mass of the links can be considered as a point mass located at the distal (outermost) end of the link. The mass values are m_1 and m_2, and the link lengths are l_1 and l_2. This manipulator is like the first two links of the arm in Exercise 3.3. Also assume that viscous friction is acting at each joint with coefficients v_1 and v_2. Write the Cartesian dynamics in frame {3}, which is located at the tip of the manipulator and has the same orientation as link frame {2}.

6.14 [18] The following equations were derived for a 2-DOF RP manipulator:

$$\tau_1 = m_1(d_1^2 + d_2)\ddot{\theta}_1 + m_2 d_2^2\ddot{\theta}_1 + 2m_2 d_2\dot{d}_2\dot{\theta}_1$$
$$+g\cos(\theta_1)[m_1(d_1 + d_2\dot{\theta}_1) + m_2(d_2 + \dot{d}_2)]$$
$$\tau_2 = m_1\dot{d}_2\ddot{\theta}_1 + m_2\ddot{d}_2 - m_1 d_1\dot{d}_2 - m_2 d_2\dot{\theta}^2 + m_2(d_2 + 1)g\sin(\theta_1).$$

Some of the terms are obviously incorrect. Indicate the incorrect terms.

6.15 [28] Derive the dynamic equations for the RP manipulator of Example 6.5, using the Newton–Euler procedure instead of the Lagrangian technique.

6.16 [25] Derive the equations of motion for the PR manipulator shown in Fig. 6.10. Neglect friction, but include gravity. (Here, $\hat{X}_0$ is upward.) The inertia tensors of the links are diagonal, with moments $I_{xx1}, I_{yy1}, I_{zz1}$ and $I_{xx2}, I_{yy2}, I_{zz2}$. The centers of mass for the links are given by

$${}^1P_{C_1} = \begin{bmatrix} 0 \\ 0 \\ -l_1 \end{bmatrix},$$

$${}^2P_{C_2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

6.17 [40] The velocity-related terms appearing in the manipulator dynamic equation can be written as a matrix-vector product—that is,

$$V(\Theta, \dot{\Theta}) = V_m(\Theta, \dot{\Theta})\dot{\Theta},$$

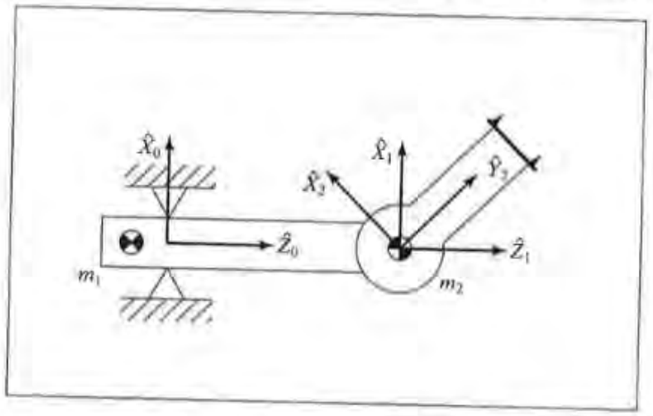


FIGURE 6.10: PR manipulator of Exercise 6.16.

where the m subscript stands for "matrix form." Show that an interesting relationship exists between the time derivative of the manipulator mass matrix and $V_m(\cdot)$, namely,

$$\dot{M}(\Theta) = 2V_m(\Theta, \dot{\Theta}) - S,$$

where S is some skew-symmetric matrix.

6.18 [15] Give two properties that any reasonable friction model (i.e., the term $F(\Theta, \dot{\Theta})$ in (6.114)) would possess.

6.19 [28] Do Exercise 6.5, using Lagrange's equations.

6.20 [28] Derive the dynamic equations of the 2-DOF manipulator of Section 6.7, using a Lagrangian formulation.

PROGRAMMING EXERCISE (PART 6)

1. Derive the dynamic equations of motion for the three-link manipulator (from Example 3.3). That is, expand Section 6.7 for the three-link case. The following numerical values describe the manipulator:

$$l_1 = l_2 = 0.5\text{m},$$
$$m_1 = 4.6\text{Kg},$$
$$m_2 = 2.3\text{Kg},$$
$$m_3 = 1.0\text{Kg},$$
$$g = 9.8\text{m/s}^2.$$

For the first two links, we assume that the mass is all concentrated at the distal end of the link. For link 3, we assume that the center of mass is located at the origin of frame {3}—that is, at the proximal end of the link. The inertia tensor for link 3 is

$${}^{C_3}I = \begin{bmatrix} 0.05 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \text{Kg-m}^2.$$

The vectors that locate each center of mass relative to the respective link frame are

$$^1P_{C_1} = l_1\hat{X}_1.$$
$$^2P_{C_2} = l_2\hat{X}_2.$$
$$^3P_{C_3} = 0.$$

2. Write a simulator for the three-link manipulator. A simple Euler-integration routine is sufficient for performing the numerical integration (as in Section 6.12). To keep your code modular, it might be helpful to define the routine

```
Procedure UPDATE(VAR tau: vec3; VAR period: real; VAR
theta, thetadot: vec3);
```

where "tau" is the torque command to the manipulator (always zero for this assignment), "period" is the length of time you wish to advance time (in seconds), and "theta" and "thetadot" are the *state* of the manipulator. Theta and thetadot are updated by "period" seconds each time you call UPDATE. Note that "period" would typically be longer than the integration step size, Δt, used in the numerical integration. For example, although the step size for numerical integration might be 0.001 second, you might wish to print out the manipulator position and velocity only each 0.1 seconds.

To test your simulation, set the joint-torque commands to zero (for all time) and perform these tests:

(a) Set the initial position of the manipulator to

$$[\theta_1\ \theta_2\ \theta_3] = [-90\ 0\ 0].$$

Simulate for a few seconds. Is the motion of the manipulator what you would expect?

(b) Set the initial position of the manipulator to

$$[\theta_1\ \theta_2\ \theta_3] = [30\ 30\ 10].$$

Simulate for a few seconds. Is the motion of the manipulator what you would expect?

(c) Introduce some viscous friction at each joint of the simulated manipulator—that is, add a term to the dynamics of each joint in the form $\tau_f = v\dot{\theta}$, where $v = 5.0$ newton-meter-seconds for each joint. Repeat test (b) above. Is the motion what you would expect?

MATLAB EXERCISE 6A

This exercise focuses on the inverse-dynamics analysis (in a resolved-rate control framework—see MATLAB Exercise 5) for the planar 2-DOF 2R robot. This robot is the first two R-joints and first two moving links of the planar 3-DOF 3R robot. (See Figures 3.6 and 3.7; the DH parameters are given in the first two rows of Figure 3.8.)

For the planar 2R robot, calculate the required joint torques (i.e., solve the inverse-dynamics problem) to provide the commanded motion at every time step in a resolved-rate control scheme. You can use either numerical Newton–Euler recursion or the analytical equations from the results of Exercise 6.2, or both.

Given: $L_1 = 1.0$ m, $L_2 = 0.5$ m; Both links are solid steel with mass density $\rho = 7806$ kg/m^3; both have the width and thickness dimensions $w = t = 5$ cm. The revolute joints are assumed to be perfect, connecting the links at their very edges (not physically possible).

The initial angles are $\Theta = \begin{Bmatrix}\theta_1\\ \theta_2\end{Bmatrix} = \begin{Bmatrix}10^\circ\\ 90^\circ\end{Bmatrix}$.

The (constant) commanded Cartesian velocity is $^0\dot{X} = {}^0\begin{Bmatrix}\dot{x}\\ \dot{y}\end{Bmatrix} = {}^0\begin{Bmatrix}0\\ 0.5\end{Bmatrix}$ (m/s).

Simulate motion for 1 sec, with a control time step of 0.01 sec.
Present five plots (each set on a separate graph, please):

1. the two joint angles (degrees) $\Theta = [\theta_1\ \ \theta_2]^T$ vs. time;
2. the two joint rates (rad/s) $\dot{\Theta} = [\dot{\theta}_1\ \ \dot{\theta}_2]^T$ vs. time;
3. the two joint accelerations (rad/s^2) $\ddot{\Theta} = [\ddot{\theta}_1\ \ \ddot{\theta}_2]^T$ vs. time;
4. the three Cartesian components of ${}^0_HT$, $X = [x\ \ y\ \ \phi]^T$ (rad is fine for ϕ so it will fit) vs. time;
5. the two inverse dynamics joint torques (Nm) $\mathrm{T} = [\tau_1\ \ \tau_2]^T$ vs. time.

Carefully label (by hand is fine!) each component on each plot. Also, label the axis names and units.

Perform this simulation twice. The first time, ignore gravity (the motion plane is normal to the effect of gravity); the second time, consider gravity g in the negative Y direction.

MATLAB EXERCISE 6B

This exercise focuses on the inverse-dynamics solution for the planar 3-DOF, 3R robot (of Figures 3.6 and 3.7; the DH parameters are given in Figure 3.8) for a motion snapshot in time only. The following fixed-length parameters are given: $L_1 = 4$, $L_2 = 3$, and $L_3 = 2$ (m). For dynamics, we must also be given mass and moment-of-inertia information: $m_1 = 20$, $m_2 = 15$, $m_3 = 10$ (kg), ${}^CI_{ZZ1} = 0.5$, ${}^CI_{ZZ2} = 0.2$, and ${}^CI_{ZZ3} = 0.1$ (kgm^2). Assume that the CG of each link is in its geometric center. Also, assume that gravity acts in the $-Y$ direction in the plane of motion. For this exercise, ignore actuator dynamics and the joint gearing.

a) Write a MATLAB program to implement the recursive Newton–Euler inverse-dynamics solution (i.e., given the commanded motion, calculate the required driving joint torques) for the following motion snapshot in time:

$$\Theta = \begin{Bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{Bmatrix} = \begin{Bmatrix}10^\circ\\ 20^\circ\\ 30^\circ\end{Bmatrix}\quad \dot{\Theta} = \begin{Bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{Bmatrix} = \begin{Bmatrix}1\\ 2\\ 3\end{Bmatrix}\text{ (rad/s)}\ddot{\Theta} = \begin{Bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\end{Bmatrix} = \begin{Bmatrix}0.5\\ 1\\ 1.5\end{Bmatrix}\text{ (rad/s}^2\text{)}$$

b) Check your results in (a) by means of the Corke MATLAB Robotics Toolbox. Try functions *rne()* and *gravload()*.

MATLAB EXERCISE 6C

This exercise focuses on the forward-dynamics solution for the planar 3-DOF, 3R robot (parameters from MATLAB Exercise 6B) for motion over time. In this case, ignore gravity (i.e., assume that gravity acts in a direction normal to the plane of motion). Use the Corke MATLAB Robotics Toolbox to solve the forward-dynamics problem (i.e.,

given the commanded driving joint torques, calculate the resulting robot motion) for the following constant joint torques and the given initial joint angles and initial joint rates:

$$T = \begin{Bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{Bmatrix} = \begin{Bmatrix} 20 \\ 5 \\ 1 \end{Bmatrix} \text{ (Nm, constant)} \quad \Theta_0 = \begin{Bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{Bmatrix} = \begin{Bmatrix} -60^\circ \\ 90^\circ \\ 30^\circ \end{Bmatrix}$$

$$\dot{\Theta}_0 = \begin{Bmatrix} \dot{\theta}_{10} \\ \dot{\theta}_{20} \\ \dot{\theta}_{30} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \text{ (rad/s)}$$

Perform this simulation for 4 seconds. Try function *fdyn()*.

Present two plots for the resulting robot motion (each set on a separate graph, please):

1. the three joint angles (degrees) $\Theta = \{\theta_1 \ \theta_2 \ \theta_3\}^T$ vs. time;
2. the three joint rates (rad/s) $\dot{\Theta} = \{\dot{\theta}_1 \ \dot{\theta}_2 \ \dot{\theta}_3\}^T$ vs. time.

Carefully label (by hand is fine!) each component on each plot. Also, label the axis names and units.

CHAPTER 7

Trajectory generation

7.1 INTRODUCTION

In this chapter, we concern ourselves with methods of computing a trajectory that describes the desired motion of a manipulator in multidimensional space. Here, **trajectory** refers to a time history of position, velocity, and acceleration for each degree of freedom.

This problem includes the human-interface problem of how we wish to *specify* a trajectory or path through space. In order to make the description of manipulator motion easy for a human user of a robot system, the user should not be required to write down complicated functions of space and time to specify the task. Rather, we must allow the capability of specifying trajectories with simple descriptions of the desired motion, and let the system figure out the details. For example, the user might want to be able to specify nothing more than the desired goal position and orientation of the end-effector and leave it to the system to decide on the exact shape of the path to get there, the duration, the velocity profile, and other details.

We also are concerned with how trajectories are *represented* in the computer after they have been planned. Finally, there is the problem of actually computing the trajectory from the internal representation—or *generating* the trajectory. Generation occurs at *run time*; in the most general case, position, velocity, and acceleration are computed. These trajectories are computed on digital computers, so the trajectory points are computed at a certain rate, called the **path-update rate**. In typical manipulator systems, this rate lies between 60 and 2000 Hz.

7.2 GENERAL CONSIDERATIONS IN PATH DESCRIPTION AND GENERATION

For the most part, we will consider motions of a manipulator as motions of the tool frame, $\{T\}$, relative to the station frame, $\{S\}$. This is the same manner

CHAPTER 8

Manipulator-mechanism design

8.1 INTRODUCTION

In previous chapters, we have seen that the particular structure of a manipulator influences kinematic and dynamic analysis. For example, some kinematic configurations will be easy to solve; others will have no closed-form kinematic solution. Likewise, the complexity of the dynamic equations can vary greatly with the kinematic configuration and the mass distribution of the links. In coming chapters, we will see that manipulator control depends not only on the rigid-body dynamics, but also upon the friction and flexibility of the drive systems.

The tasks that a manipulator can perform will also vary greatly with the particular design. Although we have generally dealt with the robot manipulator as an abstract entity, its performance is ultimately limited by such pragmatic factors as load capacity, speed, size of workspace, and repeatability. For certain applications, the overall manipulator size, weight, power consumption, and cost will be significant factors.

This chapter discusses some of the issues involved in the design of the manipulator. In general, methods of design and even the evaluation of a finished design are partially subjective topics. It is difficult to narrow the spectrum of design choices with many hard and fast rules.

The elements of a robot system fall roughly into four categories:

1. The manipulator, including its internal or **proprioceptive** sensors;
2. the end-effector, or **end-of-arm tooling**;
3. external sensors and effectors, such as vision systems and part feeders; and
4. the controller.

The breadth of engineering disciplines encompassed forces us to restrict our attention only to the design of the manipulator itself.

In developing a manipulator design, we will start by examining the factors likely to have the greatest overall effect on the design and then consider more detailed questions. Ultimately, however, designing a manipulator is an iterative process. More often than not, problems that arise in the solving of a design detail will force rethinking of previous higher level design decisions.

8.2 BASING THE DESIGN ON TASK REQUIREMENTS

Although robots are nominally "universally programmable" machines capable of performing a wide variety of tasks, economies and practicalities dictate that different manipulators be designed for particular types of tasks. For example, large robots capable of handling payloads of hundreds of pounds do not generally have the capability to insert electronic components into circuit boards. As we shall see, not only the size, but the number of joints, the arrangement of the joints, and the types of actuation, sensing, and control will all vary greatly with the sort of task to be performed.

Number of degrees of freedom

The number of degrees of freedom in a manipulator should match the number required by the task. Not all tasks require a full six degrees of freedom.

The most common such circumstance occurs when the end-effector has an axis of symmetry. Figure 8.1 shows a manipulator positioning a grinding tool in two different ways. In this case, the orientation of the tool with respect to the axis of the tool, $\hat{Z}_T$, is immaterial, because the grinding wheel is spinning at several hundred RPM. To say that we can position this 6-DOF robot in an infinity of ways for this task (rotation about $\hat{Z}_T$ is a free variable), we say that the robot is **redundant** for this task. Arc welding, spot welding, deburring, glueing, and polishing provide other examples of tasks that often employ end-effectors with at least one axis of symmetry.

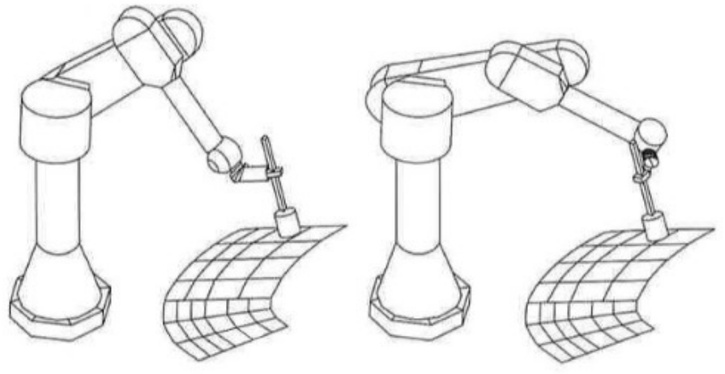


FIGURE 8.1: A 6-DOF manipulator with a symmetric tool contains a redundant degree of freedom.

In analyzing the symmetric-tool situation, it is sometimes helpful to imagine a *fictitious joint* whose axis lies along the axis of symmetry. In positioning any end-effector to a specific pose, we need a total of six degrees of freedom. Because one of these six is our fictitious joint, the actual manipulator need not have more than five degrees of freedom. If a 5-DOF robot were used in the application of Fig. 8.1, then we would be back to the usual case in which only a finite number of different solutions are available for positioning the tool. Quite a large percentage of existing industrial robots are 5-DOF, in recognition of the relative prevalence of symmetric-tool applications.

Some tasks are performed in domains that, fundamentally, have fewer than six degrees of freedom. Placement of components on circuit boards provides a common example of this. Circuit boards generally are planar and contain parts of various heights. Positioning parts on a planar surface requires three degrees of freedom (x, y, and θ); in order to lift and insert the parts, a fourth motion normal to the plane is added (z).

Robots with fewer than six degrees of freedom can also perform tasks in which some sort of active positioning device presents the parts. In welding pipes, for example, a tilt/roll platform, shown in Fig. 8.2, often presents the parts to be welded. In counting the number of degrees of freedom between the pipes and the end-effector, the tilt/roll platform accounts for two. This, together with the fact that arc welding is a symmetric-tool task, means that, in theory, a 3-DOF manipulator could be used. In practice, realities such as the need to avoid collisions with the workpiece generally dictate the use of a robot with more degrees of freedom.

Parts with an axis of symmetry also reduce the required degrees of freedom for the manipulator. For example, cylindrical parts can in many cases be picked up and inserted independent of the orientation of the gripper with respect to the axis of the cylinder. Note, however, that after the part is grasped, the orientation of the part about its symmetric axis must fail to matter for *all* subsequent operations, because its orientation is not guaranteed.

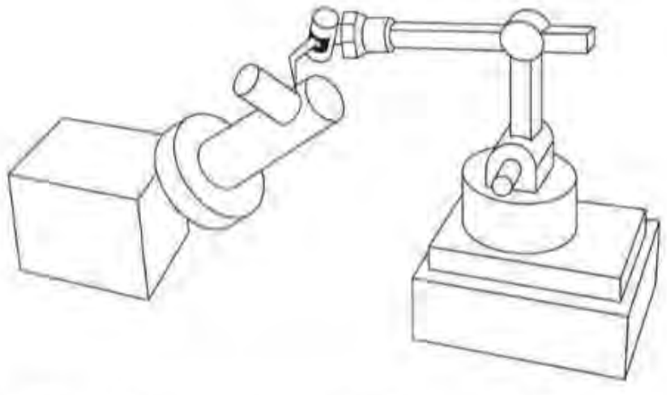


FIGURE 8.2: A tilt/roll platform provides two degrees of freedom to the overall manipulator system.

Workspace

In performing tasks, a manipulator has to reach a number of workpieces or fixtures. In some cases, these can be positioned as needed to suit the workspace of the manipulator. In other cases, a robot can be installed in a fixed environment with rigid workspace requirements. **Workspace** is also sometimes called **work volume** or **work envelope**.

The overall scale of the task sets the required workspace of the manipulator. In some cases, the details of the shape of the workspace and the location of workspace singularities will be important considerations.

The intrusion of the manipulator itself in the workspace can sometimes be a factor. Depending on the kinematic design, operating a manipulator in a given application could require more or less space around the fixtures in order to avoid collisions. Restricted environments can affect the choice of kinematic configuration.

Load capacity

The **load capacity** of a manipulator depends upon the sizing of its structural members, power-transmission system, and actuators. The load placed on actuators and drive system is a function of the configuration of the robot, the percentage of time supporting a load, and dynamic loading due to inertial- and velocity-related forces.

Speed

An obvious goal in design has been for faster and faster manipulators. High speed offers advantages in many applications when a proposed robotic solution must compete on economic terms with hard automation or human workers. For some applications, however, the process itself limits the speed rather than the manipulator. This is the case with many welding and spray-painting applications.

An important distinction is that between the maximum end-effector speed and the overall **cycle time** for a particular task. For pick-and-place applications, the manipulator must accelerate and decelerate to and from the *pick* and *place* locations within some positional accuracy bounds. Often, the acceleration and deceleration phases take up most of the cycle time. Hence, acceleration capability, not just peak speed, is very important.

Repeatability and accuracy

High repeatability and accuracy, although desirable in any manipulator design, are expensive to achieve. For example, it would be absurd to design a paint-spraying robot to be accurate to within 0.001 inches when the spray spot diameter is 8 inches ±2 inches. To a large extent, accuracy of a particular model of industrial robot depends upon the details of its manufacture rather than on its design. High accuracy is achieved by having good knowledge of the link (and other) parameters. Making it possible are accurate measurements after manufacture or careful attention to tolerances during manufacturing.

8.3 KINEMATIC CONFIGURATION

Once the required number of degrees of freedom has been decided upon, a particular configuration of joints must be chosen to realize those freedoms. For serial kinematic linkages, the number of joints equals the required number of degrees of freedom. Most manipulators are designed so that the last $n-3$ joints orient the end-effector and have axes that intersect at the **wrist point**, and the first three joints position this wrist point. Manipulators with this design could be said to be composed of a **positioning structure** followed by an **orienting structure** or **wrist**. As we saw in Chapter 4, these manipulators always have closed-form kinematic solutions. Although other configurations exist that possess closed-form kinematic solutions, almost every industrial manipulator belongs to this **wrist-partitioned** class of mechanisms. Furthermore, the positioning structure is almost without exception designed to be kinematically simple, having link twists equal to 0° or ±90° and having many of the link lengths and offsets equal to zero.

It has become customary to classify manipulators of the wrist-partitioned, kinematically simple class according to the design of their first three joints (the positioning structure). The following paragraphs briefly describe the most common of these classifications.

Cartesian

A **Cartesian manipulator** has perhaps the most straightforward configuration. As shown in Fig. 8.3, joints 1 through 3 are prismatic, mutually orthogonal, and correspond to the $\hat{X}$, $\hat{Y}$, and $\hat{Z}$ Cartesian directions. The inverse kinematic solution for this configuration is trivial.

This configuration produces robots with very stiff structures. As a consequence, very large robots can be built. These large robots, often called **gantry robots**, resemble overhead gantry cranes. Gantry robots sometimes manipulate entire automobiles or inspect entire aircraft.

The other advantages of Cartesian manipulators stem from the fact that the first three joints are *decoupled*. This makes them simpler to design and prevents kinematic singularities due to the first three joints.

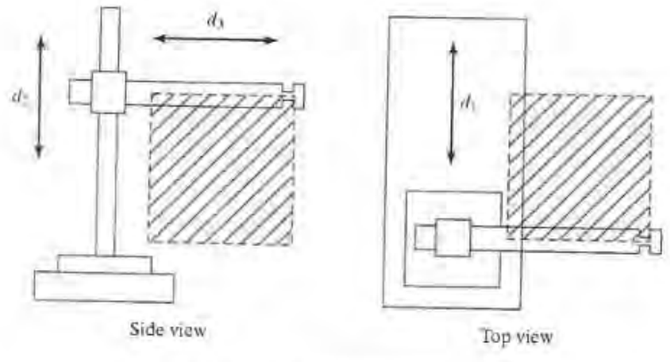


FIGURE 8.3: A Cartesian manipulator.

Their primary disadvantage is that all of the feeders and fixtures associated with an application must lie "inside" the robot. Consequently, application workcells for Cartesian robots become very machine dependent. The size of the robot's support structure limits the size and placement of fixtures and sensors. These limitations make retrofitting Cartesian robots into existing workcells extremely difficult.

Articulated

Figure 8.4 shows an **articulated manipulator**, sometimes also called a **jointed, elbow**, or **anthropomorphic** manipulator. A manipulator of this kind typically consists of two "shoulder" joints (one for rotation about a vertical axis and one for elevation out of the horizontal plane), an "elbow" joint (whose axis is usually parallel to the shoulder elevation joint), and two or three wrist joints at the end of the manipulator. Both the PUMA 560 and the Motoman L-3, which we studied in earlier chapters, fall into this class.

Articulated robots minimize the intrusion of the manipulator structure into the workspace, making them capable of reaching into confined spaces. They require much less overall structure than Cartesian robots, making them less expensive for applications needing smaller workspaces.

SCARA

The **SCARA**[1] configuration, shown in Fig. 8.5, has three parallel revolute joints (allowing it to move and orient in a plane), with a fourth prismatic joint for moving the end-effector normal to the plane. The chief advantage is that the first three joints don't have to support any of the weight of the manipulator or the load. In addition, link 0 can easily house the actuators for the first two joints. The actuators can be made very large, so the robot can move very fast. For example, the Adept

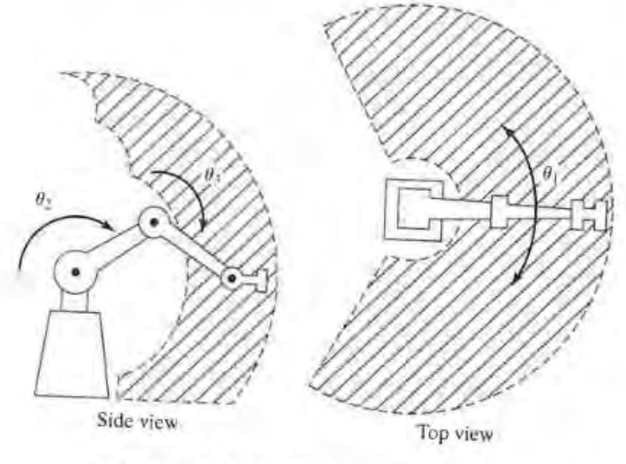


FIGURE 8.4: An articulated manipulator.

[1] SCARA stands for "selectively compliant assembly robot arm."

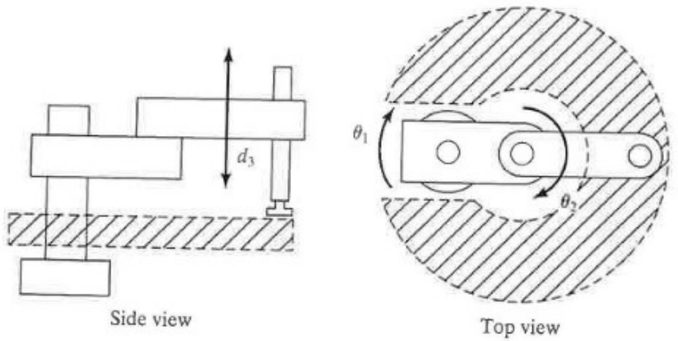


FIGURE 8.5: A SCARA manipulator.

One SCARA manipulator can move at up to 30 feet per second, about 10 times faster than most articulated industrial robots [1]. This configuration is best suited to planar tasks.

Spherical

The spherical configuration in Fig. 8.6 has many similarities to the articulated manipulator, but with the elbow joint replaced by a prismatic joint. This design is better suited to some applications than is the elbow design. The link that moves prismatically might telescope—or even "stick out the back" when retracted.

Cylindrical

Cylindrical manipulators (Fig. 8.7) consist of a prismatic joint for translating the arm vertically, a revolute joint with a vertical axis, another prismatic joint orthogonal to the revolute joint axis, and, finally, a wrist of some sort.

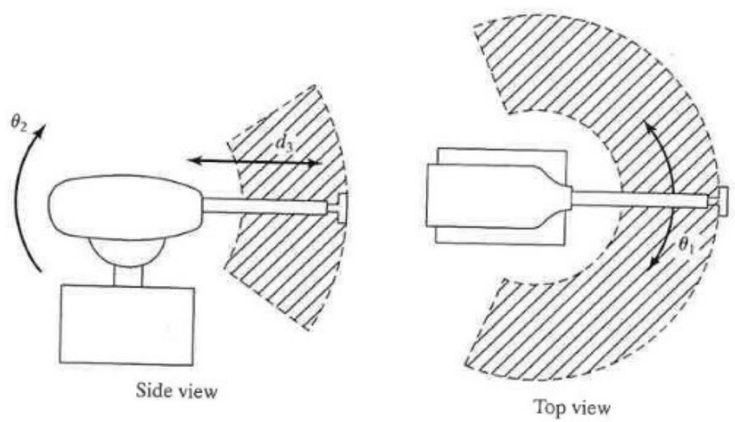


FIGURE 8.6: A spherical manipulator.

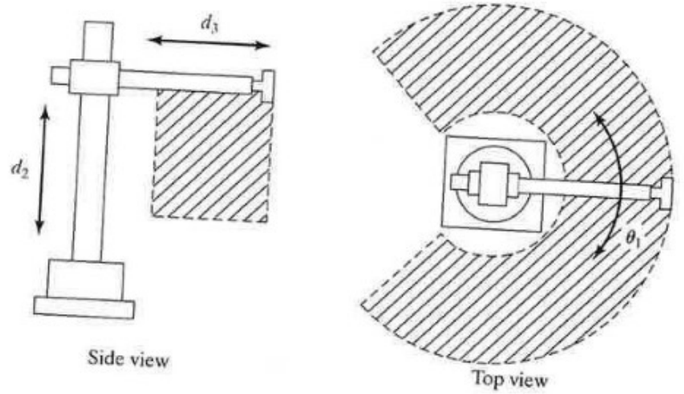


FIGURE 8.7: A cylindrical manipulator.

Wrists

The most common wrist configurations consist of either two or three revolute joints with orthogonal, intersecting axes. The first of the wrist joints usually forms joint 4 of the manipulator.

A configuration of three orthogonal axes will guarantee that any orientation can be achieved (assuming no joint-angle limits) [2]. As was stated in Chapter 4, any manipulator with three consecutive intersecting axes will possess a closed-form kinematic solution. Therefore, a three-orthogonal-axis wrist can be located at the end of the manipulator in any desired orientation with no penalty. Figure 8.8 is a schematic of one possible design of such a wrist, which uses several sets of bevel gears to drive the mechanism from remotely located actuators.

In practice, it is difficult to build a three-orthogonal-axis wrist not subject to rather severe joint-angle limitations. An interesting design used in several robots

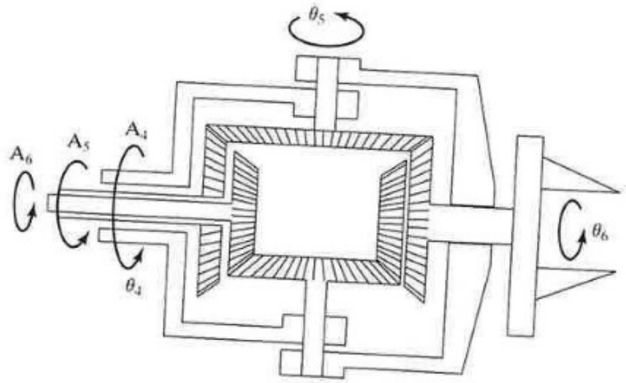


FIGURE 8.8: An orthogonal-axis wrist driven by remotely located actuators via three concentric shafts.

manufactured by Cincinatti Milacron (Fig. 1.4) employs a wrist that has three intersecting but nonorthogonal axes. In this design (called the "three roll wrist"), all three joints of the wrist can rotate continuously without limits. The nonorthogonality of the axes creates, however, a set of orientations that are impossible to reach with this wrist. This set of unattainable orientations is described by a cone within which the third axis of the wrist cannot lie. (See Exercise 8.11.) However, the wrist can be mounted to link 3 of the manipulator in such a way that the link structure occupies this cone and so would be block access anyway. Figure 8.9 shows two drawings of such a wrist [24].

Some industrial robots have wrists that do *not* have intersecting axes. This implies that a closed-form kinematic solution might not exist. If, however, the wrist is mounted on an articulated manipulator in such a way that the joint-4 axis is parallel to the joint-2 and -3 axes, as in Fig. 8.10, there *will* be a closed-form kinematic solution. Likewise, a nonintersecting-axis wrist mounted on a Cartesian robot yields a closed-form-solvable manipulator.

Typically, 5-DOF welding robots use two-axis wrists oriented as shown in Fig. 8.11. Note that, if the robot has a symmetric tool, this "fictitious joint" must follow the rules of wrist design. That is, in order to reach all orientations, the tool must be mounted with its axis of symmetry orthogonal to the joint-5 axis. In the worst case, when the axis of symmetry is parallel to the joint-5 axis, the fictitious sixth axis is in a permanently singular configuration.

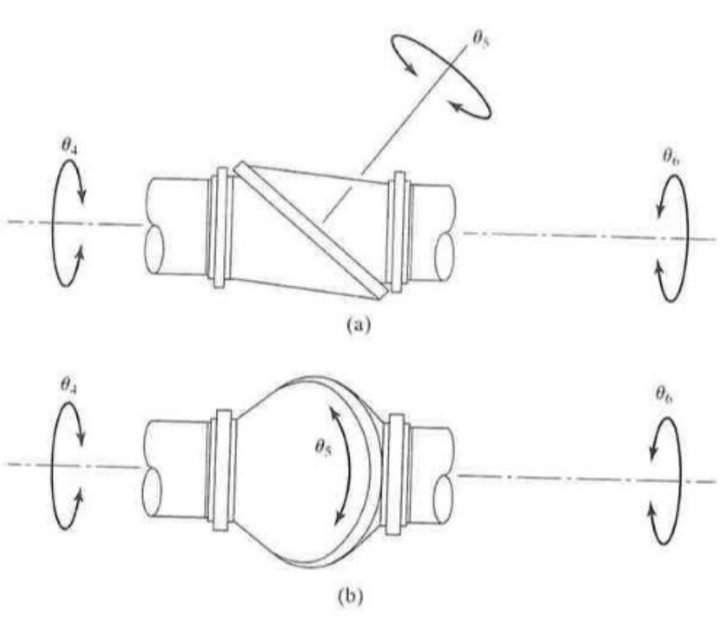


FIGURE 8.9: Two views of a nonorthogonal-axis wrist [24]. From *International Encyclopedia of Robotics*, by R. Dorf and S. Nof (editors). From "Wrists" by M. Rosheim. John C. Wiley and Sons, Inc., New York, NY ©1988. Reprinted by permission.

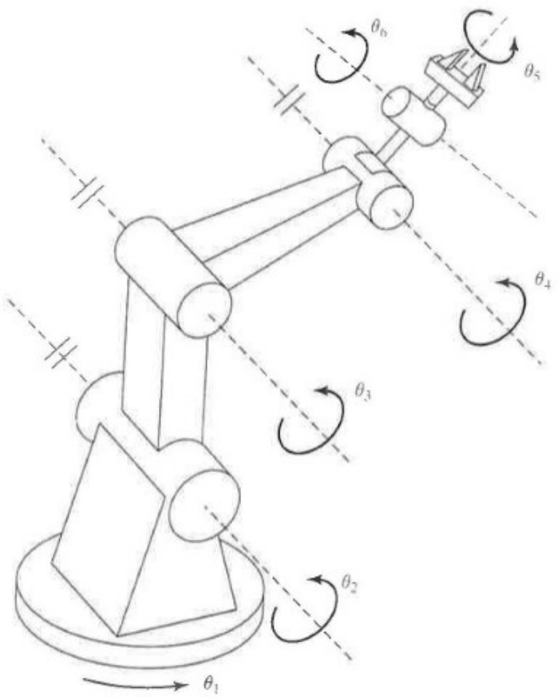


FIGURE 8.10: A manipulator with a wrist whose axes do not intersect. However, this robot does possess a closed-form kinematic solution.

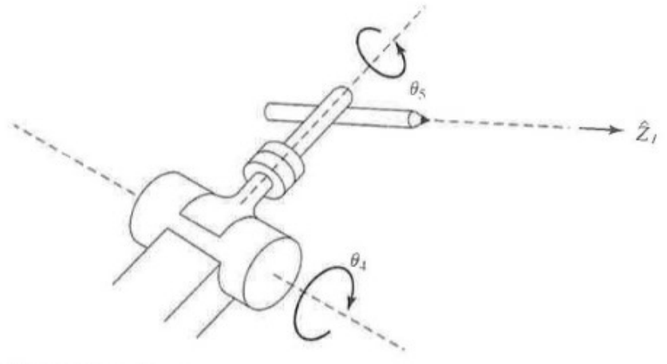


FIGURE 8.11: Typical wrist design of a 5-DOF welding robot.

8.4 QUANTITATIVE MEASURES OF WORKSPACE ATTRIBUTES

Manipulator designers have proposed several interesting quantitative measures of various workspace attributes.

Efficiency of design in terms of generating workspace

Some designers noticed that it seemed to take more material to build a Cartesian manipulator than to build an articulated manipulator of similar workspace volume. To get a quantitative handle on this, we first define the **length sum** of a manipulator as

$$L = \sum_{i=1}^{N} (a_{i-1} + d_i), \tag{8.1}$$

where a_{i-1} and d_i are the link length and joint offset as defined in Chapter 3. Thus, the length sum of a manipulator gives a rough measure of the "length" of the complete linkage. Note that, for prismatic joints, d_i must here be interpreted as a constant equal to the length of travel between the joint-travel limits.

In [3], the **structural length index**, Q_L, is defined as the ratio of the manipulator's length sum to the cube root of the workspace volume—that is,

$$Q_L = L/\sqrt[3]{w}, \tag{8.2}$$

where L is given in (8.1) and W is the volume of the manipulator's workspace. Hence, Q_L attempts to index the relative amount of structure (linkage length) required by different configurations to generate a given work volume. Thus, a good design would be one in which a manipulator with a small length sum nonetheless possessed a large workspace volume. Good designs have a low Q_L.

Considering just the positioning structure of a Cartesian manipulator (and therefore the workspace of the wrist point), the value of Q_L is minimized when all three joints have the same length of travel. This minimal value is $Q_L = 3.0$. On the other hand, an ideal articulated manipulator, such as the one in Fig. 8.4, has $Q_L = \frac{1}{\sqrt[3]{4\pi/3}} \cong 0.62$. This helps quantify our earlier statement that articulated manipulators are superior to other configurations in that they have minimal intrusion into their own workspace. Of course, in any real manipulator structure, the figure just given would be made somewhat larger by the effect of joint limits in reducing the workspace volume.

EXAMPLE 8.1

A SCARA manipulator like that of Fig. 8.5 has links 1 and 2 of equal length $l/2$, and the range of motion of the prismatic joint 3 is given by d_3. Assume for simplicity that the joint limits are absent, and find Q_L. What value of d_3 minimizes Q_L and what is this minimal value?

The length sum of this manipulator is $L = l/2 + l/2 + d_3 = l + d_3$, and the workspace volume is that of a right cylinder of radius l and height d_3; therefore,

$$Q_L = \frac{l + d_3}{\sqrt[3]{\pi l^2 d_3}}. \tag{8.3}$$

Minimizing Q_L as a function of the ratio d_3/l gives $d_3 = l/2$ as optimal [3]. The corresponding minimal value of Q_L is 1.29.

Designing well-conditioned workspaces

At singular points, a manipulator effectively loses one or more degrees of freedom, so certain tasks may not be able to be performed at that point. In fact, in the neighborhood of singular points (including workspace-boundary singularities), actions of the manipulator could fail to be **well-conditioned**. In some sense, the farther the manipulator is away from singularities, the better able it is to move uniformly and apply forces uniformly in all directions. Several measures have been suggested for quantifying this effect. The use of such measures at design time might yield a manipulator design with a maximally large well-conditioned subspace of the workspace.

Singular configurations are given by

$$\det(J(\Theta)) = 0, \tag{8.4}$$

so it is natural to use the determinant of the Jacobian in a measure of manipulator dexterity. In [4], the **manipulability measure**, w, is defined as

$$w = \sqrt{\det(J(\Theta)J^T(\Theta))}, \tag{8.5}$$

which, for a nonredundant manipulator, reduces to

$$w = |\det(J(\Theta))| \tag{8.6}$$

A good manipulator design has large areas of its workspace characterized by high values of w.

Whereas velocity analysis motivated (8.6), other researchers have proposed manipulability measures based on acceleration analysis or force-application capability. Asada [5] suggested an examination of the eigenvalues of the Cartesian mass matrix

$$M_x(\Theta) = J^{-T}(\Theta)M(\Theta)J^{-1}(\Theta) \tag{8.7}$$

as a measure of how well the manipulator can accelerate in various Cartesian directions. He suggests a graphic representation of this measure as an **inertia ellipsoid**, given by

$$X^T M_x(\Theta) X = 1, \tag{8.8}$$

the equation of an n-dimensional ellipse, where n is the dimension of X. The axes of the ellipsoid given in (8.8) lie in the directions of the eigenvectors of $M_x(\Theta)$, and the reciprocals of the square roots of the corresponding eigenvalues provide the lengths of the axes of the ellipsoid. Well-conditioned points in the manipulator workspace are characterized by inertia ellipsoids that are spherical (or nearly so).

Figure 8.12 shows graphically the properties of a planar two-link manipulator. In the center of the workspace, the manipulator is well conditioned, as is indicated by nearly circular ellipsoids. At workspace boundaries, the ellipses flatten, indicating the manipulator's difficulty in accelerating in certain directions.

Other measures of workspace conditioning have been proposed in [6–8, 25].

8.5 REDUNDANT AND CLOSED-CHAIN STRUCTURES

In general, the scope of this book is limited to manipulators that are serial-chain linkages of six or fewer joints. In this section, however, we briefly discuss manipulators outside of this class.

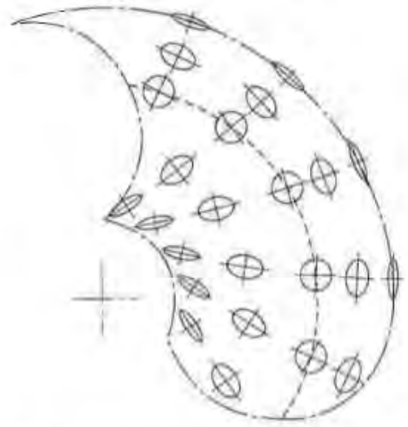


FIGURE 8.12: Workspace of a 2-DOF planar arm, showing inertia ellipsoids, from [5] (© 1984 IEEE). The dashed line indicates a locus of isotropic points in the workspace. Reprinted by permission.

Micromanipulators and other redundancies

General spatial positioning capability requires only six degrees of freedom, but there are advantages to having even more controllable joints.

One use for these extra freedoms is already finding some practical application [9,10] and is of growing interest in the research community: a **micromanipulator**. A micromanipulator is generally formed by several fast, precise degrees of freedom located near the distal end of a "conventional" manipulator. The conventional manipulator takes care of large motions, while the micromanipulator, whose joints generally have a small range of motion, accomplishes fine motion and force control.

Additional joints can also help a mechanism avoid singular configurations, as is suggested in [11, 12]. For example, any three-degree-of-freedom wrist will suffer from singular configurations (when all three axes lie in a plane), but a four-degree-of-freedom wrist can effectively avoid such configurations [13–15].

Figure 8.13 shows two configurations suggested [11, 12] for seven-degree-of-freedom manipulators.

A major potential use of redundant robots is in avoiding collisions while operating in cluttered work environments. As we have seen, a six-degree-of-freedom manipulator can reach a given position and orientation in only a finite number of ways. The addition of a seventh joint allows an infinity of ways, permitting the desire to avoid obstacles to influence the choice.

Closed-loop structures

Although we have considered only serial-chain manipulators in our analysis, some manipulators contain **closed-loop structures**. For example, the Motoman L-3 robot described in Chapters 3 and 4 possesses closed-loop structures in the drive mechanism of joints 2 and 3. Closed-loop structures offer a benefit: increased stiffness of

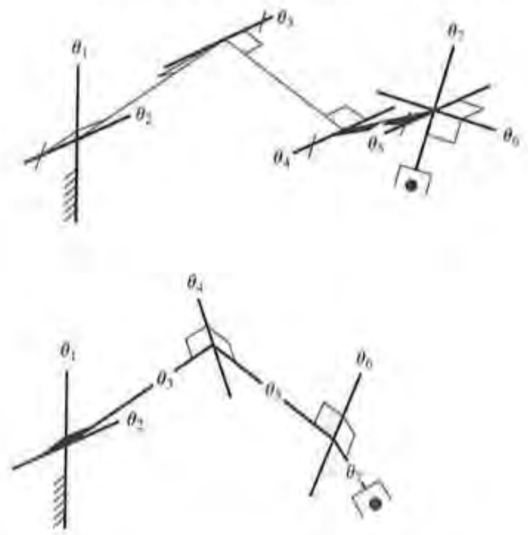


FIGURE 8.13: Two suggested seven-degree-of-freedom manipulator designs [3].

the mechanism [16]. On the other hand, closed-loop structures generally reduce the allowable range of motion of the joints and thus decrease the workspace size.

Figure 8.14 depicts a **Stewart mechanism**, a closed-loop alternative to the serial 6-DOF manipulator. The position and orientation of the "end-effector" is controlled by the lengths of the six linear actuators which connect it to the base. At the base end, each actuator is connected by a two-degree-of-freedom universal joint. At the end-effector, each actuator is attached with a three-degree-of-freedom ball-and-socket joint. It exhibits characteristics common to most closed-loop mechanisms: it can be made very stiff, but the links have a much more limited range of motion than do serial linkages. The Stewart mechanism, in particular, demonstrates an interesting reversal in the nature of the forward and inverse kinematic solutions: the inverse solution is quite simple, whereas the forward solution is typically quite complex, sometimes lacking a closed-form formulation. (See Exercises 8.7 and 8.12.)

In general, the number of degrees of freedom of a closed-loop mechanism is not obvious. The total number of degrees of freedom can be computed by means of **Grübler's** formula [17],

$$F = 6(l - n - 1) + \sum_{i=1}^{n} f_i. \tag{8.9}$$

where F is the total number of degrees of freedom in the mechanism, l is the number of links (including the base), n is the total number of joints, and f_i is the number of degrees of freedom associated with the ith joint. A planar version of Grübler's formula (when all objects are considered to have three degrees of freedom if unconstrained) is obtained by replacing the 6 in (8.9) with a 3.

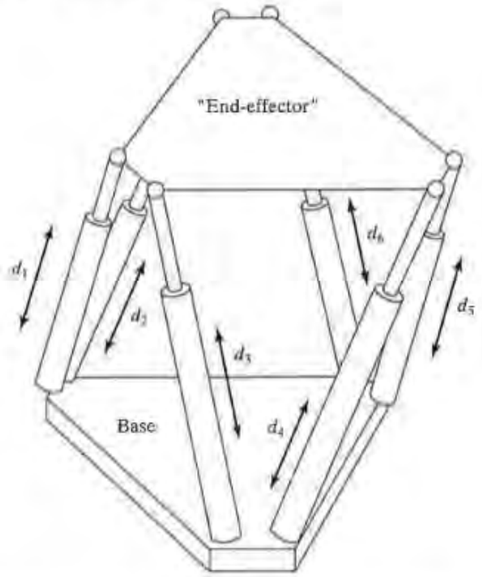


FIGURE 8.14: The Stewart mechanism is a six-degree-of-freedom fully parallel manipulator.

EXAMPLE 8.2

Use Grübler's formula to verify that the Stewart mechanism (Fig. 8.14) indeed has six degrees of freedom.

The number of joints is 18 (6 universal, 6 ball and socket, and 6 prismatic in the actuators). The number of links is 14 (2 parts for each actuator, the end-effector, and the base). The sum of all the joint freedoms is 36. Using Grübler's formula, we can verify that the total number of degrees of freedom is six:

$$F = 6(14 - 18 - 1) + 36 = 6. \tag{8.10}$$

8.6 ACTUATION SCHEMES

Once the general kinematic structure of a manipulator has been chosen, the next most important matter of concern is the actuation of the joints. Typically, the actuator, reduction, and transmission are closely coupled and must be designed together.

Actuator location

The most straightforward choice of actuator location is at or near the joint it drives. If the actuator can produce enough torque or force, its output can attach directly to the joint. This arrangement, known as a **direct-drive** configuration [18], offers

the advantages of simplicity in design and superior controllability—that is, with no transmission or reduction elements between the actuator and the joint, the joint motions can be controlled with the same fidelity as the actuator itself.

Unfortunately, many actuators are best suited to relatively high speeds and low torques and therefore require a **speed-reduction system**. Furthermore, actuators tend to be rather heavy. If they can be located remotely from the joint and toward the base of the manipulator, the overall inertia of the manipulator can be reduced considerably. This, in turn, reduces the size needed for the actuators. To realize these benefits, a **transmission system** is needed to transfer the motion from the actuator to the joint.

In a joint-drive system with a remotely mounted actuator, the reduction system could be placed either at the actuator or at the joint. Some arrangements combine the functions of transmission and reduction. Aside from added complexity, the major disadvantage of reduction and transmission systems is that they introduce additional friction and flexibility into the mechanism. When the reduction is at the joint, the transmission will be working at higher speeds and lower torques. Lower torque means that flexibility will be less of a problem. However, if the weight of the reducer is significant, some of the advantage of remotely mounted actuators is lost.

In Chapter 3, details were given for the actuation scheme of the Yasukawa Motoman L-3, which is typical of a design in which actuators are mounted remotely and resulting joint motions are coupled. Equations (3.16) show explicitly how actuator motions cause joint motions. Note, for example, that motion of actuator 2 causes motion of joints 2, 3, and 4.

The optimal distribution of reduction stages throughout the transmission will depend ultimately on the flexibility of the transmission, the weight of the reduction system, the friction associated with the reduction system, and the ease of incorporating these components into the overall manipulator design.

Reduction and transmission systems

Gears are the most common element used for reduction. They can provide for large reductions in relatively compact configurations. Gear pairs come in various configurations for parallel shafts (spur gears), orthogonal intersecting shafts (bevel gears), skew shafts (worm gears or cross helical gears), and other configurations. Different types of gears have different load ratings, wear characteristics, and frictional properties.

The major disadvantages of using gearing are added **backlash** and friction. Backlash, which arises from the imperfect meshing of gears, can be defined as the maximum angular motion of the output gear when the input gear remains fixed. If the gear teeth are meshed tightly to eliminate backlash, there can be excessive amounts of friction. Very precise gears and very precise mounting minimize these problems, but also increase cost.

The **gear ratio**, η, describes the speed-reducing and torque-increasing effects of a gear pair. For speed-reduction systems, we will define $\eta > 1$; then the relationships between input and output speeds and torques are given by

$$\begin{aligned} \dot{\theta}_o &= (1/\eta)\dot{\theta}_i \\ \tau_o &= \eta\tau_i. \end{aligned} \tag{8.11}$$

where $\dot{\theta}_o$ and $\dot{\theta}_i$ are output and input speeds, respectively, and τ_o and τ_i are output and input torques, respectively.

The second broad class of reduction elements includes flexible bands, cables, and belts. Because all of these elements must be flexible enough to bend around pulleys, they also tend to be flexible in the longitudinal direction. The flexibility of these elements is proportional to their length. Because these systems are flexible, there must be some mechanism for preloading the loop to ensure that the belt or cable stays engaged on the pulley. Large preloads can add undue strain to the flexible element and introduce excessive friction.

Cables or flexible bands can be used either in a closed loop or as single-ended elements that are always kept in tension by some sort of preload. In a joint that is spring loaded in one direction, a single-ended cable could be used to pull against it. Alternately, two active single-ended systems can oppose each other. This arrangement eliminates the problem of excessive preloads but adds more actuators.

Roller chains are similar to flexible bands but can bend around relatively small pulleys while retaining a high stiffness. As a result of wear and high loads on the pins connecting the links, toothed-belt systems are more compact than roller chains for certain applications.

Band, cable, belt, and chain drives have the ability to combine transmission with reduction. As is shown in Fig. 8.15, when the input pulley has radius r_1 and the output pulley has radius r_2, the "gear" ratio of the transmission system is

$$\eta = \frac{r_2}{r_1}. \tag{8.12}$$

Lead screws or ball-bearing screws provide another popular method of getting a large reduction in a compact package (Fig. 8.16). Lead screws are very stiff and can support very large loads, and have the property that they transform rotary motion into linear motion. Ball-bearing screws are similar to lead screws, but instead of having the nut threads riding directly on the screw threads, a recirculating circuit of ball bearings rolls between the sets of threads. Ball-bearings screws have very low friction and are usually backdrivable.

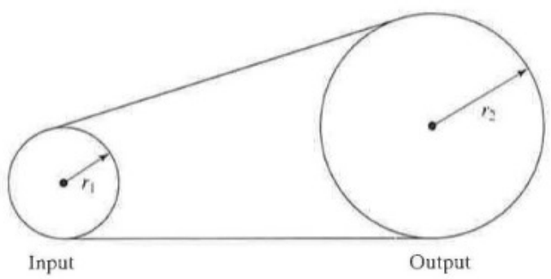


FIGURE 8.15: Band, cable, belt, and chain drives have the ability to combine transmission with reduction.

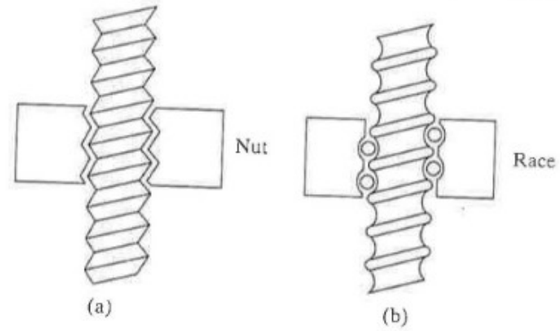


FIGURE 8.16: Lead screws (a) and ball-bearing screws (b) combine a large reduction and transformation from rotary to linear motion.

8.7 STIFFNESS AND DEFLECTIONS

An important goal for the design of most manipulators is overall stiffness of the structure and the drive system. Stiff systems provide two main benefits. First, because typical manipulators do not have sensors to measure the tool frame location directly, it is calculated by using the forward kinematics based on sensed joint positions. For an accurate calculation, the links cannot sag under gravity or other loads. In other words, we wish our Denavit–Hartenberg description of the linkages to remain fixed under various loading conditions. Second, flexibilities in the structure or drive train will lead to **resonances**, which have an undesirable effect on manipulator performance. In this section, we consider issues of stiffness and the resulting deflections under loads. We postpone further discussion of resonances until Chapter 9.

Flexible elements in parallel and in series

As can be easily shown (see Exercise 8.21), the combination of two flexible members of stiffness k_1 and k_2 "connected in parallel" produces the net stiffness

$$k_{\text{parallel}} = k_1 + k_2; \tag{8.13}$$

"connected in series," the combination produces the net stiffness

$$\frac{1}{k_{\text{series}}} = \frac{1}{k_1} + \frac{1}{k_2}. \tag{8.14}$$

In considering transmission systems, we often have the case of one stage of reduction or transmission in series with a following stage of reduction or transmission; hence, (8.14) becomes useful.

Shafts

A common method for transmitting rotary motion is through shafts. The torsional stiffness of a round shaft can be calculated [19] as

$$k = \frac{G\pi d^4}{32l}, \tag{8.15}$$

where d is the shaft diameter, l is the shaft length, and G is the shear modulus of elasticity (about 7.5×10^{10} Nt/m^2 for steel, and about a third as much for aluminum).

Gears

Gears, although typically quite stiff, introduce compliance into the drive system. An approximate formula to estimate the stiffness of the output gear (assuming the input gear is fixed) is given in [20] as

$$k = C_g b r^2, \tag{8.16}$$

where b is the face width of the gears, r is the radius of the output gear, and $C_g = 1.34 \times 10^{10}$ Nt/m^2 for steel.

Gearing also has the effect of changing the effective stiffness of the drive system by a factor of η^2. If the stiffness of the transmission system prior to the reduction (i.e., on the input side) is k_i, so that

$$\tau_i = k_i \delta\theta_i, \tag{8.17}$$

and the stiffness of the output side of the reduction is k_o, so that

$$\tau_o = k_o \delta\theta_o, \tag{8.18}$$

then we can compute the relationship between k_i and k_o (under the assumption of a perfectly rigid gear pair) as

$$k_o = \frac{\tau_o}{\delta\theta_o} = \frac{\eta k_i \delta\theta_i}{(1/\eta)\delta\theta_i} = \eta^2 k_i. \tag{8.19}$$

Hence, a gear reduction has the effect of increasing the stiffness by the square of the gear ratio.

EXAMPLE 8.3

A shaft with torsional stiffness equal to 500.0 Nt-m/radian is connected to the input side of a gear set with $\eta = 10$, whose output gear (when the input gear is fixed) exhibits a stiffness of 5000.0 Nt m/radian. What is the output stiffness of the combined drive system?

Using (8.14) and (8.19), we have

$$\frac{1}{k_{\text{series}}} = \frac{1}{5000.0} + \frac{1}{10^2(500.0)}, \tag{8.20}$$

or

$$k_{\text{series}} = \frac{50000}{11} \cong 4545.4 \text{ Nt m/radian}. \tag{8.21}$$

When a relatively large speed reduction is the last element of a multielement transmission system, the stiffnesses of the preceding elements can generally be ignored.

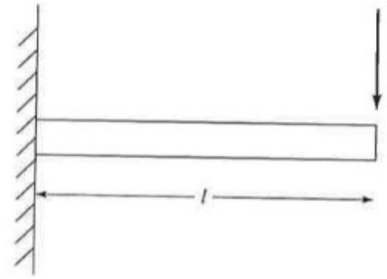


FIGURE 8.17: Simple cantilever beam used to model the stiffness of a link to an end load.

Belts

In such a belt drive as that of Fig. 8.15, stiffness is given by

$$k = \frac{AE}{l}, \tag{8.22}$$

where A is the cross-sectional area of the belt, E is the modulus of elasticity of the belt, and l is the length of the free belt between pulleys plus one-third of the length of the belt in contact with the pulleys [19].

Links

As a rough approximation of the stiffness of a link, we might model a single link as a cantilever beam and calculate the stiffness at the end point, as in Fig. 8.17. For a round hollow beam, this stiffness is given by [19]

$$k = \frac{3\pi E(d_o^4 - d_i^4)}{64 l^3}, \tag{8.23}$$

where d_i and d_o are the inner and outer diameters of the tubular beam, l is the length, and E is the modulus of elasticity (about 2×10^{11} Nt/m^2 for steel, and about a third as much for aluminum). For a square-cross-section hollow beam, this stiffness is given by

$$k = \frac{E(w_o^4 - w_i^4)}{4 l^3}, \tag{8.24}$$

where w_i and w_o are the outer and inner widths of the beam (i.e., the wall thickness is $w_o - w_i$).

EXAMPLE 8.4

A square-cross-section link of dimensions $5 \times 5 \times 50$ cm with a wall thickness of 1 cm is driven by a set of rigid gears with $\eta = 10$, and the input of the gears is driven by a shaft having diameter 0.5 cm and length 30 cm. What deflection is caused by a force of 100 Nt at the end of the link?

Using (8.24), we calculate the stiffness of the link as

$$k_{\text{link}} = \frac{(2 \times 10^{11})(0.05^4 - 0.04^4)}{4(0.5)} \cong 3.69 \times 10^5. \tag{8.25}$$

Hence, for a load of 100 Nt, there is a deflection in the link itself of

$$\delta x = \frac{100}{k_{\text{link}}} \cong 2.7 \times 10^{-4} \text{ m}, \tag{8.26}$$

or 0.027 cm.

Additionally, 100 Nt at the end of a 50-cm link is placing a torque of 50 Nt-m on the output gear. The gears are rigid, but the flexibility of the input shaft is

$$k_{\text{shaft}} = \frac{(7.5 \times 10^{10})(3.14)(5 \times 10^{-3})^4}{(32)(0.3)} \cong 15.3 \text{ Nt m/radian}, \tag{8.27}$$

which, viewed from the output gear, is

$$k'_{\text{shaft}} = (15.3)(10^2) = 1530.0 \text{ Nt-m/radian}. \tag{8.28}$$

Loading with 50 Nt-m causes an angular deflection of

$$\delta\theta = \frac{50.0}{1530.0} \cong 0.0326 \text{ radian}. \tag{8.29}$$

so the total linear deflection at the tip of the link is

$$\delta x \cong 0.027 + (0.0326)(50) = 0.027 + 1.630 = 1.657 \text{ cm}. \tag{8.30}$$

In our solution, we have assumed that the shaft and link are made of steel. The stiffness of both members is linear in E, the modulus of elasticity, so, for aluminum elements, we can multiply our result by about 3.

In this section, we have examined some simple formulas for estimating the stiffness of gears, shafts, belts, and links. They are meant to give some guidance in sizing structural members and transmission elements. However, in practical applications, many sources of flexibility are very difficult to model. Often, the drive train introduces significantly more flexibility than the link of a manipulator. Furthermore, many sources of flexibility in the drive system have not been considered here (bearing flexibility, flexibility of the actuator mounting, etc.). Generally, any attempt to predict stiffness analytically results in an overly high prediction, because many sources are not considered.

Finite-element techniques can be used to predict the stiffness (and other properties) of more realistic structural elements more accurately. This is an entire field in itself [21] and is beyond the scope of this book.

Actuators

Among various actuators, **hydraulic cylinders** or **vane actuators** were originally the most popular for use in manipulators. In a relatively compact package, they

can produce enough force to drive joints without a reduction system. The speed of operation depends upon the pump and accumulator system, usually located remotely from the manipulator. The position control of hydraulic systems is well understood and relatively straightforward. All of the early industrial robots and many modern large industrial robots use hydraulic actuators.

Unfortunately, hydraulics require a great deal of equipment, such as pumps, accumulators, hoses, and servo valves. Hydraulic systems also tend to be inherently messy, making them unsuitable for some applications. With the advent of more advanced robot-control strategies, in which actuator forces must be applied accurately, hydraulics proved disadvantageous, because of the friction contributed by their seals.

Pneumatic cylinders possess all the favorable attributes of hydraulics, and they are cleaner than hydraulics—air seeps out instead of hydraulic fluid. However, pneumatic actuators have proven difficult to control accurately, because of the compressibility of air and the high friction of the seals.

Electric motors are the most popular actuator for manipulators. Although they don't have the power-to-weight ratio of hydraulics or pneumatics, their controllability and ease of interface makes them attractive for small-to-medium-sized manipulators.

Direct current (DC) brush motors (Fig. 8.18) are the most straightforward to interface and control. The current is conducted to the windings of the rotor via brushes, which make contact with the revolving commutator. Brush wear and friction can be problems. New magnetic materials have made high peak torques possible. The limiting factor on the torque output of these motors is the overheating

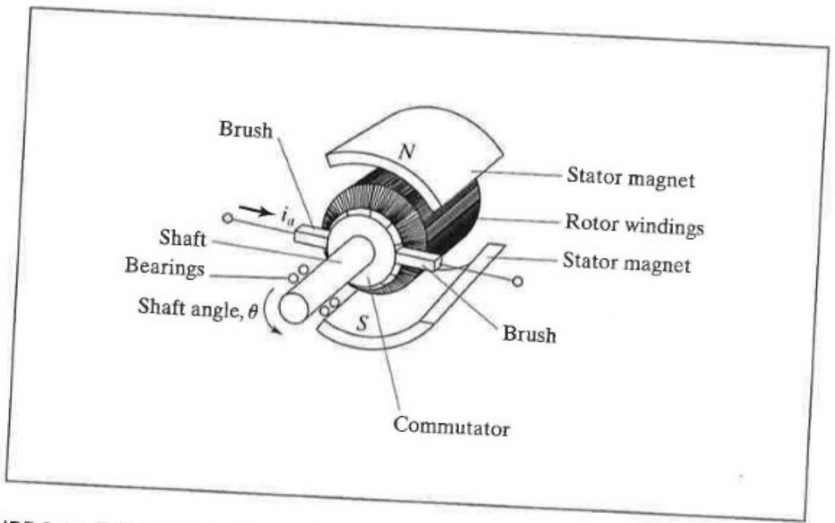


FIGURE 8.18: DC brush motors are among the actuators occurring most frequently in manipulator design. Franklin, Powell, Emami-Naeini, *Feedback Control of Dynamic Systems*, © 1988, Addison-Wesley, Reading, MA. Reprinted with permission.

of the windings. For short duty cycles, high torques can be achieved, but only much lower torques can be sustained over long periods of time.

Brushless motors solve brush wear and friction problems. Here, the windings remain stationary and the magnetic field piece rotates. A sensor on the rotor detects the shaft angle and is then used by external electronics to perform the commutation. Another advantage of brushless motors is that the winding is on the outside, attached to the motor case, affording it much better cooling. Sustained torque ratings tend to be somewhat higher than for similar-sized brush motors.

Alternating current (AC) motors and stepper motors have been used infrequently in industrial robotics. Difficulty of control of the former and low torque ability of the latter have limited their use.

8.8 POSITION SENSING

Virtually all manipulators are servo-controlled mechanisms—that is, the force or torque command to an actuator is based on the error between the sensed position of the joint and the desired position. This requires that each joint have some sort of position-sensing device.

The most common approach is to locate a position sensor directly on the shaft of the actuator. If the drive train is stiff and has no backlash, the true joint angles can be calculated from the actuator shaft positions. Such **co-located** sensor and actuator pairs are easiest to control.

The most popular position-feedback device is the **rotary optical encoder**. As the encoder shaft turns, a disk containing a pattern of fine lines interrupts a light beam. A photodetector turns these light pulses into a binary waveform. Typically, there are two such channels, with wave pulse trains 90 degrees out of phase. The shaft angle is determined by counting the number of pulses, and the direction of rotation is determined by the relative phase of the two square waves. Additionally, encoders generally emit an **index pulse** at one location, which can be used to set a home position in order to compute an absolute angular position.

Resolvers are devices that output two analog signals—one the sine of the shaft angle, the other the cosine. The shaft angle is computed from the relative magnitude of the two signals. The resolution is a function of the quality of the resolver and the amount of noise picked up in the electronics and cabling. Resolvers are often more reliable than optical encoders, but their resolution is lower. Typically, resolvers cannot be placed directly at the joint without additional gearing to improve the resolution.

Potentiometers provide the most straightforward form of position sensing. Connected in a bridge configuration, they produce a voltage proportional to the shaft position. Difficulties with resolution, linearity, and noise susceptibility limit their use.

Tachometers are sometimes used to provide an analog signal proportional to the shaft velocity. In the absence of such velocity sensors, the velocity feedback is derived by taking differences of sensed position over time. This **numerical differentiation** can introduce both noise and a time lag. Despite these potential problems, most manipulators are without direct velocity sensing.

8.9 FORCE SENSING

A variety of devices have been designed to measure forces of contact between a manipulator's end-effector and the environment that it touches. Most such sensors make use of sensing elements called **strain gauges**, of either the semiconductor or the metal-foil variety. These strain gauges are bonded to a metal structure and produce an output proportional to the strain in the metal. In this type of force-sensor design, the issues the designer must address include the following:

1. How many sensors are needed to resolve the desired information?
2. How are the sensors mounted relative to each other on the structure?
3. What structure allows good sensitivity while maintaining stiffness?
4. How can protection against mechanical overload be built into the device?

There are three places where such sensors are usually placed on a manipulator:

1. At the joint actuators. These sensors measure the torque or force output of the actuator/reduction itself. These are useful for some control schemes, but usually do not provide good sensing of contact between the end-effector and the environment.
2. Between the end-effector and last joint of the manipulator. These sensors are usually referred to as **wrist sensors**. They are mechanical structures instrumented with strain gauges, which can measure the forces and torques acting on the end-effector. Typically, these sensors are capable of measuring from three to six components of the force/torque vector acting on the end-effector.
3. At the "fingertips" of the end-effector. Usually, these **force-sensing fingers** have built-in strain gauges to measure from one to four components of force acting at each fingertip.

As an example, Fig. 8.19 is a drawing of the internal structure of a popular style of wrist-force sensor designed by Scheinman [22]. Bonded to the cross-bar structure of the device are eight pairs of semiconductor strain gauges. Each pair is wired in a voltage-divider arrangement. Each time the wrist is queried, eight analog voltages are digitized and read into the computer. Calibration schemes have been designed with which to arrive at a constant 6×8 **calibration matrix** that maps these eight strain measurements into the force–torque vector, $\mathcal{F}$, acting on the end-effector. The sensed force–torque vector can be transformed to a reference frame of interest, as we saw in Example 5.8.

Force-sensor design issues

Use of strain gauges to measure force relies on measuring the deflection of a stressed **flexure**. Therefore, one of the primary design trade-offs is between the stiffness and the sensitivity of the sensor. A stiffer sensor is inherently less sensitive.

The stiffness of the sensor also affects the construction of **overload protection**. Strain gauges can be damaged by impact loading and therefore must be protected against such overloads. Transducer damage can be prevented by having **limit stops**,

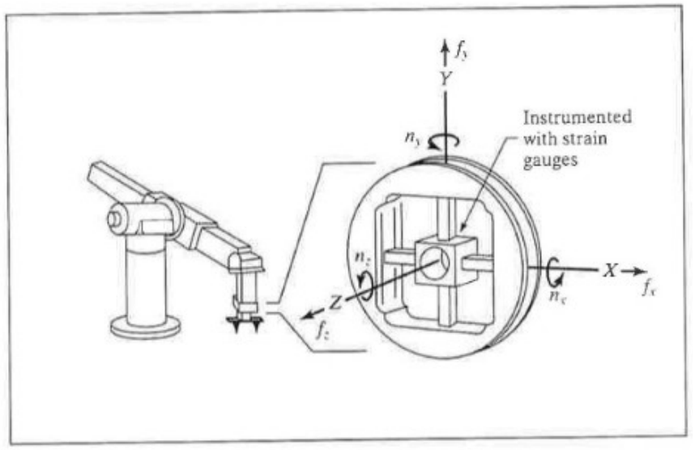


FIGURE 8.19: The internal structure of a typical force-sensing wrist.

which prevent the flexures from deflecting past a certain point. Unfortunately, a very stiff sensor might deflect only a few ten-thousandths of an inch. Manufacturing limit stops with such small clearances is very difficult. Consequently, for many types of transducers, a certain amount of flexibility *must* be built-in in order to make possible effective limit stops.

Eliminating **hysteresis** is one of the most cumbersome restrictions in the sensor design. Most metals used as flexures, if not overstrained, have very little hysteresis. However, bolted, press-fit, or welded joints near the flexure introduce hysteresis. Ideally, the flexure and the material near it are made from a single piece of metal.

It is also important to use differential measurements to increase the linearity and disturbance rejection of torque sensors. Different physical configurations of transducers can eliminate influences due to temperature effects and off-axis forces.

Foil gauges are relatively durable, but they produce a very small resistance change at full strain. Eliminating noise in the strain-gauge cabling and amplification electronics is of crucial importance for a good dynamic range.

Semiconductor strain gauges are much more susceptible to damage through overload. In their favor, they produce a resistance change about seventy times that of foil gauges for a given strain. This makes the task of signal processing much simpler for a given dynamic range.

BIBLIOGRAPHY

[1] W. Rowe, Editor, *Robotics Technical Directory 1986*, Instrument Society of America, Research Triangle Park, NC, 1986.

[2] R. Vijaykumar and K. Waldron, "Geometric Optimization of Manipulator Structures for Working Volume and Dexterity," *International Journal of Robotics Research*, Vol. 5, No. 2, 1986.

[3] K. Waldron, "Design of Arms," in *The International Encyclopedia of Robotics*, R. Dorf and S. Nof, Editors, John Wiley and Sons, New York, 1988.

[4] T. Yoshikawa, "Manipulability of Robotic Mechanisms," *The International Journal of Robotics Research*, Vol. 4, No. 2, MIT Press, Cambridge, MA, 1985.

[5] H. Asada, "Dynamic Analysis and Design of Robot Manipulators Using Inertia Ellipsoids," *Proceedings of the IEEE International Conference on Robotics*, Atlanta, March 1984.

[6] J.K. Salisbury and J. Craig, "Articulated Hands: Force Control and Kinematic Issues," *The International Journal of Robotics Research*, Vol. 1, No. 1, 1982.

[7] O. Khatib and J. Burdick, "Optimization of Dynamics in Manipulator Design: The Operational Space Formulation," *International Journal of Robotics and Automation*, Vol. 2, No. 2, IASTED, 1987.

[8] T. Yoshikawa, "Dynamic Manipulability of Robot Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Louis, March 1985.

[9] J. Trevelyan, P. Kovesi, and M. Ong, "Motion Control for a Sheep Shearing Robot," *The 1st International Symposium of Robotics Research*, MIT Press, Cambridge, MA, 1984.

[10] P. Marchal, J. Cornu, and J. Detriche, "Self Adaptive Arc Welding Operation by Means of an Automatic Joint Following System," *Proceedings of the 4th Symposium on Theory and Practice of Robots and Manipulators*, Zaburow, Poland, September 1981.

[11] J.M. Hollerbach, "Optimum Kinematic Design for a Seven Degree of Freedom Manipulator," *Proceedings of the 2nd International Symposium of Robotics Research*, Kyoto, Japan, August 1984.

[12] K. Waldron and J. Reidy, "A Study of Kinematically Redundant Manipulator Structure," *Proceedings of the IEEE Robotics and Automation Conference*, San Francisco, April 1986.

[13] V. Milenkovic, "New Nonsingular Robot Wrist Design," *Proceedings of the Robots 11 / 17th ISIR Conference*, SME, 1987.

[14] E. Rivin, *Mechanical Design of Robots*, McGraw-Hill, New York, 1988.

[15] T. Yoshikawa, "Manipulability of Robotic Mechanisms," in *Proceedings of the 2nd International Symposium on Robotics Research*, Kyoto, Japan, 1984.

[16] M. Leu, V. Dukowski, and K. Wang, "An Analytical and Experimental Study of the Stiffness of Robot Manipulators with Parallel Mechanisms," in *Robotics and Manufacturing Automation*, M. Donath and M. Leu, Editors, ASME, New York, 1985.

[17] K. Hunt, *Kinematic Geometry of Mechanisms*, Cambridge University Press, Cambridge, MA, 1978.

[18] H. Asada and K. Youcef-Toumi, *Design of Direct Drive Manipulators*, MIT Press, Cambridge, MA, 1987.

[19] J. Shigley, *Mechanical Engineering Design*, 3rd edition, McGraw-Hill, New York, 1977.

[20] D. Welbourne, "Fundamental Knowledge of Gear Noise—A Survey," *Proceedings of the Conference on Noise and Vibrations of Engines and Transmissions*, Institute of Mechanical Engineers, Cranfield, UK, 1979.

[21] O. Zienkiewicz, *The Finite Element Method*, 3rd edition, McGraw-Hill, New York, 1977.

[22] V. Scheinman, "Design of a Computer Controlled Manipulator," M.S. Thesis, Mechanical Engineering Department, Stanford University, 1969.

[23] K. Lau. N. Dagalakis, and D. Meyers, "Testing," in *The International Encyclopedia of Robotics*, R. Dorf and S. Nof, Editors, John Wiley and Sons, New York, 1988.

[24] M. Roshiem, "Wrists," in *The International Encyclopedia of Robotics*, R. Dorf and S. Nof, Editors, John Wiley and Sons, New York, 1988.

[25] A. Bowling and O. Khatib, "Robot Acceleration Capability: The Actuation Efficiency Measure," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, April 2000.

EXERCISES

8.1 [15] A robot is to be used for positioning a laser cutting device. The laser produces a pinpoint, nondivergent beam. For general cutting tasks, how many degrees of freedom does the positioning robot need? Justify your answer.

8.2 [15] Sketch a possible joint configuration for the laser-positioning robot of Exercise 8.1, assuming that it will be used primarily for cutting at odd angles through 1-inch-thick, 8 × 8-foot plates.

8.3 [17] For a spherical robot like that of Fig. 8.6, if joints 1 and 2 have no limits and joint 3 has lower limit l and upper limit u, find the structural length index, Q_L, for the wrist point of this robot.

8.4 [25] A steel shaft of length 30 cm and diameter 0.2 cm drives the input gear of a reduction having $\eta = 8$. The output gear drives a steel shaft having length 30 cm and diameter 0.3 cm. If the gears introduce no compliance of their own, what is the overall stiffness of the transmission system?

8.5 [20] In Fig. 8.20, a link is driven through a shaft after a gear reduction. Model the link as rigid with mass of 10 Kg located at a point 30 cm from the shaft axis. Assume that the gears are rigid and that the reduction, η, is large. The shaft is steel and must be 30 cm long. If the design specifications call for the center of link mass to undergo accelerations of 2.0 g, what should the shaft diameter be to limit dynamic deflections to 0.1 radian at the joint angle?

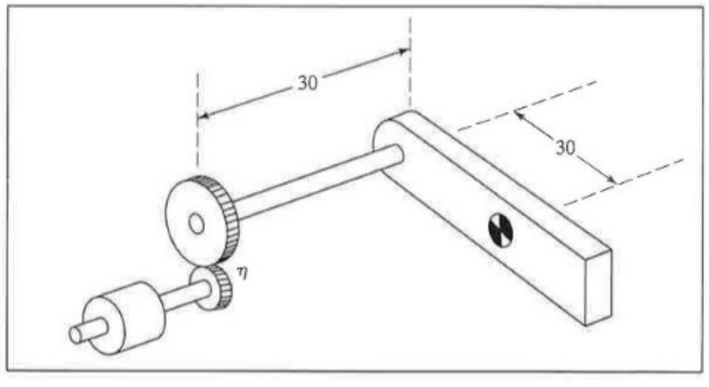


FIGURE 8.20: A link actuated through a shaft after a gear reduction.

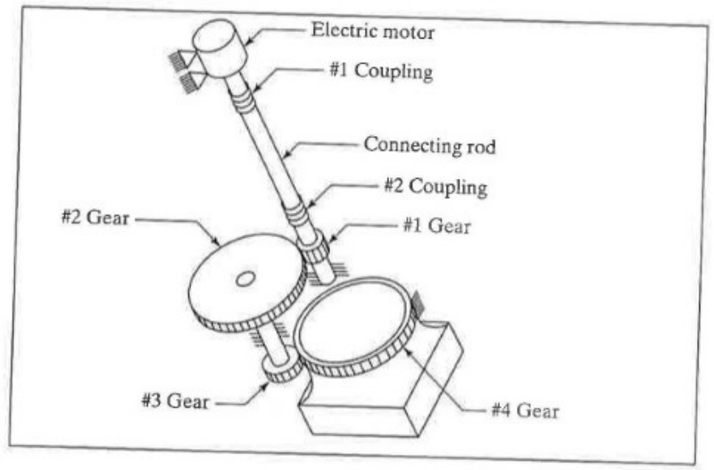


FIGURE 8.21: Simplified version of the drive train of joint 4 of the PUMA 560 manipulator (from [23]). From *International Encyclopedia of Robotics*, by R. Dorf and S. Nof, editors. From "Testing," by K. Law, N. Dagalakis, and D. Myers.

8.6 [15] If the output gear exhibits a stiffness of 1000 Nt-m/radian with input gear locked and the shaft has stiffness of 300 Nt-m/radian, what is the combined stiffness of the drive system in Fig. 8.20?

8.7 [43] Pieper's criteria for serial-link manipulators state that the manipulator will be solvable if three consecutive axes intersect at a single point or are parallel. This is based on the idea that inverse kinematics can be decoupled by looking at the position of the wrist point independently from the orientation of the wrist frame. Propose a similar result for the Stewart mechanism in Fig. 8.14, to allow the forward kinematic solution to be similarly decoupled.

8.8 [20] In the Stewart mechanism of Fig. 8.14, if the 2-DOF universal joints at the base were replaced with 3-DOF ball-and-socket joints, what would the total number of degrees of freedom of the mechanism be? Use Grübler's formula.

8.9 [22] Figure 8.21 shows a simplified schematic of the drive system of joint 4 of the PUMA 560 [23]. The torsional stiffness of the couplings is 100 Nt-m/radian each, that of the shaft is 400 Nt-m/radian, and each of the reduction pairs has been measured to have output stiffness of 2000 Nt-m/radian with its input gears fixed. Both the first and second reductions have $\eta = 6$.[2] Assuming the structure and bearing are perfectly rigid, what is the stiffness of the joint (i.e., when the motor's shaft is locked)?

8.10 [25] What is the error if one approximates the answer to Exercise 8.9 by considering just the stiffness of the final speed-reduction gearing?

8.11 [20] Figure 4.14 shows an orthogonal-axis wrist and a nonorthogonal wrist. The orthogonal-axis wrist has link twists of magnitude 90°; the nonorthogonal wrist has link twists of ϕ and $180° - \phi$ in magnitude. Describe the set of orientations that are *unattainable* with the nonorthogonal mechanism. Assume that all axes can turn 360° and that links can pass through one another if need be (i.e., workspace is not limited by self-collision).

[2] None of the numerical values in this exercise is meant to be realistic!

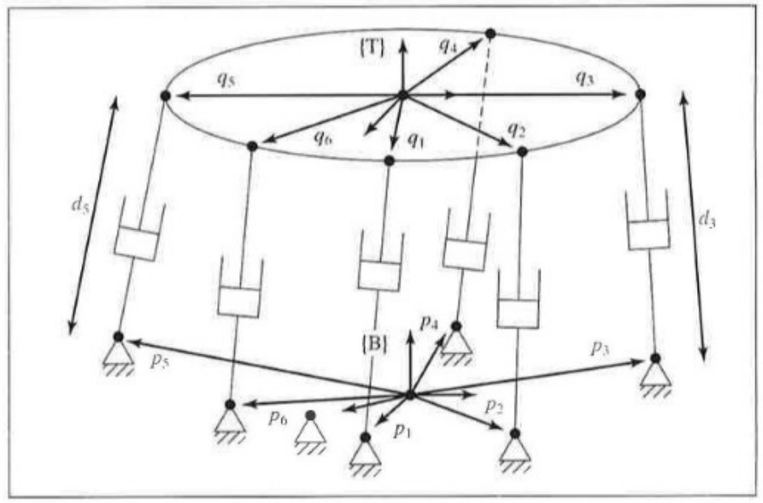


FIGURE 8.22: Stewart mechanism of Exercise 8.12.

8.12 [18] Write down a general inverse-kinematic solution for the Stewart mechanism shown in Fig. 8.22. Given the location of $\{T\}$ relative to the base frame $\{B\}$, solve for the joint-position variables d_1 through d_6. The ${}^B p_i$ are 3×1 vectors which locate the base connections of the linear actuators relative to frame $\{B\}$. The ${}^T q_i$ are 3×1 vectors which locate the upper connections of the linear actuators relative to the frame $\{T\}$.

8.13 [20] The planar two-link of example 5.3 has the determinant of its Jacobian given by

$$\det(J(\Theta)) = l_1 l_2 s_2. \tag{8.31}$$

If the sum of the two link lengths, $l_1 + l_2$, is constrained to be equal to a constant, what should the relative lengths be in order to maximize the manipulator's manipulability as defined by (8.6)?

8.14 [28] For a SCARA robot, given that the sum of the link lengths of link 1 and link 2 must be constant, what is the optimal choice of relative length in terms of the manipulability index given in (8.6)? Solving Exercise 8.13 first could be helpful.

8.15 [35] Show that the manipulability measure defined in (8.6) is also equal to the product of the eigenvalues of $J(\Theta)$.

8.16 [15] What is the torsional stiffness of a 40-cm aluminum rod with radius 0.1 cm?

8.17 [5] What is the effective "gear" reduction, η, of a belt system having an input pulley of radius 2.0 cm and an output pulley of radius 12.0 cm?

8.18 [10] How many degrees of freedom are required in a manipulator used to place cylindrical-shaped parts on a flat plane? The cylindrical parts are perfectly symmetrical about their main axes.

8.19 [25] Figure 8.23 shows a three-fingered hand grasping an object. Each finger has three single-degree-of-freedom joints. The contact points between fingertips and the object are modeled as "point contact"—that is, the position is fixed, but the relative orientation is free in all three degrees of freedom. Hence, these point contacts can be replaced by 3-DOF ball-and-socket joints for the purposes of analysis. Apply Grübler's formula to compute how many degrees of freedom the overall system possesses.

8.20 [23] Figure 8.24 shows an object connected to the ground with three rods. Each rod is connected to the object with a 2-DOF universal joint and to the ground with a 3-DOF ball-and-socket joint. How many degrees of freedom does the system possess?

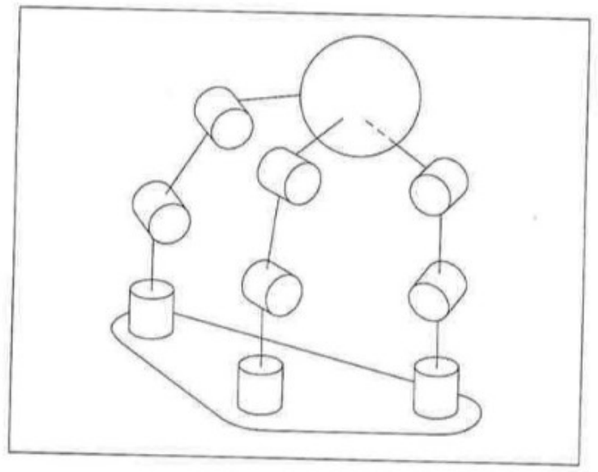


FIGURE 8.23: A three-fingered hand in which each finger has three degrees of freedom grasps an object with "point contact."

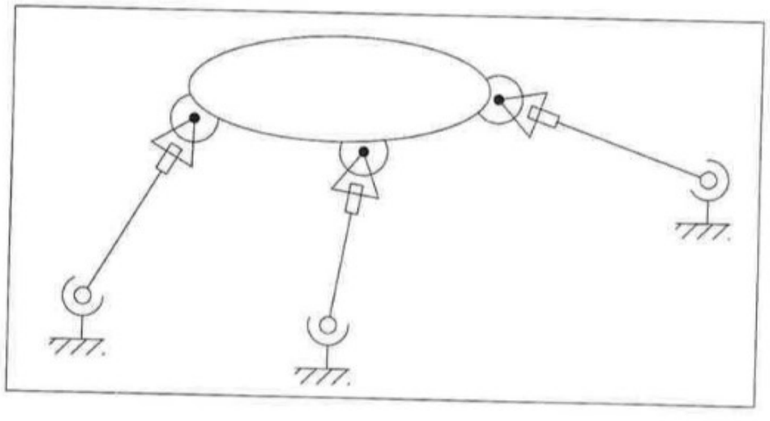


FIGURE 8.24: Closed loop mechanism of Exercise 8.20.

8.21 [18] Verify that, if two transmission systems are connected serially, then the equivalent stiffness of the overall system is given by (8.14). It is perhaps simplest to think of the serial connection of two linear springs having stiffness coefficients

k_1 and k_2 and of the resulting equations:

$$f = k_1 \delta x_1,$$
$$f = k_2 \delta x_2, \qquad (8.32)$$
$$f = k_{\text{sum}}(\delta x_1 + \delta x_2).$$

8.22 [20] Derive a formula for the stiffness of a belt-drive system in terms of the pulley radii (r_1 and r_2) and the center-to-center distance between the pulleys, d_c. Start from (8.22).

PROGRAMMING EXERCISE (PART 8)

1. Write a program to compute the determinant of a 3 × 3 matrix.
2. Write a program to move the simulated three-link robot in 20 steps in a straight line and constant orientation from

$$ {}^0_3T = \begin{bmatrix} 0.25 \\ 0.0 \\ 0.0 \end{bmatrix} $$

to

$$ {}^0_3T = \begin{bmatrix} 0.95 \\ 0.0 \\ 0.0 \end{bmatrix} $$

in increments of 0.05 meter. At each location, compute the manipulability measure for the robot at that configuration (i.e., the determinant of the Jacobian). List, or, better yet, make a plot of the values as a function of the position along the $\hat{X}_0$ axis. Generate the preceding data for two cases:

(a) $l_1 = l_2 = 0.5$ meter, and
(b) $l_1 = 0.625$ meter, $l_2 = 0.375$ meter.

Which manipulator design do you think is better? Explain your answer.

MATLAB EXERCISE 8

Section 8.5 introduced the concept of kinematically redundant robots. This exercise deals with the resolved-rate control simulation for a kinematically redundant robot. We will focus on the planar 4-DOF 4R robot with one degree of kinematic redundancy (four joints to provide three Cartesian motions: two translations and one rotation). This robot is obtained by adding a fourth R-joint and a fourth moving link L_4 to the planar 3-DOF, 3R robot (of Figures 3.6 and 3.7; the DH parameters can be extended by adding one row to Figure 3.8).

For the planar 4R robot, derive analytical expressions for the 3 × 4 Jacobian matrix; then, perform resolved-rate control simulation in MATLAB (as in MATLAB Exercise 5). The form of the velocity equation is again ${}^k\dot{X} = {}^kJ\dot{\Theta}$; however, this equation cannot be inverted by means of the normal matrix inverse, because the Jacobian matrix is nonsquare (three equations, four unknowns, infinite solutions to $\dot{\Theta}$). Therefore, let us use the Moore–Penrose pseudoinverse J^* of the Jacobian matrix: $J^* = J^T(JJ^T)^{-1}$. For the resulting commanded relative joint rates for the resolved-rate algorithm, $\dot{\Theta} = {}^kJ^{*k}\dot{X}$,

choose the minimum-norm solution from the infinite possibilities (i.e., this specific $\dot{\Theta}$ is as small as possible to satisfy the commanded Cartesian velocities ${}^k\dot{X}$).

This solution represents the particular solution only—that is, there exists a homogeneous solution to optimize performance (such as avoiding manipulator singularities or avoiding joint limits) in addition to satisfying the commanded Cartesian motion. Performance optimization is beyond the scope of this exercise.

Given: $L_1 = 1.0\,m$, $L_2 = 1.0$ m, $L_3 = 0.2\,m$, $L_4 = 0.2\,m$.

The initial angles are:

$$\Theta = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{Bmatrix} = \begin{Bmatrix} -30^\circ \\ 70^\circ \\ 30^\circ \\ 40^\circ \end{Bmatrix}.$$

The (constant) commanded Cartesian velocity is

$$ {}^0\dot{X} = {}^0 \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \omega_z \end{Bmatrix} = {}^0 \begin{Bmatrix} -0.2 \\ -0.2 \\ 0.2 \end{Bmatrix} \text{ (m/s, rad/s).} $$

Simulate resolved-rate motion, for the particular solution only, for 3 sec, with a control time step of 0.1 sec. Also, in the same loop, animate the robot to the screen during each time step, so that you can watch the simulated motion to verify that it is correct.

a) Present four plots (each set on a separate graph, please):

1. the four joint angles (degrees) $\Theta = \{\theta_1 \ \theta_2 \ \theta_3 \ \theta_4\}^T$ vs. time;
2. the four joint rates (rad/s) $\dot{\Theta} = \{\dot{\theta}_1 \ \dot{\theta}_2 \ \dot{\theta}_3 \ \dot{\theta}_4\}^T$ vs. time;
3. the joint-rate Euclidean norm $\|\dot{\Theta}\|$ (vector magnitude) vs. time;
4. the three Cartesian components of ${}^0_HT$, $X = \{x \ y \ \phi\}^T$ (rad is fine for ϕ so that it will fit) vs. time.

Carefully label (by hand is fine!) each component on each plot. Also, label the axis names and units.

b) Check your Jacobian matrix results for the initial and final joint-angle sets by means of the Corke MATLAB Robotics Toolbox. Try function *jacob0()*. **Caution**: The toolbox Jacobian functions are for motion of {4} with respect to {0}, not for {*H*} with respect to {0} as in the problem assignment. The preceding function gives the Jacobian result in {0} coordinates; *jacobn()* would give results in {4} coordinates.

CHAPTER 9

Linear control of manipulators

9.1 INTRODUCTION

Armed with the previous material, we now have the means to calculate joint-position time histories that correspond to desired end-effector motions through space. In this chapter, we begin to discuss how to cause the manipulator actually to perform these desired motions.

The control methods that we will discuss fall into the class called **linear-control** systems. Strictly speaking, the use of linear-control techniques is valid only when the system being studied can be modeled mathematically by *linear* differential equations. For the case of manipulator control, such linear methods must essentially be viewed as approximate methods, for, as we have seen in Chapter 6, the dynamics of a manipulator are more properly represented by a *nonlinear* differential equation. Nonetheless, we will see that it is often reasonable to make such approximations, and it also is the case that these linear methods are the ones most often used in current industrial practice.

Finally, consideration of the linear approach will serve as a basis for the more complex treatment of nonlinear control systems in Chapter 10. Although we approach linear control as an approximate method for manipulator control, the justification for using linear controllers is not only empirical. In Chapter 10, we will prove that a certain linear controller leads to a reasonable control system even without resorting to a linear approximation of manipulator dynamics. Readers familiar with linear-control systems might wish to skip the first four sections of the current chapter.

9.2 FEEDBACK AND CLOSED-LOOP CONTROL

We will model a manipulator as a mechanism that is instrumented with sensors at each joint to measure the joint angle and that has an actuator at each joint to apply a torque on the neighboring (next higher) link.[1]. Although other physical arrangements of sensors are sometimes used, the vast majority of robots have a position sensor at each joint. Sometimes velocity sensors (tachometers) are also present at the joints. Various actuation and transmission schemes are prevalent in industrial robots, but many of these can be modeled by supposing that there is a single actuator at each joint.

We wish to cause the manipulator joints to follow prescribed position trajectories, but the actuators are commanded in terms of torque, so we must use some kind of **control system** to compute appropriate actuator commands that will realize this desired motion. Almost always, these torques are determined by using **feedback** from the joint sensors to compute the torque required.

Figure 9.1 shows the relationship between the trajectory generator and the physical robot. The robot accepts a vector of joint torques, τ, from the control system. The manipulator's sensors allow the controller to read the vectors of joint positions, Θ, and joint velocities, $\dot{\Theta}$. All signal lines in Fig. 9.1 carry $N \times 1$ vectors (where N is the number of joints in the manipulator).

Let's consider what algorithm might be implemented in the block labeled "control system" in Fig. 9.1. One possibility is to use the dynamic equation of the robot (as studied in Chapter 6) to calculate the torques required for a particular trajectory. We are given Θ_d, $\dot{\Theta}_d$, and $\ddot{\Theta}_d$ by the trajectory generator, so we could use (6.59) to compute

$$\tau = M(\Theta_d)\ddot{\Theta}_d + V(\Theta_d, \dot{\Theta}_d) + G(\Theta_d). \tag{9.1}$$

This computes the torques that our model dictates would be required to realize the desired trajectory. If our dynamic model were complete and accurate and no "noise" or other disturbances were present, continuous use of (9.1) along the desired trajectory would realize the desired trajectory. Unfortunately, imperfection in the dynamic model and the inevitable presence of disturbances make such a scheme impractical for use in real applications. Such a control technique is termed an **open-loop** scheme, because there is no use made of the feedback from the joint sensors

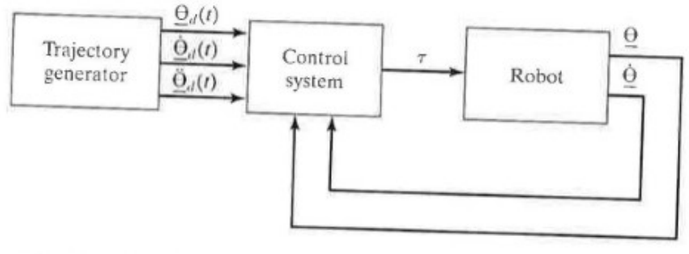


FIGURE 9.1: High-level block diagram of a robot-control system.

[1] Remember, all remarks made concerning rotational joints hold analogously for linear joints, and vice versa

(i.e., (9.1) is a function only of the desired trajectory, Θ_d, and its derivatives, and *not* a function of Θ, the actual trajectory).

Generally, the only way to build a high-performance control system is to make use of feedback from joint sensors, as indicated in Fig. 9.1. Typically, this feedback is used to compute any **servo error** by finding the difference between the desired and the actual position and that between the desired and the actual velocity:

$$E = \Theta_d - \Theta,$$
$$\dot{E} = \dot{\Theta}_d - \dot{\Theta}. \tag{9.2}$$

The control system can then compute how much torque to require of the actuators as some function of the servo error. Obviously, the basic idea is to compute actuator torques that would tend to reduce servo errors. A control system that makes use of feedback is called a **closed-loop** system. The "loop" closed by such a control system around the manipulator is apparent in Fig. 9.1.

The central problem in designing a control system is to ensure that the resulting closed-loop system meets certain performance specifications. The most basic such criterion is that the system remain **stable**. For our purposes, we will define a system to be stable if the errors remain "small" when executing various desired trajectories even in the presence of some "moderate" disturbances. It should be noted that an improperly designed control system can sometimes result in **unstable** performance, in which servo errors are enlarged instead of reduced. Hence, the first task of a control engineer is to prove that his or her design yields a stable system; the second is to prove that the closed-loop performance of the system is satisfactory. In practice, such "proofs" range from mathematical proofs based on certain assumptions and models to more empirical results, such as those obtained through simulation or experimentation.

Figure 9.1, in which all signals lines represent $N \times 1$ vectors, summarizes the fact that the manipulator-control problem is a **multi-input, multi-output (MIMO)** control problem. In this chapter, we take a simple approach to constructing a control system by treating each joint as a separate system to be controlled. Hence, for an N-jointed manipulator, we will design N independent **single-input, single-output (SISO)** control systems. This is the design approach presently adopted by most industrial-robot suppliers. This **independent joint control** approach is an approximate method in that the equations of motion (developed in Chapter 6) are not independent, but rather are highly coupled. Later, this chapter will present justification for the linear approach, at least for the case of highly geared manipulators.

9.3 SECOND-ORDER LINEAR SYSTEMS

Before considering the manipulator control problem, let's step back and start by considering a simple mechanical system. Figure 9.2 shows a block of mass m attached to a spring of stiffness k and subject to friction of coefficient b. Figure 9.2 also indicates the zero position and positive sense of x, the block's position. Assuming a frictional force proportional to the block's velocity, a free-body diagram of the forces acting on the block leads directly to the equation of motion,

$$m\ddot{x} + b\dot{x} + kx = 0. \tag{9.3}$$

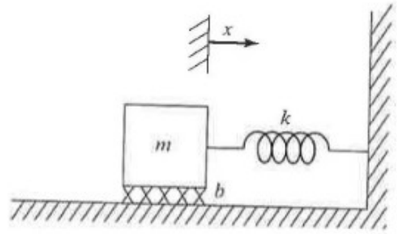


FIGURE 9.2: Spring–mass system with friction.

Hence, the open-loop dynamics of this one-degree-of-freedom system are described by a second-order linear constant-coefficient differential equation [1]. The solution to the differential equation (9.3) is a time function, $x(t)$, that specifies the motion of the block. This solution will depend on the block's **initial conditions**—that is, its initial position and velocity.

We will use this simple mechanical system as an example with which to review some basic control system concepts. Unfortunately, it is impossible to do justice to the field of control theory with only a brief introduction here. We will discuss the control problem, assuming no more than that the student is familiar with simple differential equations. Hence, we will not use many of the popular tools of the control-engineering trade. For example, **Laplace transforms** and other common techniques neither are a prerequisite nor are introduced here. A good reference for the field is [4].

Intuition suggests that the system of Fig. 9.2 might exhibit several different characteristic motions. For example, in the case of a very weak spring (i.e., k small) and very heavy friction (i.e., b large) one imagines that, if the block were perturbed, it would return to its resting position in a very slow, sluggish manner. However, with a very stiff spring and very low friction, the block might oscillate several times before coming to rest. These different possibilities arise because the character of the solution to (9.3) depends upon the values of the parameters m, b, and k.

From the study of differential equations [1], we know that the form of the solution to an equation of the form of (9.3) depends on the roots of its **characteristic equation**,

$$ms^2 + bs + k = 0. \tag{9.4}$$

This equation has the roots

$$s_1 = -\frac{b}{2m} + \frac{\sqrt{b^2 - 4mk}}{2m},$$
$$s_2 = -\frac{b}{2m} - \frac{\sqrt{b^2 - 4mk}}{2m}. \tag{9.5}$$

The location of s_1 and s_2 (sometimes called the **poles** of the system) in the real–imaginary plane dictate the nature of the motions of the system. If s_1 and s_2 are real, then the behavior of the system is sluggish and nonoscillatory. If s_1 and s_2 are complex (i.e., have an imaginary component) then the behavior of the system is

oscillatory. If we include the special limiting case between these two behaviors, we have three classes of response to study:

1. **Real and Unequal Roots**. This is the case when $b^2 > 4mk$; that is, friction dominates, and sluggish behavior results. This response is called **overdamped**.
2. **Complex Roots**. This is the case when $b^2 < 4mk$; that is, stiffness dominates, and oscillatory behavior results. This response is called **underdamped**.
3. **Real and Equal Roots**. This is the special case when $b^2 = 4mk$; that is, friction and stiffness are "balanced," yielding the fastest possible nonoscillatory response. This response is called **critically damped**.

The third case (critical damping) is generally a desirable situation: the system nulls out nonzero initial conditions and returns to its nominal position as rapidly as possible, yet without oscillatory behavior.

Real and unequal roots

It can easily be shown (by direct substitution into (9.3)) that the solution, $x(t)$, giving the motion of the block in the case of real, unequal roots has the form

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}, \tag{9.6}$$

where s_1 and s_2 are given by (9.5). The coefficients c_1 and c_2 are constants that can be computed for any given set of initial conditions (i.e., initial position and velocity of the block).

Figure 9.3 shows an example of pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are real and unequal, the system exhibits sluggish or overdamped motion.

In cases where one of the poles has a much greater magnitude than the other, the pole of larger magnitude can be neglected, because the term corresponding to it will decay to zero rapidly in comparison to the other, **dominant pole**. This same notion of dominance extends to higher order systems—for example, often a

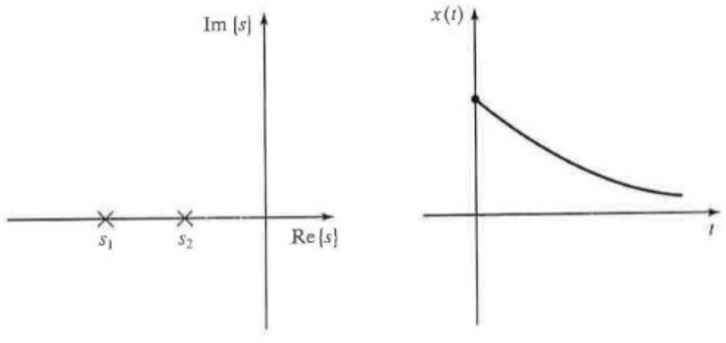


FIGURE 9.3: Root location and response to initial conditions for an overdamped system.

third-order system can be studied as a second-order system by considering only two dominant poles.

EXAMPLE 9.1

Determine the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 5$, and $k = 6$ and the block (initially at rest) is released from the position $x = -1$.

The characteristic equation is

$$s^2 + 5s + 6 = 0, \tag{9.7}$$

which has the roots $s_1 = -2$ and $s_2 = -3$. Hence, the response has the form

$$x(t) = c_1 e^{-2t} + c_2 e^{-3t}. \tag{9.8}$$

We now use the given initial conditions, $x(0) = -1$ and $\dot{x}(0) = 0$, to compute c_1 and c_2. To satisfy these conditions at $t = 0$, we must have

$$c_1 + c_2 = -1$$

and

$$-2c_1 - 3c_2 = 0, \tag{9.9}$$

which are satisfied by $c_1 = -3$ and $c_2 = 2$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = -3e^{-2t} + 2e^{-3t}. \tag{9.10}$$

Complex roots

For the case where the characteristic equation has complex roots of the form

$$s_1 = \lambda + \mu i,$$
$$s_2 = \lambda - \mu i, \tag{9.11}$$

it is still the case that the solution has the form

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}. \tag{9.12}$$

However, equation (9.12) is difficult to use directly, because it involves imaginary numbers explicitly. It can be shown (see Exercise 9.1) that **Euler's formula**,

$$e^{ix} = \cos x + i \sin x, \tag{9.13}$$

allows the solution (9.12) to be manipulated into the form

$$x(t) = c_1 e^{\lambda t} \cos(\mu t) + c_2 e^{\lambda t} \sin(\mu t). \tag{9.14}$$

As before, the coefficients c_1 and c_2 are constants that can be computed for any given set of initial conditions (i.e., initial position and velocity of the block). If we write the constants c_1 and c_2 in the form

$$c_1 = r \cos \delta,$$
$$c_2 = r \sin \delta, \tag{9.15}$$

then (9.14) can be written in the form

$$x(t) = re^{\lambda t}\cos(\mu t - \delta), \tag{9.16}$$

where

$$r = \sqrt{c_1^2 + c_2^2},$$
$$\delta = \text{Atan2}(c_2, c_1). \tag{9.17}$$

In this form, it is easier to see that the resulting motion is an oscillation whose amplitude is exponentially decreasing toward zero.

Another common way of describing oscillatory second-order systems is in terms of **damping ratio** and **natural frequency**. These terms are defined by the parameterization of the characteristic equation given by

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0, \tag{9.18}$$

where ζ is the damping ratio (a dimensionless number between 0 and 1) and ω_n is the natural frequency.[2] Relationships between the pole locations and these parameters are

$$\lambda = -\zeta\omega_n$$

and

$$\mu = \omega_n\sqrt{1-\zeta^2}. \tag{9.19}$$

In this terminology, μ, the imaginary part of the poles, is sometimes called the **damped natural frequency**. For a damped spring–mass system such as the one in Fig. 9.2, the damping ratio and natural frequency are, respectively,

$$\zeta = \frac{b}{2\sqrt{km}},$$
$$\omega_n = \sqrt{k/m}. \tag{9.20}$$

When no damping is present ($b = 0$ in our example), the damping ratio becomes zero; for critical damping ($b^2 = 4km$), the damping ratio is 1.

Figure 9.4 shows an example of pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are complex, the system exhibits oscillatory or underdamped motion.

EXAMPLE 9.2

Find the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 1$, and $k = 1$ and the block (initially at rest) is released from the position $x = -1$.

The characteristic equation is

$$s^2 + s + 1 = 0, \tag{9.21}$$

[2]The terms *damping ratio* and *natural frequency* can also be applied to overdamped systems, in which case $\zeta > 1.0$.

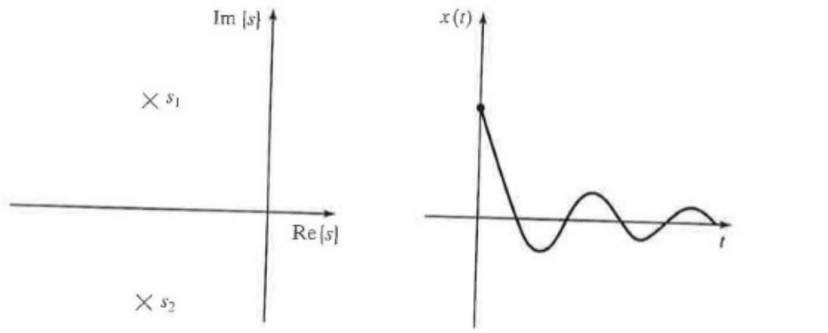


FIGURE 9.4: Root location and response to initial conditions for an underdamped system.

which has the roots $s_i = -\frac{1}{2} \pm \frac{\sqrt{3}}{2}i$. Hence, the response has the form

$$x(t) = e^{-\frac{t}{2}}\left(c_1\cos\frac{\sqrt{3}}{2}t + c_2\sin\frac{\sqrt{3}}{2}t\right). \tag{9.22}$$

We now use the given initial conditions, $x(0) = -1$ and $\dot{x}(0) = 0$, to compute c_1 and c_2. To satisfy these conditions at $t = 0$, we must have

$$c_1 = -1$$

and

$$-\frac{1}{2}c_1 - \frac{\sqrt{3}}{2}c_2 = 0, \tag{9.23}$$

which are satisfied by $c_1 = -1$ and $c_2 = \frac{\sqrt{3}}{3}$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = e^{-\frac{t}{2}}\left(-\cos\frac{\sqrt{3}}{2}t - \frac{\sqrt{3}}{3}\sin\frac{\sqrt{3}}{2}t\right). \tag{9.24}$$

This result can also be put in the form of (9.16), as

$$x(t) = \frac{2\sqrt{3}}{3}e^{-\frac{t}{2}}\cos\left(\frac{\sqrt{3}}{2}t + 120^\circ\right). \tag{9.25}$$

Real and equal roots

By substitution into (9.3), it can be shown that, in the case of real and equal roots (i.e., **repeated roots**), the solution has the form

$$x(t) = c_1e^{s_1t} + c_2te^{s_2t}, \tag{9.26}$$

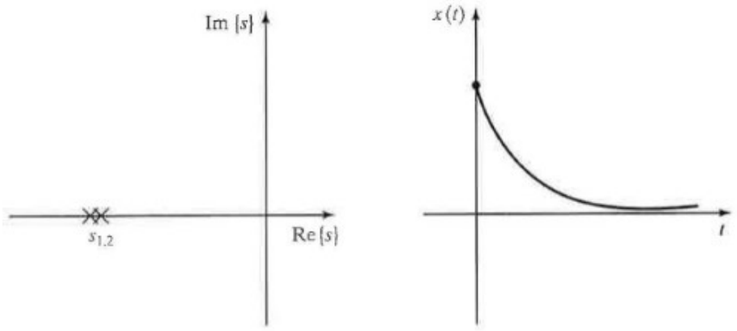


FIGURE 9.5: Root location and response to initial conditions for a critically damped system.

where, in this case, $s_1 = s_2 = -\frac{b}{2m}$, so (9.26) can be written

$$x(t) = (c_1 + c_2 t)e^{-\frac{b}{2m}t}. \tag{9.27}$$

In case it is not clear, a quick application of **l'Hôpital's rule** [2] shows that, for any c_1, c_2, and a,

$$\lim_{t\to\infty} (c_1 + c_2 t)e^{-at} = 0. \tag{9.28}$$

Figure 9.5 shows an example of pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are real and equal, the system exhibits critically damped motion, the fastest possible nonoscillatory response.

EXAMPLE 9.3

Work out the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 4$, and $k = 4$ and the block (initially at rest) is released from the position $x = -1$.

The characteristic equation is

$$s^2 + 4s + 4 = 0, \tag{9.29}$$

which has the roots $s_1 = s_2 = -2$. Hence, the response has the form

$$x(t) = (c_1 + c_2 t)e^{-2t}. \tag{9.30}$$

We now use the given initial conditions, $x(0) = -1$ and $\dot{x}(0) = 0$, to calculate c_1 and c_2. To satisfy these conditions at $t = 0$, we must have

$$c_1 = -1$$

and

$$-2c_1 + c_2 = 0, \tag{9.31}$$

which are satisfied by $c_1 = -1$ and $c_2 = -2$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = (-1 - 2t)e^{-2t}. \tag{9.32}$$

In Examples 9.1 through 9.3, all the systems were stable. For any passive physical system like that of Fig. 9.2, this will be the case. Such mechanical systems always have the properties

$$\begin{aligned} m &> 0, \\ b &> 0, \\ k &> 0. \end{aligned} \tag{9.33}$$

In the next section, we will see that the action of a control system is, in effect, to change the value of one or more of these coefficients. It will then be necessary to consider whether the resulting system is stable.

9.4 CONTROL OF SECOND-ORDER SYSTEMS

Suppose that the natural response of our second-order mechanical system is not what we wish it to be. Perhaps it is underdamped and oscillatory, and we would like it to be critically damped; or perhaps the spring is missing altogether ($k = 0$), so the system never returns to $x = 0$ if disturbed. Through the use of sensors, an actuator, and a control system, we can modify the system's behavior as desired.

Figure 9.6 shows a damped spring–mass system with the addition of an actuator with which it is possible to apply a force f to the block. A free-body diagram leads to the equation of motion,

$$m\ddot{x} + b\dot{x} + kx = f. \tag{9.34}$$

Let's also assume that we have sensors capable of detecting the block's position and velocity. We now propose a **control law** which computes the force that should be applied by the actuator as a function of the sensed feedback:

$$f = -k_p x - k_v \dot{x}. \tag{9.35}$$

Figure 9.7 is a block diagram of the closed-loop system, where the portion to the left of the dashed line is the control system (usually implemented in a computer) and that to the right of the dashed line is the physical system. Implicit in the figure are interfaces between the control computer and the output actuator commands and the input sensor information.

The control system we have proposed is a **position-regulation** system—it simply attempts to maintain the position of the block in one fixed place regardless

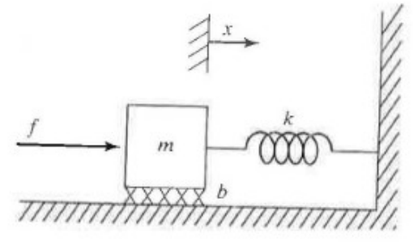


FIGURE 9.6: A damped spring–mass system with an actuator.

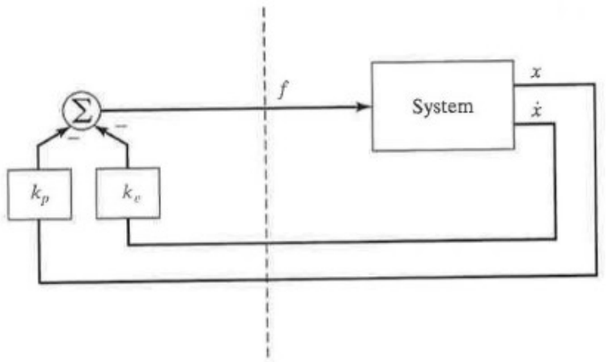


FIGURE 9.7: A closed-loop control system. The control computer (to the left of the dashed line) reads sensor input and writes actuator output commands.

of disturbance forces applied to the block. In a later section, we will construct a **trajectory-following** control system, which can cause the block to follow a desired position trajectory.

By equating the open-loop dynamics of (9.34) with the control law of (9.35), we can derive the closed-loop dynamics as

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x}, \tag{9.36}$$

or

$$m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0, \tag{9.37}$$

or

$$m\ddot{x} + b'\dot{x} + k'x = 0, \tag{9.38}$$

where $b' = b + k_v$ and $k' = k + k_p$. From (9.37) and (9.38), it is clear that, by setting the **control gains**, k_v and k_p, we can cause the closed-loop system to appear to have any second system behavior that we wish. Often, gains would be chosen to obtain critical damping (i.e., $b' = 2\sqrt{mk'}$) and some desired **closed-loop stiffness** given directly by k'.

Note that k_v and k_p could be positive or negative, depending on the parameters of the original system. However, if b' or k' became negative, the result would be an unstable control system. This instability will be obvious if one writes down the solution of the second-order differential equation (in the form of (9.6), (9.14), or (9.26)). It also makes intuitive sense that, if b' or k' is negative, servo errors tend to get magnified rather than reduced.

EXAMPLE 9.4

If the parameters of the system in Fig. 9.6 are $m = 1$, $b = 1$, and $k = 1$, find gains k_p and k_v for a position-regulation control law that results in the system's being critically damped with a closed-loop stiffness of 16.0.

If we wish k' to be 16.0, then, for critical damping, we require that $b' = 2\sqrt{mk'} = 8.0$. Now, $k = 1$ and $b = 1$, so we need

$$k_p = 15.0,$$
$$k_v = 7.0. \tag{9.39}$$

9.5 CONTROL-LAW PARTITIONING

In preparation for designing control laws for more complicated systems, let us consider a slightly different controller structure for the sample problem of Fig. 9.6. In this method, we will partition the controller into a **model-based portion** and a **servo portion**. The result is that the system's parameters (i.e., m, b, and k, in this case) appear only in the model-based portion and that the servo portion is independent of these parameters. At the moment, this distinction might not seem important, but it will become more obviously important as we consider nonlinear systems in Chapter 10. We will adopt this **control-law partitioning** approach throughout the book.

The open-loop equation of motion for the system is

$$m\ddot{x} + b\dot{x} + kx = f. \tag{9.40}$$

We wish to decompose the controller for this system into two parts. In this case, the model-based portion of the control law will make use of supposed knowledge of m, b, and k. This portion of the control law is set up such that it *reduces the system so that it appears to be a unit mass*. This will become clear when we do Example 9.5. The second part of the control law makes use of feedback to modify the behavior of the system. The model-based portion of the control law has the effect of making the system appear as a unit mass, so the design of the servo portion is very simple—gains are chosen to control a system composed of a single unit mass (i.e., no friction, no stiffness).

The model-based portion of the control appears in a control law of the form

$$f = \alpha f' + \beta. \tag{9.41}$$

where α and β are functions or constants and are chosen so that, if f' is taken as the *new input* to the system, *the system appears to be a unit mass*. With this structure of the control law, the system equation (the result of combining (9.40) and (9.41)) is

$$m\ddot{x} + b\dot{x} + kx = \alpha f' + \beta. \tag{9.42}$$

Clearly, in order to make the system appear as a unit mass from the f' input, for this particular system we should choose α and β as follows:

$$\alpha = m,$$
$$\beta = b\dot{x} + kx. \tag{9.43}$$

Making these assignments and plugging them into (9.42), we have the system equation

$$\ddot{x} = f'. \tag{9.44}$$

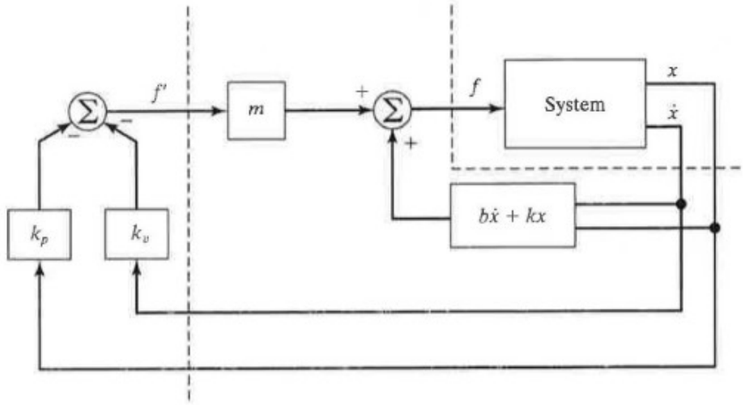


FIGURE 9.8: A closed-loop control system employing the partitioned control method.

This is the equation of motion for a unit mass. We now proceed as if (9.44) were the open-loop dynamics of a system to be controlled. We design a control law to compute f', just as we did before:

$$f' = -k_v\dot{x} - k_p x. \tag{9.45}$$

Combining this control law with (9.44) yields

$$\ddot{x} + k_v\dot{x} + k_p x = 0. \tag{9.46}$$

Under this methodology, the setting of the control gains is simple and is independent of the system parameters; that is,

$$k_v = 2\sqrt{k_p} \tag{9.47}$$

must hold for critical damping. Figure 9.8 shows a block diagram of the partitioned controller used to control the system of Fig. 9.6.

EXAMPLE 9.5

If the parameters of the system in Fig. 9.6 are $m = 1, b = 1$, and $k = 1$, find α, β, and the gains k_p and k_v for a position-regulation control law that results in the system's being critically damped with a closed-loop stiffness of 16.0.

We choose

$$\alpha = 1,$$
$$\beta = \dot{x} + x. \tag{9.48}$$

so that the system appears as a unit mass from the fictitious f' input. We then set gain k_p to the desired closed-loop stiffness and set $k_v = 2\sqrt{k_p}$ for critical damping.

This gives

$$k_p = 16.0,$$
$$k_v = 8.0. \tag{9.49}$$

9.6 TRAJECTORY-FOLLOWING CONTROL

Rather than just maintaining the block at a desired location, let us enhance our controller so that the block can be made to follow a trajectory. The trajectory is given by a function of time, $x_d(t)$, that specifies the desired position of the block. We assume that the trajectory is smooth (i.e., the first two derivatives exist) and that our trajectory generator provides x_d, $\dot{x}_d$, and $\ddot{x}_d$ at all times t. We define the servo error between the desired and actual trajectory as $e = x_d - x$. A servo-control law that will cause trajectory following is

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e. \tag{9.50}$$

We see that (9.50) is a good choice if we combine it with the equation of motion of a unit mass (9.44), which leads to

$$\ddot{x} = \ddot{x}_d + k_v\dot{e} + k_p e, \tag{9.51}$$

or

$$\ddot{e} + k_v\dot{e} + k_p e = 0. \tag{9.52}$$

This is a second-order differential equation for which we can choose the coefficients, so we can design any response we wish. (Often, critical damping is the choice made.) Such an equation is sometimes said to be written in **error space**, because it describes the evolution of errors relative to the desired trajectory. Figure 9.9 shows a block diagram of our trajectory-following controller.

If our model is perfect (i.e., our knowledge of m, b, and k), and if there is no noise and no initial error, the block will follow the desired trajectory exactly. If there is an initial error, it will be suppressed according to (9.52), and thereafter the system will follow the trajectory exactly.

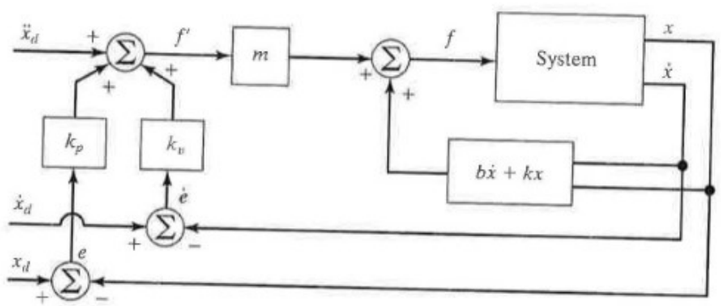


FIGURE 9.9: A trajectory-following controller for the system in Fig. 9.6.

9.7 DISTURBANCE REJECTION

One of the purposes of a control system is to provide **disturbance rejection**, that is, to maintain good performance (i.e., minimize errors) even in the presence of some external disturbances or **noise**. In Fig. 9.10, we show the trajectory-following controller with an additional input: a disturbance force f_{dist}. An analysis of our closed-loop system leads to the error equation

$$\ddot{e} + k_v\dot{e} + k_p e = f_{\text{dist}}. \tag{9.53}$$

Equation (9.53) is that of a differential equation driven by a forcing function. If it is known that f_{dist} is **bounded**—that is, that a constant a exists such that

$$\max_t f_{\text{dist}}(t) < a, \tag{9.54}$$

then the solution of the differential equation, $e(t)$, is also bounded. This result is due to a property of stable linear systems known as **bounded-input, bounded-output** or **BIBO** stability [3, 4]. This very basic result ensures that, for a large class of possible disturbances, we can at least be assured that the system remains stable.

Steady-state error

Let's consider the simplest kind of disturbance—namely, that f_{dist} is a constant. In this case, we can perform a **steady-state analysis** by analyzing the system at rest (i.e., the derivatives of all system variables are zero). Setting derivatives to zero in (9.53) yields the steady-state equation

$$k_p e = f_{\text{dist}}, \tag{9.55}$$

or

$$e = f_{\text{dist}}/k_p. \tag{9.56}$$

The value of e given by (9.56) represents a **steady-state error**. Thus, it is clear that the higher the position gain k_p, the smaller will be the steady-state error.

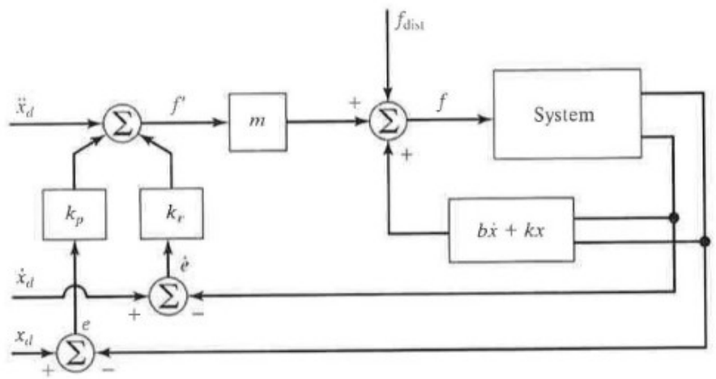


FIGURE 9.10: A trajectory-following control system with a disturbance acting.

Addition of an integral term

In order to eliminate steady-state error, a modified control law is sometimes used. The modification involves the addition of an *integral* term to the control law. The control law becomes

$$f' = \ddot{x}_d + k_v\ddot{e} + k_p\dot{e} + k_i\int e dt. \tag{9.57}$$

which results in the error equation

$$\ddot{e} + k_v\dot{e} + k_p e + k_i\int e dt = f_{\text{dist}}. \tag{9.58}$$

The term is added so that the system will have no steady-state error in the presence of constant disturbances. If $e(t) = 0$ for $t < 0$, we can write (9.58) for $t > 0$ as

$$\dddot{e} + k_v\ddot{e} + k_p\dot{e} + k_i e = \dot{f}_{\text{dist}}, \tag{9.59}$$

which, in the steady state (for a constant disturbance), becomes

$$k_i e = 0, \tag{9.60}$$

so

$$e = 0. \tag{9.61}$$

With this control law, the system becomes a third-order system, and one can solve the corresponding third-order differential equation to work out the response of the system to initial conditions. Often, k_i is kept quite small so that the third-order system is "close" to the second-order system without this term (i.e., a dominant-pole analysis can be performed). The form of control law (9.57) is called a **PID control law**, or "proportional, integral, derivative" control law [4]. For simplicity, the displayed equations generally do not show an integral term in the control laws that we develop in this book.

9.8 CONTINUOUS VS. DISCRETE TIME CONTROL

In the control systems we have discussed, we implicitly assumed that the control computer performs the computation of the control law in zero time (i.e., infinitely fast), so that the value of the actuator force f is a continuous function of time. Of course, in reality, the computation requires some time, and the resulting commanded force is therefore a discrete "staircase" function. We shall employ this approximation of a very fast control computer throughout the book. This approximation is good if the rate at which new values of f are computed is much faster than the natural frequency of the system being controlled. In the field of **discrete time control** or **digital control**, one does not make this approximation but rather takes the **servo rate** of the control system into account when analyzing the system [3].

We will generally assume that the computations can be performed quickly enough that our continuous time assumption is valid. This raises a question: How quick is quick enough? There are several points that need to be considered in choosing a sufficiently fast servo (or sample) rate:

Tracking reference inputs: The frequency content of the desired or reference input places an absolute lower bound on the sample rate. The sample rate must be at least twice the bandwidth of reference inputs. This is usually not the limiting factor.

Disturbance rejection: In disturbance rejection, an upper bound on performance is given by a continuous-time system. If the sample period is longer than the correlation time of the disturbance effects (assuming a statistical model for random disturbances), then these disturbances will not be suppressed. Perhaps a good rule of thumb is that the sample period should be 10 times shorter than the correlation time of the noise [3].

Antialiasing: Any time an analog sensor is used in a digital control scheme, there will be a problem with aliasing unless the sensor's output is strictly band limited. In most cases, sensors do not have a band limited output, and so sample rate should be chosen such that the amount of energy that appears in the aliased signal is small.

Structural resonances: We have not included bending modes in our characterization of a manipulator's dynamics. All real mechanisms have finite stiffness and so will be subject to various kinds of vibrations. If it is important to suppress these vibrations (and it often is), we must choose a sample rate at least twice the natural frequency of these resonances. We will return to the topic of resonance later in this chapter.

9.9 MODELING AND CONTROL OF A SINGLE JOINT

In this section, we will develop a simplified model of a single rotary joint of a manipulator. A few assumptions will be made that will allow us to model the resulting system as a second-order linear system. For a more complete model of an actuated joint, see [5].

A common actuator found in many industrial robots is the direct current (DC) torque motor (as in Fig. 8.18). The nonturning part of the motor (the **stator**) consists of a housing, bearings, and either permanent magnets or electromagnets. These stator magnets establish a magnetic field across the turning part of the motor (the **rotor**). The rotor consists of a shaft and windings through which current moves to power the motor. The current is conducted to the windings via brushes, which make contact with the commutator. The commutator is wired to the various windings (also called the **armature**) in such a way that torque is always produced in the desired direction. The underlying physical phenomenon [6] that causes a motor to generate a torque when current passes through the windings can be expressed as

$$F = qV \times B, \tag{9.62}$$

where charge q, moving with velocity V through a magnetic field B, experiences a force F. The charges are those of electrons moving through the windings, and the magnetic field is that set up by the stator magnets. Generally, the torque-producing ability of a motor is stated by means of a single **motor torque constant**, which relates armature current to the output torque as

$$\tau_m = k_m i_a. \tag{9.63}$$

When a motor is rotating, it acts as a generator, and a voltage develops across the armature. A second motor constant, the **back emf constant**,[3] describes the voltage generated for a given rotational velocity:

$$v = k_e \dot{\theta}_m. \tag{9.64}$$

Generally, the fact that the commutator is switching the current through various sets of windings causes the torque produced to contain some **torque ripple**. Although sometimes important, this effect can usually be ignored. (In any case, it is quite hard to model—and quite hard to compensate for, even if it is modeled.)

Motor-armature inductance

Figure 9.11 shows the electric circuit of the armature. The major components are a voltage source, v_a, the inductance of the armature windings, l_a, the resistance of the armature windings, r_a, and the generated back emf, v. The circuit is described by a first-order differential equation:

$$l_a \dot{i}_a + r_a i_a = v_a - k_e \dot{\theta}_m. \tag{9.65}$$

It is generally desirable to control the torque generated by the motor (rather than the velocity) with electronic motor driver circuitry. These drive circuits sense the current through the armature and continuously adjust the voltage source v_a so that a desired current i_a flows through the armature. Such a circuit is called a **current amplifier** motor driver [7]. In these current-drive systems, the rate at which the armature current can be commanded to change is limited by the motor inductance l_a and by an upper limit on the voltage capability of the voltage source v_a. The net effect is that of a **low-pass filter** between the requested current and output torque.

Our first simplifying assumption is that the inductance of the motor can be neglected. This is a reasonable assumption when the natural frequency of the closed-loop control system is quite low compared to the cut-off frequency of the implicit low-pass filter in the current-drive circuitry due to the inductance. This assumption, along with the assumption that torque ripple is a negligible effect, means that we can essentially command torque directly. Although there might be a scale factor (such as k_m) to contend with, we will assume that the actuator acts as a pure torque source that we can command directly.

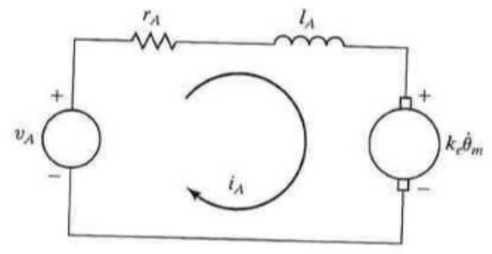


FIGURE 9.11: The armature circuit of a DC torque motor.

[3] "emf" stands for electromotive force.

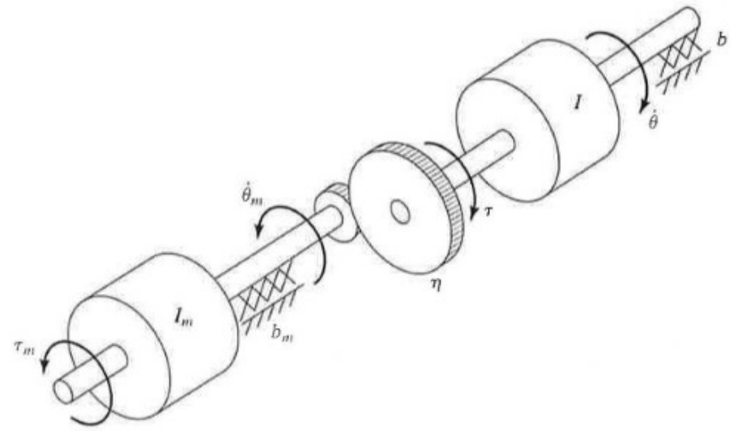


FIGURE 9.12: Mechanical model of a DC torque motor connected through gearing to an inertial load.

Effective inertia

Figure 9.12 shows the mechanical model of the rotor of a DC torque motor connected through a gear reduction to an inertial load. The torque applied to the rotor, τ_m, is given by (9.63) as a function of the current i_a flowing in the armature circuit. The gear ratio (η) causes an increase in the torque seen at the load and a reduction in the speed of the load, given by

$$\begin{aligned} \tau &= \eta \tau_m, \\ \dot{\theta} &= (1/\eta)\dot{\theta}_m, \end{aligned} \tag{9.66}$$

where $\eta > 1$. Writing a torque balance for this system in terms of torque at the rotor yields

$$\tau_m = I_m \ddot{\theta}_m + b_m \dot{\theta}_m + (1/\eta)\left(I\ddot{\theta} + b\dot{\theta}\right), \tag{9.67}$$

where I_m and I are the inertias of the motor rotor and of the load, respectively, and b_m and b are viscous friction coefficients for the rotor and load bearings, respectively. Using the relations (9.66), we can write (9.67) in terms of motor variables as

$$\tau_m = \left(I_m + \frac{I}{\eta^2}\right)\ddot{\theta}_m + \left(b_m + \frac{b}{\eta^2}\right)\dot{\theta}_m \tag{9.68}$$

or in terms of load variables as

$$\tau = (I + \eta^2 I_m)\ddot{\theta} + (b + \eta^2 b_m)\dot{\theta}. \tag{9.69}$$

The term $I + \eta^2 I_m$ is sometimes called the **effective inertia** "seen" at the output (link side) of the gearing. Likewise, the term $b + \eta^2 b_m$ can be called the **effective damping**. Note that, in a highly geared joint (i.e., $\eta \gg 1$), the inertia of the motor rotor can be a significant portion of the combined effective inertia. It is this effect that allows us to make the assumption that the effective inertia is a constant. We know

from Chapter 6 that the inertia, I, of a joint of the mechanism actually varies with configuration and load. However, in highly geared robots, the variations represent a smaller percentage than they would in a **direct-drive** manipulator (i.e., $\eta = 1$). To ensure that the motion of the robot link is never underdamped, the value used for I should be the maximum of the range of values that I takes on; we'll call this value $I_{\max}$. This choice results in a system that is critically damped or overdamped in all situations. In Chapter 10, we will deal with varying inertia directly and will not have to make this assumption.

EXAMPLE 9.6

If the apparent link inertia, I, varies between 2 and 6 Kg-m^2, the rotor inertia is $I_m = 0.01$, and the gear ratio is $\eta = 30$, what are the minimum and maximum of the effective inertia?

The minimum effective inertia is

$$I_{\min} + \eta^2 I_m = 2.0 + (900)(0.01) = 11.0; \tag{9.70}$$

the maximum is

$$I_{\max} + \eta^2 I_m = 6.0 + (900)(0.01) = 15.0. \tag{9.71}$$

Hence, we see that, as a percentage of the total effective inertia, the variation of inertia is reduced by the gearing.

Unmodeled flexibility

The other major assumption we have made in our model is that the gearing, the shafts, the bearings, and the driven link are not flexible. In reality, all of these elements have finite stiffness, and their flexibility, if modeled, would increase the order of the system. The argument for ignoring flexibility effects is that, if the system is sufficiently stiff, the natural frequencies of these **unmodeled resonances** are very high and can be neglected compared to the influence of the dominant second-order poles that we have modeled.[4] The term "unmodeled" refers to the fact that, for purposes of control-system analysis and design, we neglect these effects and use a simpler dynamic model, such as (9.69).

Because we have chosen not to model structural flexibilities in the system, we must be careful not to excite these resonances. A rule of thumb [8] is that, if the lowest structural resonance is ω_{res}, then we must limit our closed-loop natural frequency according to

$$\omega_n \leq \frac{1}{2}\omega_{\text{res}}. \tag{9.72}$$

This provides some guidance on how to choose gains in our controller. We have seen that increasing gains leads to faster response and lower steady-state error, but we now see that unmodeled structural resonances limit the magnitude of gains. Typical industrial manipulators have structural resonances in the range from 5 Hz to 25 Hz [8]. Recent designs using direct-drive arrangements that do not contain flexibility

[4] This is basically the same argument we used to neglect the pole due to the motor inductance. Including it would also have raised the order of the overall system.

introduced by reduction and transmission systems have their lowest structural resonances as high as 70 Hz [9].

EXAMPLE 9.7

Consider the system of Fig. 9.7 with the parameter values $m = 1$, $b = 1$, and $k = 1$. Additionally, it is known that the lowest unmodeled resonance of the system is at 8 radians/second. Find α, β, and gains k_p and k_v for a position-control law so the system is critically damped, doesn't excite unmodeled dynamics, and has as high a closed-loop stiffness as possible.

We choose

$$\alpha = 1,$$
$$\beta = \dot{x} + x, \tag{9.73}$$

so that the system appears as a unit mass from the fictitious f' input. Using our rule of thumb (9.72), we choose the closed-loop natural frequency to be $\omega_n = 4$ radians/second. From (9.18) and (9.46), we have $k_p = \omega_n^2$, so

$$k_p = 16.0,$$
$$k_v = 8.0. \tag{9.74}$$

Estimating resonant frequency

The same sources of structural flexibility discussed in Chapter 8 give rise to resonances. In each case where a structural flexibility can be identified, an approximate analysis of the resulting vibration is possible if we can describe the effective mass or inertia of the flexible member. This is done by approximating the situation by a simple spring–mass system, which, as given in (9.20), exhibits the natural frequency

$$\omega_n = \sqrt{k/m}, \tag{9.75}$$

where k is the stiffness of the flexible member and m is the equivalent mass displaced in vibrations.

EXAMPLE 9.8

A shaft (assumed massless) with a stiffness of 400 Nt-m/radian drives a rotational inertia of 1 Kg-m^2. If the shaft stiffness was neglected in the modeling of the dynamics, what is the frequency of this unmodeled resonance?

Using (9.75), we have

$$\omega_{\text{res}} = \sqrt{400/1} = 20 \text{ rad/second} = 20/(2\pi)\text{Hz} \cong 3.2 \text{ Hz}. \tag{9.76}$$

For the purposes of a rough estimate of the lowest resonant frequency of beams and shafts, [10] suggests using a **lumped model** of the mass. We already

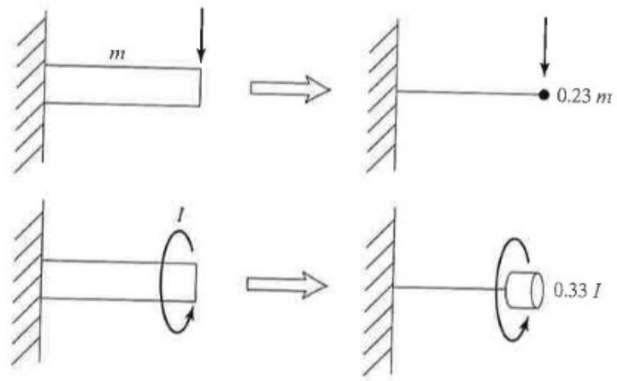


FIGURE 9.13: Lumped models of beams for estimation of lowest lateral and torsional resonance.

have formulas for estimating stiffness at the ends of beams and shafts; these lumped models provide the effective mass or inertia needed for our estimation of resonant frequency. Figure 9.13 shows the results of an energy analysis [10] which suggests that a beam of mass m be replaced by a point mass at the end of 0.23 m and, likewise, that a distributed inertia of I be replaced by a lumped 0.33 I at the end of the shaft.

EXAMPLE 9.9

A link of mass 4.347 Kg has an end-point lateral stiffness of 3600 Nt/m. Assuming the drive system is completely rigid, the resonance due to the flexibility of the link will limit control gains. What is ω_{res}?

The 4.347 Kg mass is distributed along the link. Using the method of Fig. 9.13, the effective mass is $(0.23)(4.347) \cong 1.0$ Kg. Hence, the vibration frequency is

$$\omega_{\text{res}} = \sqrt{3600/1.0} = 60 \text{ radians/second} = 60/(2\pi)\text{Hz} \cong 9.6 \text{ Hz}. \tag{9.77}$$

The inclusion of structural flexibilities in the model of the system used for control-law synthesis is required if we wish to achieve closed-loop bandwidths higher than that given by (9.75). The resulting system models are of high order, and the control techniques applicable to this situation become quite sophisticated. Such control schemes are currently beyond the state of the art of industrial practice but are an active area of research [11, 12].

Control of a single joint

In summary, we make the following three major assumptions:

1. The motor inductance l_a can be neglected.
2. Taking into account high gearing, we model the effective inertia as a constant equal to $I_{\max} + \eta^2 I_m$.
3. Structural flexibilities are neglected, except that the lowest structural resonance ω_{res} is used in setting the servo gains.

With these assumptions, a single joint of a manipulator can be controlled with the partitioned controller given by

$$\alpha = I_{\max} + \eta^2 I_m,$$
$$\beta = (b + \eta^2 b_m)\dot{\theta}, \tag{9.78}$$
$$\tau' = \ddot{\theta}_d + k_v \dot{e} + k_p e. \tag{9.79}$$

The resulting system closed-loop dynamics are

$$\ddot{e} + k_v \dot{e} + k_p e = \tau_{\text{dist}}, \tag{9.80}$$

where the gains are chosen as

$$k_p = \omega_n^2 = \frac{1}{4}\omega_{\text{res}}^2,$$
$$k_v = 2\sqrt{k_p} = \omega_{\text{res}}. \tag{9.81}$$

9.10 ARCHITECTURE OF AN INDUSTRIAL-ROBOT CONTROLLER

In this section, we briefly look at the architecture of the control system of the Unimation PUMA 560 industrial robot. As shown in Fig. 9.14, the hardware architecture is that of a two-level hierarchy, with a DEC LSI-11 computer serving as the top-level "master" control computer passing commands to six Rockwell 6503 microprocessors.[5] Each of these microprocessors controls an individual joint with a PID control law not unlike that presented in this chapter. Each joint of the PUMA 560 is instrumented with an incremental optical encoder. The encoders are interfaced to an up/down counter, which the microprocessor can read to obtain the current joint position. There are no tachometers in the PUMA 560; rather, joint positions are differenced on subsequent servo cycles to obtain an estimate of joint velocity. In order to command torques to the DC torque motors, the microprocessor

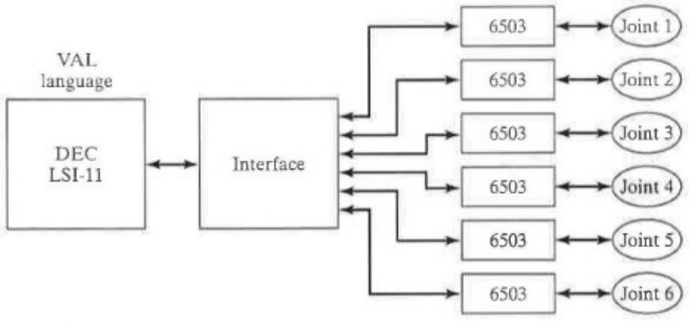


FIGURE 9.14: Hierarchical computer architecture of the PUMA 560 robot-control system.

[5]These simple 8-bit computers are already old technology. It is common these days for robot controllers to be based on 32-bit microprocessors.

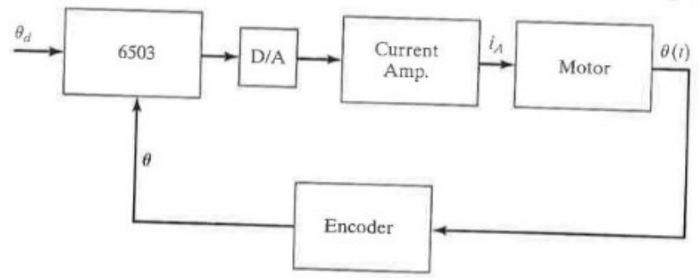


FIGURE 9.15: Functional blocks of the joint-control system of the PUMA 560.

is interfaced to a digital-to-analog converter (DAC) so that motor currents can be commanded to the current-driver circuits. The current flowing through the motor is controlled in analog circuitry by adjusting the voltage across the armature as needed to maintain the desired armature current. A block diagram is shown in Fig. 9.15.

Each 28 milliseconds, the LSI-11 computer sends a new position command (**set-point**) to the joint microprocessors. The joint microprocessors are running on a 0.875 millisecond cycle. In this time, they interpolate the desired position set-point, compute the servo error, compute the PID control law, and command a new value of torque to the motors.

The LSI-11 computer carries out all the "high-level" operations of the overall control system. First of all, it takes care of interpreting the VAL (Unimation's robot programming language) program commands one by one. When a motion command is interpreted, the LSI-11 must perform any needed inverse kinematic computations, plan a desired trajectory, and begin generating trajectory via points every 28 milliseconds for the joint controllers.

The LSI-11 is also interfaced to such standard peripherals as the terminal and a floppy disk drive. In addition, it is interfaced to a **teach pendant**. A teach pendant is a handheld button box that allows the operator to move the robot around in a variety of modes. For example, the PUMA 560 system allows the user to move the robot incrementally in joint coordinates or in Cartesian coordinates from the teach pendant. In this mode, teach-pendant buttons cause a trajectory to be computed "on the fly" and passed down to the joint-control microprocessors.

BIBLIOGRAPHY

[1] W. Boyce and R. DiPrima, *Elementary Differential Equations*, 3rd edition, John Wiley and Sons, New York, 1977.

[2] E. Purcell, *Calculus with Analytic Geometry*, Meredith Corporation, New York, 1972.

[3] G. Franklin and J.D. Powell, *Digital Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1980.

[4] G. Franklin, J.D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1986.

[5] J. Luh, "Conventional Controller Design for Industrial Robots—a Tutorial," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 3, June 1983.

[6] D. Halliday and R. Resnik, *Fundamentals of Physics*, Wiley, New York 1970.

[7] Y. Koren and A. Ulsoy, "Control of DC Servo-Motor Driven Robots," *Proceedings of Robots 6 Conference*, SME, Detroit, March 1982.

[8] R.P. Paul, *Robot Manipulators*, MIT Press, Cambridge, MA, 1981.

[9] H. Asada and K. Youcef-Toumi, *Direct-Drive Robots—Theory and Practice*, MIT Press, Cambridge, MA, 1987.

[10] J. Shigley, *Mechanical Engineering Design*, 3rd edition, McGraw-Hill, New York, 1977.

[11] W. Book, "Recursive Lagrangian Dynamics of Flexible Manipulator Arms," *The International Journal of Robotics Research*, Vol. 3, No. 3, 1984.

[12] R. Cannon and E. Schmitz, "Initial Experiments on the End-Point Control of a Flexible One Link Robot," *The International Journal of Robotics Research*, Vol. 3, No. 3, 1984.

[13] R.J. Nyzen, "*Analysis and Control of an Eight-Degree-of-Freedom Manipulator*," Ohio University Master's Thesis, Mechanical Engineering, Dr. Robert L. Williams II, Advisor, August 1999.

[14] R.L. Williams II, "*Local Performance Optimization for a Class of Redundant Eight-Degree-of-Freedom Manipulators*," **NASA Technical Paper 3417**, NASA Langley Research Center, Hampton, VA, March 1994.

EXERCISES

9.1 [20] For a second-order differential equation with complex roots

$$s_1 = \lambda + \mu i,$$
$$s_2 = \lambda - \mu i,$$

show that the general solution

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t},$$

can be written

$$x(t) = c_1 e^{\lambda t} \cos(\mu t) + c_2 e^{\lambda t} \sin(\mu t).$$

9.2 [13] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 2$, $b = 6$, and $k = 4$ and the block (initially at rest) is released from the position $x = 1$.

9.3 [13] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 2$, and $k = 1$ and the block (initially at rest) is released from the position $x = 4$.

9.4 [13] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 4$, and $k = 5$ and the block (initially at rest) is released from the position $x = 2$.

9.5 [15] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 7$, and $k = 10$ and the block is released from the position $x = 1$ with an initial velocity of $x = 2$.

9.6 [15] Use the (1, 1) element of (6.60) to compute the variation (as a percentage of the maximum) of the inertia "seen" by joint 1 of this robot as it changes configuration. Use the numerical values

$$l_1 = l_2 = 0.5 \text{ m},$$
$$m_1 = 4.0 \text{ Kg},$$
$$m_2 = 2.0 \text{ Kg}.$$

Consider that the robot is direct drive and that the rotor inertia is negligible.

9.7 [17] Repeat Exercise 9.6 for the case of a geared robot (use $\eta = 20$) and a rotor inertia of $I_m = 0.01$ Kg m^2.

9.8 [18] Consider the system of Fig. 9.6 with the parameter values $m = 1$, $b = 4$, and $k = 5$. The system is also known to possess an unmodeled resonance at $\omega_{res} = 6.0$ radians/second. Determine the gains k_v and k_p that will critically damp the system with as high a stiffness as is reasonable.

9.9 [25] In a system like that of Fig. 9.12, the inertial load, I, varies between 4 and 5 Kg-m^2. The rotor inertia is $I_m = 0.01$ Kg-m^2, and the gear ratio is $\eta = 10$. The system possesses unmodeled resonances at 8.0, 12.0, and 20.0 radians/second. Design α and β of the partitioned controller and give the values of k_p and k_v such that the system is never underdamped and never excites resonances, but is as stiff as possible.

9.10 [18] A designer of a direct-drive robot suspects that the resonance due to beam flexibility of the link itself will be the cause of the lowest unmodeled resonance. If the link is approximately a square-cross-section beam of dimensions $5 \times 5 \times 50$ cm with a 1-cm wall thickness and a total mass of 5 Kg, estimate ω_{res}.

9.11 [15] A direct-drive robot link is driven through a shaft of stiffness 1000 Nt-m/radian. The link inertia is 1 Kg-m^2. Assuming the shaft is massless, what is ω_{res}?

9.12 [18] A shaft of stiffness 500 Nt-m/radian drives the input of a rigid gear pair with $\eta = 8$. The output of the gears drives a rigid link of inertia 1 Kg-m^2. What is the ω_{res} caused by flexibility of the shaft?

9.13 [25] A shaft of stiffness 500 Nt-m/radian drives the input of a rigid gear pair with $\eta = 8$. The shaft has an inertia of 0.1 Kg-m^2. The output of the gears drives a rigid link of inertia 1 Kg-m^2. What is the ω_{res} caused by flexibility of the shaft?

9.14 [28] In a system like that of Fig. 9.12, the inertial load, I, varies between 4 and 5 Kg-m^2. The rotor inertia is $I_m = 0.01$ Kg-m^2, and the gear ratio is $\eta = 10$. The system possesses an unmodeled resonance due to an end-point stiffness of the link of 2400 Nt-m/radian. Design α and β of the partitioned controller, and give the values of k_p and k_v such that the system is never underdamped and never excites resonances, but is as stiff as possible.

9.15 [25] A steel shaft of length 30 cm and diameter 0.2 cm drives the input gear of a reduction of $\eta = 8$. The rigid output gear drives a steel shaft of length 30 cm and diameter 0.3 cm. What is the range of resonant frequencies observed if the load inertia varies between 1 and 4 Kg-m^2?

PROGRAMMING EXERCISE (PART 9)

We wish to simulate a simple trajectory-following control system for the three-link planar arm. This control system will be implemented as an independent-joint PD (proportional plus derivative) control law. Set the servo gains to achieve closed-loop stiffnesses of 175.0, 110.0, and 20.0 for joints 1 through 3 respectively. Try to achieve approximate critical damping.

Use the simulation routine **UPDATE** to simulate a discrete-time servo running at 100 Hz—that is, calculate the control law at 100 Hz, not at the frequency of the numerical integration process. Test the control scheme on the following tests:

1. Start the arm at $\Theta = (60, -110, 20)$ and command it to stay there until *time* = 3.0, when the set-points should instantly change to $\Theta = (60, -50, 20)$. That is, give a step input of 60 degrees to joint 2. Record the error–time history for each joint.
2. Control the arm to follow the cubic-spline trajectory from Programming Exercise Part 7. Record the error–time history for each joint.

MATLAB EXERCISE 9

This exercise focuses on linearized independent joint-control simulation for the shoulder joint (joint 2) of the NASA eight-axis AAI ARMII (Advanced Research Manipulator II) manipulator arm—see [14]. Familiarity with linear classical feedback-control systems, including block diagrams and Laplace transforms, is assumed. We will use Simulink, the graphical user interface of MATLAB.

Figure 9.16 shows a linearized open-loop system-dynamics model for the ARMII electromechanical shoulder joint/link, actuated by an armature-controller DC servomotor. The open-loop input is reference voltage V_{ref} (boosted to armature voltage via an amplifier), and the output of interest is the load shaft angle ThetaL. The figure also shows the feedback-control diagram, where the load-shaft angle is sensed via an optical encoder and provided as feedback to the PID controller. The table describes all system parameters and variables.

If we reflect the load shaft inertia and damping to the motor shaft, the effective polar inertia and damping coefficient are $J = J_M + J_L(t)/n^2$ and $C = C_M + C_L/n^2$. By virtue of the large gear ratio n, these effective values are not much different from the motor-shaft values. Thus, the gear ratio allows us to ignore variations in the configuration-dependent load-shaft inertia $J_L(t)$ and just set a reasonable average value.

The ARMII shoulder joint constant parameters are given in the accompanying table [13]. Note that we can use the English units directly, because their effect cancels out inside the control diagram. Also, we can directly use deg units for the angle. Develop a Simulink model to simulate the single-joint control model from the model and feedback-control diagram shown; use the specific parameters from the table. For the nominal case, determine the PID gains by trial and error for "good" performance (reasonable percent overshoot, rise time, peak time, and settling time). Simulate the resulting motion for moving this shoulder joint for a step input of 0 to 60 deg. Plot the simulated load-angle value over time, plus the load-shaft angular velocity over time. In addition, plot the

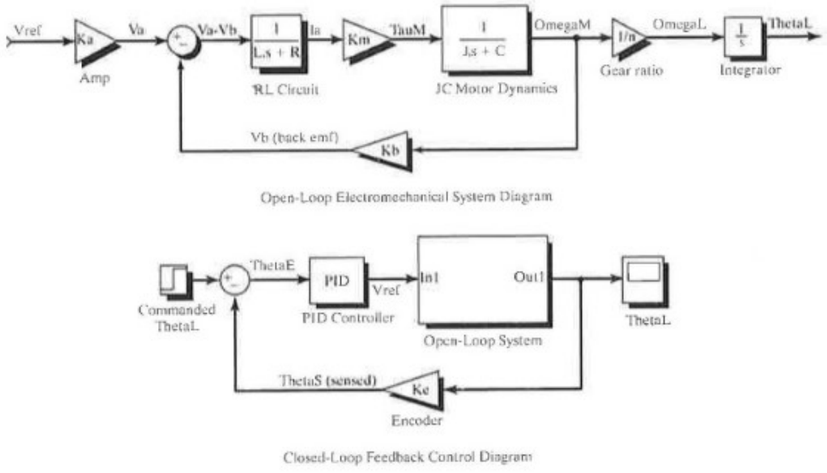


FIGURE 9.16: Linearized open-loop system-dynamics model for the ARMII electromechanical shoulder joint/link, actuated by an armature-controller DC servomotor.

TABLE 9.1: ARMII shoulder joint constant parameters.

$V_a(t)$	armature voltage	$\tau_M(t)$	generated motor torque	$\tau_L(t)$	load torque
$L = 0.0006H$	armature inductance	$\theta_M(t)$	motor shaft angle	$\theta_L(t)$	load shaft angle
$R = 1.40\Omega$	armature resistance	$\omega_M(t)$	motor shaft velocity	$\omega_L(t)$	load shaft velocity
$i_a(t)$	armature current	$J_M = 0.00844$ lb_f-in-s^2	lumped motor polar inertia	$J_L(t) = 1$ lb_f-in-s^2	lumped load polar inertia
$V_b(t)$	back emf voltage	$C_M = 0.00013$ lb_f-in/deg/s	motor shaft viscous damping coefficient	$C_L = 0.5$ lb_f-in/deg/s	load shaft viscous damping coefficient
$K_a = 12$	amplifier gain	$n = 200$	gear ratio	$g = 0$ in/s^2	gravity (ignore gravity at first)
$K_b = 0.00867$ V/deg/s	back emf constant	$K_M = 4.375$ lb_f-in/A	torque constant	$K_e = 1$	encoder transfer function

control effort—that is, the armature voltage V_a over time. (On the same graph, also give the back emf V_b.)

Now, try some changes—Simulink is so easy and enjoyable to change:

1) The step input is frustrating for controller design, so try a ramped step input instead: Ramp from 0 to 60 deg in 1.5 sec, then hold the 60-deg command for all time greater than 1.5 sec. Redesign PID gains and restimulate.
2) Investigate whether the inductor L is significant in this system. (The electrical system rises much faster than the mechanical system—this effect can be represented by time constants.)
3) We don't have a good estimate for the load inertia and damping (J_L and C_L). With your best PID gains from before, investigate how big these values can grow (scale the nominal parameters up equally) before they affect the system.
4) Now, include the effect of gravity as a disturbance to the motor torque T_M. Assume that the moving robot mass is 200 lb and the moving length beyond joint 2 is 6.4 feet. Test for the nominal "good" PID gains you found; redesign if necessary. The shoulder load angle θ_2 zero configuration is straight up.

CHAPTER 10

Nonlinear control of manipulators

10.1 INTRODUCTION

In the previous chapter, we made several approximations to allow a linear analysis of the manipulator-control problem. Most important among these approximations was that each joint could be considered independent and that the inertia "seen" by each joint actuator was constant. In implementations of linear controllers as introduced in the previous chapter, this approximation results in nonuniform damping throughout the workspace and other undesirable effects. In this chapter, we will introduce a more advanced control technique for which this assumption will not have to be made.

In Chapter 9, we modeled the manipulator by n independent second-order differential equations and based our controller on that model. In this chapter, we will base our controller design directly on the $n \times 1$-nonlinear vector differential equation of motion, derived in Chapter 6 for a general manipulator.

The field of nonlinear control theory is large; we must therefore restrict our attention to one or two methods that seem well suited to mechanical manipulators. Consequently, the major focus of the chapter will be one particular method, apparently first proposed in [1] and named the **computed-torque method** in [2, 3]. We will also introduce one method of stability analysis of nonlinear systems, known as **Lyapunov's** method [4].

To begin our discussion of nonlinear techniques for controlling a manipulator, we return again to a very simple single-degree-of-freedom mass–spring friction system.

10.2 NONLINEAR AND TIME-VARYING SYSTEMS

In the preceding development, we dealt with a linear constant-coefficient differential equation. This mathematical form arose because the mass–spring friction system of Fig. 9.6 was modeled as a linear time-invariant system. For systems whose parameters vary in time or systems that by nature are nonlinear, solutions are more difficult.

When nonlinearities are not severe, **local linearization** can be used to derive linear models that are approximations of the nonlinear equations in the neighborhood of an **operating point**. Unfortunately, the manipulator-control problem is not well suited to this approach, because manipulators constantly move among regions of their workspaces so widely separated that no linearization valid for all regions can be found.

Another approach is to move the operating point with the manipulator as it moves, always linearizing about the desired position of the manipulator. The result of this sort of *moving linearization* is a linear, but time-varying, system. Although this quasi-static linearization of the original system is useful in some analysis and design techniques, we will not make use of it in our control-law synthesis procedure. Rather, we will deal with the nonlinear equations of motion directly and will not resort to linearizations in deriving a controller.

If the spring in Fig. 9.6 were not linear but instead contained a nonlinear element, we could consider the system quasi-statically and, at each instant, figure out where the poles of the system are located. We would find that the poles "move" around in the real–imaginary plane as a function of the position of the block. Hence, we could not select fixed gains that would keep the poles in a desirable location (for example, at critical damping). So we may be tempted to consider a more complicated control law, in which the gains are time-varying (actually, varying as a function of the block's position) in such a manner that the system is always critically damped. Essentially, this would be done by computing k_p such that the combination of the nonlinear effect of the spring would be exactly cancelled by a nonlinear term in the control law so that the overall stiffness would stay a constant at all times. Such a control scheme might be called a **linearizing** control law, because it uses a nonlinear control term to "cancel" a nonlinearity in the controlled system, so that the overall closed loop system is linear.

We will now return to our partitioned control law and see that it can perform this linearizing function. In our partitioned control-law scheme, the servo law remains the same as always, but the model-based portion now will contain a model of the nonlinearity. Thus, the model-based portion of the control performs a linearization function. This is best shown in an example.

EXAMPLE 10.1

Consider the nonlinear spring characteristic shown in Fig. 10.1. Rather than the usual linear spring relationship, $f = kx$, this spring is described by $f = qx^3$. If this spring is part of the physical system shown in Fig. 9.6, construct a control law to keep the system critically damped with a stiffness of k_{CL}.

The open-loop equation is

$$m\ddot{x} + b\dot{x} + qx^3 = f. \tag{10.1}$$

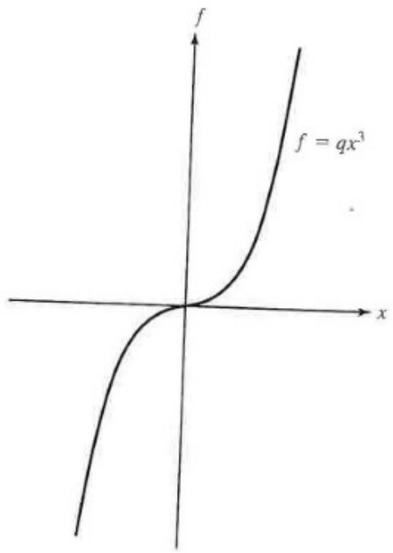


FIGURE 10.1: The force-vs.-distance characteristic of a nonlinear spring.

The model-based portion of the control is $f = \alpha f' + \beta$, where now we use

$$\alpha = m,$$
$$\beta = b\dot{x} + qx^3; \tag{10.2}$$

the servo portion is, as always

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e, \tag{10.3}$$

where the values of the gains are calculated from some desired performance specification. Figure 10.2 shows a block diagram of this control system. The resulting closed-loop system maintains poles in fixed locations.

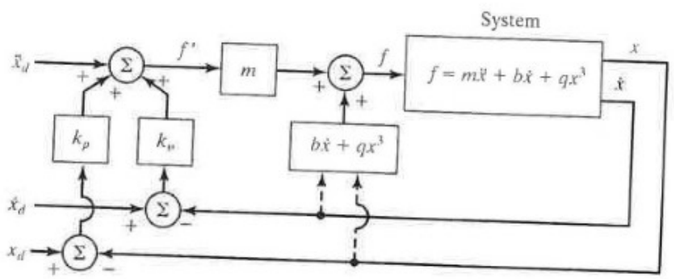


FIGURE 10.2: A nonlinear control system for a system with a nonlinear spring.

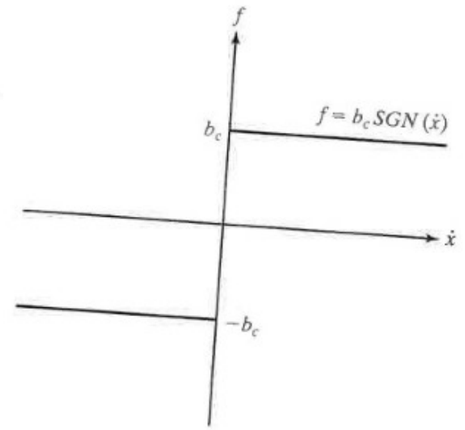


FIGURE 10.3: The force-vs.-velocity characteristic of Coulomb friction.

EXAMPLE 10.2

Consider the nonlinear friction characteristic shown in Fig. 10.3. Whereas linear friction is described by $f = b\dot{x}$, this **Coulomb friction** is described by $f = b_c sgn(\dot{x})$. For most of today's manipulators, the friction of the joint in its bearing (be it rotational or linear) is modeled more accurately by this nonlinear characteristic than by the simpler, linear model. If this type of friction is present in the system of Fig. 9.6, design a control system that uses a nonlinear model-based portion to damp the system critically at all times.

The open-loop equation is

$$m\ddot{x} + b_c sgn(\dot{x}) + kx = f. \tag{10.4}$$

The partitioned control law is $f = \alpha f' + \beta$, where

$$\alpha = m,$$
$$\beta = b_c sgn(\dot{x}) + kx, \tag{10.5}$$
$$f' = \ddot{x}_d + k_v\dot{e} + k_p e,$$

where the values of the gains are calculated from some desired performance specification.

EXAMPLE 10.3

Consider the single-link manipulator shown in Fig. 10.4. It has one rotational joint. The mass is considered to be located at a point at the distal end of the link, and so the moment of inertia is ml^2. There is Coulomb and viscous friction acting at the joint, and there is a load due to gravity.

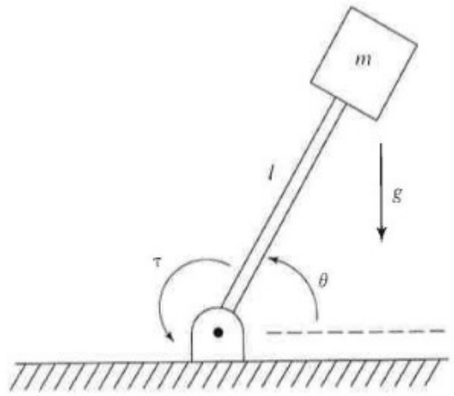


FIGURE 10.4: An inverted pendulum or a one-link manipulator.

The model of the manipulator is

$$\tau = ml^2\ddot{\theta} + \upsilon\dot{\theta} + csgn(\dot{\theta}) + mlg\cos(\theta). \tag{10.6}$$

As always, the control system has two parts, the linearizing model-based portion and the servo-law portion.

The model-based portion of the control is $f = \alpha f' + \beta$, where

$$\alpha = ml^2.$$
$$\beta = \upsilon\dot{\theta} + csgn(\dot{\theta}) + mlg\cos(\theta); \tag{10.7}$$

the servo portion is, as always,

$$f' = \ddot{\theta}_d + k_\upsilon\dot{e} + k_p e, \tag{10.8}$$

where the values of the gains are calculated from some desired performance specification.

We have seen that, in certain simple cases, it is not difficult to design a nonlinear controller. The general method used in the foregoing simple examples is the same method we will use for the problem of manipulator control:

1. Compute a nonlinear model-based control law that "cancels" the nonlinearities of the system to be controlled.
2. Reduce the system to a linear system that can be controlled with the simple linear servo law developed for the unit mass.

In some sense, the linearizing control law implements an *inverse model* of the system being controlled. The nonlinearities in the system cancel those in the inverse model; this, together with the servo law, results in a linear closed-loop system. Obviously, to do this cancelling, we must know the parameters and the structure of the nonlinear system. This is often a problem in practical application of this method.

10.3 MULTI-INPUT, MULTI-OUTPUT CONTROL SYSTEMS

Unlike the simple examples we have discussed in this chapter so far, the problem of controlling a manipulator is a multi-input, multi-output (MIMO) problem. That is, we have a *vector* of desired joint positions, velocities, and accelerations, and the control law must compute a *vector* of joint-actuator signals. Our basic scheme, partitioning the control law into a model-based portion and a servo portion, is still applicable, but it now appears in a matrix–vector form. The control law takes the form

$$F = \alpha F' + \beta, \tag{10.9}$$

where, for a system of n degrees of freedom, F, F', and β are $n \times 1$ vectors and α is an $n \times n$ matrix. Note that the matrix α is not necessarily diagonal, but rather is chosen to **decouple** the n equations of motion. If α and β are correctly chosen, then, from the F' input, the system appears to be n independent unit masses. For this reason, in the multidimensional case, the model-based portion of the control law is called a **linearizing and decoupling** control law. The servo law for a multidimensional system becomes

$$F' = \ddot{X}_d + K_\upsilon\dot{E} + K_p E, \tag{10.10}$$

where K_υ and K_p are now $n \times n$ matrices, which are generally chosen to be diagonal with constant gains on the diagonal. E and $\dot{E}$ are $n \times 1$ vectors of the errors in position and velocity, respectively.

10.4 THE CONTROL PROBLEM FOR MANIPULATORS

In the case of manipulator control, we developed a model and the corresponding equations of motion in Chapter 6. As we saw, these equations are quite complicated. The rigid-body dynamics have the form

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta), \tag{10.11}$$

where $M(\Theta)$ is the $n \times n$ inertia matrix of the manipulator, $V(\Theta, \dot{\Theta})$ is an $n \times 1$ vector of centrifugal and Coriolis terms, and $G(\Theta)$ is an $n \times 1$ vector of gravity terms. Each element of $M(\Theta)$ and $G(\Theta)$ is a complicated function that depends on Θ, the position of all the joints of the manipulator. Each element of $V(\Theta, \dot{\Theta})$ is a complicated function of both Θ and $\dot{\Theta}$.

Additionally, we could incorporate a model of friction (or other non-rigid-body effects). Assuming that our model of friction is a function of joint positions and velocities, we add the term $F(\Theta, \dot{\Theta})$ to (10.11), to yield the model

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}). \tag{10.12}$$

The problem of controlling a complicated system like (10.12) can be handled by the partitioned controller scheme we have introduced in this chapter. In this case, we have

$$\tau = \alpha\tau' + \beta, \tag{10.13}$$

where τ is the $n \times 1$ vector of joint torques. We choose

$$\alpha = M(\Theta),$$
$$\beta = V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}), \tag{10.14}$$

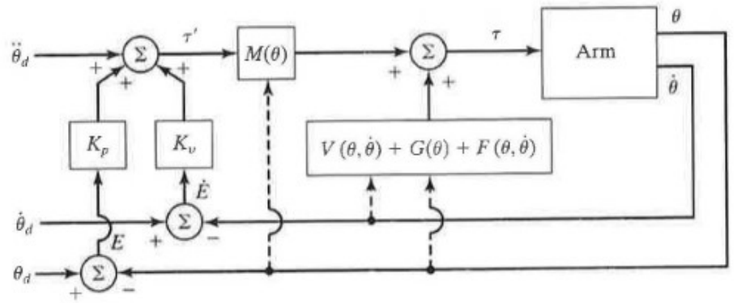


FIGURE 10.5: A model-based manipulator-control system.

with the servo law

$$\tau' = \ddot{\Theta}_d + K_v \dot{E} + K_p E, \tag{10.15}$$

where

$$E = \Theta_d - \Theta. \tag{10.16}$$

The resulting control system is shown in Fig. 10.5.

Using (10.12) through (10.15), it is quite easy to show that the closed-loop system is characterized by the error equation

$$\ddot{E} + K_v \dot{E} + K_p E = 0. \tag{10.17}$$

Note that this vector equation is decoupled: The matrices K_v and K_p are diagonal, so that (10.17) could just as well be written on a joint-by-joint basis as

$$\ddot{e}_i + k_{vi}\dot{e} + k_{pi}e = 0. \tag{10.18}$$

The ideal performance represented by (10.17) is unattainable in practice, for many reasons, the most important two being

1. The discrete nature of a digital-computer implementation, as opposed to the ideal continuous-time control law implied by (10.14) and (10.15).
2. Inaccuracy in the manipulator model (needed to compute (10.14)).

In the next section, we will (at least partially) address these two issues.

10.5 PRACTICAL CONSIDERATIONS

In developing the decoupling and linearizing control in the last few sections, we have implicitly made a few assumptions that rarely are true in practice.

Time required to compute the model

In all our considerations of the partitioned-control-law strategy, we have implicitly assumed that the entire system was running in continuous time and that the computations in the control law require zero time for their computation. Given any amount of computation, with a large enough computer we can do the computations sufficiently

fast that this is a reasonable approximation: however, the expense of the computer could make the scheme economically unfeasible. In the manipulator-control case, the entire dynamic equation of the manipulator, (10.14), must be computed in the control law. These computations are quite involved: consequently, as was discussed in Chapter 6, there has been a great deal of interest in developing fast computational schemes to compute them in an efficient way. As computer power becomes more and more affordable, control laws that require a great deal of computation will become more practical. Several experimental implementations of nonlinear-model-based control laws have been reported [5–9], and partial implementations are beginning to appear in industrial controllers.

As was discussed in Chapter 9, almost all manipulator-control systems are now performed in digital circuitry and are run at a certain **sampling rate**. This means that the position (and possibly other) sensors are read at discrete points in time. From the values read, an actuator command is computed and sent to the actuator. Thus, reading sensors and sending actuator commands are not done continuously, but rather at a finite sampling rate. To analyze the effect of delay due to computation and finite sample rate, we must use tools from the field of **discrete-time control**. In discrete time, differential equations turn into difference equations, and a complete set of tools has been developed to answer questions about stability and pole placement for these systems. Discrete-time control theory is beyond the scope of this book, although, for researchers working in the area of manipulator control, many of the concepts from discrete-time systems are essential. (See [10].)

Although important, ideas and methods from discrete-time control theory are often difficult to apply to the case of nonlinear systems. Whereas we have managed to write a complicated differential equation of motion for the manipulator dynamic equation, a discrete-time equivalent is impossible to obtain in general because, for a general manipulator, the only way to solve for the motion of the manipulator for a given set of initial conditions, an input, and a finite interval is by numerical integration (as we saw in Chapter 6). Discrete-time models are possible if we are willing to use series solutions to the differential equations, or if we make approximations. However, if we need to make approximations to develop a discrete model, then it is not clear whether we have a better model than we have when just using the continuous model and making the continuous-time approximation. Suffice it to say that analysis of the discrete-time manipulator-control problem is difficult, and usually simulation is resorted to in order to judge the effect that a certain sample rate will have on performance.

We will generally assume that the computations can be performed quickly enough and often enough that the continuous-time approximation is valid.

Feedforward nonlinear control

The use of **feedforward control** has been proposed as a method of using a nonlinear dynamic model in a control law without the need for complex and time-consuming computations to be performed at servo rates [11]. In Fig. 10.5, the model-based control portion of the control law is "in the servo loop" in that signals "flow" through that black box with each tick of the servo clock. If we wish to select a sample

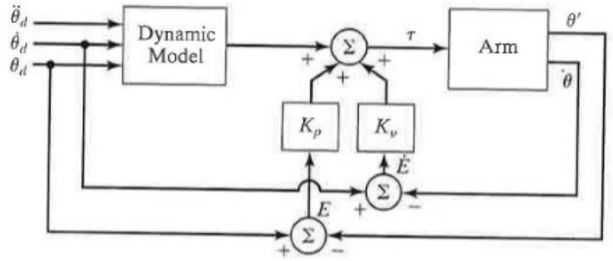


FIGURE 10.6: Control scheme with the model-based portion "outside" the servo loop.

rate of 200 Hz, then the dynamic model of the manipulator must be computed at this rate. Another possible control system is shown in Fig. 10.6. Here, the model-based control is "outside" the servo loop. Hence, it is possible to have a fast inner servo loop, consisting simply of multiplying errors by gains, with the model-based torques added at a slower rate.

Unfortunately, the feedforward scheme of Fig. 10.6 does not provide complete decoupling. If we write the system equations,[1] we will find that the error equation of this system is

$$\ddot{E} + M^{-1}(\Theta)K_v\dot{E} + M^{-1}(\Theta)K_pE = 0. \tag{10.19}$$

Clearly, as the configuration of the arm changes, the effective closed-loop gain changes, and the quasi-static poles move around in the real–imaginary plane. However, equation (10.19) could be used as a starting point for designing a **robust controller**—one that finds a good set of constant gains such that, despite the "motion" of the poles, they are guaranteed to remain in reasonably favorable locations. Alternatively, one might consider schemes in which variable gains are precomputed which change with configuration of the robot, so that the system's quasi-static poles remain in fixed positions.

Note that, in the system of Fig. 10.6, the dynamic model is computed as a function of the desired path only, so when the desired path is known in advance, values could be computed "off-line" before motion begins. At run time, the precomputed torque histories would then be read out of memory. Likewise, if time-varying gains are computed, they too could be computed beforehand and stored. Hence, such a scheme could be quite inexpensive computationally at run time and thus achieve a high servo rate.

Dual-rate computed-torque implementation

Figure 10.7 shows the block diagram of a possible practical implementation of the decoupling and linearizing position-control system. The dynamic model is expressed in its *configuration space* form so that the dynamic parameters of the manipulator will appear as functions of manipulator position only. These functions might then

[1] We have used the simplifying assumptions $M(\Theta_d) \cong M(\Theta)$, $V(\Theta_d, \dot{\Theta}_d) \cong (V(\Theta, \dot{\Theta}), G(\Theta_d) \cong G(\Theta)$, and $F(\Theta_d, \dot{\Theta}_d) \cong F(\Theta, \dot{\Theta})$.

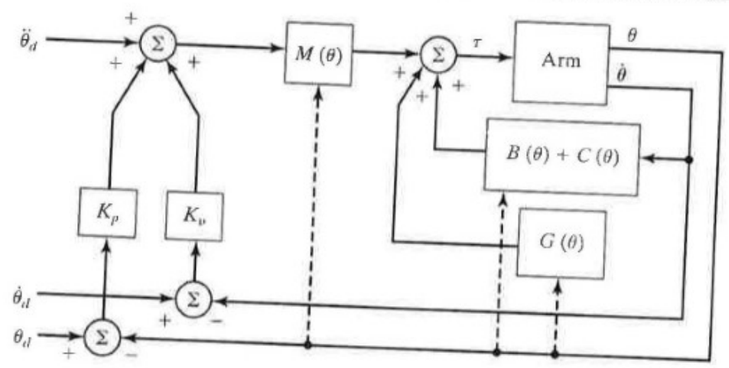


FIGURE 10.7: An implementation of the model-based manipulator-control system.

be computed by a *background* process or by a second control computer [8] or be looked up in a precomputed table [12]. In this architecture, the dynamic parameters can be updated at a rate slower than the rate of the closed-loop servo. For example, the background computation might proceed at 60 Hz while the closed-loop servo was running at 250 Hz.

Lack of knowledge of parameters

The second potential difficulty encountered in employing the computed-torque control algorithm is that the manipulator dynamic model is often not known accurately. This is particularly true of certain components of the dynamics, such as friction effects. In fact, it is usually extremely difficult to know the structure of the friction model, let alone the parameter values [13]. Finally, if the manipulator has some portion of its dynamics that is not repeatable—because, for example, it changes as the robot ages—it is difficult to have good parameter values in the model at all times.

By nature, most robots will be picking up various parts and tools. When a robot is holding a tool, the inertia and the weight of the tool change the dynamics of the manipulator. In an industrial situation, the mass properties of the tools might be known—in this case, they can be accounted for in the modeled portion of the control law. When a tool is grasped, the inertia matrix, total mass, and center of mass of the last link of the manipulator can be updated to new values that represent the combined effect of the last link plus tool. However, in many applications, the mass properties of objects that the manipulator picks up are not generally known, so maintenance of an accurate dynamic model is difficult.

The simplest possible nonideal situation is one in which we still assume a perfect model implemented in continuous time, but with external noise acting to disturb the system. In Fig. 10.8, we indicate a vector of disturbance torques acting at the joints. Writing the system error equation with inclusion of these unknown disturbances, we arrive at

$$\ddot{E} + K_v\dot{E} + K_pE = M^{-1}(\Theta)\tau$$

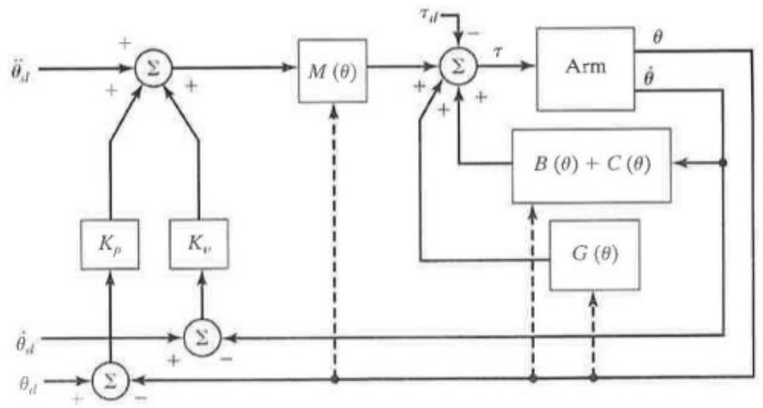


FIGURE 10.8: The model-based controller with an external disturbance acting.

where τ_d is the vector of disturbance torques at the joints. The left-hand side of (10.20) is uncoupled, but, from the right-hand side, we see that a disturbance on any particular joint will introduce errors at all the other joints, because $M(\Theta)$ is not, in general, diagonal.

Some simple analyses might be performed on the basis of (10.20). For example, it is easy to compute the steady-state servo error due to a constant disturbance as

$$E = K_p^{-1} M^{-1}(\Theta)\tau_d. \tag{10.21}$$

When our model of the manipulator dynamics is not perfect, analysis of the resulting closed-loop system becomes more difficult. We define the following notation: $\hat{M}(\Theta)$ is our model of the manipulator inertia matrix, $M(\Theta)$. Likewise, $\hat{V}(\Theta, \dot{\Theta})$, $\hat{G}(\Theta)$, and $\hat{F}(\Theta, \dot{\Theta})$ are our models of the velocity terms, gravity terms, and friction terms of the actual mechanism. Perfect knowledge of the model would mean that

$$\begin{aligned} \hat{M}(\Theta) &= M(\Theta), \\ \hat{V}(\Theta, \dot{\Theta}) &= V(\Theta, \dot{\Theta}), \\ \hat{G}(\Theta) &= G(\Theta), \\ \hat{F}(\Theta, \dot{\Theta}) &= F(\Theta, \dot{\Theta}). \end{aligned} \tag{10.22}$$

Therefore, although the manipulator dynamics are given by

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}), \tag{10.23}$$

our control law computes

$$\begin{aligned} \tau &= \alpha\tau' + \beta, \\ \alpha &= \hat{M}(\Theta), \\ \beta &= \hat{V}(\Theta, \dot{\Theta}) + \hat{G}(\Theta) + \hat{F}(\Theta, \dot{\Theta}). \end{aligned} \tag{10.24}$$

Decoupling and linearizing will not, therefore, be perfectly accomplished when parameters are not known exactly. Writing the closed-loop equation for the system, we have

$$\begin{aligned} &\ddot{E} + K_v\dot{E} + K_p E \\ &\quad = \hat{M}^{-1}[(M - \hat{M})\ddot{\Theta} + (V - \hat{V}) + (G - \hat{G}) + (F - \hat{F})], \end{aligned} \tag{10.25}$$

where the arguments of the dynamic functions are not shown for brevity. Note that, if the model were exact, so that (10.22) were true, then the right-hand side of (10.25) would be zero and the errors would disappear. When the parameters are not known exactly, the mismatch between actual and modeled parameters will cause servo errors to be excited (possibly even resulting in an unstable system [21]) according to the rather complicated equation (10.25).

Discussion of stability analysis of a nonlinear closed-loop system is deferred until Section 10.7.

10.6 CURRENT INDUSTRIAL-ROBOT CONTROL SYSTEMS

Because of the problems with having good knowledge of parameters, it is not clear whether it makes sense to go to the trouble of computing a complicated model-based control law for manipulator control. The expense of the computer power needed to compute the model of the manipulator at a sufficient rate might not be worthwhile, especially when lack of knowledge of parameters could nullify the benefits of such an approach. Manufacturers of industrial robots have decided, probably for economic reasons, that attempting to use a complete manipulator model in the controller is not worthwhile. Instead, present-day manipulators are controlled with very simple control laws that generally are completely error driven and are implemented in architectures such as those studied in Section 9.10. An industrial robot with a high-performance servo system is shown in Fig. 10.9.

Individual-joint PID control

Most industrial robots nowadays have a control scheme that, in our notation, would be described by

$$\begin{aligned} \alpha &= I, \\ \beta &= 0, \end{aligned} \tag{10.26}$$

where I is the $n \times n$ identity matrix. The servo portion is

$$\tau' = \ddot{\Theta}_d + K_v\dot{E} + K_p E + K_i \int E dt, \tag{10.27}$$

where K_v, K_p, and K_i are constant diagonal matrices. In many cases, $\ddot{\Theta}_d$ is not available, and this term is simply set to zero. That is, most simple robot controllers do not use a model-based component *at all* in their control law. This type of PID control scheme is simple because each joint is controlled as a separate control system. Often, one microprocessor per joint is used to implement (10.27), as was discussed in Section 9.10.

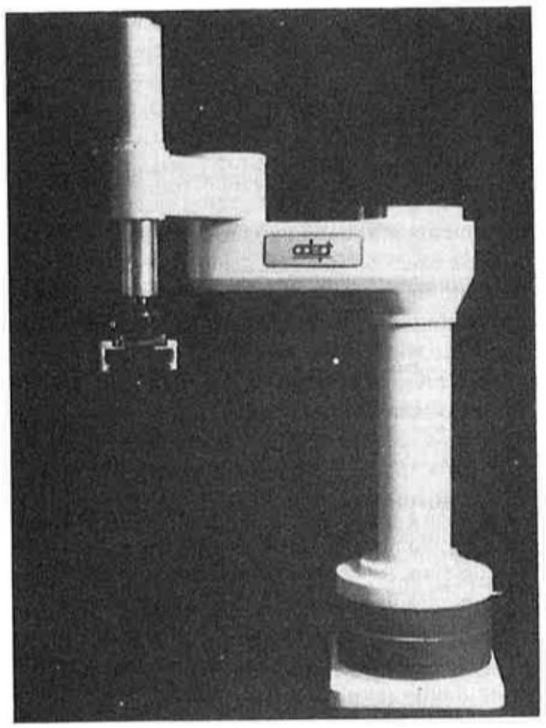


FIGURE 10.9: The Adept One, a direct-drive robot by Adept Technology, Inc.

The performance of a manipulator controlled in this way is not simple to describe. No decoupling is being done, so the motion of each joint affects the other joints. These interactions cause errors, which are suppressed by the error-driven control law. It is impossible to select fixed gains that will critically damp the response to disturbances for all configurations. Therefore, "average" gains are chosen, which approximate critical damping in the center of the robot's workspace. In various extreme configurations of the arm, the system becomes either underdamped or overdamped. Depending on the details of the mechanical design of the robot, these effects could be fairly small; then control would be good. In such systems, it is important to keep the gains as high as possible, so that the inevitable disturbances will be suppressed quickly.

Addition of gravity compensation

The gravity terms will tend to cause static positioning errors, so some robot manufacturers include a gravity model, $G(\theta)$, in the control law (that is, $\beta = \hat{G}(\Theta)$ in our notation). The complete control law takes the form

$$\tau' = \ddot{\Theta}_d + K_v \dot{E} + K_p E + K_i \int E dt + \hat{G}(\Theta). \tag{10.28}$$

Such a control law is perhaps the simplest example of a model-based controller. Because (10.28) can no longer be implemented on a strict joint-by-joint basis, the controller architecture must allow communication between the joint controllers or must make use of a central processor rather than individual-joint processors.

Various approximations of decoupling control

There are various ways to simplify the dynamic equations of a particular manipulator [3,14]. After the simplification, an approximate decoupling and linearizing law can be derived. A usual simplification might be to disregard components of torque due to the velocity terms—that is, to model only the inertial and gravity terms. Often, friction models are not included in the controller, because friction is so hard to model correctly. Sometimes, the inertia matrix is simplified so that it accounts for the major coupling between axes but not for minor cross-coupling effects. For example, [14] presents a simplified version of the PUMA 560's mass matrix that requires only about 10% of the calculations needed to compute the complete mass matrix, yet is accurate to within 1%.

10.7 LYAPUNOV STABILITY ANALYSIS

In Chapter 9, we examined linear control systems analytically to evaluate stability and also performance of the dynamic response in terms of damping and closed-loop bandwidth. The same analyses are valid for a nonlinear system that has been decoupled and linearized by means of a perfect model-based nonlinear controller, because the overall resulting system is again linear. However, when decoupling and linearizing are not performed by the controller, or are incomplete or inaccurate, the overall closed-loop system remains nonlinear. For nonlinear systems, stability and performance analysis is much more difficult. In this section, we introduce one method of stability analysis that is applicable to both linear and nonlinear systems.

Consider the simple mass–spring friction system originally introduced in Chapter 9, whose equation of motion is

$$m\ddot{x} + b\dot{x} + kx = 0. \tag{10.29}$$

The total energy of the system is given by

$$v = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2. \tag{10.30}$$

where the first term gives the kinetic energy of the mass and the second term gives the potential energy stored in the spring. Note that the value, v, of the system energy is always nonnegative (i.e., it is positive or zero). Let's find out the *rate* of change of the total energy by differentiating (10.30) with respect to time, to obtain

$$\dot{v} = m\dot{x}\ddot{x} + kx\dot{x}. \tag{10.31}$$

Substituting (10.29) for $m\ddot{x}$ in (10.31) yields

$$\dot{v} = -b\dot{x}^2, \tag{10.32}$$

which we note is always nonpositive (because $b > 0$). Thus, energy is always leaving the system, unless $\dot{x} = 0$. This implies that, however initially perturbed, the system

will lose energy until it comes to rest. Investigating possible resting positions by means of a steady-state analysis of (10.29) yields

$$kx = 0. \tag{10.33}$$

or

$$x = 0. \tag{10.34}$$

Hence, by means of an energy analysis, we have shown that the system of (10.29) with any initial conditions (i.e., any initial energy) will eventually come to rest at the equilibrium point. This stability proof by means of an energy analysis is a simple example of a more general technique called **Lyapunov stability analysis** or **Lyapunov's second** (or **direct**) **method**, after a Russian mathematician of the 19th century [15].

An interesting feature of this method of stability analysis is that we can conclude stability without solving for the solution of the differential equation governing the system. However, while Lyapunov's method is useful for examining *stability*, it generally does not provide any information about the transient response or *performance* of the system. Note that our energy analysis yielded no information on whether the system was overdamped or underdamped or on how long it would take the system to suppress a disturbance. It is important to distinguish between stability and performance: A stable system might nonetheless exhibit control performance unsatisfactory for its intended use.

Lyapunov's method is somewhat more general than our example indicated. It is one of the few techniques that can be applied directly to nonlinear systems to investigate their stability. As a means of quickly getting an idea of Lyapunov's method (in sufficient detail for our needs), we will look at an extremely brief introduction to the theory and then proceed directly to several examples. A more complete treatment of Lyapunov theory can be found in [16, 17].

Lyapunov's method is concerned with determining the stability of a differential equation

$$\dot{X} = f(X). \tag{10.35}$$

where X is $m \times 1$ and $f(\cdot)$ could be nonlinear. Note that higher order differential equations can always be written as a set of first-order equations in the form (10.35). To prove a system stable by Lyapunov's method, one is required to propose a generalized energy function $v(X)$ that has the following properties:

1. $v(X)$ has continuous first partial derivatives, and $v(X) > 0$ for all X except $v(0) = 0$.
2. $\dot{v}(X) \leq 0$. Here, $\dot{v}(X)$ means the change in $v(X)$ along all system trajectories.

These properties might hold only in a certain region, or they might be global, with correspondingly weaker or stronger stability results. The intuitive idea is that a positive definite "energy-like" function of state is shown to always decrease or remain constant—hence, the system is stable in the sense that the size of the state vector is bounded.

When $\dot{v}(X)$ is strictly less than zero, asymptotic convergence of the state to the zero vector can be concluded. Lyapunov's original work was extended in an important way by LaSalle and Lefschetz [4], who showed that, in certain situations, even when $\dot{v}(X) \leq 0$ (note equality included), asymptotic stability can be shown. For our purposes, we can deal with the case $\dot{v}(X) = 0$ by performing a steady-state analysis in order to learn whether the stability is asymptotic or the system under study can "get stuck" somewhere other than $v(X) = 0$.

A system described by (10.35) is said to be **autonomous** because the function $f(\cdot)$ is not an explicit function of time. Lyapunov's method also extends to **nonautonomous** systems, in which time is an argument of the nonlinear function. See [4, 17] for details.

EXAMPLE 10.4

Consider the linear system

$$\dot{X} = -AX, \tag{10.36}$$

where A is $m \times m$ and positive definite. Propose the **candidate Lyapunov function**

$$v(X) = \frac{1}{2}X^T X, \tag{10.37}$$

which is continuous and everywhere nonnegative. Differentiating yields

$$\begin{aligned} \dot{v}(X) &= X^T \dot{X} \\ &= X^T(-AX) \\ &= -X^T AX, \end{aligned} \tag{10.38}$$

which is everywhere nonpositive because A is a positive definite matrix. Hence, (10.37) is indeed a Lyapunov function for the system of (10.36). The system is asymptotically stable because $\dot{v}(X)$ can be zero only at $X = 0$; everywhere else, X must decrease.

EXAMPLE 10.5

Consider a mechanical spring–damper system in which both the spring and damper are nonlinear:

$$\ddot{x} + b(\dot{x}) + k(x) = 0. \tag{10.39}$$

The functions $b(\cdot)$ and $k(\cdot)$ are first- and third-quadrant continuous functions such that

$$\begin{aligned} \dot{x}b(\dot{x}) &> 0 \text{ for } \dot{x} \neq 0, \\ xk(x) &> 0 \text{ for } x \neq 0. \end{aligned} \tag{10.40}$$

Once having proposed the Lyapunov function

$$v(x, \dot{x}) = \frac{1}{2}\dot{x}^2 + \int_0^x k(\lambda)d\lambda, \tag{10.41}$$

we are led to

$$\begin{aligned}\dot{v}(x, \dot{x}) &= \dot{x}\ddot{x} + k(x)\dot{x}, \\ &= -\dot{x}b(\dot{x}) - k(x)\dot{x} + k(x)\dot{x}, \\ &= -\dot{x}b(\dot{x}).\end{aligned} \tag{10.42}$$

Hence, $\dot{v}(\cdot)$ is nonpositive but is only semidefinite, because it is not a function of x but only of $\dot{x}$. In order to conclude asymptotic stability, we have to ensure that it is not possible for the system to "get stuck" with nonzero x. To study all trajectories for which $\dot{x} = 0$, we must consider

$$\ddot{x} = -k(x), \tag{10.43}$$

for which $x = 0$ is the only solution. Hence, the system will come to rest only if $x = \dot{x} = \ddot{x} = 0$.

EXAMPLE 10.6

Consider a manipulator with dynamics given by

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \tag{10.44}$$

and controlled with the control law

$$\tau = K_p E - K_d \dot{\Theta} + G(\Theta), \tag{10.45}$$

where K_p and K_d are diagonal gain matrices. Note that this controller does not force the manipulator to follow a trajectory, but moves the manipulator to a goal point along a path specified by the manipulator dynamics and then regulates the position there. The resulting closed-loop system obtained by equating (10.44) and (10.45) is

$$M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + K_d\dot{\Theta} + K_p\Theta = K_p\Theta_d; \tag{10.46}$$

it can be proven globally asymptotically stable by Lyapunov's method [18, 19].

Consider the candidate Lyapunov function

$$v = \frac{1}{2}\dot{\Theta}^T M(\Theta)\dot{\Theta} + \frac{1}{2}E^T K_p E. \tag{10.47}$$

The function (10.47) is always positive or zero, because the manipulator mass matrix, $M(\Theta)$, and the position gain matrix, K_p, are positive definite matrices. Differentiating (10.47) yields

$$\begin{aligned}\dot{v} &= \frac{1}{2}\dot{\Theta}^T \dot{M}(\Theta)\dot{\theta} + \dot{\theta}^T M(\theta)\ddot{\Theta} - E^T K_p \dot{\Theta} \\ &= \frac{1}{2}\dot{\Theta}^T \dot{M}(\Theta)\dot{\Theta} - \dot{\Theta}^T K_d \dot{\Theta} - \dot{\Theta}^T V(\Theta, \dot{\Theta}) \\ &= -\dot{\Theta}^T K_d \dot{\Theta},\end{aligned} \tag{10.48}$$

which is nonpositive as long as K_d is positive definite. In taking the last step in (10.48), we have made use of the interesting identity

$$\frac{1}{2}\dot{\Theta}^T \dot{M}(\Theta)\dot{\Theta} = \dot{\Theta}^T V(\Theta, \dot{\Theta}), \tag{10.49}$$

which can be shown by investigation of the structure of Lagrange's equations of motion [18–20]. (See also Exercise 6.17.)

Next, we investigate whether the system can get "stuck" with nonzero error. Because $\dot{v}$ can remain zero only along trajectories that have $\dot{\Theta} = 0$ and $\ddot{\Theta} = 0$, we see from (10.46) that, in this case,

$$K_p E = 0, \tag{10.50}$$

and because K_p is nonsingular, we have

$$E = 0. \tag{10.51}$$

Hence, control law (10.45) applied to the system (10.44) achieves global asymptotic stability.

This proof is important in that it explains, to some extent, why today's industrial robots work. Most industrial robots use a simple error-driven servo, occasionally with gravity models, and so are quite similar to (10.45).

See Exercises 10.11 through 10.16 for more examples of nonlinear manipulator-control laws that can be proven stable by Lyapunov's method. Recently, Lyapunov theory has become increasingly prevalent in robotics research publications [18–25].

10.8 CARTESIAN-BASED CONTROL SYSTEMS

In this section, we introduce the notion of **Cartesian-based control**. Although such approaches are not currently used in industrial robots, there is activity at several research institutions on such schemes.

Comparison with joint-based schemes

In all the control schemes for manipulators we have discussed so far, we assumed that the desired trajectory was available in terms of time histories of joint position, velocity, and acceleration. Given that these desired inputs were available, we designed **joint-based control** schemes, that is, schemes in which we develop trajectory errors by finding the difference between desired and actual quantities expressed in joint space. Very often, we wish the manipulator end-effector to follow straight lines or other path shapes described in Cartesian coordinates. As we saw in Chapter 7, it is possible to compute the time histories of the joint-space trajectory that correspond to Cartesian straight-line paths. Figure 10.10 shows this approach to manipulator-trajectory control. A basic feature of the approach is the **trajectory-conversion** process, which is used to compute the joint trajectories. This is then followed by some kind of joint-based servo scheme such as we have been studying.

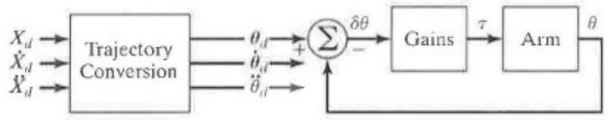


FIGURE 10.10: A joint-based control scheme with Cartesian-path input.

The trajectory-conversion process is quite difficult (in terms of computational expense) if it is to be done analytically. The computations that would be required are

$$\begin{aligned} \Theta_d &= INVKIN(\chi_d), \\ \dot{\Theta}_d &= J^{-1}(\Theta)\dot{\chi}_d, \\ \ddot{\Theta}_d &= \dot{J}^{-1}(\Theta)\dot{\chi}_d + J^{-1}(\Theta)\ddot{\chi}_d. \end{aligned} \tag{10.52}$$

To the extent that such a computation is done at all in present-day systems, usually just the solution for Θ_d is performed, by using the inverse kinematics, and then the joint velocities and accelerations are computed numerically by first and second differences. However, such numerical differentiation tends to amplify noise and introduces a lag unless it can be done with a noncausal filter.[2] Therefore, we are interested in either finding a less computationally expensive way of computing (10.52) or suggesting a control scheme in which this information is not needed.

An alternative approach is shown in Fig. 10.11. Here, the sensed position of the manipulator is immediately transformed by means of the kinematic equations into a Cartesian description of position. This Cartesian description is then compared to the desired Cartesian position in order to form errors in Cartesian space. Control schemes based on forming errors in Cartesian space are called **Cartesian-based control** schemes. For simplicity, velocity feedback is not shown in Fig. 10.11, but it would be present in any implementation.

The trajectory-conversion process is replaced by some kind of coordinate conversion inside the servo loop. Note that Cartesian-based controllers must perform many computations in the loop; the kinematics and other transformations are now "inside the loop." This can be a drawback of the Cartesian-based methods; the resulting system could run at a lower sampling frequency compared to joint-based

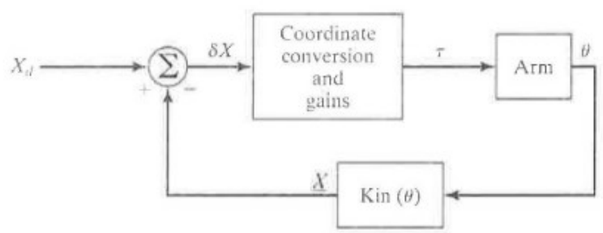


FIGURE 10.11: The concept of a Cartesian-based control scheme.

[2]Numerical differentiation introduces a lag unless it can be based on past, present, and future values. When the entire path is preplanned, this kind of noncausal numerical differentiation can be done.

systems (given the same size of computer). This would, in general, degrade the stability and disturbance-rejection capabilities of the system.

Intuitive schemes of Cartesian control

One possible control scheme that comes to mind rather intuitively is shown in Fig. 10.12. Here, Cartesian position is compared to the desired position to form an error, δX, in Cartesian space. This error, which may be presumed small if the control system is doing its job, may be mapped into a small displacement in joint space by means of the inverse Jacobian. The resulting errors in joint space, $\delta\theta$, are then multiplied by gains to compute torques that will tend to reduce these errors. Note that Fig. 10.12 shows a simplified controller in which, for clarity, the velocity feedback has not been shown. It could be added in a straightforward manner. We will call this scheme the **inverse-Jacobian controller.**

Another scheme which could come to mind is shown in Fig. 10.13. Here, the Cartesian error vector is multiplied by a gain to compute a Cartesian force vector. This can be thought of as a Cartesian force which, if applied to the end-effector of the robot, would push the end-effector in a direction that would tend to reduce the Cartesian error. This Cartesian force vector (actually a force–moment vector) is then mapped through the Jacobian transpose in order to compute the equivalent joint torques that would tend to reduce the observed errors. We will call this scheme the **transpose-Jacobian controller.**

The inverse-Jacobian controller and the transpose-Jacobian controller have both been arrived at intuitively. We cannot be sure that such arrangements would be stable, let alone perform well. It is also curious that the schemes are extremely similar, except that the one contains the Jacobian's inverse, the other its transpose. Remember, the inverse is not equal to the transpose in general (only in the case of a strictly Cartesian manipulator does $J^T = J^{-1}$). The exact dynamic performance

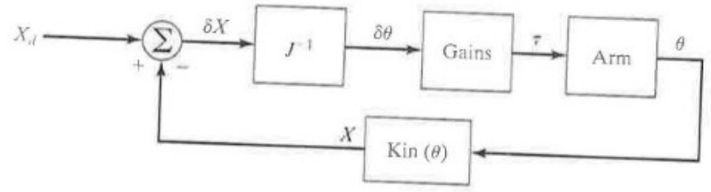


FIGURE 10.12: The inverse-Jacobian Cartesian-control scheme.

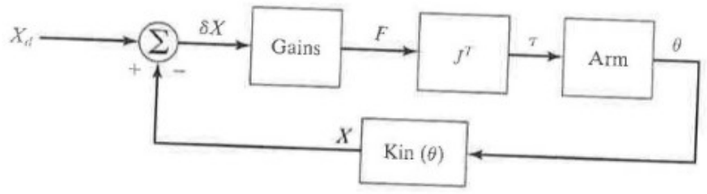


FIGURE 10.13: The transpose-Jacobian Cartesian-control scheme.

of such systems (if expressed in a second-order error-space equation, for example) is very complicated. It turns out that both schemes will work (i.e., can be made stable), but not well (i.e., performance is not good over the entire workspace). Both can be made stable by appropriate gain selection, including some form of velocity feedback (which was not shown in Figs. 10.12 and 10.13). While both will work, neither is *correct*, in the sense that we cannot choose fixed gains that will result in fixed closed-loop poles. The dynamic response of such controllers will vary with arm configuration.

Cartesian decoupling scheme

For Cartesian-based controllers, like joint-based controllers, good performance would be characterized by constant error dynamics over all configurations of the manipulator. Errors are expressed in Cartesian space in Cartesian-based schemes, so this means that we would like to design a system which, over all possible configurations, would suppress Cartesian errors in a critically damped fashion.

Just as we achieved good control with a joint-based controller that was based on a linearizing and decoupling model of the arm, we can do the same for the Cartesian case. However, we must now write the dynamic equations of motion of the manipulator in terms of Cartesian variables. This can be done, as was discussed in Chapter 6. The resulting form of the equations of motion is quite analogous to the joint-space version. The rigid-body dynamics can be written as

$$\mathcal{F} = M_x(\Theta)\ddot{\chi} + V_x(\Theta, \dot{\Theta}) + G_x(\Theta). \tag{10.53}$$

where $\mathcal{F}$ is a fictitious force–moment vector acting on the end-effector of the robot and χ is an appropriate Cartesian vector representing position and orientation of the end-effector [8]. Analogous to the joint-space quantities, $M_x(\Theta)$ is the mass matrix in Cartesian space, $V_x(\Theta, \dot{\Theta})$ is a vector of velocity terms in Cartesian space, and $G_x(\Theta)$ is a vector of gravity terms in Cartesian space.

Just as we did in the joint-based case, we can use the dynamic equations in a decoupling and linearizing controller. Because (10.53) computes $\mathcal{F}$, a fictitious Cartesian force vector which should be applied to the hand, we will also need to use the transpose of the Jacobian in order to implement the control—that is, after $\mathcal{F}$ is calculated by (10.53), we cannot actually cause a Cartesian force to be applied to the end-effector; we instead compute the joint torques needed to effectively balance the system if we were to apply this force:

$$\tau = J^T(\Theta)\mathcal{F}. \tag{10.54}$$

Figure 10.14 shows a Cartesian arm-control system using complete dynamic decoupling. Note that the arm is preceded by the Jacobian transpose. Notice that the controller of Fig. 10.14 allows Cartesian paths to be described directly, with no need for trajectory conversion.

As in the joint-space case, a practical implementation might best be achieved through use of a dual-rate control system. Figure 10.15 shows a block diagram of a Cartesian-based decoupling and linearizing controller in which the dynamic parameters are written as functions of manipulator position only. These dynamic parameters are updated at a rate slower than the servo rate by a background

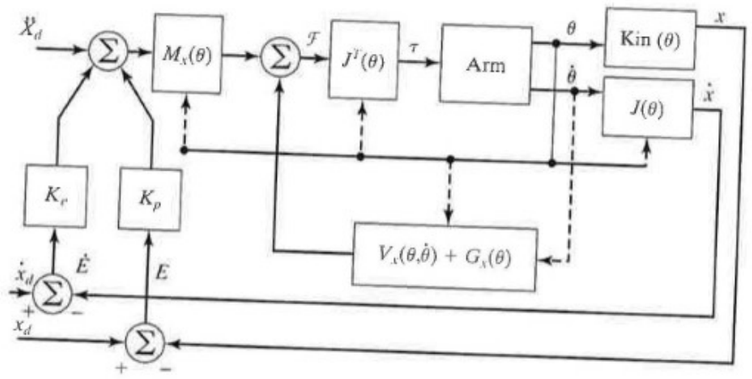


FIGURE 10.14: The Cartesian model-based control scheme.

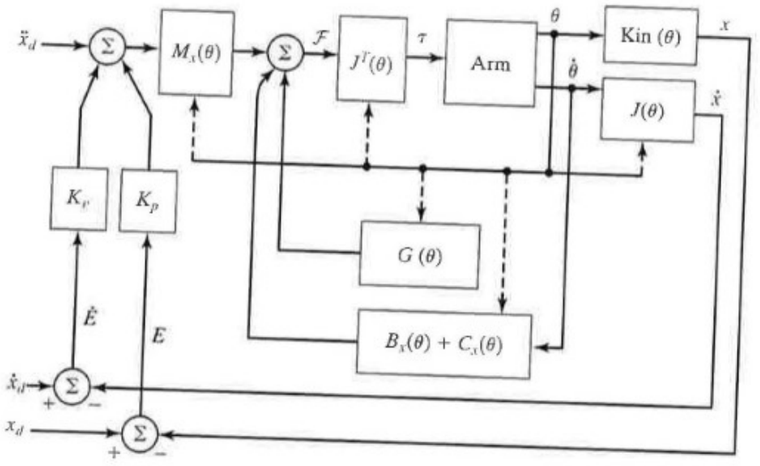


FIGURE 10.15: An implementation of the Cartesian model-based control scheme.

process or a second control computer. This is appropriate, because we desire a fast servo (perhaps running at 500 Hz or even higher) to maximize disturbance rejection and stability. The dynamic parameters are functions of manipulator position only, so they need be updated at a rate related only to how fast the manipulator is changing configuration. The parameter-update rate probably need not be higher than 100 Hz [8].

10.9 ADAPTIVE CONTROL

In the discussion of model-based control, it was noted that, often, parameters of the manipulator are not known exactly. When the parameters in the model do not

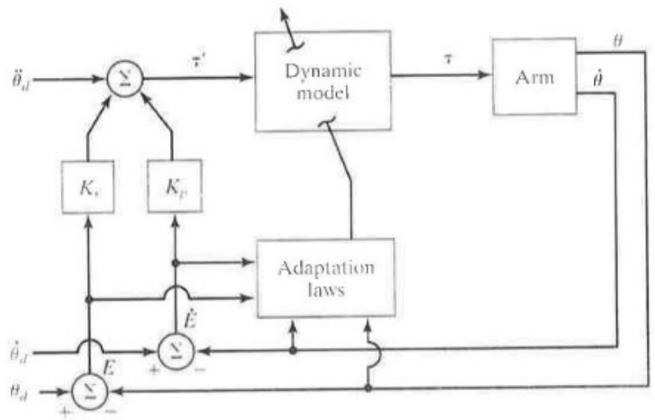


FIGURE 10.16: The concept of an adaptive manipulator controller.

match the parameters of the real device, servo errors will result, as is made explicit in (10.25). These servo errors could be used to drive some adaptation scheme that attempts to update the values of the model parameters until the errors disappear. Several such adaptive schemes have been proposed.

An ideal adaptive scheme might be like the one in Fig. 10.16. Here, we are using a model-based control law as developed in this chapter. There is an adaptation process that, given observations of manipulator state and servo errors, readjusts the parameters in the nonlinear model until the errors disappear. Such a system would *learn* its own dynamic properties. The design and analysis of adaptive schemes are beyond the scope of this book. A method that possesses exactly the structure shown in Fig. 10.16 and has been proven globally stable is presented in [20, 21]. A related technique is that of [22].

BIBLIOGRAPHY

[1] R.P. Paul, "Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm," Technical Report AIM-177, Stanford University Artificial Intelligence Laboratory, 1972.

[2] B. Markiewicz, "Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator," Jet Propulsion Laboratory Technical Memo 33-601, March 1973.

[3] A. Bejczy, "Robot Arm Dynamics and Control," Jet Propulsion Laboratory Technical Memo 33-669, February 1974.

[4] J. LaSalle and S. Lefschetz, *Stability by Liapunov's Direct Method with Applications*, Academic Press, New York, 1961.

[5] P.K. Khosla, "Some Experimental Results on Model-Based Control Schemes," IEEE Conference on Robotics and Automation, Philadelphia, April 1988.

[6] M. Leahy, K. Valavanis, and G. Saridis, "The Effects of Dynamic Models on Robot Control," IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[7] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd Edition, Springer-Verlag, London, 2000.

[8] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 1, 1987.

[9] C. An, C. Atkeson, and J. Hollerbach, "Model-Based Control of a Direct Drive Arm, Part II: Control," IEEE Conference on Robotics and Automation, Philadelphia, April 1988.

[10] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 2nd edition, Addison-Wesley, Reading, MA, 1989.

[11] A. Liegeois, A. Fournier, and M. Aldon, "Model Reference Control of High Velocity Industrial Robots," *Proceedings of the Joint Automatic Control Conference*, San Francisco, 1980.

[12] M. Raibert, "Mechanical Arm Control Using a State Space Memory," SME paper MS77-750, 1977.

[13] B. Armstrong, "Friction: Experimental Determination, Modeling and Compensation," IEEE Conference on Robotics and Automation, Philadelphia, April 1988.

[14] B. Armstrong, O. Khatib, and J. Burdick, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm," IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[15] A.M. Lyapunov, "On the General Problem of Stability of Motion," (in Russian), Kharkov Mathematical Society, Soviet Union, 1892.

[16] C. Desoer and M. Vidyasagar, *Feedback Systems: Input–Output Properties*, Academic Press, New York, 1975.

[17] M. Vidyasagar, *Nonlinear Systems Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[18] S. Arimoto and F. Miyazaki, "Stability and Robustness of PID Feedback Control for Robot Manipulators of Sensory Capability," Third International Symposium of Robotics Research, Gouvieux, France, July 1985.

[19] D. Koditschek, "Adaptive Strategies for the Control of Natural Motion," *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, December 1985.

[20] J. Craig, P. Hsu, and S. Sastry, "Adaptive Control of Mechanical Manipulators," IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[21] J. Craig, *Adaptive Control of Mechanical Manipulators*, Addison-Wesley, Reading, MA, 1988.

[22] J.J. Slotine and W. Li, "On the Adaptive Control of Mechanical Manipulators," *The International Journal of Robotics Research*, Vol. 6, No. 3, 1987.

[23] R. Kelly and R. Ortega, "Adaptive Control of Robot Manipulators: An Input–Output Approach," IEEE Conference on Robotics and Automation, Philadelphia, 1988.

[24] H. Das, J.J. Slotine, and T. Sheridan, "Inverse Kinematic Algorithms for Redundant Systems," IEEE Conference on Robotics and Automation, Philadelphia, 1988.

[25] T. Yabuta, A. Chona, and G. Beni, "On the Asymptotic Stability of the Hybrid Position/Force Control Scheme for Robot Manipulators," IEEE Conference on Robotics and Automation, Philadelphia, 1988.

EXERCISES

10.1 [15] Give the nonlinear control equations for an α,β-partitioned controller for the system

$$\tau = (2\sqrt{\theta} + 1)\ddot{\theta} + 3\dot{\theta}^2 - \sin(\theta).$$

Choose gains so that this system is always critically damped with $k_{CL} = 10$.

10.2 [15] Give the nonlinear control equations for an α,β-partitioned controller for the system

$$\tau = 5\theta\dot{\theta} + 2\ddot{\theta} - 13\dot{\theta}^3 + 5.$$

Choose gains so that this system is always critically damped with $k_{CL} = 10$.

10.3 [19] Draw a block diagram showing a joint-space controller for the two-link arm from Section 6.7, such that the arm is critically damped over its entire workspace. Show the equations inside the blocks of a block diagram.

10.4 [20] Draw a block diagram showing a Cartesian-space controller for the two-link arm from Section 6.7, such that the arm is critically damped over its entire workspace. (See Example 6.6.) Show the equations inside the blocks of a block diagram.

10.5 [18] Design a trajectory-following control system for the system whose dynamics are given by

$$\tau_1 = m_1 l_1^2 \ddot{\theta}_1 + m_1 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2,$$

$$\tau_2 = m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) + \upsilon_2 \dot{\theta}_2.$$

Do you think these equations could represent a real system?

10.6 [17] For the control system designed for the one-link manipulator in Example 10.3, give an expression for the steady-state position error as a function of error in the mass parameter. Let $\psi_m = m - \hat{m}$. The result should be a function of $l, g, \theta, \psi_m, \hat{m}$, and k_p. For what position of the manipulator is this at a maximum?

10.7 [26] For the two-degree-of-freedom mechanical system of Fig. 10.17, design a controller that can cause x_1 and x_2 to follow trajectories and suppress disturbances in a critically damped fashion.

10.8 [30] Consider the dynamic equations of the two-link manipulator from Section 6.7 in configuration-space form. Derive expressions for the sensitivity of the computed torque value versus small deviations in Θ. Can you say something about how often the dynamics should be recomputed in a controller like that of Fig. 10.7 as a function of average joint velocities expected during normal operations?

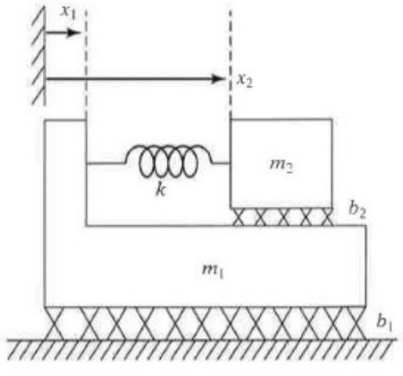


FIGURE 10.17: Mechanical system with two degrees of freedom.

10.9 [32] Consider the dynamic equations of the two-link manipulator from Example 6.6 in Cartesian configuration-space form. Derive expressions for the sensitivity of the computed torque value versus small deviations in Θ. Can you say something about how often the dynamics should be recomputed in a controller like that of Fig. 10.15 as a function of average joint velocities expected during normal operations?

10.10 [15] Design a control system for the system

$$f = 5x\dot{x} + 2\ddot{x} - 12.$$

Choose gains so that this system is always critically damped with a closed-loop stiffness of 20.

10.11 [20] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -K_p\Theta - M(\Theta)K_v\dot{\Theta} + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$ where k_v is a scalar and I_n is the $n \times n$ identity matrix. *Hint*: This is similar to example 10.6.

10.12 [20] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -K_p\Theta - \hat{M}(\Theta)K_v\dot{\Theta} + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$ where k_v is a scalar and I_n is the $n \times n$ identity matrix. The matrix $\hat{M}(\Theta)$ is a positive definite estimate of the manipulator mass matrix. *Hint*: This is similar to example 10.6.

10.13 [25] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -M(\Theta)[K_p\Theta + K_v\dot{\Theta}] + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$ where k_v is a scalar and I_n is the $n \times n$ identity matrix. *Hint*: This is similar to example 10.6.

10.14 [25] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -\hat{M}(\Theta)[K_p\Theta + K_v\dot{\Theta}] + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$, where k_v is a scalar and I_n is the $n \times n$ identity matrix. The matrix $\hat{M}(\Theta)$ is a positive definite estimate of the manipulator mass matrix. *Hint*: This is similar to example 10.6.

10.15 [28] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -K_p\Theta - K_v\dot{\Theta}$$

yields a stable nonlinear system. Show that stability is not asymptotic and give an expression for the steady-state error. *Hint*: This is similar to Example 10.6.

10.16 [30] Prove the global stability of the Jacobian-transpose Cartesian controller introduced in Section 10.8. Use an appropriate form of velocity feedback to stabilize the system. *Hint*: See [18].

10.17 [15] Design a trajectory-following controller for a system with dynamics given by

$$f = ax^2\dot{x}\ddot{x} + b\dot{x}^2 + c\sin(x),$$

such that errors are suppressed in a critically damped fashion over all configurations.

10.18 [15] A system with open-loop dynamics given by

$$\tau = m\ddot{\theta} + b\dot{\theta}^2 + c\dot{\theta}$$

is controlled with the control law

$$\tau = m[\ddot{\theta}_d + k_v\dot{e} + k_p e] + \sin(\theta).$$

Give the differential equation that characterizes the closed-loop action of the system.

PROGRAMMING EXERCISE (PART 10)

Repeat Programming Exercise Part 9, and use the same tests, but with a new controller that uses a complete dynamic model of the 3-link to decouple and linearize the system. For this case, use

$$K_p = \begin{bmatrix} 100.0 & 0.0 & 0.0 \\ 0.0 & 100.0 & 0.0 \\ 0.0 & 0.0 & 100.0 \end{bmatrix}.$$

Choose a diagonal K_v that guarantees critical damping over all configurations of the arm. Compare the results with those obtained with the simpler controller used in Programming Exercise Part 9.

CHAPTER 11

Force control of manipulators

11.1 INTRODUCTION

Position control is appropriate when a manipulator is following a trajectory through space, but when any contact is made between the end-effector and the manipulator's environment, mere position control might not suffice. Consider a manipulator washing a window with a sponge. The compliance of the sponge might make it possible to regulate the force applied to the window by controlling the position of the end-effector relative to the glass. If the sponge is very compliant or the position of the glass is known very accurately, this technique could work quite well.

If, however, the stiffness of the end-effector, tool, or environment is high, it becomes increasingly difficult to perform operations in which the manipulator presses against a surface. Instead of washing with a sponge, imagine that the manipulator is scraping paint off a glass surface, using a rigid scraping tool. If there is any uncertainty in the position of the glass surface or any error in the position of the manipulator, this task would become impossible. Either the glass would be broken, or the manipulator would wave the scraping tool over the glass with no contact taking place.

In both the washing and scraping tasks, it would be more reasonable not to specify the position of the plane of the glass, but rather *to specify a force that is to be maintained normal to the surface.*

More so than in previous chapters, in this chapter we present methods that are not yet employed by industrial robots, except in an extremely simplified way. The major thrust of the chapter is to introduce the **hybrid position/force controller**, which is one formalism through which industrial robots might someday be controlled in order to perform tasks requiring force control. However, regardless of which method(s) emerge as practical for industrial application, many of the concepts introduced in this chapter will certainly remain valid.