Robot Dynamics and Control

Second Edition

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

January 28, 2004

Contents

Chapter 1

INTRODUCTION

1.1 Robotics

Robotics is a relatively young field of modern technology that crosses traditional engineering boundaries. Understanding the complexity of robots and their applications requires knowledge of electrical engineering, mechanical engineering, systems and industrial engineering, computer science, economics, and mathematics. New disciplines of engineering, such as manufacturing engineering, applications engineering, and knowledge engineering have emerged to deal with the complexity of the field of robotics and factory automation.

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

The science of robotics has grown tremendously over the past twenty years, fueled by rapid advances in computer and sensor technology as well as theoretical advances in control and computer vision. In addition to the topics listed above, robotics encompasses several areas not covered in this text such as locomotion, including wheeled and legged robots, flying and swimming robots, grasping, artificial intelligence, computer architectures, programming languages, and computer-aided design. A complete treatment of the discipline of robotics would require several volumes. Nevertheless, at the present time, the vast majority of robot applications deal with industrial robot arms operating in structured factory environments so that a first introduction to the subject of robotics must include a rigorous treatment of the topics in this text.

1.2 History of Robotics

The term **robot** was first introduced into our vocabulary by the Czech playwright Karel Capek in his 1920 play *Rossum's Universal Robots*, the word *robota* being the Czech word for work. Since then the term has been applied to a great variety of mechanical devices, such as teleoperators, underwater vehicles, autonomous land rovers, etc. Virtually anything that

operates with some degree of autonomy, usually under computer control, has at some point been called a robot. In this text the term robot will mean a computer controlled industrial manipulator of the type shown in Figure 1.1. This type of robot is essentially a mechanical arm operating under computer control. Such devices, though far from the robots of science fiction, are nevertheless extremely complex electro-mechanical systems whose analytical description requires advanced methods, and which present many challenging and interesting research problems.

Figure 1.1: The ABB IRB6600 Robot. Photo courtesy of ABB

An official definition of such a robot comes from the **Robot Institute of America** (RIA): *A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.*

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

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

The robot, as we have defined it, was born out of the marriage of two earlier technologies: that of **teleoperators** and **numerically controlled milling machines**. Teleoperators, or master-slave devices, were developed during the second world war to handle radioactive materials. Computer numerical control (CNC) was developed because of the high precision required in the machining of certain items, such as components of high performance aircraft. The first robots essentially combined the mechanical linkages of the teleoperator with the autonomy and programmability of CNC machines. Several milestones on the road to present day robot technology are listed below.

Milestones in the History of Robotics

1947 — the first servoed electric powered teleoperator is developed

1948 — a teleoperator is developed incorporating force feedback

1949 — research on numerically controlled milling machine is initiated

1954 — George Devol designs the first programmable robot

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

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

1961 — the first robot incorporating force feedback is developed

1963 — the first robot vision system is developed

1971 — the Stanford Arm is developed at Stanford University

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

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

1975 — Unimation Inc. registers its first financial profit

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

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

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

1979 — the SCARA robot design is introduced in Japan

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

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

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

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

1988 — Stäubli Group purchases Unimation from Westinghouse

1988 — the IEEE Robotics and Automation Society is formed

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

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

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

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

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

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

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

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

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

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

It should be pointed out that the important applications of robots are by no means limited to those industrial jobs where the robot is directly replacing a human worker. There are many other applications of robotics in areas where the use of humans is impractical or undesirable. Among these are undersea and planetary exploration, satellite retrieval and repair, the defusing of explosive devices, and work in radioactive environments. Finally, prostheses, such as artificial limbs, are themselves robotic devices requiring methods of analysis and design similar to those of industrial manipulators.

1.3 Components and Structure of Robots

1.3.1 Symbolic Representation of Robots

Robot Manipulators are composed of **links** connected by **joints** into a **kinematic chain**. Joints are typically rotary (revolute) or linear (prismatic). A **revolute** joint is like a hinge and allows relative rotation between two links. A **prismatic** joint allows a linear relative motion between two links. We use the convention (R) for representing revolute joints and (P) for prismatic joints and draw them as shown in Figure 1.2.

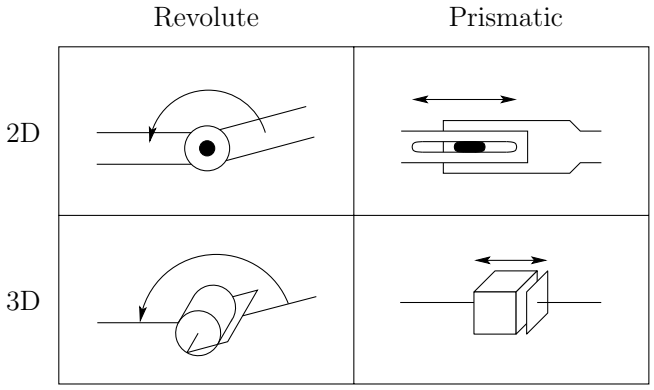


Figure 1.2: Symbolic representation of robot joints.

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

1.3.2 Degrees of Freedom and Workspace

The number of joints determines the **degrees-of-freedom** (DOF) of the manipulator. Typically, a manipulator should possess at least six independent DOF: three for **positioning** and three for **orientation**. With fewer than six DOF the arm cannot reach every point in its work environment with arbitrary orientation. Certain applications such as reaching around or behind obstacles require more than six DOF. The difficulty of controlling a manipulator increases rapidly with the number of links. A manipulator having more than six links is referred to as a **kinematically redundant** manipulator.

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

1.3.3 Classification of Robots

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

Power Source

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

Application Area

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

Method of Control

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

Servo controlled robots are further classified according to the method that the controller uses to guide the end-effector. The simplest type of robot in this class is the **point-to-point** robot. A point-to-point robot can be taught a discrete set of points but there is no control on the path of the end-effector in between taught points. Such robots are usually taught a series of points with a teach pendant. The points are then stored and played back. Point-to-point robots are severely limited in their range of applications. In **continuous path** robots, on the other hand, the entire path of the end-effector can be controlled. For example, the robot end-effector can be taught to follow a straight line between two points or even to

follow a contour such as a welding seam. In addition, the velocity and/or acceleration of the end-effector can often be controlled. These are the most advanced robots and require the most sophisticated computer controllers and software development.

Geometry

Most industrial manipulators at the present time have six or fewer degrees-of-freedom. These manipulators are usually classified kinematically on the basis of the first three joints of the arm, with the wrist being described separately. The majority of these manipulators fall into one of five geometric types: **articulate (RRR)**, **spherical (RRP)**, **SCARA (RRP)**, **cylindrical (RPP)**, or **cartesian (PPP)**.

We discuss each of these in detail below. Each of these five configurations are **serial link** robots. A sixth and fundamentally distinct class of manipulators is the so-called **parallel robot**. In a parallel configuration the links are arranged in a closed rather than open kinematic chain. We include a discussion of the parallel robot for completeness as parallel robots are becoming increasingly common.

1.3.4 Common Kinematic Arrangements

Articulated Configuration (RRR)

The articulated manipulator is also called a **revolute**, or **anthropomorphic** manipulator. The ABB IRB1400 articulated arm is shown in Figure 1.3. A common revolute joint design

Figure 1.3: The ABB IRB1400 Robot. Photo courtesy of ABB

is the **parallelogram linkage** such as the Motoman SK16, shown in Figure 1.4. In both of these arrangements joint axis z_2 is parallel to z_1 and both z_1 and z_2 are perpendicular to z_0. The structure and terminology associated with the elbow manipulator are shown in Figure 1.5. Its workspace is shown in Figure 1.6. The revolute configuration provides for relatively large freedom of movement in a compact space. The parallelogram linkage, although less dextrous typically than the elbow manipulator configuration, nevertheless

Figure 1.4: The Motoman SK16 manipulator.

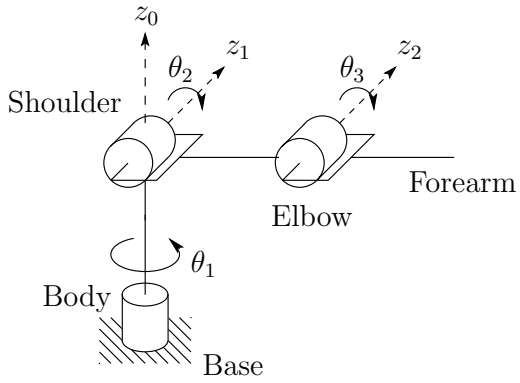


Figure 1.5: Structure of the elbow manipulator.

has several advantages that make it an attractive and popular design. The most notable feature of the parallelogram linkage configuration is that the actuator for joint 3 is located on link 1. Since the weight of the motor is born by link 1, links 2 and 3 can be made more lightweight and the motors themselves can be less powerful. Also the dynamics of the parallelogram manipulator are simpler than those of the elbow manipulator, thus making it easier to control.

Spherical Configuration (RRP)

By replacing the third or elbow joint in the revolute configuration by a prismatic joint one obtains the spherical configuration shown in Figure 1.7. The term **spherical configuration** derives from the fact that the spherical coordinates defining the position of the end-effector with respect to a frame whose origin lies at the intersection of the axes z_1 and z_2 are the same as the first three joint variables. Figure 1.8 shows the Stanford Arm, one of the most well-known spherical robots. The workspace of a spherical manipulator is shown in

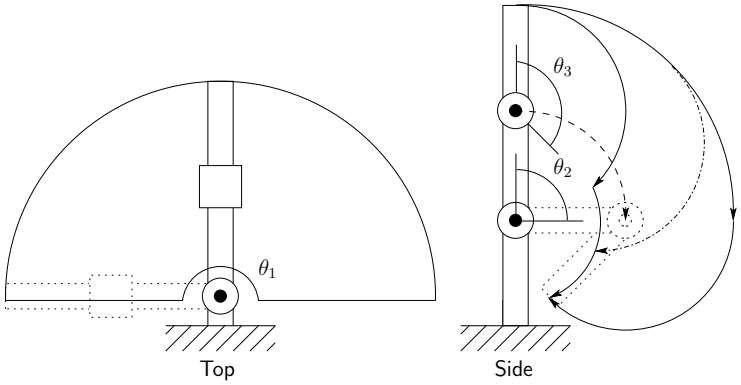


Figure 1.6: Workspace of the elbow manipulator.

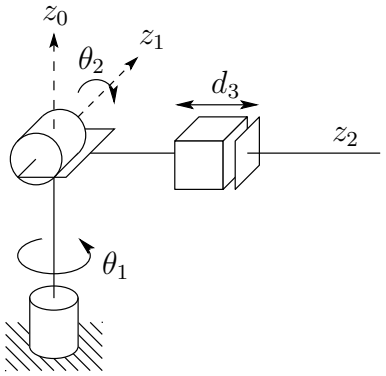


Figure 1.7: The spherical manipulator configuration.

Figure 1.9.

SCARA Configuration (RRP)

The so-called **SCARA** (for **S**elective **C**ompliant **A**rticulated **R**obot for **A**ssembly) shown in Figure 1.10 is a popular configuration, which, as its name suggests, is tailored for assembly operations. Although the SCARA has an RRP structure, it is quite different from the spherical configuration in both appearance and in its range of applications. Unlike the spherical design, which has z_0, z_1, z_2 mutually perpendicular, the SCARA has z_0, z_1, z_2 parallel. Figure 1.11 shows the Epson E2L653S, a manipulator of this type. The SCARA manipulator workspace is shown in Figure 1.12.

Cylindrical Configuration (RPP)

The cylindrical configuration is shown in Figure 1.13. The first joint is revolute and produces a rotation about the base, while the second and third joints are prismatic. As the name

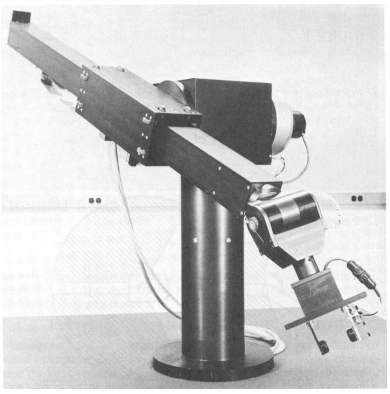


Figure 1.8: The Stanford Arm. Photo courtesy of the Coordinated Science Lab, University of Illinois at Urbana-Champaign.

suggests, the joint variables are the cylindrical coordinates of the end-effector with respect to the base. A cylindrical robot, the Seiko RT3300, is shown in Figure 1.14, with its workspace shown in Figure 1.15.

Cartesian configuration (PPP)

A manipulator whose first three joints are prismatic is known as a cartesian manipulator, shown in Figure 1.16.

For the Cartesian manipulator the joint variables are the Cartesian coordinates of the end-effector with respect to the base. As might be expected the kinematic description of this manipulator is the simplest of all configurations. Cartesian configurations are useful for table-top assembly applications and, as gantry robots, for transfer of material or cargo. An example of a cartesian robot, from Epson-Seiko, is shown in Figure 1.17. The workspace of a Cartesian manipulator is shown in Figure 1.18.

Parallel Manipulator

A **parallel manipulator** is one in which the links form a closed chain. More specifically, a parallel manipulator has two or more independent kinematic chains connecting the base to the end-effector. Figure 1.19 shows the ABB IRB 940 Tricept robot, which has a parallel configuration. The closed chain kinematics of parallel robots can result in greater structural rigidity, and hence greater accuracy, than open chain robots. The kinematic description of parallel robots fundamentally different from that of serial link robots and therefore requires different methods of analysis.

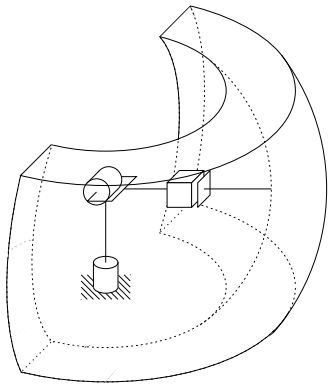


Figure 1.9: Workspace of the spherical manipulator.

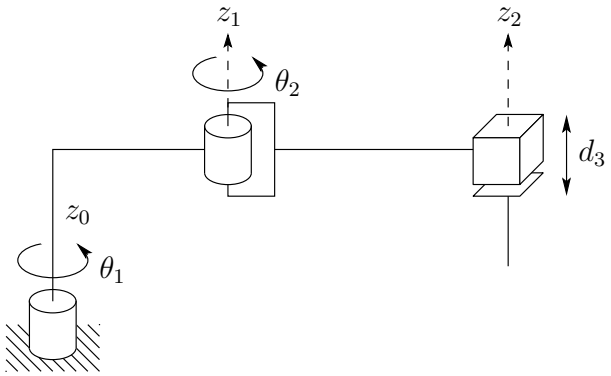


Figure 1.10: The SCARA (Selective Compliant Articulated Robot for Assembly).

1.3.5 Robotic Systems

A robot manipulator should be viewed as more than just a series of mechanical linkages. The mechanical arm is just one component to an overall **Robotic System**, shown in Figure 1.20, which consists of the **arm**, **external power source**, **end-of-arm tooling**, **external and internal sensors**, **computer interface**, and **control computer**. Even the programmed software should be considered as an integral part of the overall system, since the manner in which the robot is programmed and controlled can have a major impact on its performance and subsequent range of applications.

Figure 1.11: The Epson E2L653S SCARA Robot. Photo Courtesy of Epson.

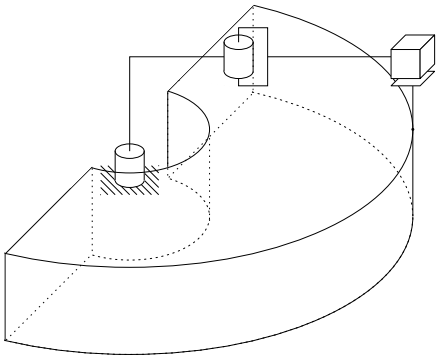


Figure 1.12: Workspace of the SCARA manipulator.

1.3.6 Accuracy and Repeatability

The **accuracy** of a manipulator is a measure of how close the manipulator can come to a given point within its workspace. **Repeatability** is a measure of how close a manipulator can return to a previously taught point. Most present day manipulators are highly repeatable but not very accurate. The primary method of sensing positioning errors in most cases is with position encoders located at the joints, either on the shaft of the motor that actuates the joint or on the joint itself. There is typically no direct measurement of the end-effector position and orientation. One must rely on the assumed geometry of the manipulator and its rigidity to infer (i.e., to calculate) the end-effector position from the measured joint angles. Accuracy is affected therefore by computational errors, machining accuracy in the construction of the manipulator, flexibility effects such as the bending of the links under gravitational and other loads, gear backlash, and a host of other static and dynamic effects. It is primarily for this reason that robots are designed with extremely high rigidity. Without high rigidity, accuracy can only be improved by some sort of direct

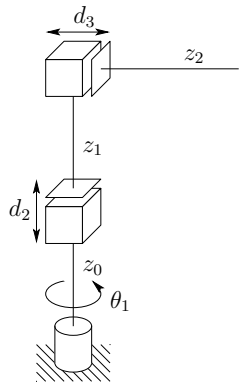


Figure 1.13: The cylindrical manipulator configuration.

Figure 1.14: The Seiko RT3300 Robot. Photo courtesy of Seiko.

sensing of the end-effector position, such as with vision.

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

In addition, as we will see in later chapters, rotational axes usually result in a large amount of kinematic and dynamic coupling among the links with a resultant accumulation of errors and a more difficult control problem. One may wonder then what the advantages of revolute joints are in manipulator design. The answer lies primarily in the increased dexterity and compactness of revolute joint designs. For example, Figure 1.21 shows that

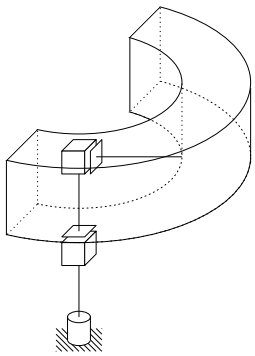


Figure 1.15: Workspace of the cylindrical manipulator.

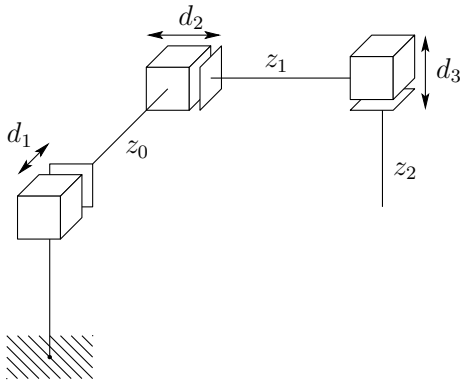


Figure 1.16: The cartesian manipulator configuration.

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

1.3.7 Wrists and End-Effectors

The **wrist** of a manipulator refers to the joints in the kinematic chain between the arm and hand. The wrist joints are nearly always all revolute. It is increasingly common to design manipulators with **spherical wrists**, by which we mean wrists whose three joint axes intersect at a common point. The spherical wrist is represented symbolically in Figure 1.22.

The spherical wrist greatly simplifies the kinematic analysis, effectively allowing one to decouple the positioning and orientation of an object to as great an extent as possible.

Figure 1.17: The Epson Cartesian Robot. Photo courtesy of Epson.

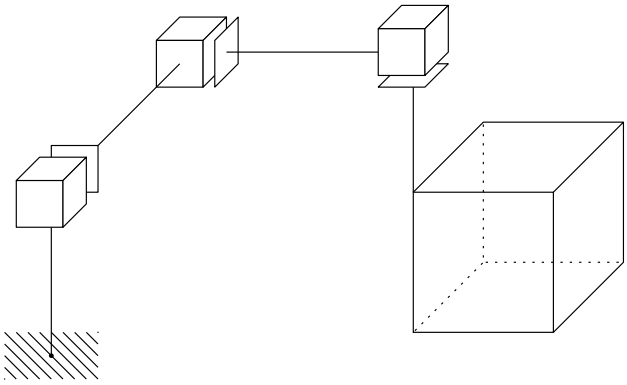


Figure 1.18: Workspace of the cartesian manipulator.

Typically therefore, the manipulator will possess three positional degrees-of-freedom, which are produced by three or more joints in the arm. The number of orientational degrees-of-freedom will then depend on the degrees-of-freedom of the wrist. It is common to find wrists having one, two, or three degrees-of-freedom depending of the application. For example, the SCARA robot shown in Figure 1.11 has four degrees-of-freedom: three for the arm, and one for the wrist, which has only a roll about the final z-axis.

It has been said that a robot is only as good as its **hand** or **end-effector**. The arm and wrist assemblies of a robot are used primarily for positioning the end-effector and any tool it may carry. It is the end-effector or tool that actually performs the work. The simplest type of end-effectors are grippers, such as shown in Figure 1.23 which usually are capable of only two actions, **opening** and **closing**. While this is adequate for materials transfer, some parts handling, or gripping simple tools, it is not adequate for other tasks such as welding, assembly, grinding, etc. A great deal of research is therefore being devoted to the design of special purpose end-effectors as well as tools that can be rapidly changed as the task dictates. There is also much research being devoted to the development of anthropomorphic hands.

Figure 1.19: The ABB IRB940 Tricept Parallel Robot. Photo courtesy of ABB.

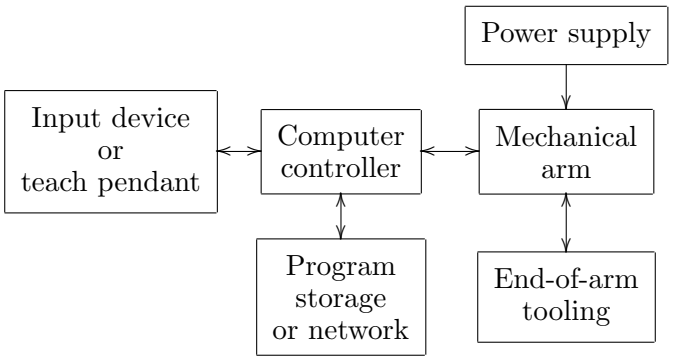


Figure 1.20: Components of a robotic system.

Such hands are being developed both for prosthetic use and for use in manufacturing. Since we are concerned with the analysis and control of the manipulator itself and not in the particular application or end-effector, we will not discuss end-effector design or the study of grasping and manipulation.

1.4 Outline of the Text

A typical application involving an industrial manipulator is shown in Figure 1.24. The manipulator is shown with a grinding tool which it must use to remove a certain amount of metal from a surface. In the present text we are concerned with the following question:

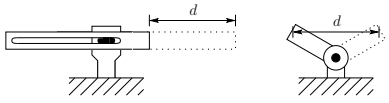


Figure 1.21: Linear vs. rotational link motion.

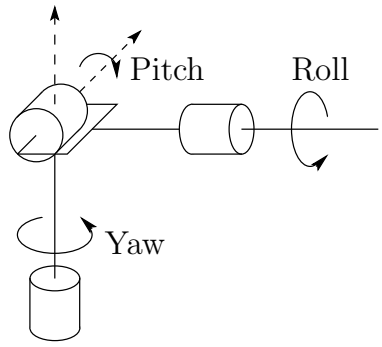


Figure 1.22: Structure of a spherical wrist.

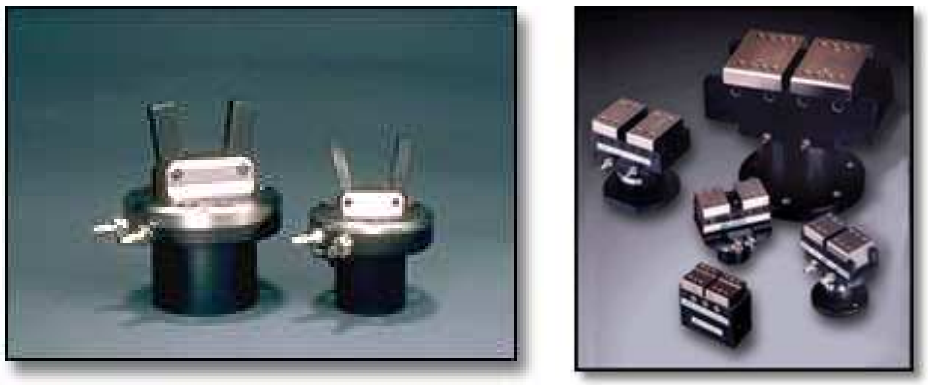


Figure 1.23: Angular Jaw and Parallel Jaw Grippers.

What are the basic issues to be resolved and what must we learn in order to be able to program a robot to perform tasks such as the above?

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

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

Problem 1: Forward Kinematics

The first problem encountered is to describe both the position of the tool and the locations A and B (and most likely the entire surface S) with respect to a common coordinate system. In Chapter 2 we give some background on representations of coordinate systems

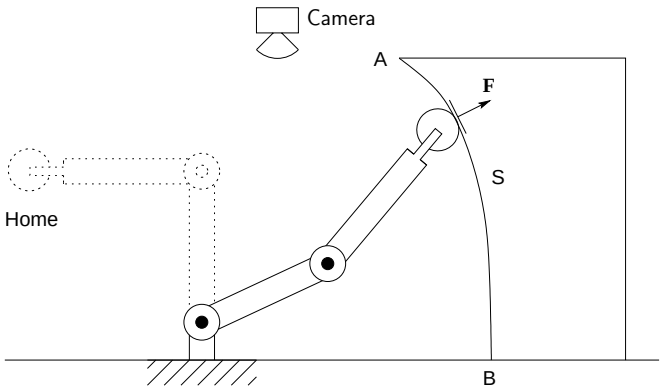


Figure 1.24: Two-link planar robot example.

and transformations among various coordinate systems.

Typically, the manipulator will be able to sense its own position in some manner using internal sensors (position encoders) located at joints 1 and 2, which can measure directly the joint angles θ_1 and θ_2. We also need therefore to express the positions A and B in terms of these joint angles. This leads to the **forward kinematics problem** studied in Chapter 3, which is to determine the position and orientation of the end-effector or tool in terms of the joint variables.

It is customary to establish a fixed coordinate system, called the **world** or **base** frame to which all objects including the manipulator are referenced. In this case we establish the base coordinate frame $o_0x_0y_0$ at the base of the robot, as shown in Figure 1.25. The

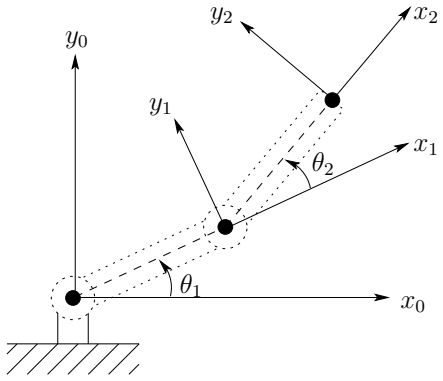


Figure 1.25: Coordinate frames for two-link planar robot.

coordinates (x, y) of the tool are expressed in this coordinate frame as

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

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

in which α_1 and α_2 are the lengths of the two links, respectively. Also the **orientation of the tool frame** relative to the base frame is given by the direction cosines of the x_2 and y_2 axes relative to the x_0 and y_0 axes, that is,

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

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

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

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

Problem 2: Inverse Kinematics

Now, given the joint angles θ_1, θ_2 we can determine the end-effector coordinates x and y. In order to command the robot to move to location B we need the inverse; that is, we need the joint variables θ_1, θ_2 in terms of the x and y coordinates of B. This is the problem of **Inverse Kinematics**. In other words, given x and y in the forward kinematic equations (1.1-1.2), we wish to solve for the joint angles. Since the forward kinematic equations are nonlinear, a solution may not be easy to find nor is there a unique solution in general. We can see, for example, in the case of a two-link planar mechanism that there may be no solution, if the given (x, y) coordinates are out of reach of the manipulator. If the given (x, y) coordinates are within the manipulator's reach there may be two solutions as shown in Figure 1.26, the so-called **elbow up** and **elbow down** configurations, or there may be exactly one solution if the manipulator must be fully extended to reach the point. There may even be an infinite number of solutions in some cases (Problem 1.25).

Consider the diagram of Figure 1.27. Using the **Law of Cosines** we see that the angle θ_2 is given by

$$\cos\theta_2 = \frac{x^2 + y^2 - \alpha_1^2 - \alpha_2^2}{2\alpha_1\alpha_2} := D. \tag{1.5}$$

We could now determine θ_2 as

$$\theta_2 = \cos^{-1}(D). \tag{1.6}$$

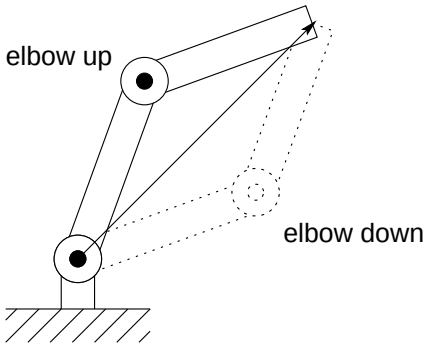


Figure 1.26: Multiple inverse kinematic solutions.

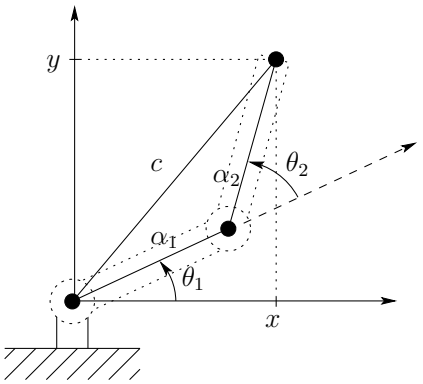


Figure 1.27: Solving for the joint angles of a two-link planar arm.

However, a better way to find θ_2 is to notice that if $\cos(\theta_2)$ is given by (1.5) then $\sin(\theta_2)$ is given as

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

and, hence, θ_2 can be found by

$$\theta_2 \quad = \quad \tan^{-1} \frac{\pm\sqrt{1 - D^2}}{D}. \tag{1.8}$$

The advantage of this latter approach is that both the elbow-up and elbow-down solutions are recovered by choosing the positive and negative signs in (1.8), respectively.

It is left as an exercise (Problem 1.19) to show that θ_1 is now given as

$$\theta_1 \quad = \quad \tan^{-1}(y/x) - \tan^{-1}\left(\frac{\alpha_2 \sin\theta_2}{\alpha_1 + \alpha_2 \cos\theta_2}\right). \tag{1.9}$$

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

Problem 3: Velocity Kinematics

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

$$\begin{aligned} \dot{x} &= -\alpha_1 \sin\theta_1 \cdot \dot{\theta}_1 - \alpha_2 \sin(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2) \\ \dot{y} &= \alpha_1 \cos\theta_1 \cdot \dot{\theta}_1 + \alpha_2 \cos(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2). \end{aligned} \tag{1.10}$$

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

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} -\alpha_1 \sin\theta_1 - \alpha_2 \sin(\theta_1+\theta_2) & -\alpha_2 \sin(\theta_1+\theta_2) \\ \alpha_1 \cos\theta_1 + \alpha_2 \cos(\theta_1+\theta_2) & \alpha_2 \cos(\theta_1+\theta_2) \end{bmatrix} \dot{\boldsymbol{\theta}} \\ &= J\dot{\boldsymbol{\theta}}. \end{aligned} \tag{1.11}$$

The matrix J defined by (1.11) is called the **Jacobian** of the manipulator and is a fundamental object to determine for any manipulator. In Chapter 5 we present a systematic procedure for deriving the Jacobian for any manipulator in the so-called **cross-product form**.

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

$$\dot{\boldsymbol{\theta}} = J^{-1}\dot{\boldsymbol{x}} \tag{1.12}$$

where J^{-1} is given by

$$J^{-1} = \frac{1}{\alpha_1\alpha_2 s_{\theta_2}} \begin{bmatrix} \alpha_2 c_{\theta_1+\theta_2} & \alpha_2 s_{\theta_1+\theta_2} \\ -\alpha_1 c_{\theta_1} - \alpha_2 c_{\theta_1+\theta_2} & -\alpha_1 s_{\theta_1} - \alpha_2 s_{\theta_1+\theta_2} \end{bmatrix} \tag{1.13}$$

in which c_θ and s_θ denote respectively $\cos\theta$ and $\sin\theta$. The determinant, $\det J$, of the Jacobian in (1.11) is $\alpha_1\alpha_2 \sin\theta_2$. The Jacobian does not have an inverse, therefore, when $\theta_2 = 0$ or π, in which case the manipulator is said to be in a **singular configuration**, such as shown in Figure 1.28 for $\theta_2 = 0$. The determination of such singular configurations is important for several reasons. At singular configurations there are infinitesimal motions that are unachievable; that is, the manipulator end-effector cannot move in certain directions. In the above cases the end effector cannot move in the direction parallel to x_2, from a singular configuration. Singular configurations are also related to the non-uniqueness of solutions of the inverse kinematics. For example, for a given end-effector position, there are in general two possible solutions to the inverse kinematics. Note that the singular configuration separates these two solutions in the sense that the manipulator cannot go from one configuration to the other without passing through the singularity. For many applications it is important to plan manipulator motions in such a way that singular configurations are avoided.

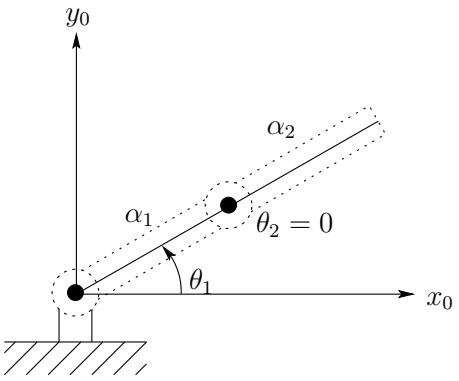


Figure 1.28: A singular configuration.

Problem 4: Path Planning and Trajectory Generation

The robot control problem is typically decomposed heirarchically into three tasks: path planning, trajectory generation, and trajectory tracking. The path planning problem, considered in Chapter 7, is to determine a path in task space (or configuration space) to move the robot to a goal position while avoiding collisions with objects in its workspace. These paths are encode position information without timing considerations, i.e. without considering velocities and accelerations along the planned paths. The tractory generation problem, considered in Chapter 8 is to generate reference trajectories that determine the time history of the manipulator along a given path or between initial and final configurations.

Problem 5: Vision

Cameras have become reliable and relatively inexpensive sensors in many robotic applications. Unlike joint sensors, which give information about the internal configuration of the robot, cameras can be used not only to measure the position of the robot but also to locate objects external to the robot in its workspace. In Chapter 6 we discuss the use of cameras to obtain position and orientation of objects.

Problem 6: Dynamics

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

Deriving the dynamic equations of motion for robots is not a simple task due to the large number of degrees of freedom and nonlinearities present in the system. In Chapter 9 we develop techniques based on Lagrangian dynamics for systematically deriving the equations of motion of such a system. In addition to the rigid links, the complete description of

robot dynamics includes the dynamics of the actuators that produce the forces and torques to drive the robot, and the dynamics of the drive trains that transmit the power from the actuators to the links. Thus, in Chapter 10 we also discuss actuator and drive train dynamics and their effects on the control problem.

Problem 7: Position Control

Control theory is used in Chapters 10 and 11 to design control algorithms for the execution of programmed tasks. The motion control problem consists of the **Tracking and Disturbance Rejection Problem**, which is the problem of determining the control inputs necessary to follow, or **track**, a desired trajectory that has been planned for the manipulator, while simultaneously **rejecting** disturbances due to unmodelled dynamic effects such as friction and noise. We detail the standard approaches to robot control based on frequency domain techniques. We also introduce the notion of **feedforward control** and the techniques of **computed torque** and **inverse dynamics** as a means for compensating the complex nonlinear interaction forces among the links of the manipulator. Robust control is introduced in Chapter 11 using the **Second Method of Lyapunov**. Chapter **??** provides some additional advanced techniques from nonlinear control theory that are useful for controlling high performance robots.

Problem 8: Force Control

Once the manipulator has reached location A. it must follow the contour S maintaining a constant force normal to the surface. Conceivably, knowing the location of the object and the shape of the contour, we could carry out this task using position control alone. This would be quite difficult to accomplish in practice, however. Since the manipulator itself possesses high rigidity, any errors in position due to uncertainty in the exact location of the surface or tool would give rise to extremely large forces at the end-effector that could damage the tool, the surface, or the robot. A better approach is to measure the forces of interaction directly and use a **force control** scheme to accomplish the task. In Chapter 12 we discuss force control and compliance and discuss the two most common approaches to force control, **hybrid control** and **impedance control**.

Chapter 2

RIGID MOTIONS AND HOMOGENEOUS TRANSFORMATIONS

A large part of robot kinematics is concerned with the establishment of various coordinate systems to represent the positions and orientations of rigid objects and with transformations among these coordinate systems. Indeed, the geometry of three-dimensional space and of rigid motions plays a central role in all aspects of robotic manipulation. In this chapter we study the operations of rotation and translation and introduce the notion of homogeneous transformations.[1] Homogeneous transformations combine the operations of rotation and translation into a single matrix multiplication, and are used in Chapter 3 to derive the so-called forward kinematic equations of rigid manipulators.

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

2.1 Representing Positions

Before developing representation schemes for points and vectors, it is instructive to distinguish between the two fundamental approaches to geometric reasoning: the *synthetic*

[1]Since we make extensive use of elementary matrix theory, the reader may wish to review Appendix A before beginning this chapter.

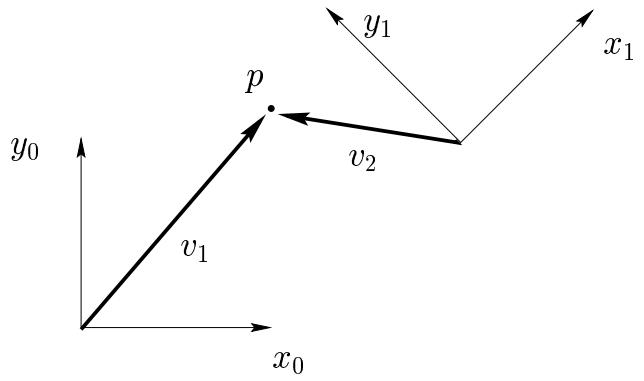


Figure 2.1: Two coordinate frames, a point p, and two vectors $\vec{v}_1$ and $\vec{v}_2$.

approach and the *analytic* approach. In the former, one reasons directly about geometric entities (e.g., points or lines), while in the latter, one represents these entities using coordinates or equations, and reasoning is performed via algebraic manipulations.

Consider Figure 2.1. Using the synthetic approach, without ever assigning coordinates to points or vectors, one can say that x_0 is perpendicular to y_0, or that $\vec{v}_1 \times \vec{v}_2$ defines a vector that is perpendicular to the plane containing $\vec{v}_1$ and $\vec{v}_2$, in this case pointing out of the page.

In robotics, one typically uses analytic reasoning, since robot tasks are often defined in a Cartesian workspace, using Cartesian coordinates. Of course, in order to assign coordinates it is necessary to specify a coordinate frame. Consider again Figure 2.1. We could specify the coordinates of the point p with respect to either frame $o_0x_0y_0$ or frame $o_1x_1y_1$. In the former case, we might assign to p the coordinate vector $(5, 6)^T$, and in the latter case $(-3, 4)^T$. So that the reference frame will always be clear, we will adopt a notation in which a superscript is used to denote the reference frame. Thus, we would write

$$p^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \qquad p^1 = \begin{bmatrix} -3 \\ 4 \end{bmatrix} \tag{2.1}$$

Geometrically, a point corresponds to a specific location in space. We stress here that $p \neq p^0$ and $p \neq p^1$, i.e., p is a geometric entity, a point in space, while both p^0 and p^1 are coordinate vectors that represent the location of this point in space with respect to coordinate frames $o_0x_0y_0$ and $o_1x_1y_1$, respectively.

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

$$o_1^0 = \begin{bmatrix} 10 \\ 5 \end{bmatrix}, \qquad o_0^1 = \begin{bmatrix} -10 \\ 5 \end{bmatrix}. \tag{2.2}$$

In cases where there is only a single coordinate frame, or in which the reference frame is obvious, we will often omit the superscript. This is a slight abuse of notation, and the

reader is advised to bear in mind the difference between the geometric entity called p and any particular coordinate vector that is assigned to represent p. The former is invariant with respect to the choice of coordinate systems, while the latter obviously depends on the choice of coordinate frames.

While a point corresponds to a specific location in space, a *vector* specifies a direction and a magnitude. Vectors can be used, for example, to represent displacements or forces. Therefore, while the point p is not equivalent to the vector $\vec{v}_1$, the displacement from the origin o_0 to the point p is given by the vector $\vec{v}_1$. In this text, we will use the term *vector* to refer to what are sometimes called *free vectors*, i.e., vectors that are not constrained to be located at a particular point in space. Under this convention, it is clear that points and vectors are not equivalent, since points refer to specific locations in space, but a vector can be moved to any location in space. Under this convention, two vectors are equal if they have the same direction and the same magnitude.

When assigning coordinates to vectors, we use the same notational convention that we used when assigning coordinates to points. Thus, $\vec{v}_1$ and $\vec{v}_2$ are geometric entities that are invariant with respect to the choice of coordinate systems, but the representation by coordinates of these vectors depends directly on the choice of reference coordinate frame. In the example of Figure 2.1, we would obtain

$$v_1^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \qquad v_1^1 = \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \qquad v_2^0 = \begin{bmatrix} -5 \\ 1 \end{bmatrix} \qquad v_2^1 = \begin{bmatrix} -3 \\ 4 \end{bmatrix}. \tag{2.3}$$

In order to perform algebraic manipulations using coordinates, it is essential that all coordinate vectors be defined with respect to the same coordinate frame. For example, an expression of the form $v_1^1 + v_2^2$ would make no sense geometrically. Thus, we see a clear need, not only for a representation system that allows points to be expressed with respect to various coordinate systems, but also for a mechanism that allows us to transform the coordinates of points that are expressed in one coordinate system into the appropriate coordinates with respect to some other coordinate frame. Such coordinate transformations and their derivations are the topic for much of the remainder of this chapter.

2.2 Representing Rotations

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

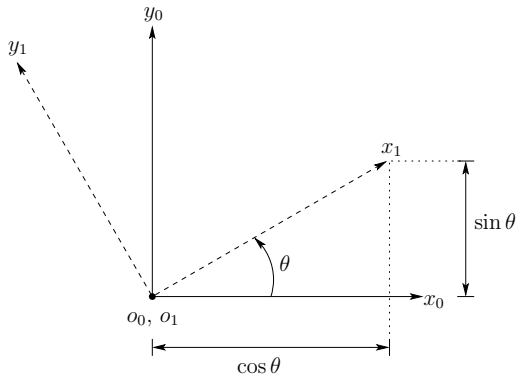


Figure 2.2: Coordinate frame $o_1x_1y_1$ is oriented at an angle θ with respect to $o_0x_0y_0$

2.2.1 Rotation in the plane

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

A slightly less obvious way to specify the orientation is to specify the coordinate vectors for the axes of frame $o_1x_1y_1$ with respect to coordinate frame $o_0x_0y_0$. In particular, we can build a matrix of the form:

$$R_1^0 = [x_1^0 | y_1^0]\,. \tag{2.4}$$

A matrix in this form is called a **rotation matrix**. Rotation matrices have a number of special properties, which we will discuss below.

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

$$x_1^0 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \qquad y_1^0 = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}, \tag{2.5}$$

which gives

$$R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{2.6}$$

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

Although we have derived the entries for R_1^0 in terms of the angle θ, it is not necessary that we do so. An alternative approach, and one that scales nicely to the three dimensional case, is to build the rotation matrix by projecting the axes of frame $o_1x_1y_1$ onto the coordinate axes of frame $o_0x_0y_0$. Recalling that the dot product of two unit vectors gives the projection of one onto the other, we obtain

$$x_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix}, \qquad y_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix}, \tag{2.7}$$

which can be combined to obtain the rotation matrix

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix}. \tag{2.8}$$

Thus the columns of R_1^0 specify the direction cosines of the coordinate axes of $o_1x_1y_1$ relative to the coordinate axes of $o_0x_0y_0$. For example, the first column $(x_1 \cdot x_0, x_1 \cdot y_0)^T$ of R_1^0 specifies the direction of x_1 relative to the frame $o_0x_0y_0$. Note that the right hand sides of these equations are defined in terms of geometric entities, and not in terms of their coordinates. Examining Figure 2.2 it can be seen that this method of defining the rotation matrix by projection gives the same result as was obtained in equation (2.6).

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

$$R_0^1 = \begin{bmatrix} x_0 \cdot x_1 & y_0 \cdot x_1 \\ x_0 \cdot y_1 & y_0 \cdot y_1 \end{bmatrix}. \tag{2.9}$$

Since the inner product is commutative, (i.e. $x_i \cdot y_j = y_j \cdot x_i$), we see that

$$R_0^1 = (R_1^0)^T. \tag{2.10}$$

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

$$(R_1^0)^T = (R_1^0)^{-1}. \tag{2.11}$$

Such a matrix is said to be **orthogonal**. The column vectors of R_1^0 are of unit length and mutually orthogonal (Problem 2-1). It can also be shown (Problem 2-2) that $\det R_1^0 = \pm 1$. If we restrict ourselves to right-handed coordinate systems, as defined in Appendix A, then $\det R_1^0 = +1$ (Problem 2-3). All rotation matrices have the properties of being orthogonal matrices with determinant $+1$. It is customary to refer to the set of **all** 2×2 rotation matrices by the symbol $SO(2)$[2]. The properties of such matrices are summarized in Figure 2.3.

[2]The notation $SO(2)$ stands for Special Orthogonal group of order 2.

Every $n \times n$ rotation matrix R has the following properties (for $n = 2, 3$):

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

Figure 2.3: Properties of Rotation Matrices

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

$$\begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T. \tag{2.12}$$

2.2.2 Rotations in three dimensions

The projection technique described above scales nicely to the three dimensional case. In three dimensions, each axis of the frame $o_1x_1y_1z_1$ is projected onto coordinate frame $o_0x_0y_0z_0$. The resulting rotation matrix is given by

not corrrect

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_1 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_1 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}. \tag{2.13}$$

As was the case for rotation matrices in two dimensions, matrices in this form are orthogonal, with determinant equal to 1. In this case, 3×3 rotation matrices belong to the group $SO(3)$. The properties listed in Figure 2.3 also apply to rotation matrices in $SO(3)$.

Example 2.1 *Suppose the frame $o_1x_1y_1z_1$ is rotated through an angle θ about the z_0-axis, and it is desired to find the resulting transformation matrix R_1^0. Note that by convention the positive sense for the angle θ is given by the right hand rule; that is, a positive rotation of θ degrees about the z-axis would advance a right-hand threaded screw along the positive z-axis. From Figure 2.4 we see that*

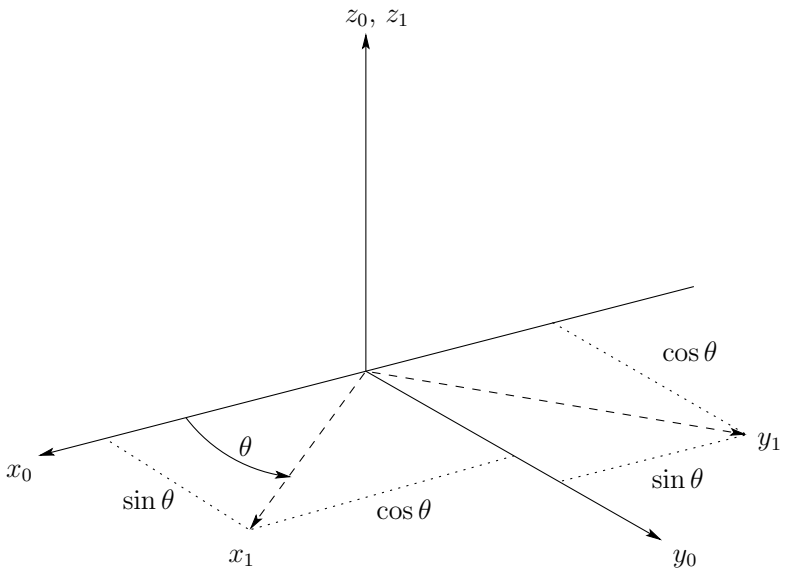


Figure 2.4: Rotation about z_0.

$$\begin{aligned} x_1 \cdot x_0 &= \cos\theta \qquad y_1 \cdot x_0 = -\sin\theta \\ x_1 \cdot y_0 &= \sin\theta \qquad y_1 \cdot y_0 = \cos\theta \end{aligned} \tag{2.14}$$

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

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

$$R_1^0 \;=\; \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.15}$$

◇

The Basic Rotation Matrices

The transformation (2.15) is called a **basic rotation matrix** (about the z-axis). In this case we find it useful to use the more descriptive notation $R_{z,\theta}$, instead of R_1^0 to denote the matrix (2.15). It is easy to verify that the basic rotation matrix $R_{z,\theta}$ has the properties

$$R_{z,0} \;=\; I \tag{2.16}$$

$$R_{z,\theta} R_{z,\phi} \;=\; R_{z,\theta+\phi} \tag{2.17}$$

which together imply

$$R_{z,\theta}{}^{-1} \;=\; R_{z,-\theta}. \tag{2.18}$$

Similarly the basic rotation matrices representing rotations about the x and y-axes are given as (Problem 2-5)

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.19}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.20}$$

which also satisfy properties analogous to (2.16)-(2.18).

Example 2.2 *Consider the frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ shown in Figure 2.5. Projecting the unit vectors x_1, y_1, z_1 onto x_0, y_0, z_0 gives the coordinates of x_1, y_1, z_1 in the $o_0x_0y_0z_0$ frame. We see that the coordinates of x_1 are $\left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)^T$, the coordinates of y_1 are $\left(\frac{1}{\sqrt{2}}, 0, \frac{-1}{\sqrt{2}}\right)^T$ and the coordinates of z_1 are $(0,1,0)^T$. The rotation matrix R_1^0 specifying the orientation of $o_1x_1y_1z_1$ relative to $o_0x_0y_0z_0$ has these as its column vectors, that is,*

$$R_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}. \tag{2.21}$$

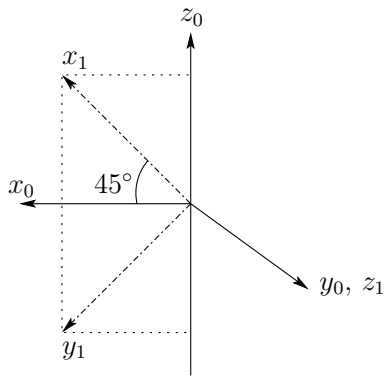


Figure 2.5: Defining the relative orientation of two frames.

◇

2.3 Rotational Transformations

Figure 2.6 shows a rigid object S to which a coordinate frame $o_1x_1y_1z_1$ is attached. Given the coordinates p^1 of the point p (i.e., given the coordinates of p with respect to the frame

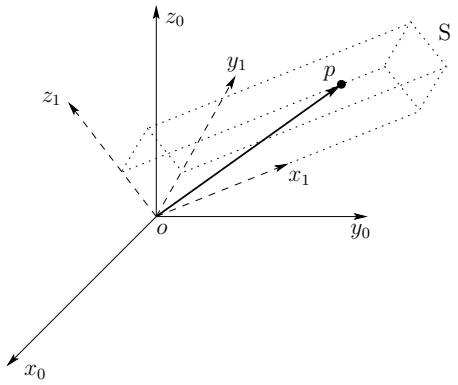


Figure 2.6: Coordinate frame attached to a rigid body.

$o_1x_1y_1z_1$), we wish to determine the coordinates of p relative to a fixed reference frame $o_0x_0y_0z_0$.

The coordinates $p^1 = (u, v, w)^t$ satisfy the equation

$$p = ux_1 + vy_1 + wz_1. \tag{2.22}$$

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

$$p^0 = \begin{bmatrix} p \cdot x_0 \\ p \cdot y_0 \\ p \cdot z_0 \end{bmatrix}. \tag{2.23}$$

Combining these two equations we obtain

$$\begin{aligned} p^0 &= \begin{bmatrix} (ux_1 + vy_1 + wz_1) \cdot x_0 \\ (ux_1 + vy_1 + wz_1) \cdot y_0 \\ (ux_1 + vy_1 + wz_1) \cdot z_0 \end{bmatrix} && (2.24) \\ &= \begin{bmatrix} ux_1 \cdot x_0 + vy_1 \cdot x_0 + wz_1 \cdot x_0 \\ ux_1 \cdot y_0 + vy_1 \cdot y_0 + wz_1 \cdot y_0 \\ ux_1 \cdot z_0 + vy_1 \cdot z_0 + wz_1 \cdot z_0 \end{bmatrix} && (2.25) \\ &= \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}. && (2.26) \end{aligned}$$

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

$$p^0 = R_1^0 p^1. \tag{2.27}$$

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

We have now seen how rotation matrices can be used to relate the orientation of one frame to another frame, and to assign coordinate representations to points and vectors. For example, given a point p in space, we have shown how a rotation matrix can be used to derive coordinates for p with respect to different coordinate frames whose orientations are related by a rotation matrix. We can also use rotation matrices to represent rigid motions that correspond to pure rotation. Consider Figure 2.7. One corner of the block in Figure 2.7(a) is located at the point p_a in space. Figure 2.7(b) shows the same block after it has been rotated about z_0 by the angle π. In Figure 2.7(b), the same corner of the block is now located at point p_b in space. It is possible to derive the coordinates for p_b given only the coordinates for p_a and the rotation matrix that corresponds to the rotation about z_0. To see how this can be accomplished, imagine that a coordinate frame is rigidly attached to the block in Figure 2.7(a), such that it is coincident with the frame $o_0x_0y_0z_0$. After the rotation by π, the block's coordinate frame, which is rigidly attached to the block, is also rotated by π. If we denote this rotated frame by $o_1x_1y_1z_1$, we obtain

$$R_1^0 = R_{z,\pi} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.28}$$

In the local coordinate frame $o_1x_1y_1z_1$, the point p_b has the coordinate representation p_b^1. To obtain its coordinates with respect to frame $o_0x_0y_0z_0$, we merely apply the coordinate transformation equation (2.27), giving

$$p_b^0 = R_{z,\pi}p_b^1. \tag{2.29}$$

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

$$p_b^0 = R_{z,\pi}p_a^0. \tag{2.30}$$

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

$$p_b^0 = Rp_a^0. \tag{2.31}$$

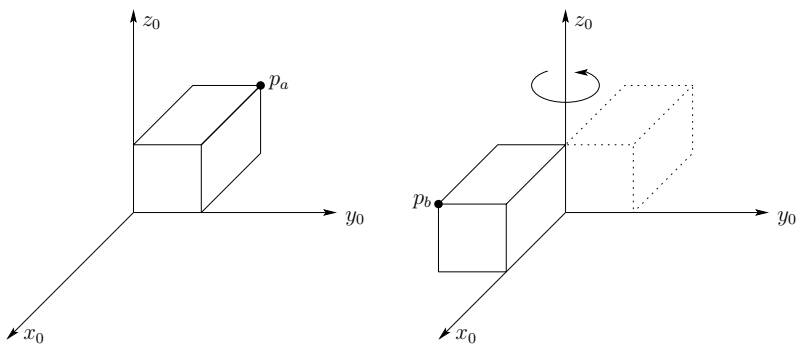


Figure 2.7: The block in (b) is obtained by rotating the block in (a) by π about z_0.

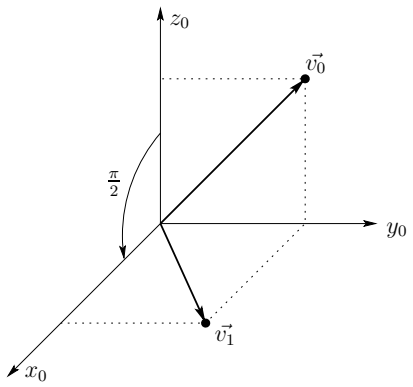


Figure 2.8: Rotating a vector about axis y_0.

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

Example 2.3 *The vector $\vec{v}$ with coordinates $v^0 = (0,1,1)^T$ is rotated about y_0 by $\frac{\pi}{2}$ as shown in Figure 2.8. The resulting vector $\vec{v}_1$ has coordinates given by*

$$v_1^0 = R_{y,\frac{\pi}{2}} v^0 \tag{2.32}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}. \tag{2.33}$$

◇

Thus, as we have now seen, a third interpretation of a rotation matrix R is as an operator acting on vectors in a fixed frame. In other words, instead of relating the coordinates of a fixed vector with respect to two different coordinate frames, the expression (2.32) can represent the coordinates in $o_0x_0y_0z_0$ of a vector $\vec{v}_1$ which is obtained from a vector $\vec{v}$ by a given rotation.

2.3.1 Summary

We have seen that a rotation matrix, either $R \in SO(3)$ or $R \in SO(2)$, can be interpreted in three distinct ways:

1. It represents a coordinate transformation relating the coordinates of a point $\boldsymbol{p}$ in two different frames.
2. It gives the orientation of a transformed coordinate frame with respect to a fixed coordinate frame.
3. It is an operator taking a vector and rotating it to a new vector in the same coordinate system.

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

2.4 Composition of Rotations

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

2.4.1 Rotation with respect to the current coordinate frame

Recall that the matrix R_1^0 in Equation (2.27) represents a rotational transformation between the frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$. Suppose we now add a third coordinate frame $o_2x_2y_2z_2$

related to the frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ by rotational transformations. As we saw above, a given point p can then be represented by coordinates specified with respect to any of these three frames: p^0, p^1 and p^2. The relationship between these representations of p is

$$\begin{aligned} p^0 &= R_1^0 p^1 && (2.34)\\ p^1 &= R_2^1 p^2 && (2.35)\\ p^0 &= R_2^0 p^2 = R_1^0 R_2^1 p^2 && (2.36) \end{aligned}$$

where each R_j^i is a rotation matrix, and equation (2.36) follows directly by substituting equation (2.35) into equation (2.34). Note that R_1^0 and R_2^0 represent rotations relative to the frame $o_0x_0y_0z_0$ while R_2^1 represents a rotation relative to the frame $o_1x_1y_1z_1$.

From equation (2.36) we can immediately infer the identity

$$R_2^0 = R_1^0 R_2^1. \tag{2.37}$$

Equation (2.37) is the composition law for rotational transformations. It states that, in order to transform the coordinates of a point p from its representation p^2 in the frame $o_2x_2y_2z_2$ to its representation p^0 in the frame $o_0x_0y_0z_0$, we may first transform to its coordinates p^1 in the frame $o_1x_1y_1z_1$ using R_2^1 and then transform p^1 to p^0 using R_1^0.

We may also interpret Equation (2.37) as follows. Suppose initially that all three of the coordinate frames coincide. We first rotate the frame $o_2x_2y_2z_2$ relative to $o_0x_0y_0z_0$ according to the transformation R_1^0. Then, with the frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ coincident, we rotate $o_2x_2y_2z_2$ relative to $o_1x_1y_1z_1$ according to the transformation R_2^1. In each case we call the frame relative to which the rotation occurs the **current frame**.

Example 2.4 *Henceforth, whenever convenient we use the shorthand notation $c_\theta = \cos\theta$, $s_\theta = \sin\theta$ for trigonometric functions. Suppose a rotation matrix R represents a rotation of ϕ degrees about the current y-axis followed by a rotation of θ degrees about the current z-axis. Refer to Figure 2.9. Then the matrix R is given by*

$$\begin{aligned} R &= R_{y,\phi} R_{z,\theta} && (2.38)\\ &= \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta & -c_\phi s_\theta & s_\phi \\ s_\theta & c_\theta & 0 \\ -s_\phi c_\theta & s_\phi s_\theta & c_\phi \end{bmatrix}. \end{aligned}$$

◇

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

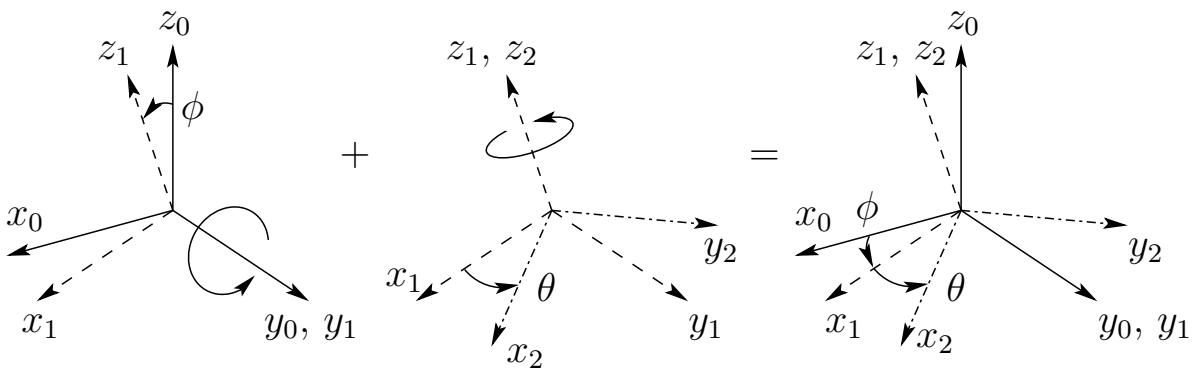


Figure 2.9: Composition of rotations about current axes.

Example 2.5 *Suppose that the above rotations are performed in the reverse order, that is, first a rotation about the current z-axis followed by a rotation about the current y-axis. Then the resulting rotation matrix is given by*

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

Comparing (2.38) and (2.39) we see that $R \neq R'$.

◇

2.4.2 Rotation with respect to a fixed frame

Many times it is desired to perform a sequence of rotations, each about a given fixed coordinate frame, rather than about successive current frames. For example we may wish to perform a rotation about x_0 followed by a rotation about the y_0 (and not y_1!). We will refer to $o_0x_0y_0z_0$ as the **fixed frame**. In this case the composition law given by equation (2.37) is not valid. It turns out that the correct composition law in this case is simply to multiply the successive rotation matrices *in the reverse order* from that given by (2.37). Note that the rotations themselves are not performed in reverse order. Rather they are performed about the fixed frame instead of about the current frame.

To see why this is so, consider the following argument. Let $o_0x_0y_0z_0$ be the reference frame. Let the frame $o_1x_1y_1z_1$ be obtained by performing a rotation with respect to the reference frame, and let this rotation be denoted by R_1^0. Now let $o_2x_2y_2z_2$ be obtained

by performing a rotation of frame $o_1x_1y_1z_1$ with respect to the reference frame (*not* with respect to $o_1x_1y_1z_1$ itself). We will, for the moment, denote this rotation about the fixed frame by the matrix R. Finally, let R_2^0 be the rotation matrix that denotes the orientation of frame $o_2x_2y_2z_2$ with respect to $o_0x_0y_0z_0$. We know that $R_2^0 \neq R_1^0 R$, since this equation applies for rotation about the current frame. Thus, we now seek to determine the matrix R_2^1 such that $R_2^0 = R_1^0 R_2^1$.

In order to find this matrix, we shall proceed as follows. First, we will rotate frame $o_1x_1y_1z_1$ to align it with the reference frame. This can be done by postmultiplication of R_1^0 by its inverse. Now, since the current frame is aligned with the reference frame, we can postmultiply by the rotation corresponding to R (i.e., now that the fixed reference frame coincides with the current frame, rotation about the current frame is equivalent to rotation about the fixed reference frame). Finally, we must undo the initial rotation, which corresponds to a postmultiplication of R_1^0. When we concatenate these operations, we obtain the following:

$$R_2^0 = R_1^0 R_2^1 \tag{2.40}$$

$$R_2^0 = R_1^0 \left[(R_1^0)^{-1} R R_1^0 \right] \tag{2.41}$$

$$R_2^0 = R R_1^0 \tag{2.42}$$

This procedure is an instance of a classical technique in engineering problem solving. When confronted with a difficult problem, if one can transform the problem into an easier problem, it is often possible to transform the solution to this easier problem into a solution to the original, more difficult problem. In our current case, we didn't know how to solve the problem of rotating with respect to the fixed reference frame. Therefore, we transformed the problem to the problem of rotating about the current frame (by using the rotation $(R_1^0)^{-1}$). We then transformed the solution for this simpler problem by applying the inverse of the rotation that we initially used to simplify the problem (i.e., we postmultiplied by R_1^0).

Example 2.6 *Suppose that a rotation matrix R represents a rotation of ϕ degrees about y_0 followed by a rotation of θ about the fixed z_0. Refer to Figure 2.10. Let p^0, p^1, and p^2 be representations of a point p. Initially the fixed and current axes are the same, namely $o_0x_0y_0z_0$ and therefore we can write as before*

$$p^0 = R_{y,\phi} p^1 \tag{2.43}$$

where $R_{y,\phi}$ is the basic rotation matrix about the y-axis. Now, since the second rotation is about the fixed frame $o_0x_0y_0z_0$ and not the current frame $o_1x_1y_1z_1$, we cannot conclude that

$$p^1 = R_{z,\theta} p^2 \tag{2.44}$$

since this would require that we interpret $R_{z,\theta}$ as being a rotation about z_1. Applying the same process as above, we first undo the previous rotation, then rotate about z_0 and finally

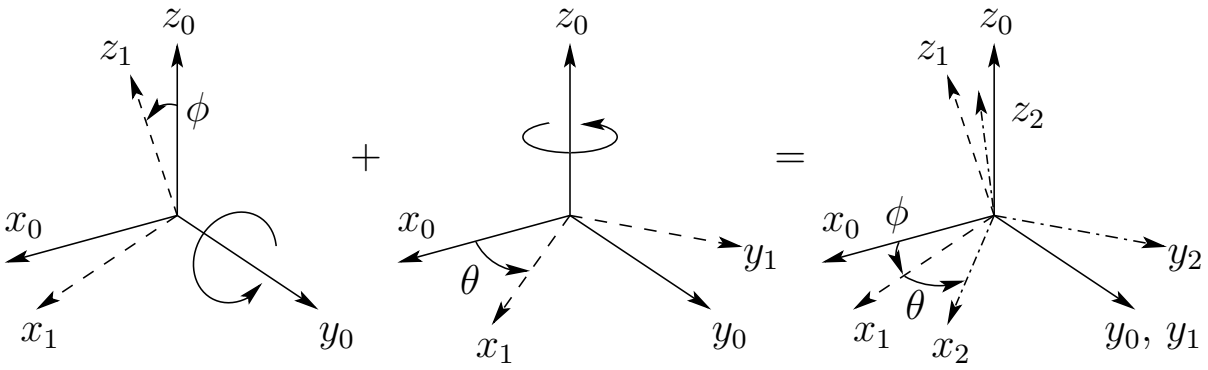


Figure 2.10: Composition of rotations about fixed axes.

reinstate the original transformation, that is,

$$p^1 = \left[R_{y,-\phi} R_{z,\theta} R_{y,\phi}\right] p^2. \tag{2.45}$$

This is the correct expression, and not (2.44). Now, substituting (2.45) into (2.43) we obtain

$$\begin{aligned} p^0 &= R_{y,\phi} p^1 \\ &= R_{y,\phi} \left[R_{y,-\phi} R_{z,\theta} R_{y,\phi}\right] p^2 \\ &= R_{z,\theta} R_{y,\phi} p^2. \end{aligned} \tag{2.46}$$

It is not necessary to remember the above derivation, only to note by comparing (2.46) with (2.38) that we obtain the same basic rotation matrices, but in the reverse order.
◇

2.4.3 Summary

We can summarize the rule of composition of rotational transformations by the following recipe. Given a fixed frame $o_0x_0y_0z_0$ a current frame $o_1x_1y_1z_1$, together with rotation matrix R_1^0 relating them, if a third frame $o_2x_2y_2z_2$ is obtained by a rotation R performed relative to the **current frame** then **postmultiply** R_1^0 by $R = R_2^1$ to obtain

$$R_2^0 = R_1^0 R_2^1. \tag{2.47}$$

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

$$R_2^0 = R R_1^0. \tag{2.48}$$

In each case R_2^0 represents the transformation between the frames $o_0x_0y_0z_0$ and $o_2x_2y_2z_2$. The frame $o_2x_2y_2z_2$ that results in (2.47) will be different from that resulting from (2.48).

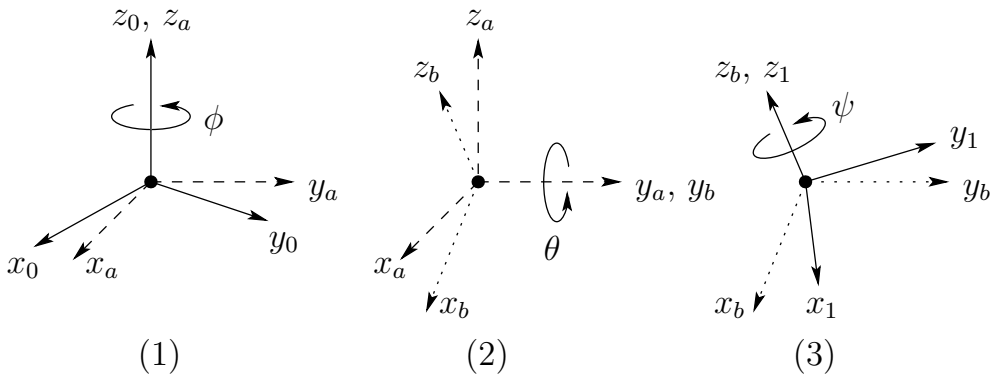


Figure 2.11: Euler angle representation.

2.5 Parameterizations of Rotations

The nine elements r_{ij} in a general rotational transformation R are not independent quantities. Indeed a rigid body possesses at most three rotational degrees-of-freedom and thus at most three quantities are required to specify its orientation. This can be easily seen by examining the constraints that govern the matrices in $SO(3)$:

$$\sum_i r_{ij}^2 = 1, \quad j \in \{1, 2, 3\} \tag{2.49}$$

$$r_{1i}r_{1j} + r_{2i}r_{2j} + r_{3i}r_{3j} = 0, \quad i \neq j. \tag{2.50}$$

Equation (2.49) follows from the fact the the columns of a rotation matrix are unit vectors, and (2.50) follows from the fact that columns of a rotation matrix are mutually orthogonal. Together, these constraints define six independent equations with nine unknowns, which implies that there are three free variables.

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

2.5.1 Euler Angles

A common method of specifying a rotation matrix in terms of three independent quantities is to use the so-called **Euler Angles**. Consider again the fixed coordinate frame $o_0x_0y_0z_0$ and the rotated frame $o_1x_1y_1z_1$ shown in Figure 2.11.

We can specify the orientation of the frame $o_1x_1y_1z_1$ relative to the frame $o_0x_0y_0z_0$ by three angles (ϕ, θ, ψ), known as Euler Angles, and obtained by three successive rotations as follows: First rotate about the z-axis by the angle ϕ. Next rotate about the current y-axis by the angle θ. Finally rotate about the current z-axis by the angle ψ. In Figure 2.11, frame $o_ax_ay_az_a$ represents the new coordinate frame after the rotation by ϕ, frame $o_bx_by_bz_b$

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

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

$$\begin{aligned} R_1^0 &= R_{z,\phi}R_{y,\theta}R_{z,\psi} && (2.51)\\ &= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}. && (2.52) \end{aligned}$$

Consider now the problem of determining the angles ϕ, θ, and ψ, given the rotation matrix

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \quad (2.53)$$

Suppose that not both of r_{13}, r_{23} are zero. Then the above equations show that $s_\theta \neq 0$, and hence that not both of r_{31}, r_{32} are zero. If not both r_{13} and r_{23} are zero, then $r_{33} \neq \pm 1$, and we have $c_\theta = r_{33}$, $s_\theta = \pm\sqrt{1 - r_{33}^2}$ so

$$\theta = A\tan\left(r_{33}, \sqrt{1 - r_{33}^2}\right) \quad (2.54)$$

or

$$\theta = A\tan\left(r_{33}, -\sqrt{1 - r_{33}^2}\right). \quad (2.55)$$

The function $\theta = A\tan(x, y)$ computes the arc tangent function, where x and y are the cosine and sine, respectively, of the angle θ. This function uses the signs of x and y to select the appropriate quadrant for the angle θ. Note that if both x and y are zero, $A\tan$ is undefined.

If we choose the value for θ given by Equation (2.54), then $s_\theta > 0$, and

$$\begin{aligned} \phi &= A\tan(r_{13}, r_{23}) && (2.56)\\ \psi &= A\tan(-r_{31}, r_{32}). && (2.57) \end{aligned}$$

If we choose the value for θ given by Equation (2.55), then $s_\theta < 0$, and

$$\begin{aligned} \phi &= A\tan(-r_{13}, -r_{23}) && (2.58)\\ \psi &= A\tan(r_{31}, -r_{32}). && (2.59) \end{aligned}$$

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

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

$$R = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & \pm 1. \end{bmatrix} \tag{2.60}$$

If $r_{33} = 1$, then $c_\theta = 1$ and $s_\theta = 0$, so that $\theta = 0$. In this case (2.52) becomes

$$\begin{bmatrix} c_\phi c_\psi - s_\phi s_\psi & -c_\phi s_\psi - s_\phi c_\psi & 0 \\ s_\phi c_\psi + c_\phi s_\psi & -s_\phi s_\psi + c_\phi c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\phi+\psi} & -s_{\phi+\psi} & 0 \\ s_{\phi+\psi} & c_{\phi+\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.61}$$

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

$$\begin{aligned} \phi + \psi &= A\tan(r_{11}, r_{21}) \\ &= A\tan(r_{11}, -r_{12}). \end{aligned} \tag{2.62}$$

Since only the sum $\phi + \psi$ can be determined in this case there are infinitely many solutions. We may take $\phi = 0$ by convention, and define ψ by (2.60). If $r_{33} = -1$, then $c_\theta = -1$ and $s_\theta = 0$, so that $\theta = \pi$. In this case (2.52) becomes

$$\begin{bmatrix} -c_{\phi-\psi} & -s_{\phi-\psi} & 0 \\ s_{\phi-\psi} & c_{\phi-\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{2.63}$$

The solution is thus

$$\phi - \psi = A\tan(-r_{11}, -r_{12}) = A\tan(-r_{21}, -r_{22}). \tag{2.64}$$

As before there are infinitely many solutions.

2.5.2 Roll, Pitch, Yaw Angles

A rotation matrix R can also be described as a product of successive rotations about the principal coordinate axes x_0, y_0, and z_0 taken in a specific order. These rotations define the **roll**, **pitch**, and **yaw** angles, which we shall also denote ϕ, θ, ψ, and which are shown in Figure 2.12. We specify the order of rotation as $x - y - z$, in other words, first a yaw about x_0 through an angle ψ, then pitch about the y_0 by an angle θ, and finally roll about the z_0 by an angle ϕ. Since the successive rotations are relative to the fixed frame, the resulting

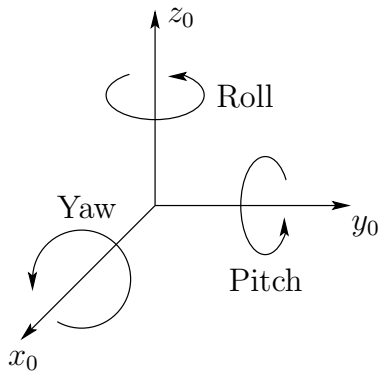


Figure 2.12: Roll, pitch, and yaw angles.

transformation matrix is given by

$$\begin{aligned} R_1^0 &= R_{z,\phi} R_{y,\theta} R_{x,\psi} \\ &= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix} \\ &= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}. \end{aligned} \tag{2.65}$$

Of course, instead of yaw-pitch-roll relative to the fixed frames we could also interpret the above transformation as roll-pitch-yaw, in that order, each taken with respect to the current frame. The end result is the same matrix (2.65).

The three angles, ϕ, θ, ψ, can be obtained for a given rotation matrix using a method that is similar to that used to derive the Euler angles above. We leave this as an exercise for the reader.

2.5.3 Axis/Angle Representation

Rotations are not always performed about the principal coordinate axes. We are often interested in a rotation about an arbitrary axis in space. This provides both a convenient way to describe rotations, and an alternative parameterization for rotation matrices. Let $\boldsymbol{k} = (k_x, k_y, k_z)^T$, expressed in the frame $o_0 x_0 y_0 z_0$, be a unit vector defining an axis. We wish to derive the rotation matrix $R_{\boldsymbol{k},\theta}$ representing a rotation of θ degrees about this axis.

There are several ways in which the matrix $R_{\boldsymbol{k},\theta}$ can be derived. Perhaps the simplest way is to rotate the vector $\boldsymbol{k}$ into one of the coordinate axes, say z_0, then rotate about z_0 by θ and finally rotate $\boldsymbol{k}$ back to its original position. This is similar to the method that we employed above to derive the equation for rotation with respect to a fixed reference frame. Referring to Figure 2.13 we see that we can rotate $\boldsymbol{k}$ into z_0 by first rotating about z_0 by

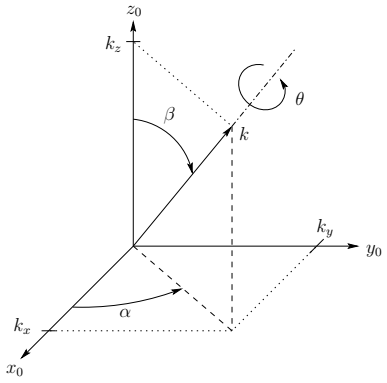


Figure 2.13: Rotation about an arbitrary axis.

$-\alpha$, then rotating about y_0 by $-\beta$. Since all rotations are performed relative to the fixed frame $o_0x_0y_0z_0$ the matrix $R_{\boldsymbol{k},\theta}$ is obtained as

$$R_{\boldsymbol{k},\theta} = R_{z,\alpha}R_{y,\beta}R_{z,\theta}R_{y,-\beta}R_{z,-\alpha}. \tag{2.66}$$

From Figure 2.13, since $\boldsymbol{k}$ is a unit vector, we see that

$$\sin\alpha = \frac{k_y}{\sqrt{k_x^2+k_y^2}} \tag{2.67}$$

$$\cos\alpha = \frac{k_x}{\sqrt{k_x^2+k_y^2}} \tag{2.68}$$

$$\sin\beta = \sqrt{k_x^2+k_y^2} \tag{2.69}$$

$$\cos\beta = k_z. \tag{2.70}$$

Note that the final two equations follow from the fact that $\boldsymbol{k}$ is a unit vector. Substituting (2.67)-(2.70) into (2.66) we obtain after some lengthy calculation (Problem 2.10)

$$R_{\boldsymbol{k},\theta} = \left[\begin{array}{c|c|c} k_x^2v_\theta + c_\theta & k_xk_yv_\theta - k_zs_\theta & k_xk_zv_\theta + k_ys_\theta \\ k_xk_yv_\theta + k_zs_\theta & k_y^2v_\theta + c_\theta & k_yk_zv_\theta - k_xs_\theta \\ k_xk_zv_\theta - k_ys_\theta & k_yk_zv_\theta + k_xs_\theta & k_z^2v_\theta + c_\theta \end{array}\right] \tag{2.71}$$

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

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

$$R = R_{\boldsymbol{k},\theta} \tag{2.72}$$

where $\boldsymbol{k}$ is a unit vector defining the axis of rotation, and θ is the angle of rotation about $\boldsymbol{k}$. Equation (2.72) is called the **axis/angle representation** of R. Given an arbitrary

rotation matrix R with components (r_{ij}), the equivalent angle θ and equivalent axis $\boldsymbol{k}$ are given by the expressions

$$\begin{aligned} \theta &= \cos^{-1}\left(\frac{Tr(R)-1}{2}\right) \\ &= \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right) \end{aligned} \tag{2.73}$$

where Tr denotes the trace of R, and

$$\boldsymbol{k} = \frac{1}{2\sin\theta}\begin{bmatrix} \boldsymbol{r}_{32}-\boldsymbol{r}_{23} \\ \boldsymbol{r}_{13}-\boldsymbol{r}_{31} \\ \boldsymbol{r}_{21}-\boldsymbol{r}_{12} \end{bmatrix}. \tag{2.74}$$

These equations can be obtained by direct manipulation of the entries of the matrix given in equation (2.71). The axis/angle representation is not unique since a rotation of $-\theta$ about $-\boldsymbol{k}$ is the same as a rotation of θ about $\boldsymbol{k}$, that is,

$$R_{\boldsymbol{k},\theta} = R_{-\boldsymbol{k},-\theta}. \tag{2.75}$$

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

Example 2.7 *Suppose R is generated by a rotation of 90° about z_0 followed by a rotation of 30° about y_0 followed by a rotation of 60° about x_0. Then*

$$\begin{aligned} R &= R_{x,60}R_{y,30}R_{z,90} \\ &= \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{4} & -\frac{3}{4} \\ \frac{\sqrt{3}}{2} & \frac{1}{4} & \frac{\sqrt{3}}{4} \end{bmatrix}. \end{aligned} \tag{2.76}$$

We see that $Tr(R) = 0$ and hence the equivalent angle is given by (2.73) as

$$\theta = \cos^{-1}\left(-\frac{1}{2}\right) = 120^\circ. \tag{2.77}$$

The equivalent axis is given from (2.74) as

$$\boldsymbol{k} = \left(\frac{1}{\sqrt{3}}, \frac{1}{2\sqrt{3}} - \frac{1}{2}, \frac{1}{2\sqrt{3}} + \frac{1}{2}\right)^T. \tag{2.78}$$

$\diamond$

The above axis/angle representation characterizes a given rotation by four quantities, namely the three components of the equivalent axis $\boldsymbol{k}$ and the equivalent angle θ. However, since the equivalent axis $\boldsymbol{k}$ is given as a unit vector only two of its components are independent. The third is constrained by the condition that $\boldsymbol{k}$ is of unit length. Therefore, only

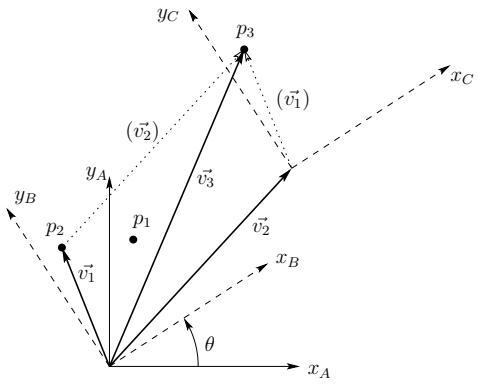


Figure 2.14: Homogeneous transformations in two dimensions.

three independent quantities are required in this representation of a rotation R. We can represent the equivalent angle/axis by a single vector r as

$$\boldsymbol{r} \quad = \quad (r_x, r_y, r_z)^T = (\theta k_x, \theta k_y, \theta k_z)^T. \tag{2.79}$$

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

2.6 Homogeneous Transformations

We have seen how to represent both positions and orientations. In this section we combine these two concepts to define homogeneous transformations.

Consider Figure 2.14. In this figure, frame $o_1x_1y_1$ is obtained by rotating frame $o_0x_0y_0$ by angle θ, and frame $o_2x_2y_2$ is obtained by subsequently translating frame $o_1x_1y_1$ by the displacement $\vec{v}_2$. If we consider the point p_1 as being rigidly attached to coordinate frame $o_0x_0y_0$ as these transformations are performed, then p_2 is the location of p_1 after the rotation, and p_3 is the location of p_1 after the translation. If we are given the coordinates of the point p_3 with respect to frame $o_2x_2y_2$, and if we know the rotation and translation that are applied to obtain frame $o_2x_2y_2$, it is straightforward to compute the coordinates of the point p_3 with respect to $o_0x_0y_0$. To see this, note the point p_3 is displaced by the vector $\vec{v}_3$ from the origin of $o_0x_0y_0$. Further, we see that $\vec{v}_3 = \vec{v}_1 + \vec{v}_2$. Therefore, to solve our problem, we need only find coordinate assignments for the vectors $\vec{v}_1$ and $\vec{v}_2$ with respect to frame $o_0x_0y_0$. Once we have these coordinate assignments, we can compute the coordinates for $\vec{v}_3$ with respect to $o_0x_0y_0$ using the equation $v_3^0 = v_1^0 + v_2^0$.

We can obtain coordinates for the vector $\vec{v}_1$ by applying the rotation matrix to the coordinates that represent p_2 in frame $o_1x_1y_1$,

$$v_1^0 = R_1^0 p_2^1 \tag{2.80}$$
$$= R_2^0 p_3^2, \tag{2.81}$$

where the second equality follows because the orientations of $o_1x_1y_1$ and $o_2x_2y_2$ are the same and because $p_2^1 = p_3^2$. If we denote the coordinate assignment for $\vec{v}_2$ by d_2^0 (which denotes the displacement of the origin of $o_2x_2y_2$, expressed relative to $o_0x_0y_0$), we obtain

$$p_3^0 = R_2^0 p_3^2 + d_2^0. \tag{2.82}$$

Note that no part of the derivation above was dependent on the fact that we used a two-dimensional space. This same derivation can be applied in three dimensions to obtain the following rule for coordinate transformations.
If frame $o_1x_1y_1z_1$ is obtained from frame $o_0x_0y_0z_0$ by first applying a rotation specified by R_1^0 followed by a translation given (with respect to $o_0x_0y_0z_0$) by d_1^0, then the coordinates p^0 are given by

$$p^0 = R_1^0 p^1 + d_1^0. \tag{2.83}$$

In this text, we will consider only geometric relationships between two coordinate systems that can be expressed as the combination of a pure rotation and a pure translation.

Definition 2.1 *A transformation of the form given in Equation (2.83) is said to define a* **rigid motion** *if R is orthogonal.*

Note that the definition of a rigid motion includes **reflections** when $\det R = -1$. In our case we will never have need for the most general rigid motion, so we assume always that $R \in SO(3)$.

If we have the two rigid motions

$$p^0 = R_1^0 p^1 + d_1^0 \tag{2.84}$$

and

$$p^1 = R_2^1 p^2 + d_2^1 \tag{2.85}$$

then their composition defines a third rigid motion, which we can describe by substituting the expression for p^1 from (2.85) into (2.84)

$$p^0 = R_1^0 R_2^1 p^2 + R_1^0 d_2^1 + d_1^0. \tag{2.86}$$

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

$$p^0 = R_2^0 p^2 + d_2^0. \tag{2.87}$$

Comparing Equations (2.86) and (2.87) we have the relationships

$$R_2^0 = R_1^0 R_2^1 \tag{2.88}$$
$$d_2^0 = d_1^0 + R_1^0 d_2^1. \tag{2.89}$$

Equation (2.88) shows that the orientation transformations can simply be multiplied together and Equation (2.89) shows that the vector from the origin o_0 to the origin o_2 has coordinates given by the sum of d_1^o (the vector from o_0 to o_1 expressed with respect to $o_0x_0y_0z_0$) and $R_1^0 d_2^1$ (the vector from o_1 to o_2, expressed in the orientation of the coordinate system $o_0x_0y_0z_0$).

A comparison of this with the matrix identity

$$\begin{bmatrix} R_1^0 & d_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_2^1 & d_1^2 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R_1^0 R_2^1 & R_1^0 d_1^2 + d_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.90}$$

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

$$H = \begin{bmatrix} R & \boldsymbol{d} \\ \mathbf{0} & 1 \end{bmatrix}; R \in SO(3). \tag{2.91}$$

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

$$H^{-1} = \begin{bmatrix} R^T & -R^T\boldsymbol{d} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.92}$$

Transformation matrices of the form (2.91) are called **homogeneous transformations**. In order to represent the transformation (2.83) by a matrix multiplication, one needs to augment the vectors p^0 and p^1 by the addition of a fourth component of 1 as follows. Set

$$P^0 = \begin{bmatrix} p^0 \\ 1 \end{bmatrix} \tag{2.93}$$
$$P^1 = \begin{bmatrix} p^1 \\ 1 \end{bmatrix}. \tag{2.94}$$

The vectors P^0 and P^1 are known as **homogeneous representations** of the vectors p^0 and p^1, respectively. It can now be seen directly that the transformation (2.83) is equivalent to the (homogeneous) matrix equation

$$P^0 = H_1^0 P^1 \tag{2.95}$$

The set of all 4×4 matrices H of the form (2.91) is denoted by $E(3)$.[3] A set of **basic homogeneous transformations** generating $E(3)$ is given by

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.96}$$

[3]The notation $E(3)$ stands for Euclidean group of order 3.

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad Rot_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.97}$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad Rot_{x,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.98}$$

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

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

$$H_1^0 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_x & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{s} & \boldsymbol{a} & \boldsymbol{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.99}$$

In the above equation $\boldsymbol{n} = (n_x, n_y, n_z)^T$ is a vector representing the direction of x_1 in the $o_0x_0y_0z_0$ system, $\boldsymbol{s} = (s_x, s_y, s_z)^T$ represents the direction of y_1, and $\boldsymbol{a} = (a_x, a_y, a_z)^T$ represents the direction of z_1. The vector $\boldsymbol{d} = (d_x, d_y, d_z)^T$ represents the vector from the origin o_0 to the origin o_1 expressed in the frame $o_0x_0y_0z_0$. The rationale behind the choice of letters $\boldsymbol{n}$, $\boldsymbol{s}$ and $\boldsymbol{a}$ is explained in Chapter 3. *NOTE: The same interpretation regarding composition and ordering of transformations holds for 4×4 homogeneous transformations as for 3×3 rotations.*

Example 2.8 *The homogeneous transformation matrix H that represents a rotation of α degrees about the current x-axis followed by a translation of b units along the current x-axis, followed by a translation of d units along the current z-axis, followed by a rotation of θ degrees about the current z-axis, is given by*

$$\begin{aligned} H &= Rot_{x,\alpha}\, Trans_{x,b}\, Trans_{z,d} Rot_{z,\theta} \qquad (2.100) \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_\theta & -s_\theta & 0 & b \\ c_\alpha s_\alpha & c_\alpha c_\theta & -s_\alpha & -s_\alpha d \\ s_\alpha s_\theta & s_\alpha c_\theta & c_\alpha & c_\alpha d \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

◇

The homogeneous representation (2.91) is a special case of homogeneous coordinates, which have been extensively used in the field of computer graphics. There, one is, in addition, interested in scaling and/or perspective transformations. The most general homogeneous transformation takes the form

$$H = \left[\begin{array}{c|c} R_{3\times 3} & \boldsymbol{d}_{3\times 1} \\ \hline \boldsymbol{f}_{1\times 3} & s_{1\times 1} \end{array}\right] = \left[\begin{array}{c|c} Rotation & Translation \\ \hline perspective & scale\ factor \end{array}\right]. \tag{2.101}$$

For our purposes we always take the last row vector of H to be $(0, 0, 0, 1)$, although the more general form given by (2.101) could be useful, for example, for interfacing a vision system into the overall robotic system or for graphic simulation.

Chapter 3

FORWARD KINEMATICS: THE DENAVIT-HARTENBERG CONVENTION

In this chapter we develop the **forward** or **configuration kinematic equations** for rigid robots. The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. Stated more formally, the forward kinematics problem is to determine the position and orientation of the end-effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints. The forward kinematics problem is to be contrasted with the **inverse** kinematics problem, which will be studied in the next chapter, and which is concerned with determining values for the joint variables that achieve a desired position and orientation for the end-effector of the robot.

3.1 Kinematic Chains

As described in Chapter 1, a robot manipulator is composed of a set of links connected together by various joints. The joints can either be very simple, such as a revolute joint or a prismatic joint, or else they can be more complex, such as a ball and socket joint. (Recall that a revolute joint is like a hinge and allows a relative rotation about a single axis, and a prismatic joint permits a linear motion along a single axis, namely an extension or retraction.) The difference between the two situations is that, in the first instance, the joint has only a single degree-of-freedom of motion: the angle of rotation in the case of a revolute joint, and the amount of linear displacement in the case of a prismatic joint. In contrast, a ball and socket joint has two degrees-of-freedom. In this book it is assumed throughout that all joints have only a single degree-of-freedom. Note that the assumption

does not involve any real loss of generality, since joints such as a ball and socket joint (two degrees-of-freedom) or a spherical wrist (three degrees-of-freedom) can always be thought of as a succession of single degree-of-freedom joints with links of length zero in between.

With the assumption that each joint has a single degree-of-freedom, the action of each joint can be described by a single real number: the angle of rotation in the case of a revolute joint or the displacement in the case of a prismatic joint. The objective of forward kinematic analysis is to determine the *cumulative* effect of the entire set of joint variables. In this chapter we will develop a set of conventions that provide a systematic procedure for performing this analysis. It is, of course, possible to carry out forward kinematics analysis even without respecting these conventions, as we did for the two-link planar manipulator example in Chapter 1. However, the kinematic analysis of an n-link manipulator can be extremely complex and the conventions introduced below simplify the analysis considerably. Moreover, they give rise to a universal language with which robot engineers can communicate.

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

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

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

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

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

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

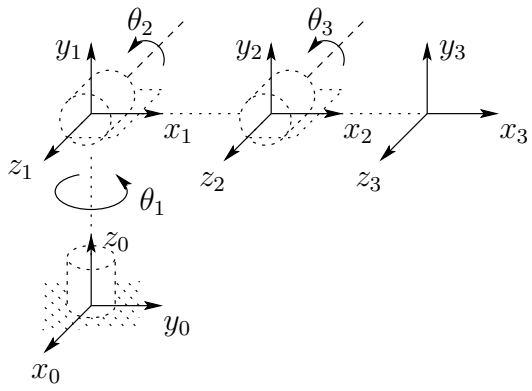


Figure 3.1: Coordinate frames attached to elbow manipulator.

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

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

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

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

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

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

Each homogeneous transformation A_i is of the form

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

Hence

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

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

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

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

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

These expressions will be useful in Chapter 5 when we study Jacobian matrices.

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

3.2 Denavit Hartenberg Representation

While it is possible to carry out all of the analysis in this chapter using an arbitrary frame attached to each link, it is helpful to be systematic in the choice of these frames. A commonly used convention for selecting frames of reference in robotic applications is the Denavit-Hartenberg, or D-H convention. In this convention, each homogeneous transformation A_i is represented as a product of four basic transformations

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

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

From Chapter 2 one can see that an arbitrary homogeneous transformation matrix can be characterized by six numbers, such as, for example, three numbers to specify the

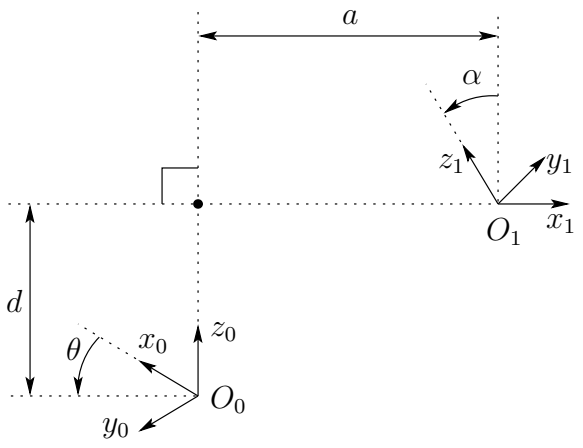


Figure 3.2: Coordinate frames satisfying assumptions DH1 and DH2.

fourth column of the matrix and three Euler angles to specify the upper left 3×3 rotation matrix. In the D-H representation, in contrast, there are only *four* parameters. How is this possible? The answer is that, while frame i is required to be rigidly attached to link i, we have considerable freedom in choosing the origin and the coordinate axes of the frame. For example, it is not necessary that the origin, o_i, of frame i be placed at the physical end of link i. In fact, it is not even necessary that frame i be placed within the physical link; frame i could lie in free space — so long as frame i is *rigidly attached* to link i. By a clever choice of the origin and the coordinate axes, it is possible to cut down the number of parameters needed from six to four (or even fewer in some cases). In Section 3.2.1 we will show why, and under what conditions, this can be done, and in Section 3.2.2 we will show exactly how to make the coordinate frame assignments.

3.2.1 Existence and uniqueness issues

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

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

(DH2) The axis x_1 intersects the axis z_0

as shown in Figure 3.2. Under these conditions, we claim that there exist unique numbers a, d, θ, α such that

$$A = Rot_{z,\theta}\text{Trans}_{z,d}\text{Trans}_{x,a}Rot_{x,\alpha}. \tag{3.11}$$

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

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

and let $\boldsymbol{r}_i$ denote the i^{th} column of the rotation matrix R_1^0. We will now examine the implications of the two DH constraints.

If (DH1) is satisfied, then x_1 is perpendicular to z_0 and we have $x_1 \cdot z_0 = 0$. Expressing this constraint with respect to $o_0x_0y_0z_0$, using the fact that $\boldsymbol{r}_1$ is the representation of the unit vector x_1 with respect to frame 0, we obtain

$$\begin{aligned} 0 &= x_1^0 \cdot z_0^0 && (3.13)\\ &= [r_{11}, r_{21}, r_{31}]^T \cdot [0,0,1]^T && (3.14)\\ &= r_{31}. && (3.15) \end{aligned}$$

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

$$R_1^0 = R_{x,\theta}R_{x,\alpha} = \begin{bmatrix} c_\theta & -s_\theta c_\alpha & s_\theta s_\alpha \\ s_\theta & c_\theta c_\alpha & -c_\theta s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix}. \tag{3.16}$$

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

$$\begin{aligned} r_{11}^2 + r_{21}^2 &= 1, \\ r_{32}^2 + r_{33}^2 &= 1 \end{aligned} \tag{3.17}$$

Hence there exist unique θ, α such that

$$(r_{11}, r_{21}) = (c_\theta, s_\theta), \qquad (r_{33}, r_{32}) = (c_\alpha, s_\alpha). \tag{3.18}$$

Once θ and α are found, it is routine to show that the remaining elements of R_1^0 must have the form shown in (3.16), using the fact that R_1^0 is a rotation matrix.

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

$$\begin{aligned} o_1^0 &= o_0^0 + dz_0^0 + ax_1^0 && (3.19)\\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + d\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + a\begin{bmatrix} c_\theta \\ s_\theta \\ 0 \end{bmatrix} && (3.20)\\ &= \begin{bmatrix} ac_\theta \\ as_\theta \\ d \end{bmatrix}. && (3.21) \end{aligned}$$

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

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

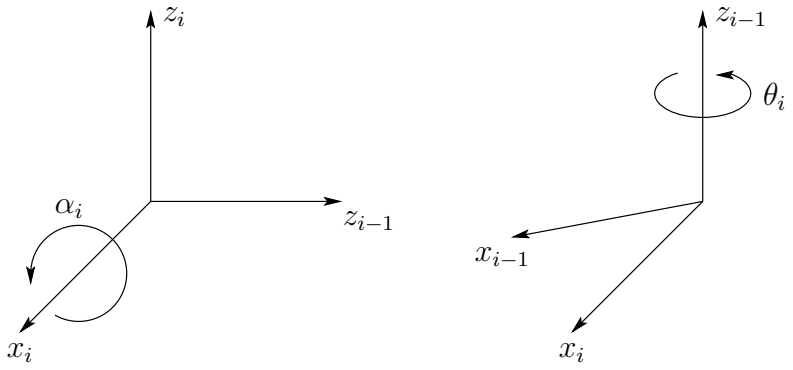


Figure 3.3: Positive sense for α_i and θ_i.

parameter d is the distance between the origin o_0 and the intersection of the x_1 axis with z_0 measured along the z_0 axis. Finally, θ is the angle between x_0 and x_1 measured in a plane normal to z_0. These physical interpretations will prove useful in developing a procedure for assigning coordinate frames that satisfy the constraints (DH1) and (DH2), and we now turn our attention to developing such a procedure.

3.2.2 Assigning the coordinate frames

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

To start, note that the choice of z_i is arbitrary. In particular, from (3.16), we see that by choosing α_i and θ_i appropriately, we can obtain any arbitrary direction for z_i. Thus, for our first step, we assign the axes $z_0, \ldots, z_{n-1}$ in an intuitively pleasing fashion. Specifically, we assign z_i to be the axis of actuation for joint $i+1$. Thus, z_0 is the axis of actuation for joint 1, z_1 is the axis of actuation for joint 2, etc. There are two cases to consider: (i) if joint $i+1$ is revolute, z_i is the axis of revolution of joint $i+1$; (ii) if joint $i+1$ is prismatic, z_i is the axis of translation of joint $i+1$. At first it may seem a bit confusing to associate z_i with joint $i+1$, but recall that this satisfies the convention that we established in Section 3.1, namely that joint i is fixed with respect to frame i, and that when joint i is actuated, link i and its attached frame, $o_i x_i y_i z_i$, experience a resulting motion.

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

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

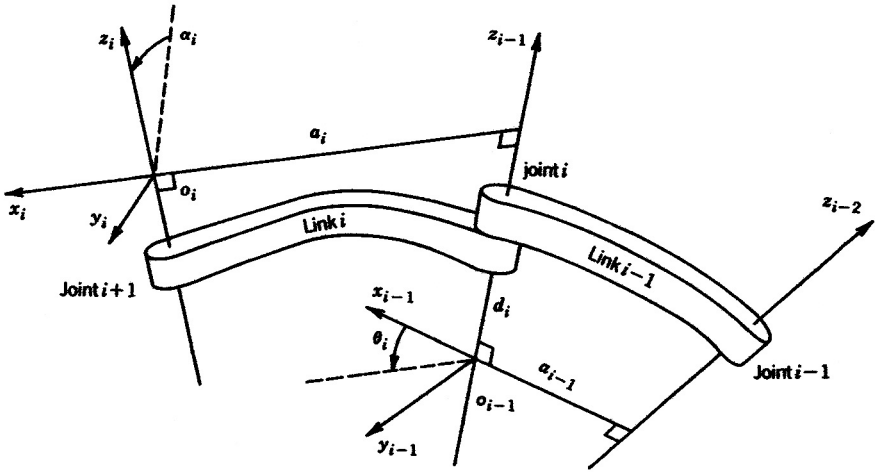


Figure 3.4: Denavit-Hartenberg frame assignment.

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

(i) z_{i-1} and z_i are not coplanar: If z_{i-l} and z_i are not coplanar, then there exists a *unique* line segment perpendicular to both z_{i-1} and z_i such that it connects both lines and it has minimum length. The line containing this common normal to z_{i-1} and z_i defines x_i,

and the point where this line intersects z_i is the origin o_i. By construction, both conditions (DH1) and (DH2) are satisfied and the vector from o_{i-1} to o_i is a linear combination of z_{i-1} and x_i. The specification of frame i is completed by choosing the axis y_i to form a right-hand frame. Since assumptions (DH1) and (DH2) are satisfied the homogeneous transformation matrix A_i is of the form (3.10).

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

(iii) z_{i-1} intersects z_i: In this case x_i is chosen normal to the plane formed by z_i and z_{i-1}. The positive direction of x_i is arbitrary. The most natural choice for the origin o_i in this case is at the point of intersection of z_i and z_{i-1}. However, any convenient point along the axis z_i suffices. Note that in this case the parameter a_i equals 0.

This constructive procedure works for frames $0, \ldots, n-l$ in an n-link robot. To complete the construction, it is necessary to specify frame n. The final coordinate system $o_n x_n y_n z_n$ is commonly referred to as the **end-effector** or **tool frame** (see Figure 3.5). The origin

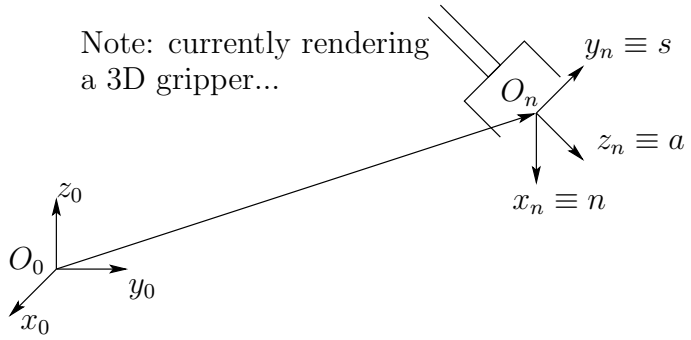


Figure 3.5: Tool frame assignment.

o_n is most often placed symmetrically between the fingers of the gripper. The unit vectors along the x_n, y_n, and z_n axes are labeled as $\boldsymbol{n}$, $\boldsymbol{s}$, and $\boldsymbol{a}$, respectively. The terminology arises from fact that the direction $\boldsymbol{a}$ is the **approach** direction, in the sense that the gripper typically approaches an object along the $\boldsymbol{a}$ direction. Similarly the $\boldsymbol{s}$ direction is the **sliding** direction, the direction along which the fingers of the gripper slide to open and close, and $\boldsymbol{n}$ is the direction **normal** to the plane formed by $\boldsymbol{a}$ and $\boldsymbol{s}$.

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

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

3.2.3 Summary

We may summarize the above procedure based on the D-H convention in the following algorithm for deriving the forward kinematics for any manipulator.

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

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

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

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

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

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

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

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

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

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

$\alpha_i =$ the angle between z_{i-1} and z_i measured about x_i (see Figure 3.3).

$\theta_i =$ the angle between x_{i-1} and x_i measured about z_{i-1} (see Figure 3.3). θ_i is variable if joint i is revolute.

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

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

3.3 Examples

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

Example 3.1 Planar Elbow Manipulator

Consider the two-link planar arm of Figure 3.6. The joint axes z_0 and z_1 are normal to

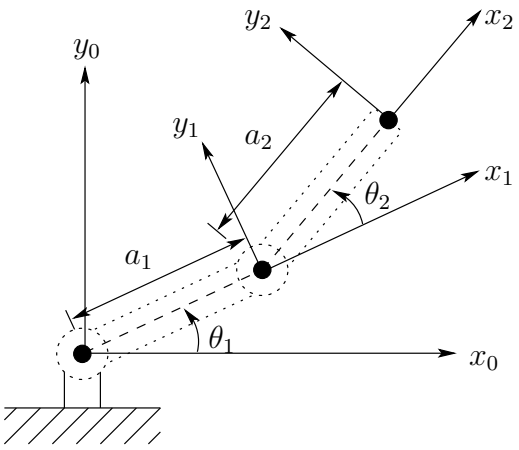


Figure 3.6: Two-link planar manipulator. The z-axes all point out of the page, and are not shown in the figure.

the page. We establish the base frame $o_0x_0y_0z_0$ as shown. The origin is chosen at the point of intersection of the z_0 axis with the page and the direction of the x_0 axis is completely arbitrary. Once the base frame is established, the $o_1x_1y_1z_1$ frame is fixed as shown by the D-H convention, where the origin o_1 has been located at the intersection of z_1 and the page. The final frame $o_2x_2y_2z_2$ is fixed by choosing the origin o_2 at the end of link 2 as shown. The link parameters are shown in Table 3.1. The A-matrices are determined from (3.10) as

Table 3.1: Link parameters for 2-link planar manipulator.

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

* variable

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.22}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

The T-matrices are thus given by

$$T_1^0 = A_1. \tag{3.24}$$

$$T_2^0 = A_1A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.25}$$

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

$$\begin{aligned} x &= a_1c_1 + a_2c_{12} \\ y &= a_1s_1 + a_2s_{12} \end{aligned} \tag{3.26}$$

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

◇

Example 3.2 Three-Link Cylindrical Robot

Consider now the three-link cylindrical robot represented symbolically by Figure 3.7. We establish o_0 as shown at joint 1. Note that the placement of the origin o_0 along z_0 as well as the direction of the x_0 axis are arbitrary. Our choice of o_0 is the most natural, but o_0 could just as well be placed at joint 2. The axis x_0 is chosen normal to the page. Next, since z_0 and z_1 coincide, the origin o_1 is chosen at joint 1 as shown. The x_1 axis is normal to the page when $\theta_1 = 0$ but, of course its direction will change since θ_1 is variable. Since z_2 and z_1 intersect, the origin o_2 is placed at this intersection. The direction of x_2 is chosen

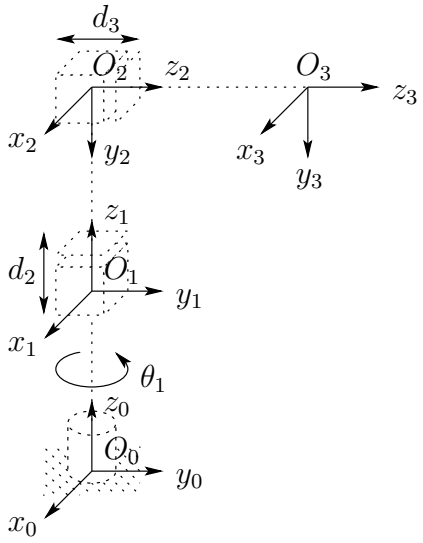


Figure 3.7: Three-link cylindrical manipulator.

Table 3.2: Link parameters for 3-link cylindrical manipulator.

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

* variable

parallel to x_1 so that θ_2 is zero. Finally, the third frame is chosen at the end of link 3 as shown.

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

are

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.27}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$T_3^0 = A_1 A_2 A_3 = \begin{bmatrix} c_1 & 0 & -s_1 & -s_1 d_3 \\ s_1 & 0 & c_1 & c_1 d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.28}$$

◇

Example 3.3 Spherical Wrist

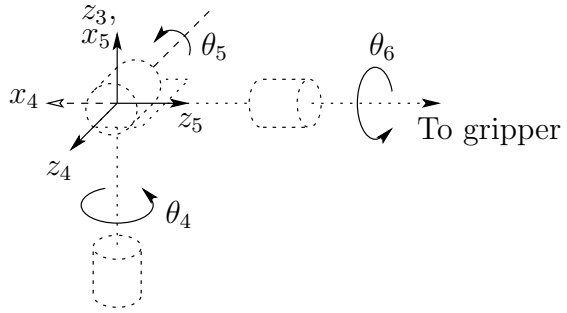


Figure 3.8: The spherical wrist frame assignment.

The spherical wrist configuration is shown in Figure 3.8, in which the joint axes z_3, z_4, z_5 intersect at o. The Denavit-Hartenberg parameters are shown in Table 3.3. The Stanford manipulator is an example of a manipulator that possesses a wrist of this type. In fact, the following analysis applies to virtually all spherical wrists.

We show now that the final three joint variables, θ_4, θ_5, θ_6 are the Euler angles ϕ, θ, ψ, respectively, with respect to the coordinate frame $o_3x_3y_3z_3$. To see this we need only compute

Table 3.3: DH parameters for spherical wrist.

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

* variable

the matrices A_4, A_5, and A_6 using Table 3.3 and the expression (3.10). This gives

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

$$A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

$$A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.31}$$

Multiplying these together yields

$$T_6^3 = A_4 A_5 A_6 = \begin{bmatrix} R_6^3 & o_6^3 \\ 0 & 1 \end{bmatrix} \tag{3.32}$$

$$= \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & c_4s_5d_6 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & s_4s_5d_6 \\ -s_5c_6 & s_5s_6 & c_5 & c_5d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Comparing the rotational part R_6^3 of T_6^3 with the Euler angle transformation (2.51) shows that $\theta_4, \theta_5, \theta_6$ can indeed be identified as the Euler angles ϕ, θ and ψ with respect to the coordinate frame $o_3x_3y_3z_3$.

◇

Example 3.4 Cylindrical Manipulator with Spherical Wrist

Suppose that we now attach a spherical wrist to the cylindrical manipulator of Example 3.3.2 as shown in Figure 3.9. Note that the axis of rotation of joint 4 is parallel to z_2 and thus coincides with the axis z_3 of Example 3.3.2. The implication of this is that we can

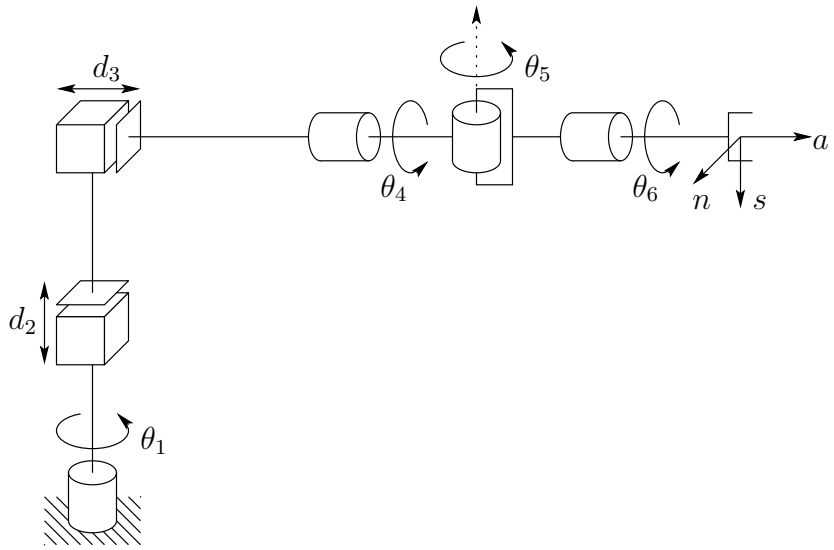


Figure 3.9: Cylindrical robot with spherical wrist.

immediately combine the two previous expression (3.28) and (3.32) to derive the forward kinematics as

$$T_6^0 = T_3^0 T_6^3 \tag{3.33}$$

with T_3^0 given by (3.28) and T_6^3 given by (3.32). Therefore the forward kinematics of this manipulator is described by

$$\begin{aligned} T_6^0 &= \begin{bmatrix} c_1 & 0 & -s_1 & -s_1 d_1 \\ s_1 & 0 & c_1 & c_1 d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 & c_4 s_5 d_6 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 & s_4 s_5 d_6 \\ -s_5 c_6 & s_5 c_6 & c_5 & c_5 d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.34}$$

where

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

Notice how most of the complexity of the forward kinematics for this manipulator results from the orientation of the end-effector while the expression for the arm position from (3.28) is fairly simple. The spherical wrist assumption not only simplifies the derivation of the forward kinematics here, but will also greatly simplify the inverse kinematics problem in the next chapter.

◇

Example 3.5 Stanford Manipulator

Consider now the Stanford Manipulator shown in Figure 3.10. This manipulator is an

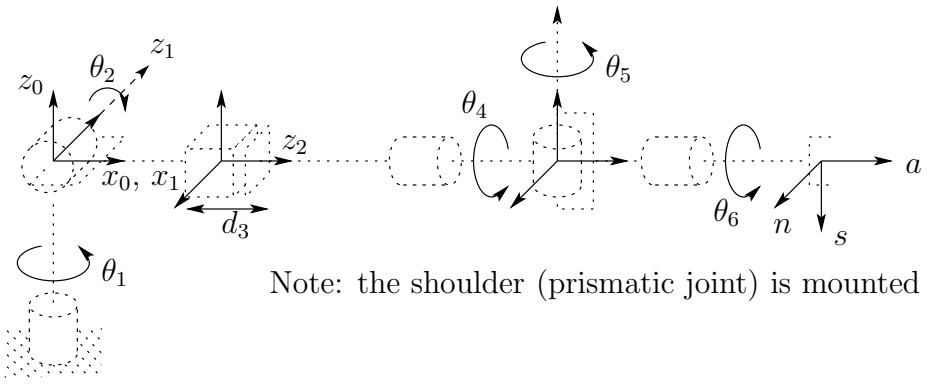


Figure 3.10: DH coordinate frame assignment for the Stanford manipulator.

example of a spherical (RRP) manipulator with a spherical wrist. This manipulator has an offset in the shoulder joint that slightly complicates both the forward and inverse kinematics problems.

Table 3.4: DH parameters for Stanford Manipulator.

Link	d_i	a_i	α_i	θ_i
1	0	0	-90	$\star$
2	d_2	0	$+90$	$\star$
3	$\star$	0	0	0
4	0	0	-90	$\star$
5	0	0	$+90$	$\star$
6	d_6	0	0	$\star$

* joint variable

We first establish the joint coordinate frames using the D-H convention as shown. The link parameters are shown in the Table 3.4.

It is straightforward to compute the matrices A_i as

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

$$A_2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.36}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.37}$$

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.38}$$

$$A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.39}$$

$$A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.40}$$

T_6^0 is then given as

$$T_6^0 = A_1 \cdots A_6 \tag{3.41}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.42}$$

where

$$\begin{aligned}
r_{11} &= c_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] - d_2(s_4c_5c_6 + c_4s_6) \\
r_{21} &= s_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] + c_1(s_4c_5c_6 + c_4s_6) \\
r_{31} &= -s_2(c_4c_5c_6 - s_4s_6) - c_2s_5c_6 \\
r_{12} &= c_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] - s_1(-s_4c_5s_6 + c_4c_6) \\
r_{22} &= -s_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] + c_1(-s_4c_5s_6 + c_4c_6) \\
r_{32} &= s_2(c_4c_5s_6 + s_4c_6) + c_2s_5s_6 \qquad (3.43) \\
r_{13} &= c_1(c_2c_4s_5 + s_2c_5) - s_1s_4s_5 \\
r_{23} &= s_1(c_2c_4s_5 + s_2c_5) + c_1s_4s_5 \\
r_{33} &= -s_2c_4s_5 + c_2c_5 \\
d_x &= c_1s_2d_3 - s_1d_2 + +d_6(c_1c_2c_4s_5 + c_1c_5s_2 - s_1s_4s_5) \\
d_y &= s_1s_2d_3 + c_1d_2 + d_6(c_1s_4s_5 + c_2c_4s_1s_5 + c_5s_1s_2) \\
d_z &= c_2d_3 + d_6(c_2c_5 - c_4s_2s_5). \qquad (3.44)
\end{aligned}$$

◇

Example 3.6 SCARA Manipulator

As another example of the general procedure, consider the SCARA manipulator of Figure 3.11. This manipulator, which is an abstraction of the AdeptOne robot of Figure 1.11, consists of an RRP arm and a one degree-of-freedom wrist, whose motion is a roll about the vertical axis. The first step is to locate and label the joint axes as shown. Since all joint axes are parallel we have some freedom in the placement of the origins. The origins are placed as shown for convenience. We establish the x_0 axis in the plane of the page as shown. This is completely arbitrary and only affects the zero configuration of the manipulator, that is, the position of the manipulator when $\theta_1 = 0$.

The joint parameters are given in Table 3.5, and the A-matrices are as follows.

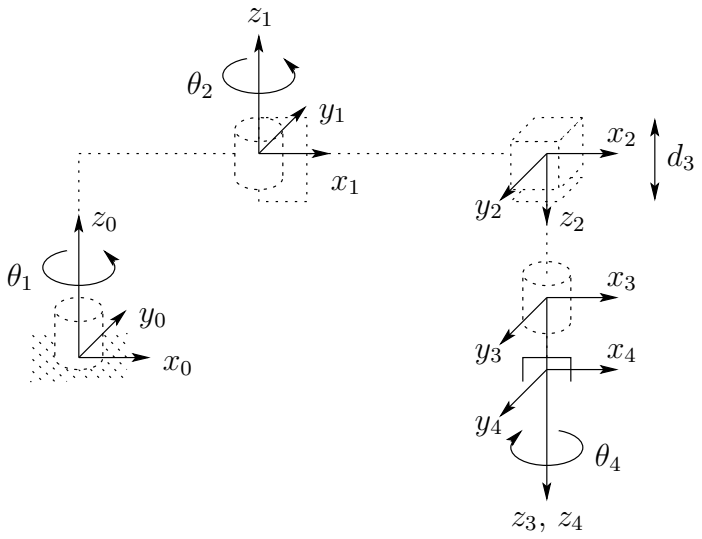


Figure 3.11: DH coordinate frame assignment for the SCARA manipulator.

Table 3.5: Joint parameters for SCARA.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	$\star$
2	a_2	180	0	$\star$
3	0	0	$\star$	0
4	0	0	d_4	$\star$

* joint variable

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.45}$$

$$A_2 = \begin{bmatrix} c_2 & s_2 & 0 & a_2c_2 \\ s_2 & -c_2 & 0 & a_2s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.46}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

$$A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.48}$$

The forward kinematic equations are therefore given by

$$T_4^0 = A_1 \cdots A_4 = \begin{bmatrix} c_{12}c_4 + s_{12}s_4 & -c_{12}s_4 + s_{12}c_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -s_{12}s_4 - c_{12}c_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.49)$$

◇

Chapter 4

INVERSE KINEMATICS

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

In this chapter, we begin by formulating the general inverse kinematics problem. Following this, we describe the principle of kinematic decoupling and how it can be used to simplify the inverse kinematics of most modern manipulators. Using kinematic decoupling, we can consider the position and orientation problems independently. We describe a geometric approach for solving the positioning problem, while we exploit the Euler angle parameterization to solve the orientation problem.

4.1 The General Inverse Kinematics Problem

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

$$H = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{4.1}$$

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

$$T_n^0(q_1, \ldots, q_n) = H \tag{4.2}$$

where

$$T_n^0(q_1, \ldots, q_n) = A_1(q_1) \cdots A_n(q_n). \tag{4.3}$$

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

Equation (4.2) results in twelve nonlinear equations in n unknown variables, which can be written as

$$T_{ij}(q_1, \ldots, q_n) = h_{ij}, \qquad i = 1, 2, 3, \quad j = 1, \ldots, 4 \tag{4.4}$$

where T_{ij}, h_{ij} refer to the twelve nontrivial entries of T_n^0 and H, respectively. (Since the bottom row of both T_n^0 and H are (0,0,0,1), four of the sixteen equations represented by (4.2) are trivial.)

Example 4.1

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

$$H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & o_x \\ r_{21} & r_{22} & r_{23} & o_y \\ r_{31} & r_{32} & r_{33} & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.5}$$

To find the corresponding joint variables θ_1, θ_2, d_3, θ_4, θ_5, and θ_6 we must solve the following simultaneous set of nonlinear trigonometric equations (cf. (3.43) and (3.44)):

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

◇

The equations in the preceding example are, of course, much too difficult to solve directly in closed form. This is the case for most robot arms. Therefore, we need to develop efficient and systematic techniques that exploit the particular kinematic structure of the manipulator. Whereas the forward kinematics problem always has a unique solution that can be obtained simply by evaluating the forward equations, the inverse kinematics problem may or may not have a solution. Even if a solution exists, it may or may not be unique.

Furthermore, because these forward kinematic equations are in general complicated nonlinear functions of the joint variables, the solutions may be difficult to obtain even when they exist.

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

$$q_k = f_k(h_{11}, \ldots, h_{34}), \qquad k = 1, \ldots, n. \tag{4.6}$$

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

The practical question of the existence of solutions to the inverse kinematics problem depends on engineering as well as mathematical considerations. For example, the motion of the revolute joints may be restricted to less than a full 360 degrees of rotation so that not all mathematical solutions of the kinematic equations will correspond to physically realizable configurations of the manipulator. We will assume that the given position and orientation is such that at least one solution of (4.2) exists. Once a solution to the mathematical equations is identified, it must be further checked to see whether or not it satisfies all constraints on the ranges of possible joint motions. For our purposes here we henceforth assume that the given homogeneous matrix H in (4.2) corresponds to a configuration within the manipulator's workspace with an attainable orientation. This then guarantees that the mathematical solutions obtained correspond to achievable configurations.

4.2 Kinematic Decoupling

Although the general problem of inverse kinematics is quite difficult, it turns out that for manipulators having six joints, with the last three joints intersecting at a point (such as the Stanford Manipulator above), it is possible to decouple the inverse kinematics problem into two simpler problems, known respectively, as **inverse position kinematics**, and **inverse orientation kinematics**. To put it another way, for a six-DOF manipulator with a spherical wrist, the inverse kinematics problem may be separated into two simpler problems, namely first finding the position of the intersection of the wrist axes, hereafter called the **wrist center**, and then finding the orientation of the wrist.

For concreteness let us suppose that there are exactly six degrees-of-freedom and that the last three joint axes intersect at a point o_c. We express (4.2) as two sets of equations representing the rotational and positional equations

$$R_6^0(q_1, \ldots, q_6) = R \tag{4.7}$$

$$o_6^0(q_1, \ldots, q_6) = o \tag{4.8}$$

where o and R are the desired position and orientation of the tool frame, expressed with respect to the world coordinate system. Thus, we are given o and R, and the inverse kinematics problem is to solve for $q_1, \ldots, q_6$.

The assumption of a spherical wrist means that the axes z_3, z_4, and z_5 intersect at o_c and hence the origins o_4 and o_5 assigned by the DH-convention will always be at the wrist center o_c. Often o_3 will also be at o_c, but this is not necessary for our subsequent development. The important point of this assumption for the inverse kinematics is that motion of the final three links about these axes will not change the position of o_c, and thus, the position of the wrist center is thus a function of only the first three joint variables.

The origin of the tool frame (whose desired coordinates are given by o) is simply obtained by a translation of distance d_6 along z_5 from o_c (see Table 3.3). In our case, z_5 and z_6 are the same axis, and the third column of R expresses the direction of z_6 with respect to the base frame. Therefore, we have

$$o = o_c^0 + d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.9}$$

Thus in order to have the end-effector of the robot at the point with coordinates given by o and with the orientation of the end-effector given by $R = (r_{ij})$, it is necessary and sufficient that the wrist center o_c have coordinates given by

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.10}$$

and that the orientation of the frame $o_6x_6y_6z_6$ with respect to the base be given by R. If the components of the end-effector position o are denoted o_x, o_y, o_z and the components of the wrist center o_c^0 are denoted x_c, y_c, z_c then (4.10) gives the relationship

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}. \tag{4.11}$$

Using Equation (4.11) we may find the values of the first three joint variables. This determines the orientation transformation R_3^0 which depends only on these first three joint variables. We can now determine the orientation of the end-effector relative to the frame $o_3x_3y_3z_3$ from the expression

$$R = R_3^0 R_6^3 \tag{4.12}$$

as

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R. \tag{4.13}$$

As we shall see in Section 4.4, the final three joint angles can then be found as a set of Euler angles corresponding to R_6^3. Note that the right hand side of (4.13) is completely known since R is given and R_3^0 can be calculated once the first three joint variables are known. The idea of kinematic decoupling is illustrated in Figure 4.1.

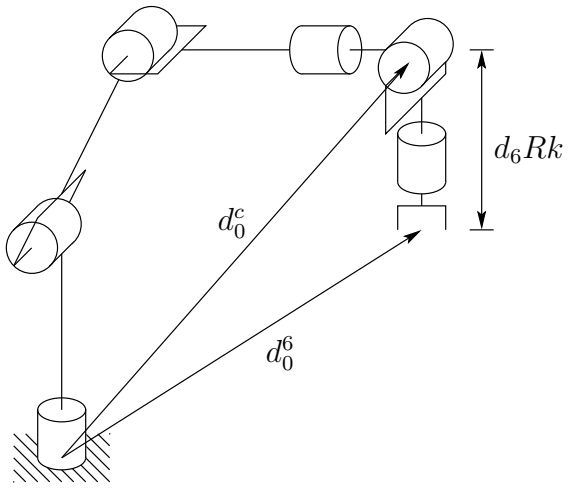


Figure 4.1: Kinematic decoupling.

Summary

For this class of manipulators the determination of the inverse kinematics can be summarized by the following algorithm.

Step 1: Find q_1, q_2, q_3 such that the wrist center o_c has coordinates given by

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.14}$$

Step 2: Using the joint variables determined in Step 1, evaluate R_3^0.

Step 3: Find a set of Euler angles corresponding to the rotation matrix

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R. \tag{4.15}$$

4.3 Inverse Position: A Geometric Approach

For the common kinematic arrangements that we consider, we can use a geometric approach to find the variables, q_1, q_2, q_3 corresponding to o_c^0 given by (4.10). We restrict our treatment to the geometric approach for two reasons. First, as we have said, most present manipulator designs are kinematically simple, usually consisting of one of the five basic configurations of Chapter 1 with a spherical wrist. Indeed, it is partly due to the difficulty of the general inverse kinematics problem that manipulator designs have evolved to their present state. Second, there are few techniques that can handle the general inverse kinematics problem for arbitrary configurations. Since the reader is most likely to encounter robot configurations

of the type considered here, the added difficulty involved in treating the general case seems unjustified. The reader is directed to the references at the end of the chapter for treatment of the general case.

In general the complexity of the inverse kinematics problem increases with the number of nonzero link parameters. For most manipulators, many of the a_i, d_i are zero, the α_i are 0 or $\pm\pi/2$, etc. In these cases especially, a geometric approach is the simplest and most natural. We will illustrate this with several important examples.

Articulated Configuration

Consider the elbow manipulator shown in Figure 4.2, with the components of o_c^0 denoted by x_c, y_c, z_c. We project o_c onto the $x_0 - y_0$ plane as shown in Figure 4.3.

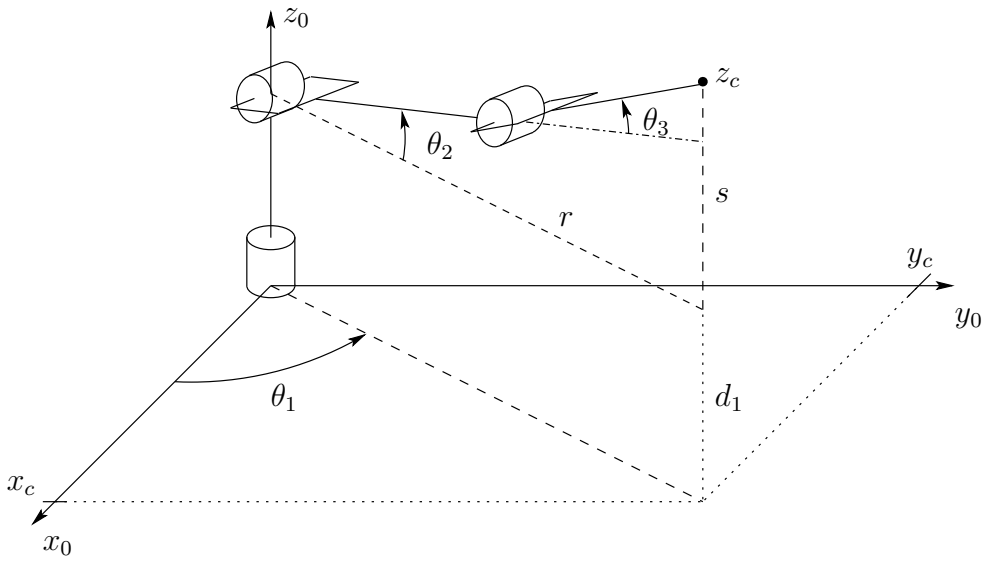


Figure 4.2: Elbow manipulator.

We see from this projection that

$$\theta_1 \quad = \quad A\tan(x_c, y_c), \tag{4.16}$$

in which $A\tan(x, y)$ denotes the two argument arctangent function. $A\tan(x, y)$ is defined for all $(x, y) \neq (0, 0)$ and equals the unique angle θ such that

$$\cos\theta = \frac{x}{(x^2 + y^2)^{\frac{1}{2}}}, \qquad \sin\theta = \frac{y}{(x^2 + y^2)^{\frac{1}{2}}}. \tag{4.17}$$

For example, $A\tan(1, -1) = -\frac{\pi}{4}$, while $A\tan(-1, 1) = +\frac{3\pi}{4}$.

Note that a second valid solution for θ_1 is

$$\theta_1 \quad = \quad \pi + A\tan(x_c, y_c). \tag{4.18}$$

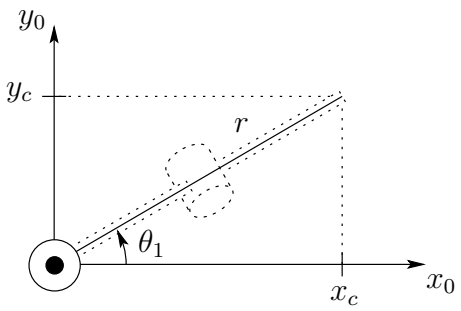


Figure 4.3: Projection of the wrist center onto $x_0 - y_0$ plane.

Of course this will, in turn, lead to different solutions for θ_2 and θ_3, as we will see below.

These solutions for θ_1, are valid unless $x_c = y_c = 0$. In this case (4.16) is undefined and the manipulator is in a singular configuration, shown in Figure 4.4. In this position the

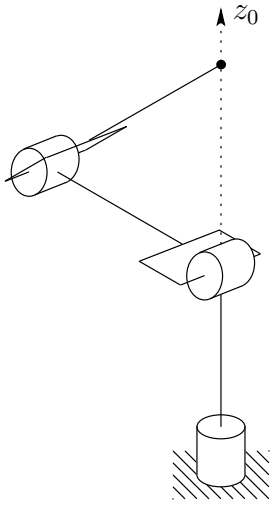


Figure 4.4: Singular configuration.

wrist center o_c intersects z_0; hence any value of θ_1 leaves o_c fixed. There are thus infinitely many solutions for θ_1 when o_c intersects z_0.

If there is an offset $d \neq 0$ as shown in Figure 4.5 then the wrist center cannot intersect z_0. In this case, depending on how the DH parameters have been assigned, we will have $d_2 = d$ or $d_3 = d$. In this case, there will, in general, be only two solutions for θ_1. These correspond to the so-called **left arm** and **right arm** configurations as shown in Figures 4.6 and 4.7. Figure 4.6 shows the left arm configuration. From this figure, we see geometrically that

$$\theta_1 \quad = \quad \phi - \alpha \tag{4.19}$$

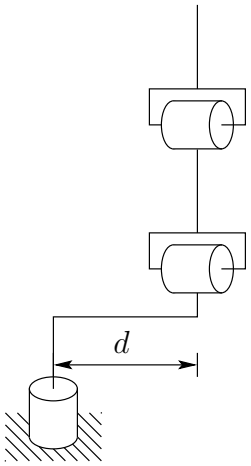


Figure 4.5: Elbow manipulator with shoulder offset.

where

$$\begin{aligned} \phi &= A\tan(x_c, y_c) && (4.20)\\ \alpha &= A\tan\left(\sqrt{r^2 - d^2}, d\right) && (4.21)\\ &= A\tan\left(\sqrt{x_c^2 + y_c^2 - d^2}, d\right). \end{aligned}$$

The second solution, given by the right arm configuration shown in Figure 4.7 is given by

$$\theta_1 = A\tan(x_c, y_c) + A\tan\left(-\sqrt{r^2 - d^2}, -d\right). \quad (4.22)$$

To see this, note that

$$\begin{aligned} \theta_1 &= \alpha + \beta && (4.23)\\ \alpha &= A\tan(x_c, y_c) && (4.24)\\ \beta &= \gamma + \pi && (4.25)\\ \gamma &= A\tan(\sqrt{r^2 - d^2}, d) && (4.26)\\ & && (4.27) \end{aligned}$$

which together imply that

$$\beta = A\tan\left(-\sqrt{r^2 - d^2}, -d\right) \quad (4.28)$$

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

To find the angles θ_2, θ_3 for the elbow manipulator, given θ_1, we consider the plane formed by the second and third links as shown in Figure 4.8. Since the motion of links

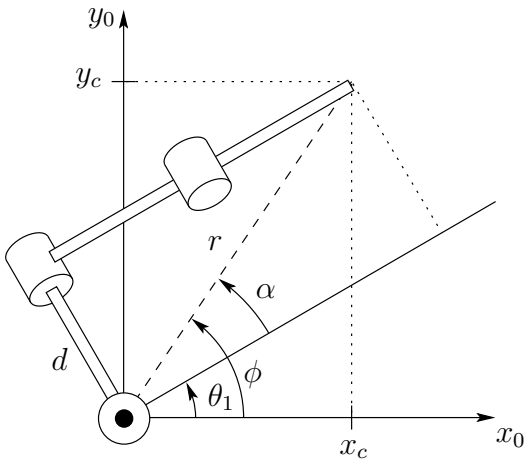


Figure 4.6: Left arm configuration.

two and three is planar, the solution is analogous to that of the two-link manipulator of Chapter 1. As in our previous derivation (cf. (1.8) and (1.9)) we can apply the law of cosines to obtain

$$\begin{aligned} \cos\theta_3 &= \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2a_3} \\ &= \frac{x_c^2 + y_c^2 - d^2 + z_c^2 - a_2^2 - a_3^2}{2a_2a_3} := D, \end{aligned} \tag{4.29}$$

since $r^2 = x_c^2 + y_c^2 - d^2$ and $s = z_c$. Hence, θ_3 is given by

$$\theta_3 = A\tan\left(D, \pm\sqrt{1 - D^2}\right). \tag{4.30}$$

Similarly θ_2 is given as

$$\begin{aligned} \theta_2 &= A\tan(r, s) - A\tan(a_2 + a_3c_3, a_3s_3) \\ &= A\tan\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c\right) - A\tan(a_2 + a_3c_3, a_3s_3). \end{aligned} \tag{4.31}$$

The two solutions for θ_3 correspond to the elbow-up position and elbow-down position, respectively.

An example of an elbow manipulator with offsets is the PUMA shown in Figure 4.9. There are four solutions to the inverse position kinematics as shown. These correspond to the situations left arm-elbow up, left arm–elbow down, right arm–elbow up and right arm–elbow down. We will see that there are two solutions for the wrist orientation thus giving a total of eight solutions of the inverse kinematics for the PUMA manipulator.

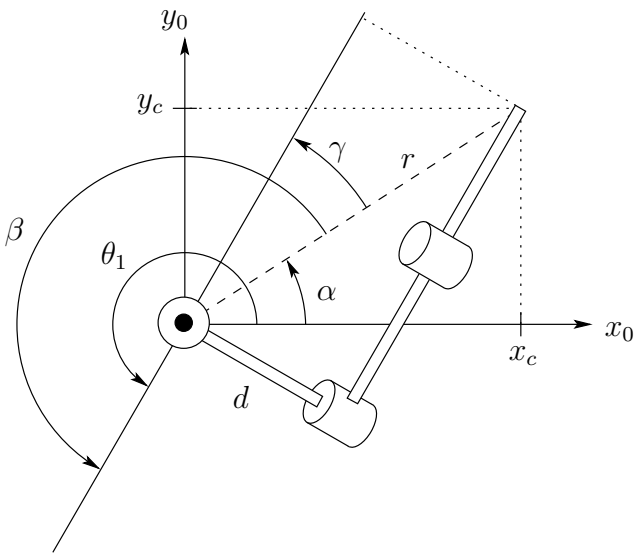


Figure 4.7: Right arm configuration.

Spherical Configuration

We next solve the inverse position kinematics for a three degree of freedom spherical manipulator shown in Figure 4.10. As in the case of the elbow manipulator the first joint variable is the base rotation and a solution is given as

$$\theta_1 \quad = \quad A\tan(x_c, y_c) \tag{4.32}$$

provided x_c and y_c are not both zero. If both x_c and y_c are zero, the configuration is singular as before and θ_1 may take on any value.

The angle θ_2 is given from Figure 4.10 as

$$\theta_2 \quad = \quad A\tan(r, s) + \frac{\pi}{2} \tag{4.33}$$

where $r^2 = x_c^2 + y_c^2$, $s = z_c - d_1$. As in the case of the elbow manipulator a second solution for θ_1 is given by

$$\theta_1 = \pi + A\tan(x_c, y_c); \tag{4.34}$$

The linear distance d_3 is found as

$$d_3 = \sqrt{r^2 + s^2} \quad = \quad \sqrt{x_c^2 + y_c^2 + (z_c - d_1)^2}. \tag{4.35}$$

The negative square root solution for d_3 is disregarded and thus in this case we obtain two solutions to the inverse position kinematics as long as the wrist center does not intersect z_0. If there is an offset then there will be left and right arm configurations as in the case of the elbow manipulator (Problem 4-12).

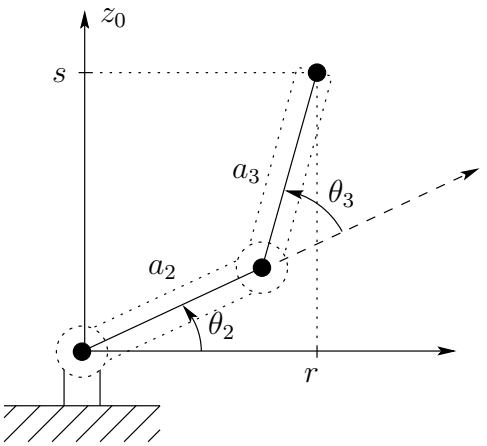


Figure 4.8: Projecting onto the plane formed by links 2 and 3.

4.4 Inverse Orientation

In the previous section we used a geometric approach to solve the inverse position problem. This gives the values of the first three joint variables corresponding to a given position of the wrist origin. The inverse orientation problem is now one of finding the values of the final three joint variables corresponding to a given orientation with respect to the frame $o_3x_3y_3z_3$. For a spherical wrist, this can be interpreted as the problem of finding a set of Euler angles corresponding to a given rotation matrix R. Recall that equation (3.32) shows that the rotation matrix obtained for the spherical wrist has the same form as the rotation matrix for the Euler transformation, given in (2.52). Therefore, we can use the method developed in Section 2.5.1 to solve for the three joint angles of the spherical wrist. In particular, we solve for the three Euler angles, ϕ, θ, ψ, using Equations (2.54) – (2.59), and then use the mapping

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

Example 4.2 Articulated Manipulator with Spherical Wrist

The DH parameters for the frame assignment shown in Figure 4.2 are summarized in Table 4.1. Multiplying the corresponding A_i matrices gives the matrix R_3^0 for the articulated or elbow manipulator as

$$R_3^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 \\ s_1c_{23} & -s_1s_{23} & -c_1 \\ s_{23} & c_{23} & 0 \end{bmatrix}. \tag{4.36}$$

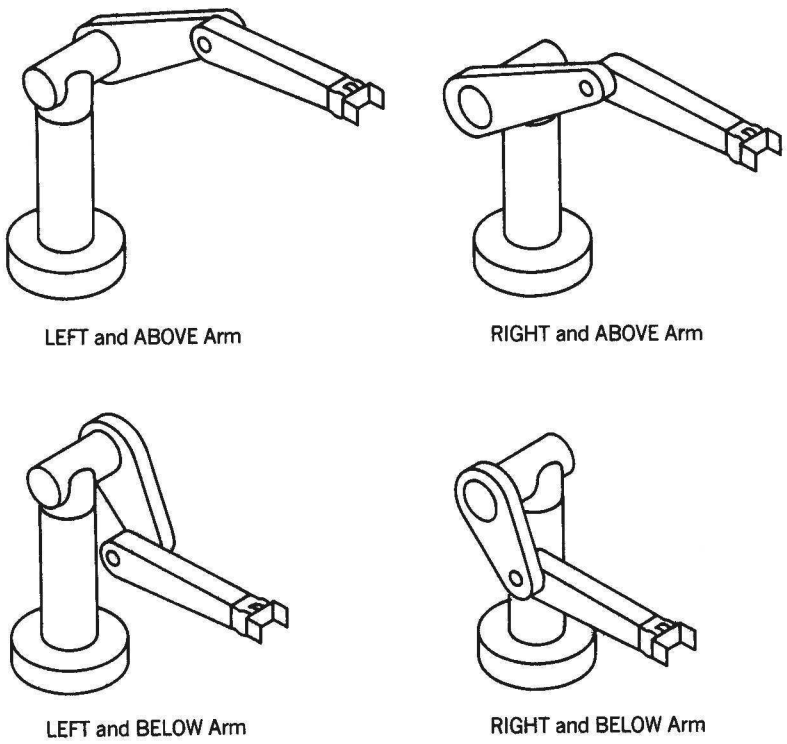


Figure 4.9: Four solutions of the inverse position kinematics for the PUMA manipulator.

Table 4.1: Link parameters for the articulated manipulator of Figure 4.2.

Link	a_i	α_i	d_i	θ_i
1	0	90	d_1	θ_1^*
2	a_2	0	0	θ_2^*
3	a_3	0	0	θ_3^*

* variable

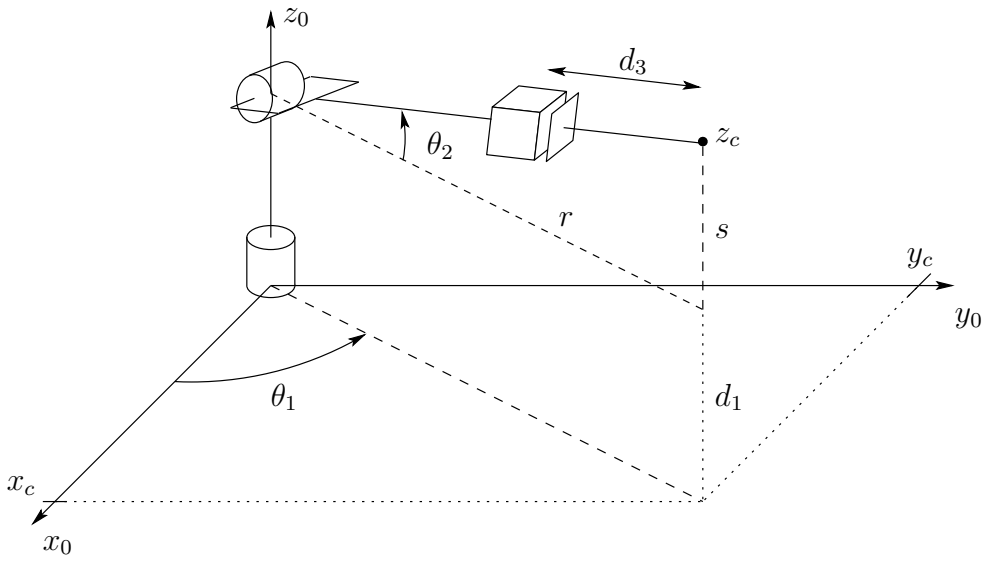


Figure 4.10: Spherical manipulator.

The matrix $R_6^3 = A_4A_5A_6$ *is given as*

$$R_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix}. \tag{4.37}$$

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

$$R_6^3 = (R_3^0)^T R \tag{4.38}$$

and the Euler angle solution can be applied to this equation. For example, the three equations given by the third column in the above matrix equation are given by

$$c_4s_5 = c_1c_{23}r_{13} + s_1c_{23}r_{23} + s_{23}r_{33} \tag{4.39}$$

$$s_4s_5 = -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33} \tag{4.40}$$

$$c_5 = s_1r_{13} - c_1r_{23}. \tag{4.41}$$

Hence, if not both of the expressions (4.39), (4.40) are zero, then we obtain θ_5 *from (2.54) and (2.55) as*

$$\theta_5 = A\tan\left(s_1r_{13} - c_1r_{23}, \pm\sqrt{1-(s_1r_{13} - c_1r_{23})^2}\right). \tag{4.42}$$

If the positive square root is chosen in (4.42), then θ_4 *and* θ_6 *are given by (2.56) and (2.57), respectively, as*

$$\theta_4 = A\tan(c_1c_{23}r_{13} + s_1c_{23}r_{23} + s_{23}r_{33}, \\ -c_1s_{23}r_{13} - s_1s_{23}r_{23} + c_{23}r_{33}) \tag{4.43}$$

$$\theta_6 = A\tan(-s_1r_{11} + c_1r_{21}, s_1r_{12} - c_1r_{22}). \tag{4.44}$$

The other solutions are obtained analogously. If $s_5 = 0$, then joint axes z_3 and z_5 are collinear. This is a singular configuration and only the sum $\theta_4 + \theta_6$ can be determined. One solution is to choose θ_4 arbitrarily and then determine θ_6 using (2.62) or (2.64).
$\diamond$

Example 4.3 Summary of Elbow Manipulator Solution

To summarize the preceding development we write down one solution to the inverse kinematics of the six degree-of-freedom elbow manipulator shown in Figure 4.2 which has no joint offsets and a spherical wrist.

Given

$$o = \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix}; \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.45}$$

then with

$$x_c = o_x - d_6 r_{13} \tag{4.46}$$
$$y_c = o_y - d_6 r_{23} \tag{4.47}$$
$$z_c = o_z - d_6 r_{33} \tag{4.48}$$

a set of D-H joint variables is given by

$$\theta_1 = A\tan(x_c, y_c) \tag{4.49}$$
$$\theta_2 = A\tan\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c\right) - A\tan(a_2 + a_3 c_3, a_3 s_3) \tag{4.50}$$
$$\theta_3 = A\tan\left(D, \pm\sqrt{1 - D^2}\right), \quad \text{where } D = \frac{x_c^2 + y_c^2 - d^2 + z_c^2 - a_2^2 - a_3^2}{2a_2 a_3} \tag{4.51}$$
$$\theta_4 = A\tan(c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33}, \; -c_1 s_{23} r_{13} - s_1 s_{23} r_{23} + c_{23} r_{33}) \tag{4.52}$$
$$\theta_5 = A\tan\left(s_1 r_{13} - c_1 r_{23}, \pm\sqrt{1 - (s_1 r_{13} - c_1 r_{23})^2}\right). \tag{4.53}$$
$$\theta_6 = A\tan(-s_1 r_{11} + c_1 r_{21}, s_1 r_{12} - c_1 r_{22}). \tag{4.54}$$

The other possible solutions are left as an exercise (Problem 4-11). $\diamond$

Example 4.4 SCARA Manipulator

As another example, we consider the SCARA manipulator whose forward kinematics is defined by T_4^0 from (3.49). The inverse kinematics is then given as the set of solutions of the equation

$$\begin{bmatrix} c_{12}c_4 + s_{12}s_4 & s_{12}c_4 - c_{12}s_4 & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12}c_4 - c_{12}s_4 & -c_{12}c_4 - s_{12}s_4 & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}. \tag{4.55}$$

We first note that, since the SCARA has only four degrees-of-freedom, not every possible H from $SE(3)$ allows a solution of (4.55). In fact we can easily see that there is no solution of (4.55) unless R is of the form

$$R = \begin{bmatrix} c_\alpha & s_\alpha & 0 \\ s_\alpha & -c_\alpha & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.56}$$

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

$$\theta_1 + \theta_2 - \theta_4 = \alpha = A\tan(r_{11}, r_{12}). \tag{4.57}$$

Projecting the manipulator configuration onto the $x_0 - y_0$ plane immediately yields the situation of Figure 4.11.

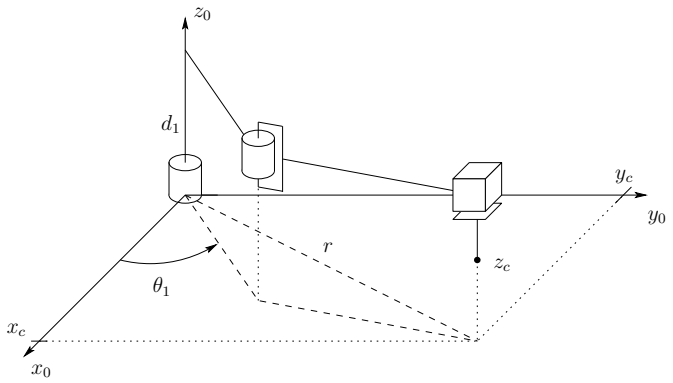


Figure 4.11: SCARA manipulator.

We see from this that

$$\theta_2 = A\tan\left(c_2, \pm\sqrt{1 - c_2}\right) \tag{4.58}$$

where

$$c_2 = \frac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{4.59}$$

$$\theta_1 = A\tan(o_x, o_y) - A\tan(a_1 + a_2c_2, a_2s_2). \tag{4.60}$$

We may then determine θ_4 from (4.57) as

$$\begin{aligned} \theta_4 &= \theta_1 + \theta_2 - \alpha \\ &= \theta_1 + \theta_2 - A\tan(r_{11}, r_{12}). \end{aligned} \tag{4.61}$$

Finally d_3 is given as

$$d_3 = o_z + d_4. \tag{4.62}$$

◇

Chapter 5

VELOCITY KINEMATICS – THE MANIPULATOR JACOBIAN

In the previous chapters we derived the forward and inverse position equations relating joint positions and end-effector positions and orientations. In this chapter we derive the velocity relationships, relating the linear and angular velocities of the end-effector (or any other point on the manipulator) to the joint velocities. In particular, we will derive the angular velocity of the end-effector frame (which gives the rate of rotation of the frame) and the linear velocity of the origin. We will then relate these velocities to the joint velocities, $\dot{q}_i$.

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

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

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

angular velocity of any point on the manipulator. This will be important when we discuss the derivation of the dynamic equations of motion in Chapter 9. We then discuss the notion of **singular configurations**. These are configurations in which the manipulator loses one or more degrees-of-freedom. We show how the singular configurations are determined geometrically and give several examples. Following this, we briefly discuss the inverse problems of determining joint velocities and accelerations for specified end-effector velocities and accelerations. We end the chapter by considering redundant manipulators. This includes discussions of the inverse velocity problem, singular value decomposition and manipulability.

5.1 Angular Velocity: The Fixed Axis Case

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

$$\omega = \dot{\theta}\boldsymbol{k} \tag{5.1}$$

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

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

$$\boldsymbol{v} = \omega \times \boldsymbol{r} \tag{5.2}$$

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

As in previous chapters, in order to specify the orientation of a rigid object, we rigidly attach a coordinate frame to the object, and then specify the orientation of the coordinate frame. Since every point on the object experiences the same angular velocity (each point sweeps out the same angle θ in a given time interval), and since each point of the body is in a fixed geometric relationship to the body-attached frame, we see that the angular velocity is a property of the attached coordinate frame itself. Angular velocity is not a property of individual points. Individual points may experience a *linear velocity* that is induced by an angular velocity, but it makes no sense to speak of a point itself rotating. Thus, in equation (5.2) $\boldsymbol{v}$ corresponds to the linear velocity of a point, while ω corresponds to the angular velocity associated with a rotating coordinate frame.

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

5.2 Skew Symmetric Matrices

In the Section 5.3 we will derive properties of rotation matrices that can be used to computing relative velocity transformations between coordinate frames. Such transformations involve derivatives of rotation matrices. By introducing the notion of a skew symmetric matrix it is possible to simplify many of the computations involved.

Definition 5.1 *A matrix S is said to be* **skew symmetric** *if and only if*

$$S^T + S = 0. \tag{5.3}$$

We denote the set of all 3×3 skew symmetric matrices by $SS(3)$ [1]. If $S \in SS(3)$ has components s_{ij}, $i, j = 1, 2, 3$ then (5.3) is equivalent to the nine equations

$$s_{ij} + s_{ji} = 0 \qquad i, j = 1, 2, 3. \tag{5.4}$$

From (5.4) we see that $s_{ii} = 0$; that is, the diagonal terms of S are zero and the off diagonal terms s_{ij}, $i \neq j$ satisfy $s_{ij} = -s_{ji}$. Thus S contains only three independent entries and every 3×3 skew symmetric matrix has the form

$$S = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}. \tag{5.5}$$

If $\boldsymbol{a} = (a_x, a_y, a_z)^T$ is a 3-vector, we define the skew symmetric matrix $S(\boldsymbol{a})$ as

$$S(\boldsymbol{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{5.6}$$

Example 5.1 *We denote by $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ the three unit basis coordinate vectors,*

$$\begin{aligned} \boldsymbol{i} &= (1, 0, 0)^T \\ \boldsymbol{j} &= (0, 1, 0)^T \\ \boldsymbol{k} &= (0, 0, 1)^T. \end{aligned}$$

[1] In the mathematical literature this set is typically denoted as $so(3)$.

The skew symmetric matrices $S(\boldsymbol{i})$, $S(\boldsymbol{j})$, and $S(\boldsymbol{k})$ are given by

$$S(\boldsymbol{i}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}; \quad S(\boldsymbol{j}) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}; \quad S(\boldsymbol{k}) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{5.7}$$

◇

Properties of Skew Symmetric Matrices

Skew symmetric matrices possess several properties that will prove useful for subsequent derivations [2] Among these properties are

1. *Linearity:*

$$S(\alpha\boldsymbol{a} + \beta\boldsymbol{b}) = \alpha S(\boldsymbol{a}) + \beta S(\boldsymbol{b}). \tag{5.8}$$

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

2.

$$S(\boldsymbol{a})\boldsymbol{p} = \boldsymbol{a} \times \boldsymbol{p} \tag{5.9}$$

for any vectors $\boldsymbol{a}$ and $\boldsymbol{p}$, where $\boldsymbol{a} \times \boldsymbol{p}$ denotes the vector cross product. Equation (5.9) can be verified by direct calculation.

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

$$R(\boldsymbol{a} \times \boldsymbol{b}) = R\boldsymbol{a} \times R\boldsymbol{b}. \tag{5.10}$$

Equation (5.10) is **not** true in general unless R is orthogonal. Equation (5.10) says that if we first rotate the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ using the rotation transformation R and then form the cross product of the rotated vectors $R\boldsymbol{a}$ and $R\boldsymbol{b}$, the result is the same as that obtained by first forming the cross product $\boldsymbol{a} \times \boldsymbol{b}$ and then rotating to obtain $R(\boldsymbol{a} \times \boldsymbol{b})$.

4.

$$RS(\boldsymbol{a})R^T = S(R\boldsymbol{a}) \tag{5.11}$$

for $R \in SO(3)$ and $\boldsymbol{a} \in \mathbf{R}^3$. This property follows easily from (5.9) and (5.10) as follows. Let $b \in \mathbf{R}^3$ be an arbitrary vector. Then

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

and the result follows.

[2] These properties are consequences of the fact that $SS(3)$ is a *Lie Algebra*, a vector space with a suitably defined product operation.

As we will see, (5.11) is one of the most useful expressions that we will derive. The left hand side of Equation (5.11) represents a similarity transformation of the matrix $S(\boldsymbol{a})$. The equation says therefore that the matrix representation of $S(\boldsymbol{a})$ in a coordinate frame rotated by R is the same as the skew symmetric matrix $S(R\boldsymbol{a})$ corresponding to the vector $\boldsymbol{a}$ rotated by R.

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

$$R(\theta)R(\theta)^T = I. \tag{5.12}$$

Differentiating both sides of (5.12) with respect to θ using the product rule gives

$$\frac{dR}{d\theta}R(\theta)^T + R(\theta)\frac{dR^T}{d\theta} = 0. \tag{5.13}$$

Let us define the matrix

$$S := \frac{dR}{d\theta}R(\theta)^T. \tag{5.14}$$

Then the transpose of S is

$$S^T = \left(\frac{dR}{d\theta}R(\theta)^T\right)^T = R(\theta)\frac{dR^T}{d\theta}. \tag{5.15}$$

Equation (5.13) says therefore that

$$S + S^T = 0. \tag{5.16}$$

In other words, the matrix S defined by (5.14) is skew symmetric. Multiplying both sides of (5.14) on the right by R and using the fact that $R^TR = I$ yields

$$\frac{dR}{d\theta} = SR(\theta). \tag{5.17}$$

Equation (5.17) is very important. It says that computing the derivative of the rotation matrix R is equivalent to a matrix multiplication by a skew symmetric matrix S. The most commonly encountered situation is the case where R is a basic rotation matrix or a product of basic rotation matrices.

Example 5.2 *If $R = R_{x,\theta}$, the basic rotation matrix given by (2.19), then direct computation shows that*

$$S = \frac{dR}{d\theta}R^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\theta & -\cos\theta \\ 0 & \cos\theta & -\sin\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{5.18}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = S(\boldsymbol{i}). \tag{5.19}$$

Thus we have shown that

$$\frac{dR_{x,\theta}}{d\theta} = S(\boldsymbol{i})R_{x,\theta}. \tag{5.20}$$

Similar computations show that

$$\frac{dR_{y,\theta}}{d\theta} = S(\boldsymbol{j})R_{y,\theta}; \qquad \frac{dR_{z,\theta}}{d\theta} = S(\boldsymbol{k})R_{z,\theta}. \tag{5.21}$$

◇

Example 5.3 *Let $R_{k,\theta}$ be a rotation about the axis defined by $\boldsymbol{k}$ as in (2.71). Note that in this example $\boldsymbol{k}$ is not the unit coordinate vector $(0,0,1)^T$. It is easy to check that $S(\boldsymbol{k})^3 = -S(\boldsymbol{k})$. Using this fact together with Problem* **??** *it follows that*

$$\frac{dR_{k,\theta}}{d\theta} = S(\boldsymbol{k})R_{k,\theta}. \tag{5.22}$$

◇

5.3 Angular Velocity: The General Case

We now consider the general case of angular velocity about an arbitrary, possibly moving, axis. Suppose that a rotation matrix R is time varying, so that $R = R(t) \in SO(3)$ for every $t \in \mathbf{R}$. Assuming that $R(t)$ is continuously differentiable as a function of t an argument identical to the one in the previous section shows that the time derivative $\dot{R}(t)$ of $R(t)$ is given by

$$\dot{R}(t) = S(t)R(t) \tag{5.23}$$

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

$$\dot{R}(t) = S(\boldsymbol{\omega}(t))R(t) \tag{5.24}$$

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

Example 5.4 *Suppose that $R(t) = R_{x,\theta(t)}$. Then $\dot{R}(t) = \frac{dR}{dt}$ is computed using the chain rule as*

$$\dot{R} = \frac{dR}{d\theta}\frac{d\theta}{dt} = \dot{\theta}S(\boldsymbol{i})R(t) = S(\boldsymbol{\omega}(t))R(t) \tag{5.25}$$

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

◇

5.4 Addition of Angular Velocities

We are often interested in finding the resultant angular velocity due to the relative rotation of several coordinate frames. We now derive the expressions for the composition of angular velocities of two moving frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ relative to the fixed frame $o_0x_0y_0z_0$. For now, we assume that the three frames share a common origin. Let the relative orientations of the frames $o_1x_1y_1z_1$ and $o_2x_2y_2z_2$ be given by the rotation matrices $R_1^0(t)$ and $R_2^1(t)$ (both time varying). As in Chapter 2,

$$R_2^0(t) = R_1^0(t)R_2^1(t). \tag{5.26}$$

Taking derivatives of both sides of (5.26) with respect to time yields

$$\dot{R}_2^0 = \dot{R}_1^0 R_2^1 + R_1^0 \dot{R}_2^1. \tag{5.27}$$

Using (5.24), the term $\dot{R}_2^0$ on the left-hand side of (5.27) can be written

$$\dot{R}_2^0 = S(\omega_2^0)R_2^0. \tag{5.28}$$

In this expression, ω_2^0 denotes the total angular velocity experienced by frame $o_2x_2y_2z_2$. This angular velocity results from the combined rotations expressed by R_1^0 and R_2^1.

The first term on the right-hand side of (5.27) is simply

$$\dot{R}_1^0 R_2^1 = S(\omega_a^0)R_1^0R_2^1 = S(\omega_a^0)R_2^0. \tag{5.29}$$

Note that in this equation, ω_a^0 denotes the angular velocity of frame $o_1x_1y_1z_1$ that results from the changing R_1^0, and this angular velocity vector is expressed relative to the coordinate system $o_0x_0y_0z_0$.

Let us examine the second term on the right hand side of (5.27). Using the expression (5.11) we have

$$\begin{aligned} R_1^0\dot{R}_2^1 &= R_1^0 S(\omega_b^1)R_2^1 && (5.30)\\ &= R_1^0 S(\omega_b^1){R_1^0}^T R_1^0 R_2^1 = S(R_1^0\omega_b^1)R_1^0R_2^1 \\ &= S(R_1^0\omega_b^1)R_2^0. && (5.31)\end{aligned}$$

Note that in this equation, ω_b^1 denotes the angular velocity of frame $o_2x_2y_2z_2$ that corresponds to the changing R_2^1, expressed relative to the coordinate system $o_1x_1y_1z_1$. Thus, the product $R_1^0\omega_b^1$ expresses this angular velocity relative to the coordinate system $o_0x_0y_0z_0$.

Now, combining the above expressions we have shown that

$$S(\omega_2^0)R_2^0 = \{S(\omega_a^0) + S(R_1^0\omega_b^1)\}R_2^0. \tag{5.32}$$

Since $S(\boldsymbol{a}) + S(\boldsymbol{b}) = S(\boldsymbol{a} + \boldsymbol{b})$, we see that

$$\omega_2^0 = \omega_a^0 + R_1^0\omega_b^1. \tag{5.33}$$

In other words, the angular velocities can be added once they are expressed relative to the same coordinate frame, in this case $o_0x_0y_0z_0$.

The above reasoning can be extended to any number of coordinate systems. In particular, suppose that we are given

$$R_n^0 = R_1^0 R_2^1 \cdots R_n^{n-1}. \tag{5.34}$$

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

$$\dot{R}_n^0 = S(\omega_n^0) R_n^0 \tag{5.35}$$

where

$$\omega_n^0 = \omega_1^0 + R_1^0\omega_2^1 + R_2^0\omega_3^2 + R_3^0\omega_4^3 + \cdots + R_{n-1}^0\omega_n^{n-1}. \tag{5.36}$$

5.5 Linear Velocity of a Point Attached to a Moving Frame

We now consider the linear velocity of a point that is rigidly attached to a moving frame. Suppose the point p is rigidly attached to the frame $o_1x_1y_1z_1$, and that $o_1x_1y_1z_1$ is rotating relative to the frame $o_0x_0y_0z_0$. Then the coordinates of p with respect to the frame $o_0x_0y_0z_0$ are given by

$$p^0 = R_1^0(t)p^1. \tag{5.37}$$

The velocity $\dot{p}^0$ is then given as

$$\begin{aligned} \dot{p}^0 &= \dot{R}_1^0(t)p^1 + R_1^0(t)\dot{p}^1 && (5.38)\\ &= S(\omega^0)R_1^0(t)p^1 && (5.39)\\ &= S(\omega^0)p^0 = \omega^0 \times p^0 \end{aligned}$$

which is the familiar expression for the velocity in terms of the vector cross product. Note that (5.39) follows from that fact that p is rigidly attached to frame $o_1x_1y_1z_1$, and therefore its coordinates relative to frame $o_1x_1y_1z_1$ do not change, giving $\dot{p}^1 = 0$.

Now suppose that the motion of the frame $o_1x_1y_1z_1$ relative to $o_0x_0y_0z_0$ is more general. Suppose that the homogeneous transformation relating the two frames is time-dependent, so that

$$H_1^0(t) = \begin{bmatrix} R_1^0(t) & o_1^0(t) \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5.40}$$

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

$$p^0 = Rp^1 + o. \tag{5.41}$$

Differentiating the above expression using the product rule gives

$$\begin{aligned} \dot{p}^0 &= \dot{R}p^1 + \dot{o} \\ &= S(\boldsymbol{\omega})Rp^1 + \dot{o} \\ &= \boldsymbol{\omega} \times \boldsymbol{r} + \boldsymbol{v} \end{aligned} \tag{5.42}$$

where $\boldsymbol{r} = Rp^1$ is the vector from o_1 to p expressed in the orientation of the frame $o_0x_0y_0z_0$, and $\boldsymbol{v}$ is the rate at which the origin o_1 is moving.

If the point p is moving relative to the frame $o_1x_1y_1z_1$, then we must add to the term $\boldsymbol{v}$ the term $R(t)\dot{p}^1$, which is the rate of change of the coordinates p^1 expressed in the frame $o_0x_0y_0z_0$.

5.6 Derivation of the Jacobian

Consider an n-link manipulator with joint variables $q_1, \ldots, q_n$. Let

$$T_n^0(\boldsymbol{q}) = \begin{bmatrix} R_n^0(\boldsymbol{q}) & o_n^0(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.43}$$

denote the transformation from the end-effector frame to the base frame, where $\boldsymbol{q} = (q_1, \ldots, q_n)^T$ is the vector of joint variables. As the robot moves about, both the joint variables q_i and the end-effector position o_n^0 and orientation R_n^0 will be functions of time. The objective of this section is to relate the linear and angular velocity of the end-effector to the vector of joint velocities $\dot{\boldsymbol{q}}(t)$. Let

$$S(\omega_n^0) = \dot{R}_n^0 (R_n^0)^T \tag{5.44}$$

define the angular velocity vector ω_n^0 of the end-effector, and let

$$v_n^0 = \dot{o}_n^0 \tag{5.45}$$

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

$$v_n^0 = J_{\boldsymbol{v}}\dot{\boldsymbol{q}} \tag{5.46}$$

$$\omega_n^0 = J_{\omega}\dot{\boldsymbol{q}} \tag{5.47}$$

where $J_{\boldsymbol{v}}$ and $J_{\boldsymbol{\omega}}$ are $3 \times n$ matrices. We may write (5.46) and (5.47) together as

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = J_n^0 \dot{\boldsymbol{q}} \tag{5.48}$$

where J_n^0 is given by

$$J_n^0 = \left[\begin{array}{c} J_{\boldsymbol{v}} \\ \hline J_{\omega} \end{array}\right]. \tag{5.49}$$

The matrix J_n^0 is called the **Manipulator Jacobian** or **Jacobian** for short. Note that J_n^0 is a $6 \times n$ matrix where n is the number of links. We next derive a simple expression for the Jacobian of any manipulator.

5.6.1 Angular Velocity

Recall from Equation (5.36) that angular velocities can be added vectorially provided that they are expressed relative to a common coordinate frame. Thus we can determine the angular velocity of the end-effector relative to the base by expressing the angular velocity contributed by each joint in the orientation of the base frame and then summing these.

If the i-th joint is revolute, then the i-th joint variable q_i equals θ_i and the axis of rotation is z_{i-1}. Following the convention that we introduced above, let ω_i^{i-1} represent the angular velocity of link i that is imparted by the rotation of joint i, expressed relative to frame $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$. This angular velocity is expressed in the frame $i-1$ by

$$\omega_i^{i-1} \quad = \quad \dot{q}_i z_{i-1}^{i-1} \quad = \quad \dot{q}_i \boldsymbol{k} \tag{5.50}$$

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

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

$$\omega_i^{i-1} \quad = \quad 0. \tag{5.51}$$

Thus, if joint i is prismatic, the angular velocity of the end-effector does not depend on q_i, which now equals d_i.

Therefore, the overall angular velocity of the end-effector, ω_n^0, in the base frame is determined by Equation (5.36) as

$$\begin{aligned} \omega_n^0 &= \rho_1 \dot{q}_1 \boldsymbol{k} + \rho_2 \dot{q}_2 R_1^0 \boldsymbol{k} + \cdots + \rho_n \dot{q}_n R_{n-1}^0 \boldsymbol{k} \\ &= \sum_{i-1}^{n} \rho_i \dot{q}_i z_{i-1}^0 \end{aligned} \tag{5.52}$$

in which ρ_i is equal to 1 if joint i is revolute and 0 if joint i is prismatic, since

$$z_{i-1}^0 \quad = \quad R_{i-1}^0 \boldsymbol{k}. \tag{5.53}$$

Of course $z_0^0 = \boldsymbol{k} = (0,0,1)^T$.

The lower half of the Jacobian J_ω, in (5.49) is thus given as

$$J_\omega \quad = \quad \left[\rho_1 z_0 \cdots \rho_n z_{n-1}\right]. \tag{5.54}$$

Note that in this equation, we have omitted the superscripts for the unit vectors along the z-axes, since these are all referenced to the world frame. In the remainder of the chapter we will follow this convention when there is no ambiguity concerning the reference frame.

5.6.2 Linear Velocity

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

$$\dot{o}_n^0 \quad = \quad \sum_{i=1}^{n} \frac{\partial o_n^0}{\partial q_i} \dot{q}_i. \tag{5.55}$$

Thus we see that the i-th column of $J_{\boldsymbol{v}}$, which we denote as $J_{\boldsymbol{v}_i}$ is given by

$$J_{\boldsymbol{v}_i} = \frac{\partial o_n^0}{\partial q_i}. \tag{5.56}$$

Furthermore this expression is just the linear velocity of the end-effector that would result if $\dot{q}_i$ were equal to one and the other $\dot{q}_j$ were zero. In other words, the i-th column of the Jacobian can be generated by holding all joints fixed but the i-th and actuating the i-th at unit velocity. We now consider the two cases (prismatic and revolute joints) separately.

(i) Case 1: Prismatic Joints

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

$$\begin{aligned}
\begin{bmatrix} R_n^0 & o_n^0 \\ \mathbf{0} & 1 \end{bmatrix} &= T_n^0 && (5.57)\\
&= T_{i-1}^0 T_i^{i-1} T_n^i && (5.58)\\
&= \begin{bmatrix} R_{i-1}^0 & o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_n^i & o_n^i \\ \mathbf{0} & 1 \end{bmatrix} && (5.59)\\
&= \begin{bmatrix} R_n^0 & R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix}, && (5.60)
\end{aligned}$$

which gives

$$o_n^0 = R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0. \tag{5.61}$$

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

$$\begin{aligned}
\frac{\partial o_n^0}{\partial q_i} &= \frac{\partial}{\partial d_i} R_{i-1}^0 o_i^{i-1} && (5.62)\\
&= R_{i-1}^0 \frac{\partial}{\partial d_i} \begin{bmatrix} a_i c_i \\ a_i s_i \\ d_i \end{bmatrix} && (5.63)\\
&= \dot{d}_i R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} && (5.64)\\
&= \dot{d}_i z_{i-1}^0, && (5.65)
\end{aligned}$$

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

$$J_{\boldsymbol{v}_i} = z_{i-1}. \tag{5.66}$$

(ii) Case 2: Revolute Joints

If joint i is revolute, then we have $q_i = \theta_i$. Starting with (5.61), and letting $q_i = \theta_i$, since R_i^0 is not constant with respect to θ_i, we obtain

$$\begin{aligned}
\frac{\partial}{\partial\theta_i} o_n^0 &= \frac{\partial}{\partial\theta_i}\left[R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1}\right] && (5.67)\\
&= \frac{\partial}{\partial\theta_i} R_i^0 o_n^i + R_{i-1}^0 \frac{\partial}{\partial\theta_i} o_i^{i-1} && (5.68)\\
&= \dot\theta_i S(z_{i-1}^0) R_i^0 o_n^i + \dot\theta_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1} && (5.69)\\
&= \dot\theta_i S(z_{i-1}^0)\left[R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1}\right] && (5.70)\\
&= \dot\theta_i S(z_{i-1}^0)(o_n^0 - o_{i-1}^0) && (5.71)\\
&= \dot\theta_i z_{i-1}^0 \times (o_n^0 - o_{i-1}^0). && (5.72)
\end{aligned}$$

The second term in (5.69) is derived as follows:

$$\begin{aligned}
R_{i-1}^0 \frac{\partial}{\partial\theta_i}\begin{bmatrix} a_i c_i \\ a_i s_i \\ d_i \end{bmatrix} &= R_{i-1}^0 \begin{bmatrix} -a_i s_i \\ a_i c_i \\ 0 \end{bmatrix}\dot\theta_i && (5.73)\\
&= R_{i-1}^0 S(\boldsymbol{k}\dot\theta_i) o_i^{i-1} && (5.74)\\
&= R_{i-1}^0 S(\boldsymbol{k}\dot\theta_i)\left(R_{i-1}^0\right)^T R_{i-1}^0 o_i^{i-1} && (5.75)\\
&= S(R_{i-1}^0 \boldsymbol{k}\dot\theta_i) R_{i-1}^0 o_i^{i-1} && (5.76)\\
&= \dot\theta_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1}. && (5.77)
\end{aligned}$$

Equation (5.74) follows by straightforward computation. Thus

$$J_{\boldsymbol{v}_i} = z_{i-1} \times (o_n - o_{i-1}), \tag{5.78}$$

in which we have, following our convention, omitted the zero superscripts. Figure 5.1 illustrates a second interpretation of (5.78). As can be seen in the figure, $o_n - o_{i-1} = \boldsymbol{r}$ and $z_{i-1} = \omega$ in the familiar expression $\boldsymbol{v} = \omega \times \boldsymbol{r}$.

Combining the Angular and Linear Jacobians

As we have seen in the preceding section, the upper half of the Jacobian $J_{\boldsymbol{v}}$ is given as

$$J_{\boldsymbol{v}} = \left[J_{\boldsymbol{v}_1} \cdots J_{\boldsymbol{v}_n}\right] \tag{5.79}$$

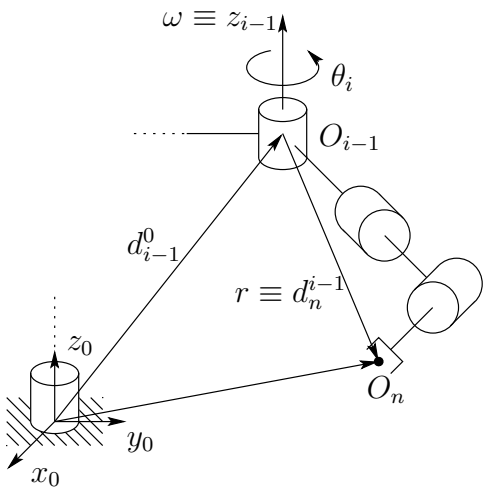


Figure 5.1: Motion of the end-effector due to link i.

where the i-th column $J_{\boldsymbol{v}_i}$ is

$$J_{\boldsymbol{v}_i} = z_{i-1} \times (o_n - o_{i-1}) \tag{5.80}$$

if joint i is revolute and

$$J_{\boldsymbol{v}_i} = z_{i-1} \tag{5.81}$$

if joint i is prismatic.

The lower half of the Jacobian is given as

$$J_\omega = [J_{\omega_1} \cdots J_{\omega_n}] \tag{5.82}$$

where the i-th column J_{ω_i} is

$$J_{\omega_i} = z_{i-1} \tag{5.83}$$

if joint i is revolute and

$$J_{\omega_i} = 0 \tag{5.84}$$

if joint i is prismatic.

Now putting the upper and lower halves of the Jacobian together we have shown that the Jacobian for an n-link manipulator is of the form

$$J = [J_1 J_2 \cdots J_n] \tag{5.85}$$

where the i-th column J_i is given by

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{5.86}$$

if joint i is revolute and

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{5.87}$$

if joint i is prismatic.

The above formulas make the determination of the Jacobian of any manipulator simple since all of the quantities needed are available once the forward kinematics are worked out. Indeed the only quantities needed to compute the Jacobian are the unit vectors z_i and the coordinates of the origins $o_1, \ldots, o_n$. A moment's reflection shows that the coordinates for z_i w.r.t. the base frame are given by the first three elements in the third column of T_i^0 while o_i is given by the first three elements of the fourth column of T_i^0. Thus only the third and fourth columns of the T matrices are needed in order to evaluate the Jacobian according to the above formulas.

The above procedure works not only for computing the velocity of the end-effector but also for computing the velocity of any point on the manipulator. This will be important in Chapter 9 when we will need to compute the velocity of the center of mass of the various links in order to derive the dynamic equations of motion.

Example 5.5 *Consider the three-link planar manipulator of Figure 5.2. Suppose we wish*

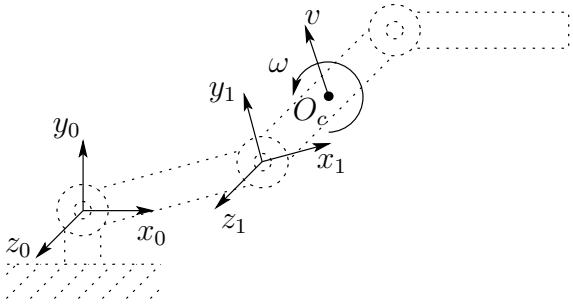


Figure 5.2: Finding the velocity of link 2 of a 3-link planar robot.

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

$$\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} = [J_1 \; J_2 \; J_3]\dot{q} \tag{5.88}$$

where the columns of the Jacobian are determined using the above formula with o_c in place of o_n. Thus we have

$$\begin{aligned} J_1 &= z_0 \times (o_c - o_0) \\ J_2 &= z_1 \times (o_c - o_1) \end{aligned} \tag{5.89}$$

and

$$J_3 \;=\; 0$$

since the velocity of the second link is unaffected by motion of link 3[3]*. Note that in this case the vector o_c must be computed as it is not given directly by the T matrices (Problem 5-1).* ◇

5.7 Examples

Example 5.6 *Consider the two-link planar manipulator of Example 3.1. Since both joints are revolute the Jacobian matrix, which in this case is 6×2, is of the form*

$$J(q) \;=\; \begin{bmatrix} z_0 \times (o_2 - o_0) & z_1 \times (o_2 - o_1) \\ z_0 & z_1 \end{bmatrix}. \tag{5.90}$$

The various quantities above are easily seen to be

$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix} \quad o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix} \tag{5.91}$$

$$z_0 = z_1 \;=\; \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.92}$$

Performing the required calculations then yields

$$J \;=\; \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}. \tag{5.93}$$

It is easy to see how the above Jacobian compares with the expression (1.1) derived in Chapter 1. The first two rows of (5.92) are exactly the 2×2 Jacobian of Chapter 1 and give the linear velocity of the origin o_2 relative to the base. The third row in (5.93) is the linear velocity in the direction of z_0, which is of course always zero in this case. The last three rows represent the angular velocity of the final frame, which is simply a rotation about the vertical axis at the rate $\dot{\theta}_1 + \dot{\theta}_2$.

[3]Note that we are treating only kinematic effects here. Reaction forces on link 2 due to the motion of link 3 will influence the motion of link 2. These dynamic effects are treated by the methods of Chapter 9.

◇

Example 5.7 Stanford Manipulator *Consider the Stanford manipulator of Example 3.3.5 with its associated Denavit-Hartenberg coordinate frames. Note that joint 3 is prismatic and that $o_3 = o_4 = o_5$ as a consequence of the spherical wrist and the frame assignment. Denoting this common origin by o we see that the Jacobian is of the form*

$$J = \begin{bmatrix} z_0 \times (o_6 - o_0) & z_1 \times (o_6 - o_1) & z_2 & z_3 \times (o_6 - o) & z_4 \times (o_6 - o) & z_5 \times (o_6 - o) \\ z_0 & z_1 & \mathbf{0} & z_3 & z_4 & z_5 \end{bmatrix}.$$

Now, using the A-matrices given by the expressions (3.35)-(3.40) and the T-matrices formed as products of the A-matrices, these quantities are easily computed as follows: First, o_j is given by the first three entries of the last column of $T_j^0 = A_1 \cdots A_j$, with $o_0 = (0,0,0)^T = o_1$. The vector z_j is given as

$$z_j = R_j^0 \boldsymbol{k} \tag{5.94}$$

where R_j^0 is the rotational part of T_j^0. Thus it is only necessary to compute the matrices T_j^0 to calculate the Jacobian. Carrying out these calculations one obtains the following expressions for the Stanford manipulator:

$$o_6 = (d_x, d_y, d_z)^T = \begin{bmatrix} c_1 s_2 d_3 - s_1 d_2 + d_6(c_1 c_2 c_4 s_5 + c_1 c_5 s_2 - s_1 s_4 s_5) \\ s_1 s_2 d_3 - c_1 d_2 + d_6(c_1 s_4 s_5 + c_2 c_4 s_1 s_5 + c_5 s_1 s_2) \\ c_2 d_3 + d_6(c_2 c_5 - c_4 s_2 s_5) \end{bmatrix} \tag{5.95}$$

$$o_3 = \begin{bmatrix} c_1 s_2 d_3 - s_1 d_2 \\ s_1 s_2 d_3 + c_1 d_2 \\ c_2 d_3 \end{bmatrix}. \tag{5.96}$$

The z_i are given as

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad z_1 = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \tag{5.97}$$

$$z_2 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \qquad z_3 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \tag{5.98}$$

$$z_4 = \begin{bmatrix} -c_1 c_2 s_4 - s_1 c_4 \\ -s_1 c_2 s_4 + c_1 c_4 \\ s_2 s_4 \end{bmatrix} \tag{5.99}$$

$$z_5 = \begin{bmatrix} c_1 c_2 c_4 s_5 - s_1 s_4 s_5 + c_1 s_2 c_5 \\ s_1 c_2 c_4 s_5 + c_1 s_4 s_5 + s_1 s_2 c_5 \\ -s_2 c_4 s_5 + c_2 c_5 \end{bmatrix}. \tag{5.100}$$

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

◇

Example 5.8 SCARA Manipulator *We will now derive the Jacobian of the SCARA manipulator of Example 3.3.6. This Jacobian is a* 6×4 *matrix since the SCARA has only four degrees-of-freedom. As before we need only compute the matrices* $T_j^0 = A_1 \dots A_j$*, where the A-matrices are given by (3.45)-(3.48).*

Since joints 1,2, and 4 are revolute and joint 3 is prismatic, and since $o_4 - o_3$ *is parallel to* z_3 *(and thus,* $z_3 \times (o_4 - o_3) = 0$*), the Jacobian is of the form*

$$J = \begin{bmatrix} z_0 \times (o_4 - o_0) & z_1 \times (o_4 - o_1) & z_2 & 0 \\ z_0 & z_1 & 0 & z_3 \end{bmatrix}. \tag{5.101}$$

Performing the indicated calculations, one obtains

$$o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix} \qquad o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix} \tag{5.102}$$

$$o_4 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_2 + a_2 s_{12} \\ d_3 - d_4 \end{bmatrix}. \tag{5.103}$$

Similarly $z_0 = z_1 = \boldsymbol{k}$*, and* $z_2 = z_3 = -\boldsymbol{k}$*. Therefore the Jacobian of the SCARA Manipulator is*

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}. \tag{5.104}$$

◇

5.8 The Analytical Jacobian

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

$$X = \begin{bmatrix} d(q) \\ \alpha(q) \end{bmatrix} \tag{5.105}$$

denote the end-effector pose, where $d(q)$ is the usual vector from the origin of the base frame to the origin of the end-effector frame and α denotes a minimal representation for the orientation of the end-effector frame relative to the base frame. For example, let $\alpha =$

$[\phi, \theta, \psi]^T$ be a vector of Euler angles as defined in Chapter 2. Then we look for an expression of the form

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q} \tag{5.106}$$

to define the analytical Jacobian.

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

$$\dot{R} = S(\omega)R \tag{5.107}$$

where ω, defining the angular velocity is given by

$$\omega = \begin{bmatrix} c_\psi s_\theta \dot{\phi} - s_\psi \dot{\theta} \\ s_\psi s_\theta \dot{\psi} + c_\psi \theta \\ \dot{\psi} + c_\theta \dot{\psi} \end{bmatrix} \tag{5.108}$$

$$= \begin{bmatrix} c_\psi s_\theta & -s_\psi & 0 \\ s_\psi s_\theta & c_\psi & 0 \\ c_\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T(\alpha)\dot{\alpha} \tag{5.109}$$

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

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ \omega \end{bmatrix} = J(q)\dot{q} \tag{5.110}$$

yields

$$J(q)\dot{q} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{d} \\ T(\alpha)\dot{\alpha} \end{bmatrix} \tag{5.111}$$

$$= \begin{bmatrix} I & 0 \\ 0 & T(\alpha) \end{bmatrix} \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} \tag{5.112}$$

$$= \begin{bmatrix} I & 0 \\ 0 & T(\alpha) \end{bmatrix} J_a \tag{5.113}$$

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

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & T(\alpha)^{-1} \end{bmatrix} J(q) \tag{5.114}$$

provided $det(T(\alpha)) \neq 0$.

In the next section we discuss the notion of Jacobian singularities, which are configurations where the Jacobian loses rank. Singularities of the matrix $T(\alpha)$ are called *representational singularities*. It can easily be shown (Problem X) that $T(\alpha)$ is invertible provided $s_\theta \neq 0$. This means that the singularities of the analytical Jacobian include the singularities of the geometric Jacobian, J, as defined in the next section, together with the representational singularities.

5.9 Singularities

The $6 \times n$ Jacobian $J(\boldsymbol{q})$ defines a mapping

$$\dot{\boldsymbol{X}} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{5.115}$$

between the vector $\dot{\boldsymbol{q}}$ of joint velocities and the vector $\dot{\boldsymbol{X}} = (\boldsymbol{v}, \boldsymbol{\omega})^T$ of end-effector velocities. Infinitesimally this defines a linear transformation

$$\boldsymbol{dX} = J(\boldsymbol{q})\boldsymbol{dq} \tag{5.116}$$

between the differentials $\boldsymbol{dq}$ and $\boldsymbol{dX}$. These differentials may be thought of as defining directions in $\mathbf{R}^6$, and $\mathbf{R}^n$, respectively.

Since the Jacobian is a function of the configuration $\boldsymbol{q}$, those configurations for which the rank of J decreases are of special significance. Such configurations are called **singularities** or **singular configurations**. Identifying manipulator singularities is important for several reasons.

1. Singularities represent configurations from which certain directions of motion may be unattainable.

2. At singularities, bounded end-effector velocities may correspond to unbounded joint velocities.

3. At singularities, bounded end-effector forces and torques may correspond to unbounded joint torques. (We will see this in Chapter 12).

4. Singularities usually (but not always) correspond to points on the boundary of the manipulator workspace, that is, to points of maximum reach of the manipulator.

5. Singularities correspond to points in the manipulator workspace that may be unreachable under small perturbations of the link parameters, such as length, offset, etc.

6. Near singularities there will not exist a unique solution to the inverse kinematics problem. In such cases there may be no solution or there may be infinitely many solutions.

Example 5.9 *Consider the two-dimensional system of equations*

$$\boldsymbol{dX} = J\boldsymbol{dq} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{dq} \tag{5.117}$$

that corresponds to the two equations

$$dx = dq_1 + dq_2 \tag{5.118}$$

$$dy = 0. \tag{5.119}$$

In this case the rank of J is one and we see that for any values of the variables dq_1 and dq_2 there is no change in the variable dy. Thus any vector $\boldsymbol{dX}$ having a nonzero second component represents an unattainable direction of instantaneous motion.

◇

5.9.1 Decoupling of Singularities

We saw in Chapter 3 that a set of forward kinematic equations can be derived for any manipulator by attaching a coordinate frame rigidly to each link in any manner that we choose, computing a set of homogeneous transformations relating the coordinate frames, and multiplying them together as needed. The D-H convention is merely a systematic way to do this. Although the resulting equations are dependent on the coordinate frames chosen, the manipulator configurations themselves are geometric quantities, independent of the frames used to describe them. Recognizing this fact allows us to decouple the determination of singular configurations, for those manipulators with spherical wrists, into two simpler problems. The first is to determine so-called **arm singularities**, that is, singularities resulting from motion of the arm, which consists of the first three or more links, while the second is to determine the **wrist singularities** resulting from motion of the spherical wrist.

For the sake of argument, suppose that $n = 6$, that is, the manipulator consists of a 3-DOF arm with a 3-DOF spherical wrist. In this case the Jacobian is a 6×6 matrix and a configuration $\boldsymbol{q}$ is singular if and only if

$$\det J(\boldsymbol{q}) \quad = \quad 0. \tag{5.120}$$

If we now partition the Jacobian J into 3×3 blocks as

$$J = [J_P \mid J_O] \quad = \quad \left[\begin{array}{c|c} J_{11} & J_{12} \\ \hline J_{21} & J_{22} \end{array} \right] \tag{5.121}$$

then, since the final three joints are always revolute

$$J_O \quad = \quad \begin{bmatrix} z_3 \times (o_6 - o_3) & z_4 \times (o_6 - o_4) & z_5 \times (o_6 - o_5) \\ z_3 & z_4 & z_5 \end{bmatrix}. \tag{5.122}$$

Since the wrist axes intersect at a common point o, if we choose the coordinate frames so that $o_3 = o_4 = o_5 = o_6 = o$, then J_O becomes

$$J_O \quad = \quad \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix} \tag{5.123}$$

and the i-th column J_i of J_p is

$$J_i \quad = \quad \begin{bmatrix} z_{i-1} \times (o - o_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{5.124}$$

if joint i is revolute and

$$J_i \quad = \quad \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{5.125}$$

if joint i is prismatic. In this case the Jacobian matrix has the block triangular form

$$J \quad = \quad \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix} \tag{5.126}$$

with determinant

$$\det J \quad = \quad \det J_{11} \det J_{22} \tag{5.127}$$

where J_{11} and J_{22} are each 3×3 matrices. J_{11} has i-th column $z_{i-1} \times (o - o_{i-1})$ if joint i is revolute, and z_{i-1} if joint i is prismatic, while

$$J_{22} \quad = \quad [z_3 \; z_4 \; z_5]. \tag{5.128}$$

Therefore the set of singular configurations of the manipulator is the union of the set of arm configurations satisfying $\det J_{11} = 0$ and the set of wrist configurations satisfying $\det J_{22} = 0$. *Note that this form of the Jacobian does not necessarily give the correct relation between the velocity of the end-effector and the joint velocities.* It is intended only to simplify the determination of singularities.

5.9.2 Wrist Singularities

We can now see from (5.128) that a spherical wrist is in a singular configuration whenever the vectors z_3, z_4 and z_5 are linearly dependent. Referring to Figure 5.3 we see that this

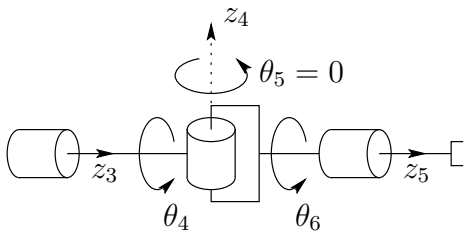


Figure 5.3: Spherical wrist singularity.

happens when the joint axes z_3 and z_5 are collinear. In fact, whenever two revolute joint axes anywhere are collinear, a singularity results since an equal and opposite rotation about the axes results in no net motion of the end-effector. This is the only singularity of the spherical wrist, and is unavoidable without imposing mechanical limits on the wrist design to restrict its motion in such a way that z_3 and z_5 are prevented from lining up.

5.9.3 Arm Singularities

In order to investigate arm singularities we need only to compute J_{11} according to (5.124) and (5.125), which is the same formula derived previously with the wrist center o in place of o_6.

Example 5.10 Elbow Manipulator Singularities *Consider the three-link articulated manipulator with coordinate frames attached as shown in Figure 5.4. It is left as an exercise*

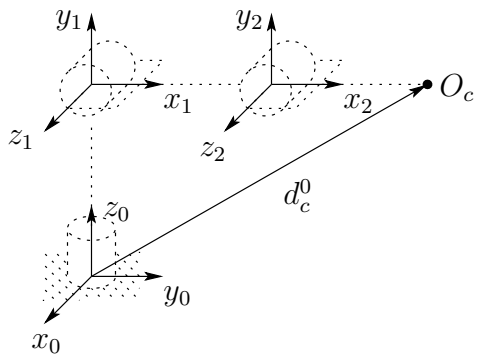


Figure 5.4: Elbow manipulator.

(Problem **??***) to show that*

$$J_{11} = \begin{bmatrix} -a_2 s_1 c_2 - a_3 s_1 c_{23} & -a_2 s_2 c_1 - a_3 s_{23} c_1 & -a_3 c_1 s_{23} \\ a_2 c_1 c_2 + a_3 c_1 c_{23} & -a_2 s_1 s_2 - a_3 s_1 s_{23} & -a_3 s_1 s_{23} \\ 0 & a_2 c_2 + a_3 c_{23} & a_3 c_{23} \end{bmatrix} \tag{5.129}$$

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

$$\det J_{11} = a_2 a_3 s_3 (a_2 c_2 + a_3 c_{23}). \tag{5.130}$$

We see from (5.130) that the elbow manipulator is in a singular configuration whenever

$$s_3 = 0, \quad \textit{that is, } \theta_3 = 0 \textit{ or } \pi \tag{5.131}$$

and whenever

$$a_2 c_2 + a_3 c_{23} = 0. \tag{5.132}$$

The situation of (5.131) is shown in Figure 5.5 and arises when the elbow is fully ex-

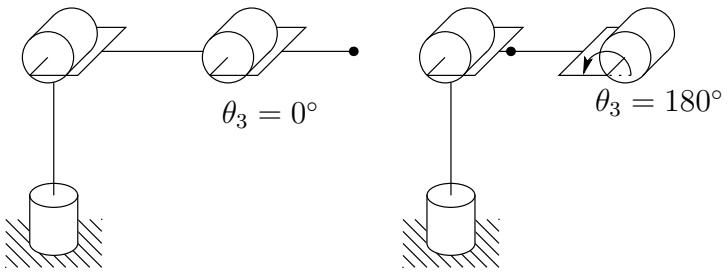


Figure 5.5: Elbow singularities of the elbow manipulator.

tended or fully retracted as shown. The second situation (5.132) is shown in Figure 5.6.

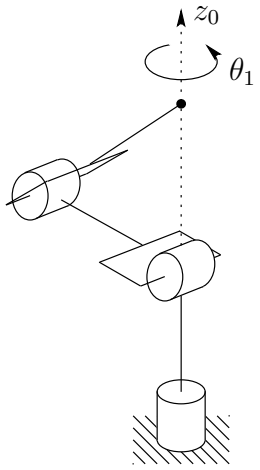


Figure 5.6: Singularity of the elbow manipulator with no offsets.

This configuration occurs when the wrist center intersects the axis of the base rotation, z_0. As we saw in Chapter 4, there are infinitely many singular configurations and infinitely many solutions to the inverse position kinematics when the wrist center is along this axis. For an elbow manipulator with an offset, as shown in Figure 5.7, the wrist center cannot

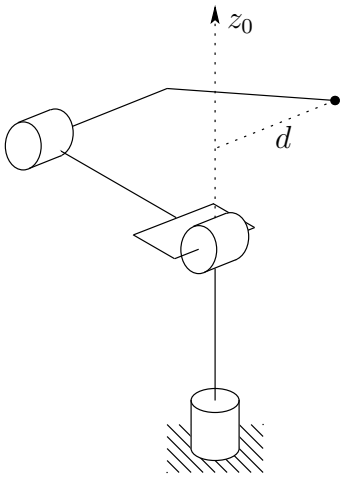


Figure 5.7: Elbow manipulator with shoulder offset.

intersect z_0, which corroborates our earlier statement that points reachable at singular configurations may not be reachable under arbitrarily small perturbations of the manipulator parameters, in this case an offset in either the elbow or the shoulder.

◇

Example 5.11 Spherical Manipulator *Consider the spherical arm of Figure 5.8. This*

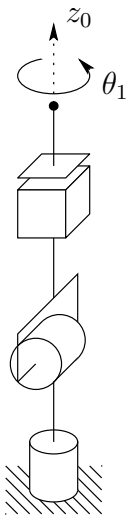


Figure 5.8: Singularity of spherical manipulator with no offsets.

manipulator is in a singular configuration when the wrist center intersects z_0 as shown since, as before, any rotation about the base leaves this point fixed.
◇

Example 5.12 SCARA Manipulator *We have already derived the complete Jacobian for the the SCARA manipulator. This Jacobian is simple enough to be used directly rather than deriving the modified Jacobian from this section. Referring to Figure 5.9 we can see*

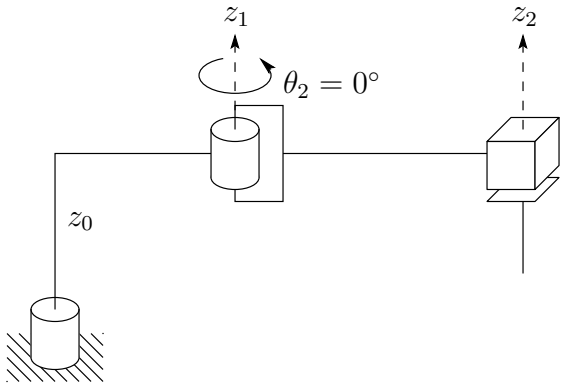


Figure 5.9: SCARA manipulator singularity.

geometrically that the only singularity of the SCARA arm is when the elbow is fully extended or fully retracted. Indeed, since the portion of the Jacobian of the SCARA governing arm

singularities is given as

$$J_{11} = \begin{bmatrix} \alpha_1 & \alpha_3 & 0 \\ \alpha_2 & \alpha_4 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5.133}$$

where

$$\alpha_1 = -a_1 s_1 - a_2 s_{12} \tag{5.134}$$
$$\alpha_2 = a_1 c_1 + a_2 c_{12}$$
$$\alpha_3 = -a_1 s_{12}$$
$$\alpha_4 = a_1 c_{12} \tag{5.135}$$

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

$$s_2 = 0, \quad \textit{which implies} \quad \theta_2 = 0, \pi. \tag{5.136}$$

◇

5.10 Inverse Velocity and Acceleration

It is perhaps a bit surprising that the inverse velocity and acceleration relationships are conceptually simpler than inverse position. Recall from (5.115) that the joint velocities and the end-effector velocities are related by the Jacobian as

$$\dot{\boldsymbol{X}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{5.137}$$

Thus the inverse velocity problem becomes one of solving the system of linear equations (5.137), which is conceptually simple.

Differentiating (5.137) yields the acceleration equations

$$\ddot{\boldsymbol{X}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \left(\frac{d}{dt}J(\boldsymbol{q})\right)\dot{\boldsymbol{q}}. \tag{5.138}$$

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

$$\boldsymbol{b} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{5.139}$$

where

$$\boldsymbol{b} = \ddot{\boldsymbol{X}} - \frac{d}{dt}J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{5.140}$$

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

$$\dot{q} = J(\boldsymbol{q})^{-1}\dot{\boldsymbol{X}} \tag{5.141}$$

and

$$\ddot{\boldsymbol{q}} = J(\boldsymbol{q})^{-1}\boldsymbol{b} \tag{5.142}$$

provided $\det J(\boldsymbol{q}) \neq 0$. In the next section, we address the case of manipulators with more than 6-DOF.

5.11 Redundant Robots and Manipulability

In this section we briefly address the topic of *redundant manipulators*. Informally, a redundant manipulator is one that is equipped with more internal degrees of freedom than are required to perform a specified task. For example, a three link planar arm is redundant for the task of positioning in the plane. As we have briefly seen in Chapter 4, in such cases there in no unique solution for the inverse kinematics problem. Further, the Jacobian matrix for a redundant manipulator is not square, and thus cannot be inverted to solve the inverse velocity problem.

In this section, we begin by giving a brief and general introduction to the subject of redundant manipulators. We then turn our attention to the inverse velocity problem. To address this problem, we will introduce the concept of a pseudoinverse and the Singular Value Decomposition. We end the section by introducing *manipulability*, a measure that can be used to quantify the quality of the internal configuration of a manipulator, and can therefore be used in an optimization framework to aid in the solution for the inverse kinematics problem.

5.11.1 Redundant Manipulators

A precise definition of what is meant by the term *redundant* requires that we specify a task, and the number of degrees of freedom required to perform that task. In previous chapters, we have dealt primarily with positioning tasks. In these cases, the task was determined by specifying the position, orientation or both for the end effector or some tool mounted at the end effector. For these kinds of positioning tasks, the number of degrees of freedom for the task is equal to the number of parameters required to specify the position and orientation information. For example, if the task involves positioning the end effector in a 3D workspace, then the task can be specified by an element of $\Re^3 \times SO(3)$. As we have seen in Chapter 2, $\Re^3 \times SO(3)$ can be parameterized by $(x, y, z, \phi, \theta, \psi)$, i.e., using six parameters. Thus, for this task, the task space is six-dimensional. A manipulator is said to be redundant when its number of internal degrees of freedom (or joints) is greater than the dimension of the task space. Thus, for the 3D position and orientation task, any manipulator with more than six joints would be redundant.

A simpler example is a three-link planar arm performing the task of positioning the end effector in the plane. Here, the task can be specified by $(x, y) \in \Re^2$, and therefore the task space is two-dimensional. The forward kinematic equations for this robot are given by

$$\begin{aligned} x &= a_1 c_1 + a_2 c_{12} + a_3 c_{123} \\ y &= a_1 s_1 + a_2 s_{12} + a_3 s_{123}. \end{aligned}$$

Clearly, since there are three variables $(\theta_1, \theta_2, \theta_3)$ and only two equations, it is not possible to solve uniquely for $\theta_1, \theta_2, \theta_3$ given a specific (x, y).

The Jacobian for this manipulator is given by

$$\mathbf{J} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & -a_3 s_{123} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & a_3 c_{123} \end{bmatrix}. \tag{5.143}$$

When using the relationship $\dot{\boldsymbol{x}} = \mathbf{J}\dot{\boldsymbol{q}}$ to for $\dot{\boldsymbol{q}}$, we have a system of two linear equations in three unknowns. Thus there are also infinitely many solutions to this system, and the inverse velocity problem cannot be solved uniquely. We now turn our attention to the specifics of dealing with these inverse problems.

5.11.2 The Inverse Velocity Problem for Redundant Manipulators

We have seen in Section 5.10 that the inverse velocity problem is easily solved when the Jacobian is square with nonzero determinant. However, when the Jacobian is not square, as is the case for redundant manipulators, the method of Section 5.10 cannot be used, since a nonsquare matrix cannot be inverted. To deal with the case when $m < n$, we use the following result from linear algebra.

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

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

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

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

It is now easy to demonstrate that a solution to (5.137) is given by

$$\dot{\boldsymbol{q}} = \mathbf{J}^+\dot{\boldsymbol{x}} + (\mathbf{I} - \mathbf{J}^+\mathbf{J})\boldsymbol{b} \tag{5.144}$$

in which $\boldsymbol{b} \in \Re^n$ is an arbitrary vector. To see this, multiply this solution by $\mathbf{J}$:

$$\begin{aligned}
\mathbf{J}\dot{\boldsymbol{q}} &= \mathbf{J}\left[\mathbf{J}^{+}\dot{\boldsymbol{x}} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\boldsymbol{b}\right] \\
&= \mathbf{J}\mathbf{J}^{+}\dot{\boldsymbol{x}} + \mathbf{J}(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\boldsymbol{b} \\
&= \mathbf{J}\mathbf{J}^{+}\dot{\boldsymbol{x}} + (\mathbf{J} - \mathbf{J}\mathbf{J}^{+}\mathbf{J})\boldsymbol{b} \\
&= \dot{\boldsymbol{x}} + (\mathbf{J} - \mathbf{J})\boldsymbol{b} \\
&= \dot{\boldsymbol{x}}.
\end{aligned}$$

In general, for $m < n$, $(\mathbf{I} - \mathbf{J}^{+}\mathbf{J}) \neq 0$, and all vectors of the form $(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\mathbf{b}$ lie in the null space of $\mathbf{J}$, i.e., if $\dot{\boldsymbol{q}}_n$ is a joint velocity vector such that $\dot{\boldsymbol{q}}_n = (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\mathbf{b}$, then when the joints move with velocity $\dot{\boldsymbol{q}}_n$, the end effector will remain fixed since $\mathbf{J}\dot{\boldsymbol{q}}_n = \mathbf{0}$. Thus, if $\dot{\boldsymbol{q}}$ is a solution to (5.137), then so is $\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}_n$ with $\dot{\boldsymbol{q}}_n = (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\mathbf{b}$, for any value of $\mathbf{b}$. If the goal is to minimize the resulting joint velocities, we choose $\mathbf{b} = \mathbf{0}$. To see this, apply the triangle inequality to obtain

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

5.11.3 Singular Value Decomposition (SVD)

For robots that are not redundant, the Jacobian matrix is square, and we can use tools such as the determinant, eigenvalues and eigenvectors to analyze its properties. However, for redundant robots, the Jacobian matrix is not square, and these tools simply do not apply. Their generalizations are captured by the Singular Value Decomposition (SVD) of a matrix, which we now introduce.

As we described above, for $\mathbf{J} \in \Re^{m \times n}$, we have $\mathbf{J}\mathbf{J}^T \in \Re^{m \times m}$. This square matrix has eigenvalues and eigenvectors that satisfy

$$\mathbf{J}\mathbf{J}^T\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \tag{5.145}$$

in which λ_i and $\boldsymbol{u}_i$ are corresponding eigenvalue and eigenvector pairs for $\mathbf{J}\mathbf{J}^T$. We can rewrite this equation to obtain

$$\begin{aligned}
\mathbf{J}\mathbf{J}^T\boldsymbol{u}_i - \lambda_i \boldsymbol{u}_i &= \mathbf{0} \\
(\mathbf{J}\mathbf{J}^T - \lambda_i \mathbf{I})\boldsymbol{u}_i &= \mathbf{0}.
\end{aligned} \tag{5.146}$$

The latter equation implies that the matrix $(\mathbf{J}\mathbf{J}^T - \lambda_i \mathbf{I})$ is singular, and we can express this in terms of its determinant as

$$\det(\mathbf{J}\mathbf{J}^T - \lambda_i \mathbf{I}) = 0. \tag{5.147}$$

We can use (5.147) to find the eigenvalues $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_m \geq 0$ for $\mathbf{J}\mathbf{J}^T$. The *singular values* for the Jacobian matrix $\mathbf{J}$ are given by the square roots of the eigenvalues of $\mathbf{J}\mathbf{J}^T$,

$$\sigma_i = \sqrt{\lambda_i}. \tag{5.148}$$

The singular value decomposition of the matrix $\mathbf{J}$ is then given by

$$\mathbf{J} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{5.149}$$

in which

$$\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_m], \ \mathbf{V} = [\mathbf{v}_1\mathbf{v}_2 \dots \mathbf{v}_n] \tag{5.150}$$

are orthogonal matrices, and $\Sigma \in \mathbf{R}^{m\times n}$.

$$\Sigma = \left[\begin{array}{cccc|c} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & . & & 0 \\ & & & . & \\ & & & \sigma_m & \end{array}\right]. \tag{5.151}$$

We can compute the SVD of $\mathbf{J}$ as follows. We begin by finding the singular values, σ_i, of $\mathbf{J}$ using (5.147) and (5.148). These singular values can then be used to find the eigenvectors $\mathbf{u}_1, \cdots \mathbf{u}_m$ that satisfy

$$\mathbf{J}\mathbf{J}^T\mathbf{u}_i = \sigma_i^2\mathbf{u}_i. \tag{5.152}$$

These eigenvectors comprise the matrix $\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_m]$. The system of equations (5.152) can be written as

$$\mathbf{J}\mathbf{J}^T\mathbf{U} = \mathbf{U}\Sigma_m^2 \tag{5.153}$$

if we define the matrix Σ_m as

$$\Sigma_m = \left[\begin{array}{cccc} \sigma_1 & & & \\ & \sigma_2 & & \\ & & . & \\ & & & . \\ & & & \sigma_m \end{array}\right].$$

Now, define

$$\mathbf{V}_m = \mathbf{J}^T\mathbf{U}\Sigma_m^{-1} \tag{5.154}$$

and let $\mathbf{V}$ be any orthogonal matrix that satisfies $\mathbf{V} = [\mathbf{V}_m \mid \mathbf{V}_{n-m}]$ (note that here $\mathbf{V}_{n-m}$ contains just enough columns so that the matrix $\mathbf{V}$ is an $n \times n$ matrix). It is a simple

matter to combine the above equations to verify (5.149):

$$\begin{aligned}
\mathbf{U}\Sigma\mathbf{V}^T &= \mathbf{U}\left[\Sigma_m \mid \mathbf{0}\right]\begin{bmatrix} \mathbf{V}_m^T \\ \mathbf{V}_{n-m}^T \end{bmatrix} && (5.155)\\
&= \mathbf{U}\Sigma_m\mathbf{V}_m^T && (5.156)\\
&= \mathbf{U}\Sigma_m\left(\mathbf{J}^T\mathbf{U}\Sigma_m^{-1}\right)^T && (5.157)\\
&= \mathbf{U}\Sigma_m(\Sigma_m^{-1})^T\mathbf{U}^T\mathbf{J} && (5.158)\\
&= \mathbf{U}\Sigma_m\Sigma_m^{-1}\mathbf{U}^T\mathbf{J} && (5.159)\\
&= \mathbf{U}\mathbf{U}^T\mathbf{J} && (5.160)\\
&= \mathbf{J}. && (5.161)
\end{aligned}$$

Here, (5.155) follows immediately from our construction of the matrices $\mathbf{U}$, $\mathbf{V}$ and Σ_m. Equation (5.157) is obtained by substituting (5.154) into (5.156). Equation (5.159) follows because Σ_m^{-1} is a diagonal matrix, and thus symmetric. Finally, (5.161) is obtained using the fact that $\mathbf{U}^T = \mathbf{U}^{-1}$, since $\mathbf{U}$ is orthogonal.

It is a simple matter construct the right pseudoinverse of $\mathbf{J}$ using the SVD,

$$\mathbf{J}^+ = \mathbf{V}\Sigma^+\mathbf{U}^T$$

in which

$$\Sigma^+ = \left[\begin{array}{cccc|c} \sigma_1^{-1} & & & & \\ & \sigma_2^{-1} & & & \\ & & . & & 0 \\ & & & . & \\ & & & \sigma_m^{-1} & \end{array}\right]^T.$$

5.11.4 Manipulability

For a specific value of $\boldsymbol{q}$, the Jacobian relationship defines the linear system given by $\dot{\boldsymbol{x}} = \mathbf{J}\dot{\boldsymbol{q}}$. We can think of $\mathbf{J}$ a scaling the input, $\dot{\boldsymbol{q}}$, to produce the output, $\dot{\boldsymbol{x}}$. It is often useful to characterize quantitatively the effects of this scaling. Often, in systems with a single input and a single output, this kind of characterization is given in terms of the so called impulse response of a system, which essentially characterizes how the system responds to a unit input. In this multidimensional case, the analogous concept is to characterize the output in terms of an input that has unit norm. Consider the set of all robot tool velocities $\dot{\boldsymbol{q}}$ such that

$$\|\dot{\boldsymbol{q}}\| = (\dot{q}_1^2 + \dot{q}_2^2 + \ldots \dot{q}_m^2)^{1/2} \leq 1. \tag{5.162}$$

If we use the minimum norm solution $\dot{\boldsymbol{q}} = \mathbf{J}^+\dot{\boldsymbol{x}}$, we obtain

$$\begin{aligned}
\|\dot{\boldsymbol{q}}\| &= \dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}} \\
&= (\mathbf{J}^+\dot{\boldsymbol{x}})^T\mathbf{J}^+\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T(\mathbf{J}^+)^T\mathbf{J}^+\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T(\mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1})^T\mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T[(\mathbf{J}\mathbf{J}^T)^{-1}]^T\mathbf{J}\mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\boldsymbol{x}} \leq 1. \qquad (5.163)
\end{aligned}$$

This final inequality gives us a quantitative characterization of the scaling that is effected by the Jacobian. In particular, if the manipulator Jacobian is full rank, i.e., rank $\mathbf{J} = m$, then (5.163) defines an m-dimensional ellipsoid that is known as the *manipulability ellipsoid*. If the input (i.e., joint velocity) vector has unit norm, then the output (i.e., task space velocity) will lie within the ellipsoid given by (5.163). We can more easily see that (5.163) defines an ellipsoid by replacing $\mathbf{J}$ by its SVD to obtain

$$\begin{aligned}
\dot{\boldsymbol{x}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\boldsymbol{x}}^T &= \dot{\boldsymbol{x}}^T[\mathbf{U}\Sigma\mathbf{V}^T(\mathbf{U}\Sigma\mathbf{V}^T)^T]^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T[\mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma^T\mathbf{U}^T]^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T[\mathbf{U}\Sigma\Sigma^T\mathbf{U}^T]^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T[\mathbf{U}\Sigma_m^2\mathbf{U}^T]^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^T[\mathbf{U}\Sigma_m^{-2}\mathbf{U}^T]\dot{\boldsymbol{x}} \\
&= (\dot{\boldsymbol{x}}^T\mathbf{U})\Sigma_m^{-2}(\mathbf{U}^T\dot{\boldsymbol{x}}) \\
&= (\mathbf{U}^T\dot{\boldsymbol{x}})^T\Sigma_m^{-2}(\mathbf{U}^T\dot{\boldsymbol{x}}) \qquad (5.164)
\end{aligned}$$

in which

$$\Sigma_m^{-2} = \begin{bmatrix} \sigma_1^{-2} & & & & \\ & \sigma_2^{-2} & & & \\ & & . & & \\ & & & . & \\ & & & & \sigma_m^{-2} \end{bmatrix}.$$

If we make the substation $\mathbf{w} = \mathbf{U}^T\dot{\boldsymbol{x}}$, then (5.164) can be written as

$$\mathbf{w}^T\Sigma_m^{-2}\mathbf{w} = \sum \sigma_i^{-2}w_i^2 \leq 1 \qquad (5.165)$$

and it is clear that this is the equation for an axis-aligned ellipse in a new coordinate system that is obtained by rotation according to the orthogonal matrix $\mathbf{U}$. In the original coordinate system, the axes of the ellipsoid are given by the vectors $\sigma_i\mathbf{u_i}$. The volume of the ellipsoid is given by

$$\text{volume} = K\sigma_1\sigma_2\cdots\sigma_m,$$

in which K is a constant that depends only on the dimension, m, of the ellipsoid. The manipulability measure, as defined by Yoshikawa [**?**], is given by

$$\omega = \sigma_1\sigma_2\cdots\sigma_m. \qquad (5.166)$$

Note that the constant K is not included in the definition of manipulability, since it is fixed once the task has been defined (i.e., once the dimension of the task space has been fixed).

Now, consider the special case that the robot is not redundant, i.e., $\mathbf{J} \in \Re^{m \times m}$. Recall that the determinant of a product is equal to the product of the determinants, and that a matrix and its transpose have the same determinant. Thus, we have

$$\begin{aligned} \det \mathbf{J}\mathbf{J}^T &= \det \mathbf{J} \ \det \mathbf{J}^T \\ &= \det \mathbf{J} \ \det \mathbf{J} \\ &= (\lambda_1 \lambda_2 \cdots \lambda_m)(\lambda_1 \lambda_2 \cdots \lambda_m) \\ &= \lambda_1^2 \lambda_2^2 \cdots \lambda_m^2 \end{aligned} \tag{5.167}$$

in which $\lambda_1 \geq \lambda_2 \cdots \leq \lambda_m$ are the eigenvalues of $\mathbf{J}$. This leads to

$$\omega = \sqrt{\det \mathbf{J}\mathbf{J}^T} = |\lambda_1 \lambda_2 \cdots \lambda_m| = |\det \mathbf{J}|. \tag{5.168}$$

The manipulability, ω, has the following properties.

- In general, $\omega = 0$ holds if and only if $\text{rank}(\mathbf{J}) < m$, (i.e., when J is not full rank).
- Suppose that there is some error in the measured velocity, $\Delta\dot{\boldsymbol{x}}$. We can bound the corresponding error in the computed joint velocity, $\Delta\dot{\boldsymbol{q}}$, by

$$(\sigma_1)^{-1} \leq \frac{||\Delta\dot{\boldsymbol{q}}||}{||\Delta\dot{\boldsymbol{x}}||} \leq (\sigma_m)^{-1}. \tag{5.169}$$

Example 5.13 Two-link Planar Arm. *We can use manipulability to determine the optimal configurations in which to perform certain tasks. In some cases it is desirable to perform a task in the configuration for which the end effector has the maximum dexterity. We can use manipulability as a measure of dexterity. Consider the two-link planar arm and the task of positioning in the plane. For the two link arm, the Jacobian is given by*

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}. \tag{5.170}$$

and the manipulability is given by

$$\omega = |det \ \mathbf{J}| = a_1 a_2 |s_2|$$

Thus, for the two-link arm, the maximum manipulability is obtained for $\theta_2 = \pm\pi/2$.

Manipulability can also be used to aid in the design of manipulators. For example, suppose that we wish to design a two-link planar arm whose total link length, $a_1 + a_2$, is fixed. What values should be chosen for a_1 and a_2? If we design the robot to maximize the maximum manipulability, the we need to maximize $\omega = a_1 a_2 |s_2|$. We have already seen that the maximum is obtained when $\theta_2 = \pm\pi/2$, so we need only find a_1 and a_2 to maximize the product $a_1 a_2$. This is achieved when $a_1 = a_2$. Thus, to maximize manipulability, the link lengths should be chosen to be equal.

◇

Chapter 6

COMPUTER VISION

If a robot is to interact with its environment, then the robot must be able to sense its environment. Computer vision is one of the most powerful sensing modalities that currently exist. Therefore, in this chapter we present a number of basic concepts from the field of computer vision. It is not our intention here to cover the now vast field of computer vision. Rather, we aim to present a number of basic techniques that are applicable to the highly constrained problems that often present themselves in industrial applications. The material in this chapter, when combined with the material of previous chapters, should enable the reader to implement a rudimentary vision-based robotic manipulation system. For example, using techniques presented in this chapter, one could design a system that locates objects on a conveyor belt, and determines the positions and orientations of those objects. This information could then be used in conjunction with the inverse kinematic solution for the robot, along with various coordinate transformations, to command the robot to grasp these objects.

We begin by examining the geometry of the image formation process. This will provide us with the fundamental geometric relationships between objects in the world and their projections in an image. We then describe a calibration process that can be used to determine the values for the various camera parameters that appear in these relationships. We then consider image segmentation, the problem of dividing the image into distinct regions that correspond to the background and to objects in the scene. When there are multiple objects in the scene, it is often useful to deal with them individually; therefore, we next present an approach to component labelling. Finally, we describe how to compute the positions and orientations of objects in the image.

6.1 The Geometry of Image Formation

A digital image is a two-dimensional array of pixels that is formed by focusing light onto a two-dimensional array of sensing elements. A lens with focal length λ is used to focus the light onto the sensing array, which is often composed of CCD (charge-coupled device) sensors. The lens and sensing array are packaged together in a camera, which is connected to

a digitizer or frame grabber. In the case of analog cameras, the digitizer converts the analog video signal that is output by the camera into discrete values that are then transferred to the pixel array by the frame grabber. In the case of digital cameras, a frame grabber merely transfers the digital data from the camera to the pixel array. Associated with each pixel in the digital image is a gray level value, typically between 0 and 255, which encodes the intensity of the light incident on the corresponding sensing element.

In robotics applications, it is often sufficient to consider only the geometric aspects of image formation. Therefore in this section we will describe only the geometry of the image formation process. We will not deal with the photometric aspects of image formation (e.g., we will not address issues related to depth of field, lens models, or radiometry).

We will begin the section by assigning a coordinate frame to the imaging system. We then discuss the popular pinhole model of image formation, and derive the corresponding equations that relate the coordinates of a point in the world to its image coordinates. Finally, we describe camera calibration, the process by which all of the relevant parameters associated with the imaging process can be determined.

6.1.1 The Camera Coordinate Frame

In order to simplify many of the equations of this chapter, it often will be useful to express the coordinates of objects relative to a camera centered coordinate frame. For this purpose, we define the camera coordinate frame as follows. Define the image plane, π, as the plane that contains the sensing array. The axes x_c and y_c form a basis for the image plane, and are typically taken to be parallel to the horizontal and vertical axes (respectively) of the image. The axis z_c is perpendicular to the image plane and aligned with the optic axis of the lens (i.e., it passes through the focal center of the lens). The origin of the camera frame is located at a distance λ behind the image plane. This point is also referred to as the *center of projection*. The point at which the optical axis intersects the image plane is known as the *principal point*. This coordinate frame is illustrated in Figure 6.1.

With this assignment of the camera frame, any point that is contained in the image plane will have coordinates (u, v, λ). Thus, we can use (u, v) to parameterize the image plane, and we will refer to (u, v) as image plane coordinates.

6.1.2 Perspective Projection

The image formation process is often modeled by the pinhole lens approximation. With this approximation, the lens is considered to be an ideal pinhole, and the pinhole is located at the focal center of the lens[1]. Light rays pass through this pinhole, and intersect the image plane.

Let P be a point in the world with coordinates x, y, z (relative to the camera frame). Let p denote the projection of P onto the image plane with coordinates (u, v, λ). Under the

[1]Note that in our mathematical model, illustrated in Figure 6.1, we have placed the pinhole behind the image plane in order to simplify the model.

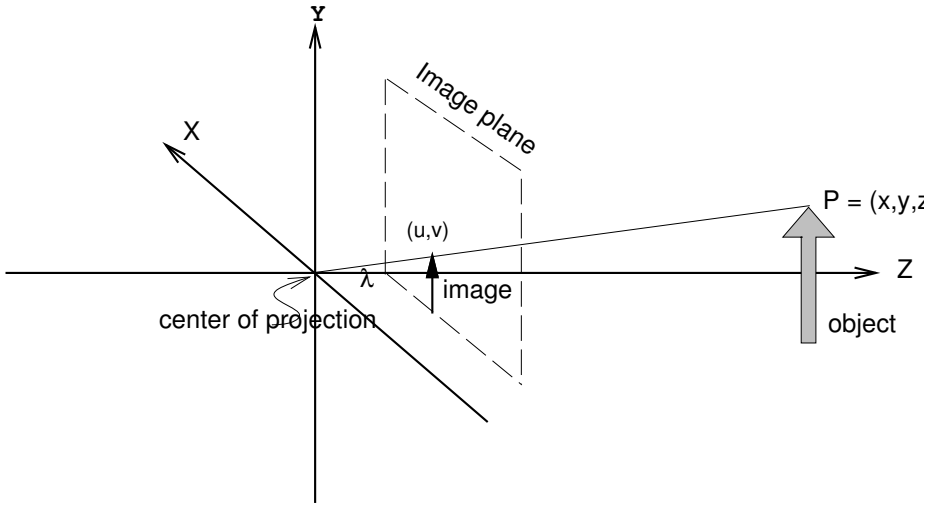


Figure 6.1: Camera coordinate frame.

pinhole assumption, P, p and the origin of the camera frame will be collinear. This can is illustrated in Figure 6.1. Thus, for some unknown positive k we have

$$k \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u \\ v \\ \lambda \end{pmatrix} \tag{6.1}$$

which can be rewritten as the system of equations:

$$kx = u, \tag{6.2}$$
$$ky = v, \tag{6.3}$$
$$kz = \lambda. \tag{6.4}$$

This gives $k = \lambda/z$, which can be substituted into (6.2) and (6.3) to obtain

$$u = \lambda\frac{x}{z}, \qquad v = \lambda\frac{y}{z}. \tag{6.5}$$

These are the well-known equations for perspective projection.

6.1.3 The Image Plane and the Sensor Array

As described above, the image is a discrete array of gray level values. We will denote the row and column indices for a pixel by the coordinates (r, c). In order to relate digital images to the 3D world, we must determine the relationship between the image plane coordinates, (u, v), and indices into the pixel array of pixels, (r, c).

We typically define the origin of this pixel array to be located at a corner of the image (rather than the center of the image). Let the pixel array coordinates of the pixel that

contains the principal point be given by (o_r, o_c). In general, the sensing elements in the camera will not be of unit size, nor will they necessarily be square. Denote the s_x and s_y the horizontal and vertical dimensions (respectively) of a pixel. Finally, it is often the case that the horizontal and vertical axes of the pixel array coordinate system point in opposite directions from the horizontal and vertical axes of the camera coordinate frame[2]. Combining these, we obtain the following relationship between image plane coordinates and pixel array coordinates,

$$-\frac{u}{s_x} = (r - o_r), \quad -\frac{v}{s_y} = (c - o_c). \tag{6.6}$$

which gives,

$$u = -s_x(r - o_r), \quad v = -s_y(c - o_c). \tag{6.7}$$

Note that the coordinates (r, c) will be integers, since they are the discrete indices into an array that is stored in computer memory. Therefore, it is not possible to obtain the exact image plane coordinates for a point from the (r, c) coordinates.

6.2 Camera Calibration

The objective of camera calibration is to determine all of the parameters that are necessary to predict the image pixel coordinates (r, c) of the projection of a point in the camera's field of view, given that the coordinates of that point with respect to the world coordinate frame are know. In other words, given the coordinates of P relative to the world coordinate frame, after we have calibrated the camera we will be able to predict (r, c), the image pixel coordinates for the projection of this point.

6.2.1 Extrinsic Camera Parameters

To this point, in our derivations of the equations for perspective projection, we have dealt only with coordinates expressed relative to the camera frame. In typical robotics applications, tasks will be expressed in terms of the world coordinate frame, and it will therefore be necessary to perform coordinate transformations. If we know the position and orientation of the camera frame relative to the world coordinate frame we have

$$x^w = R_c^w x^c + O_c^w \tag{6.8}$$

or, if we know x^w and wish to solve for x^c,

$$x^c = R_w^c(x^w - O_c^w) \tag{6.9}$$

In the remainder of this section, to simplify notation we will define

$$R = R_w^c, \quad \mathbf{T} = -R_w^c O_c^w. \tag{6.10}$$

[2]This is an artifact of our choice to place the center of projection behind the image plane. The directions of the pixel array axes may vary, depending on the frame grabber.

Thus,

$$x^c = Rx^w + \mathbf{T} \tag{6.11}$$

Cameras are typically mounted on tripods, or on mechanical positioning units. In the latter case, a popular configuration is the pan/tilt head. A pan/tilt head has two degrees of freedom: a rotation about the world z axis and a rotation about the pan/tilt head's x axis. These two degrees of freedom are analogous to the those of a human head, which can easily look up or down, and can turn from side to side. In this case, the rotation matrix R is given by

$$R = R_{z,\theta} R_{x,\alpha}, \tag{6.12}$$

where θ is the pan angle and α is the tilt angle. More precisely, θ is the angle between the world x-axis and the camera x-axis, about the world z-axis, while α is the angle between the world z-axis and the camera z-axis, about the camera x-axis.

6.2.2 Intrinsic Camera Parameters

Using the pinhole model, we obtained the following equations that map the coordinates of a point expressed with respect to the camera frame to the corresponding pixel coordinates:

$$r = -\frac{u}{s_x} + o_r, \qquad c = -\frac{v}{s_y} + o_c, \qquad u = \lambda \frac{x}{z} \qquad v = \lambda \frac{y}{z}. \tag{6.13}$$

These equations can be combined to give

$$r = -\frac{\lambda}{s_x}\frac{x}{z} + o_r, \qquad c = -\frac{\lambda}{s_y}\frac{y}{z} + o_c, \tag{6.14}$$

Thus, once we know the values of the parameters $\lambda, s_x, o_r, s_y, o_c$ we can determine (r, c) from (x, y, z), where (x, y, z) are coordinates relative to the camera frame. In fact, we don't need to know all of λ, s_x, s_y; it is sufficient to know the ratios

$$f_x = -\frac{\lambda}{s_x} \qquad f_y = -\frac{\lambda}{s_y}. \tag{6.15}$$

These parameters, f_x, o_r, f_y, o_c are known as the intrinsic parameters of the camera. They are constant for a given camera, and do not change when the camera moves.

6.2.3 Determining the Camera Parameters

The task of camera calibration is to determine the intrinsic and extrinsic parameters of the camera. We will proceed by first determining the parameters associated with the image center, and then solving for the remaining parameters.

Of all the camera parameters, o_r, o_c (the image pixel coordinates of the principal point) are the easiest to determine. This can be done by using the idea of vanishing points. Although a full treatment of vanishing points is beyond the scope of this text, the idea is simple: a set of parallel lines in the world will project onto image lines that intersect at a

single point, and this intersection point is known as a *vanishing point*. The vanishing points for three mutually orthogonal sets of lines in the world will define a triangle in the image. The orthocenter of this triangle (i.e., the point at which the three altitudes intersect) is the image principal point (a proof of this is beyond the scope of this text). Thus, a simple way to compute the principal point is to position a cube in the workspace, find the edges of the cube in the image (this will produce the three sets of mutually orthogonal parallel lines), compute the intersections of the image lines that correspond to each set of parallel lines in the world, and determine the orthocenter for the resulting triangle.

Once we know the principal point, we proceed to determine the remaining camera parameters. This is done by constructing a linear system of equations in terms of the known coordinates of points in the world and the pixel coordinates of their projections in the image. The unknowns in this system are the camera parameters. Thus, the first step in this stage of calibration is to acquire a data set of the form $r_1, c_1, x_1, y_1, z_1, r_2, c_2, x_2, y_2, z_2, \cdots r_N, c_N, x_N, y_N, z_N$, in which r_i, c_i are the image pixel coordinates of the projection of a point in the world with coordinates x_i, y_i, z_i relative to the world coordinate frame. This acquisition is often done manually, e.g., by having a robot move a small bright light to known x, y, z coordinates in the world, and then hand selecting the corresponding image point.

Once we have acquired the data set, we proceed to set up the linear system of equations. The extrinsic parameters of the camera are given by

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}. \tag{6.16}$$

With respect to the camera frame, the coordinates of a point in the world are thus given by

$$x^c = r_{11}x + r_{12}y + r_{13}z + T_x \tag{6.17}$$
$$y^c = r_{21}x + r_{22}y + r_{23}z + T_y \tag{6.18}$$
$$z^c = r_{31}x + r_{32}y + r_{33}z + T_z. \tag{6.19}$$

Combining this with (6.14) we obtain

$$r - o_r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{6.20}$$
$$c - o_c = -f_y \frac{y^c}{z^c} = -f_y \frac{r_{21}x + r_{22}y + r_{23}z + T_y}{r_{31}x + r_{32}y + r_{33}z + T_z}. \tag{6.21}$$

Since we know the coordinates of the principal point, we an simplify these equations by using the simple coordinate transformation

$$r \leftarrow r - o_r, \qquad c \leftarrow c - o_c. \tag{6.22}$$

We now write the two transformed projection equations as functions of the unknown variables: $r_{ij}, T_x, T_y, T_z, f_x, f_y$. This is done by solving each of these equations for z^c, and

setting the resulting equations to be equal to one another. In particular, for the data points r_i, c_i, x_i, y_i, z_i we have

$$r_i f_y(r_{21}x_i + r_{22}y_i + r_{23}z_i + T_y) = c_i f_x(r_{11}x_i + r_{12}y_i + r_{13}z_i + T_x). \tag{6.23}$$

We define $\alpha = f_x/f_y$ and rewrite this as:

$$r_i r_{21}x_i + r_i r_{22}y_i + r_i r_{23}z_i + r_i T_y - \alpha c_i r_{11}x_i - \alpha c_i r_{12}y_i - \alpha c_i r_{13}z_i - \alpha c_i T_x = 0. \tag{6.24}$$

We can combine the N such equations into the matrix equation

$$\begin{bmatrix} r_1x_1 & r_1y_1 & r_1z_1 & r_1 & -c_1x_1 & -c_1y_1 & -c_1z_1 & -c_1 \\ r_2x_2 & r_2y_2 & r_2z_2 & r_2 & -c_2x_2 & -c_2y_2 & -c_2z_2 & -c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ r_Nx_N & r_Ny_N & r_Nz_N & r_N & -c_Nx_N & -c_Ny_N & -c_Nz_N & -c_N \end{bmatrix} \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \\ T_y \\ \alpha r_{11} \\ \alpha r_{12} \\ \alpha r_{13} \\ \alpha T_x \end{bmatrix} = 0 \tag{6.25}$$

This is an equation of the form $A\mathbf{x} = 0$. As such, if $\bar{\mathbf{x}} = [\bar{x}_1, \cdots, \bar{x}_8]^T$ is a solution, for (6.25) we only know that this solution is some scalar multiple of the desired solution, $\mathbf{x}$, i.e.,

$$\bar{\mathbf{x}} = k[r_{21}, r_{22}, r_{23}, T_y, \alpha r_{11}, \alpha r_{12}, \alpha r_{13}, \alpha T_x]^T, \tag{6.26}$$

in which k is an unknown scale factor.

In order to solve for the true values of the camera parameters, we can exploit constraints that arise from the fact that R is a rotation matrix. In particular,

$$(\bar{x}_1^2 + \bar{x}_2^2 + \bar{x}_3^2)^{\frac{1}{2}} = (k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = |k|, \tag{6.27}$$

and likewise

$$(\bar{x}_5^2 + \bar{x}_6^2 + \bar{x}_7^2)^{\frac{1}{2}} = (\alpha^2 k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = \alpha|k| \tag{6.28}$$

(note that by definition, $\alpha > 0$).

Our next task is to determine the sign of k. Using equations (6.14) we see that $r \times x^c < 0$ (recall that we have used the coordinate transformation $r \leftarrow r - o_r$). Therefore, to determine the sign of k, we first assume that $k > 0$. If $r(r_{11}x + r_{12}y + r_{13}z + T_x) < 0$, then we know that we have made the right choice and $k > 0$; otherwise, we know that $k < 0$.

At this point, we know the values for $k, \alpha, r_{21}, r_{22}, r_{23}, r_{11}, r_{12}, r_{13}, T_x, T_Y$, and all that remains is to determine T_z, f_x, f_y. Since $\alpha = f_x/f_y$, we need only determine T_z and f_x. Returning again to the projection equations, we can write

$$r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{6.29}$$

Using an approach similar to that used above to solve for the first eight parameters, we can write this as the linear system

$$r(r_{31}x + r_{32}y + r_{33}z + T_z) = -f_x(r_{11}x + r_{12}y + r_{13}z + T_x) \tag{6.30}$$

which can easily be solved for T_Z and f_x.

6.3 Segmentation by Thresholding

Segmentation is the process by which an image is divided into meaningful components. Segmentation has been the topic of computer vision research since its earliest days, and the approaches to segmentation are far too numerous to survey here. These approaches are sometimes concerned with finding *features* in an image (e.g., edges), and sometimes concerned with partitioning the image into homogeneous regions (region-based segmentation). In many practical applications, the goal of segmentation is merely to divide the image into two regions: one region that corresponds to an object in the scene, and one region that corresponds to the background. In many industrial applications, this segmentation can be accomplished by a straight-forward thresholding approach. Pixels whose gray level is greater than the threshold are considered to belong to the object, and pixels whose gray level is less than or equal to the threshold are considered to belong to the background.

In this section we will describe an algorithm that automatically selects a threshold. This basic idea behind the algorithm is that the pixels should be divided into two groups (background and object), and that the intensities of the pixels in a particular group should all be fairly similar. To quantify this idea, we will use some standard techniques from statistics. Thus, we begin the section with a quick review of the necessary concepts from statistics and then proceed to describe the threshold selection algorithm.

6.3.1 A Brief Statistics Review

Many approaches to segmentation exploit statistical information contained in the image. In this section, we briefly review some of the more useful statistical concepts that are used by segmentation algorithms.

The basic premise for most of these statistical concepts is that the gray level value associated with a pixel in an image is a random variable that takes on values in the set $\{0, 1, \cdots N-1\}$. Let $P(z)$ denote the probability that a pixel has gray level value z. In general, we will not know this probability, but we can estimate it with the use of a *histogram*. A histogram is an array, H, that encodes the number of occurrences of each gray level value. In particular, the entry $H[z]$ is the number of times gray level value z occurs in the image. Thus, $0 \leq H[z] \leq N_{rows} \times N_{cols}$ for all z. An algorithm to compute the histogram for an image is shown in figure 6.2.

Given the histogram for the image, we estimate the probability that a pixel will have gray level z by

$$P(z) = \frac{H[z]}{N_{rows} \times N_{cols}}. \tag{6.31}$$

For $i = 0$ to $N - 1$
    $H[i] \leftarrow 0$
For $r = 0$ to $N_{rows} - 1$
    For $c = 0$ to $N_{cols} - 1$
        $Index \leftarrow Image(r, c)$
        $H[Index] \leftarrow H[Index] + 1$

Figure 6.2: Pseudo-code to compute an image histogram.

Thus, the image histogram is a scaled version of our approximation of P.

Given P, we can compute the *average*, or *mean* value of the gray level values in the image. We denote the mean by μ, and compute it by

$$\mu = \sum_{z=0}^{N-1} zP(z). \tag{6.32}$$

In many applications, the image will consist of one or more objects against some background. In such applications, it is often useful to compute the mean for each object in the image, and also for the background. This computation can be effected by constructing individual histogram arrays for each object, and for the background, in the image. If we denote by H_i the histogram for the i^{th} object in the image (where $i = 0$ denotes the background), the mean for the i^{th} object is given by

$$\mu_i = \sum_{z=0}^{N-1} z \, \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}, \tag{6.33}$$

which is a straightforward generalization of (6.32). The term

$$\frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}$$

is in fact an estimate of the probability that a pixel will have gray level value z given that the pixel is a part of object i in the image. For this reason, μ_i is sometimes called a *conditional mean*.

The mean conveys useful, but very limited information about the distribution of grey level values in an image. For example, if half or the pixels have gray value 127 and the remaining half have gray value 128, the mean will be $\mu = 127.5$. Likewise, if half or the pixels have gray value 255 and the remaining half have gray value 0, the mean will be $\mu = 127.5$. Clearly these two images are very different, but this difference is not reflected by the mean. One way to capture this difference is to compute the the average deviation of gray values from the mean. This average would be small for the first example, and large

for the second. We could, for example, use the average value of $|z - \mu|$; however, it will be more convenient mathematically to use the square of this value instead. The resulting quantity is known as the *variance*, which is defined by

$$\sigma^2 = \sum_{z=0}^{N-1} (z - \mu)^2 P(z). \tag{6.34}$$

As with the mean, we can also compute the conditional variance, σ_i^2 for each object in the image

$$\sigma_i^2 = \sum_{z=0}^{N-1} (z - \mu_i)^2 \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}. \tag{6.35}$$

6.3.2 Automatic Threshold Selection

We are now prepared to develop an automatic threshold selection algorithm. We will assume that the image consists of an object and a background, and that the background pixels have gray level values less than or equal to some threshold while the object pixels are above the threshold. Thus, for a given threshold value, z_t, we divide the image pixels into two groups: those pixels with gray level value $z \leq z_t$, and those pixels with gray level value $z > z_t$. We can compute the means and variance for each of these groups using the equations of Section 6.3.1. Clearly, the conditional means and variances depend on the choice of z_t, since it is the choice of z_t that determines which pixels will belong to each of the two groups. The approach that we take in this section is to determine the value for z_t that minimizes a function of the variances of these two groups of pixels.

In this section, it will be convenient to rewrite the conditional means and variances in terms of the pixels in the two groups. To do this, we define $q_i(z_t)$ as the probability that a pixel in the image will belong to group i for a particular choice of threshold, z_t. Since all pixels in the background have gray value less than or equal to z_t and all pixels in the object have gray value greater than z_t, we can define $q_i(z_t)$ for $i = 0, 1$ by

$$q_0(z_t) = \frac{\sum_{z=0}^{z_t} H[z]}{(N_{rows} \times N_{cols})}, \quad q_1(z_t) = \frac{\sum_{z=z_t+1}^{N-1} H[z]}{(N_{rows} \times N_{cols})}. \tag{6.36}$$

We now rewrite (6.33) as

$$\mu_i = \sum_{z=0}^{N-1} z \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]} = \sum_{z=0}^{N-1} z \frac{H_i[z]/(N_{rows} \times N_{cols})}{\sum_{z=0}^{N-1} H_i[z]/(N_{rows} \times N_{cols})}$$

Using again the fact that the two pixel groups are defined by the threshold z_t, we have

$$\frac{H_0[z]}{(N_{rows} \times N_{cols})} = \frac{P(z)}{q_0(z_t)}, \;\; z \leq z_t \qquad \text{and} \qquad \frac{H_1[z]}{(N_{rows} \times N_{cols})} = \frac{P(z)}{q_1(z_t)}, \;\; z > z_t. \tag{6.37}$$

Thus, we can write the conditional means for the two groups as

$$\mu_0(z_t) = \sum_{z=0}^{z_t} z \frac{P(z)}{q_0(z_t)}, \quad \mu_1(z_t) = \sum_{z=z_t+1}^{N-1} z \frac{P(z)}{q_1(z_t)}. \tag{6.38}$$

Similarly, we can write the equations for the conditional variances by

$$\sigma_0^2(z_t) = \sum_{z=0}^{z_t} (z - \mu_0(z_t))^2 \frac{P(z)}{q_0(z_t)}, \quad \sigma_1^2(z_t) = \sum_{z=z_t+1}^{N} (z - \mu_1(z_t))^2 \frac{P(z)}{q_1(z_t)}. \tag{6.39}$$

We now turn to the selection of z_t. If nothing is known about the true values of μ_i or σ_i^2, how can we determine the optimal value of z_t? To answer this question, recall that the variance is a measure of the average deviation of pixel intensities from the mean. Thus, if we make a good choice for z_t, we would expect that the variances $\sigma_i^2(z_t)$ would be small. This reflects the assumption that pixels belonging to the object will be clustered closely about μ_1, pixels belonging to the background will be clustered closely about μ_0. We could, therefore, select the value of z_t that minimizes the sum of these two variances. However, it is unlikely that the object and background will occupy the same number of pixels in the image; merely adding the variances gives both regions equal importance. A more reasonable approach is to weight the two variances by the probability that a pixel will belong to the corresponding region,

$$\sigma_w^2(z_t) = q_0(z_t)\sigma_0^2(z_t) + q_1(z_t)\sigma_1^2(z_t). \tag{6.40}$$

The value σ_w^2 is known as the *within-group variance*. The approach that we will take is to minimize this within-group variance.

At this point we could implement a threshold selection algorithm. The naive approach would be to simply iterate over all possible values of z_t and select the one for which $\sigma_w^2(z_t)$ is smallest. Such an algorithm performs an enormous amount of calculation, much of which is identical for different candidate values of the threshold. For example, most of the calculations required to compute $\sigma_w^2(z_t)$ are also required to compute $\sigma_w^2(z_t + 1)$; the required summations change only slightly from one iteration to the next. Therefore, we now turn our attention to an efficient algorithm.

To develop an efficient algorithm, we take two steps. First, we will derive the between-group variance, σ_b^2, which depends on the within-group variance and the variance over the entire image. The between-group variance is a bit simpler to deal with than the within-group variance, and we will show that maximizing the between-group variance is equivalent to minimizing the within-group variance. Then, we will derive a recursive formulation for the between-group variance that lends itself to an efficient implementation.

To derive the between-group variance, we begin by expanding the equation for the total variance of the image, and then simplifying and grouping terms. The variance of the gray

level values in the image is given by (6.34), which can be rewritten as

$$\begin{aligned}
\sigma^2 &= \sum_{z=0}^{N-1}(z-\mu)^2 P(z) \\
&= \sum_{z=0}^{z_t}(z-\mu)^2 P(z) + \sum_{z=z_t+1}^{N-1}(z-\mu)^2 P(z) \\
&= \sum_{z=0}^{z_t}(z-\mu_0+\mu_0-\mu)^2 P(z) + \sum_{z=z_t+1}^{N-1}(z-\mu_1+\mu_1-\mu)^2 P(z) \\
&= \sum_{z=0}^{z_t}[(z-\mu_0)^2 + 2(z-\mu_0)(\mu_0-\mu) + (\mu_0-\mu)^2]P(z) \\
&\quad + \sum_{z=z_t+1}^{N-1}[(z-\mu_1)^2 + 2(z-\mu_1)(\mu_1-\mu) + (\mu_1-\mu)^2]P(z). \qquad (6.41)
\end{aligned}$$

Note that the we have not explicitly noted the dependence on z_t here. In the remainder of this section, to simplify notation, we will refer to the group probabilities and conditional means and variances as q_i, μ_i, and σ_i^2, without explicitly noting the dependence on z_t. This last expression (6.41) can be further simplified by examining the cross-terms

$$\begin{aligned}
\sum(z-\mu_i)(\mu_i-\mu)P(z) &= \sum z\mu_i P(z) - \sum z\mu P(z) - \sum \mu_i^2 P(z) + \sum \mu_i \mu P(z) \\
&= \mu_i \sum zP(z) - \mu \sum zP(z) - \mu_i^2 \sum P(z) + \mu_i \mu \sum P(z) \\
&= \mu_i(\mu_i q_i) - \mu(\mu_i q_i) - \mu_i^2 q_i + \mu_i \mu q_i \\
&= 0,
\end{aligned}$$

in which the summations are taken for z from 0 to z_t for the background pixels (i.e., $i = 0$) and z from $z_t + 1$ to $N - 1$ for the object pixels (i.e., $i = 1$). Therefore, we can simplify (6.41) to obtain

$$\begin{aligned}
\sigma^2 &= \sum_{z=0}^{z_t}[(z-\mu_0)^2 + (\mu_0-\mu)^2]P(z) + \sum_{z=z_t+1}^{N-1}[(z-\mu_1)^2 + (\mu_1-\mu)^2]P(z) \\
&= q_0\sigma_0^2 + q_0(\mu_0-\mu)^2 + q_1\sigma_1^2 + q_1(\mu_1-\mu)^2 \\
&= \{q_0\sigma_0^2 + q_1\sigma_1^2\} + \{q_0(\mu_0-\mu)^2 + q_1(\mu_1-\mu)^2\} \\
&= \sigma_w^2 + \sigma_b^2 \qquad (6.42)
\end{aligned}$$

in which

$$\sigma_b^2 = q_0(\mu_0-\mu)^2 + q_1(\mu_1-\mu)^2. \qquad (6.43)$$

Since σ^2 does not depend on the threshold value (i.e., it is constant for a specific image), minimizing σ_w^2 is equivalent to maximizing σ_b^2. This is preferable because σ_b^2 is a function

only of the q_i and μ_i, and is thus simpler to compute than σ_w^2, which depends also on the σ_i^2. In fact, by expanding the squares in (6.43), using the facts that $q_1 = 1 - q_0$ and $\mu = q_1\mu_0 + q_1\mu_1$, we obtain

$$\sigma_b^2 = q_0(1 - q_0)(\mu_0 - \mu_1)^2. \tag{6.44}$$

The simplest algorithm to maximize σ_b^2 is to iterate over all possible threshold values, and select the one that maximizes σ_b^2. However, as discussed above, such an algorithm performs many redundant calculations, since most of the calculations required to compute $\sigma_b^2(z_t)$ are also required to compute $\sigma_b^2(z_t + 1)$. Therefore, we now turn our attention to an efficient algorithm that maximizes $\sigma_b^2(z_t)$. The basic idea for the efficient algorithm is to re-use the computations needed for $\sigma_b^2(z_t)$ when computing $\sigma_b^2(z_t + 1)$. In particular, we will derive expressions for the necessary terms at iteration $z_t + 1$ in terms of expressions that were computed at iteration z_t. We begin with the group probabilities, and determine the recursive expression for q_0 as

$$q_0(z_t + 1) = \sum_{z=0}^{z_t+1} P(z) = P(z_t + 1) + \sum_{z=0}^{z_t} P(z) = P(z_t + 1) + q_0(z_t). \tag{6.45}$$

In this expression, $P(z_t + 1)$ can be obtained directly from the histogram array, and $q_0(z_t)$ is directly available because it was computed on the previous iteration of the algorithm. Thus, given the results from iteration z_t, very little computation is required to compute the value for q_0 at iteration $z_t + 1$.

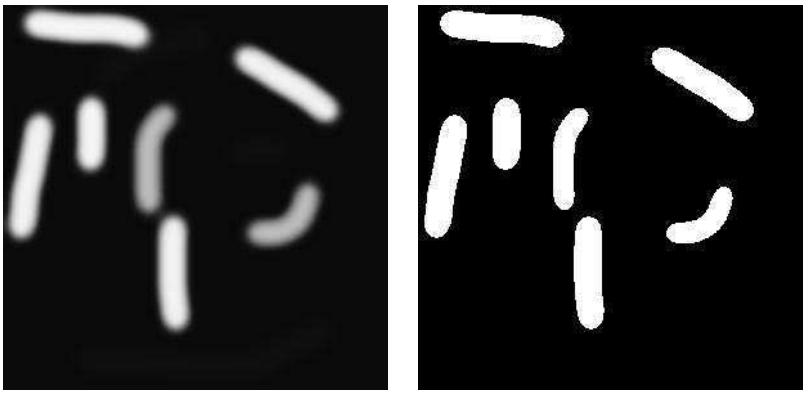
For the conditional mean $\mu_0(z_t)$ we have

$$\begin{aligned}
\mu_0(z_t + 1) &= \sum_{z=0}^{z_t+1} z\frac{P(z)}{q_0(z_t + 1)} && (6.46)\\
&= \frac{(z_t + 1)P(z_t + 1)}{q_0(z_t + 1)} + \sum_{z=0}^{z_t} z\frac{P(z)}{q_0(z_t + 1)} && (6.47)\\
&= \frac{(z_t + 1)P(z_t + 1)}{q_0(z_t + 1)} + \frac{q_0(z_t)}{q_0(z_t + 1)}\sum_{z=0}^{z_t} z\frac{P(z)}{q_0(z_t)} && (6.48)\\
&= \frac{(z_t + 1)P(z_t + 1)}{q_0(z_t + 1)} + \frac{q_0(z_t)}{q_0(z_t + 1)}\mu_0(z_t) && (6.49)
\end{aligned}$$

Again, all of the quantities in this expression are available either from the histogram, or as the results of calculations performed at iteration z_t of the algorithm.

To compute $\mu_1(z_t + 1)$, we use the relationship $\mu = q_0\mu_0 + q_1\mu_1$, which can be easily obtained using (6.32) and (6.38). Thus, we have

$$\mu_1(z_t + 1) = \frac{\mu - q_0(z_t + 1)\mu_0(z_t + 1)}{q_1(z_t + 1)} = \frac{\mu - q_0(z_t + 1)\mu_0(z_t + 1)}{1 - q_0(z_t + 1)}. \tag{6.50}$$



(a) (b)

Figure 6.3: (a) A 300×300 pixel image with 256 gray levels. (b) Thresholded version of the image in (a).

We are now equipped to construct a highly efficient algorithm that automatically selects a threshold value that minimizes the within-group variance. This algorithm simply iterates from 0 to $N-1$ (where N is the total number of gray level values), computing q_0, μ_0, μ_1 and σ_b^2 at each iteration using the recursive formulations given in (6.45), (6.49), (6.50) and (6.44). The algorithm returns the value of z_t for which σ_b^2 is largest. Figure **??** shows a grey level image and the binary, thresholded image that results from the application of this algorithm.

6.4 Connected Components

It is often the case that multiple objects will be present in a single image. When this occurs, after thresholding there will be multiple connected components with gray level values that are above the threshold. In this section, we will first make precise the notion of a connected component, and then describe an algorithm that assigns a unique label to each connected component, i.e., all pixels within a single connected component have the same label, but pixels in different connected components have different labels.

In order to define what is meant by a connected component, it is first necessary to define what is meant by connectivity. For our purposes, it is sufficient to say that a pixel, A, with image pixel coordinates (r, c) is adjacent to four pixels, those with image pixel coordinates $(r-1, c)$, $(r+1, c)$, $(r, c+1)$, and $(r, c-1)$. In other words, each image pixel A (except those at the edges of the image) has four neighbors: the pixel directly above, directly below, directly to the right and directly to the left of pixel A. This relationship is sometimes referred to as *4-connectivity*, and we say that two pixels are 4-connected if they are adjacent by this definition. If we expand the definition of adjacency to include those pixels that are diagonally adjacent (i.e., the pixels with coordinates $(r-1, c-1)$, $(r-1, c+1)$,

$(r+1, c-1)$, and $(r+1, c+1)$), then we say that adjacent pixels are 8-connected. In this text, we will consider only the case of 4-connectivity.

A connected component is a collection of pixels, S, such that for any two pixels, say P and P', in S, there is a 4-connected path between them and this path is contained in S. Intuitively, this definition means that it is possible to move from P to P' by "taking steps" only to adjacent pixels without ever leaving the region S. The purpose of a component labeling algorithm is to assign a unique label to each such S.

There are many component labeling algorithms that have been developed over the years. Here, we describe a simple algorithm that requires two passes over the image. This algorithm performs two raster scans of the image (note: a raster scan visits each pixel in the image by traversing from left to right, top to bottom, in the same way that one reads a page of text). On the first raster scan, when an object pixel P, (i.e., a pixel whose gray level is above the threshold value), is encountered, its previously visited neighbors (i.e., the pixel immediately above and the pixel immediately to the left of P) are examined, and if they have gray value that is below the threshold (i.e., they are background pixels), a new label is given to P. This is done by using a global counter that is initialized to zero, and is incremented each time a new label is needed. If either of these two neighbors have already received labels, then P is given the smaller of these, and in the case when both of the neighbors have received labels, an equivalence is noted between those two labels. For example, in Figure 6.4, after the first raster scan labels (2,3,4) are noted as equivalent. In the second raster scan, each pixel's label is replaced by the smallest label to which it is equivalent. Thus, in the example of Figure 6.4, at the end of the second raster scan labels 3 and 4 have been replaced by the label 2.

After this algorithm has assigned labels to the components in the image, it is not necessarily the case that the labels will be the consecutive integers $(1, 2, \cdots)$. Therefore, a second stage of processing is sometimes used to relabel the components to achieve this. In other cases, it is desirable to give each component a label that is very different from the labels of the other components. For example, if the component labelled image is to be displayed, it is useful to increase the contrast, so that distinct components will actually appear distinct in the image (a component with the label 2 will appear almost indistinguishable from a component with label 3 if the component labels are used as pixel gray values in the displayed component labelled image). The results of applying this process to the image in Figure 6.3 are shown in Figure 6.5.

When there are multiple connected object components, it is often useful to process each component individually. For example, we might like to compute the sizes of the various components. For this purpose, it is useful to introduce the *indicator function* for a component. The indicator function for component i, denoted by $\mathcal{I}_i$, is a function that takes on the value 1 for pixels that are contained in component i, and the value 0 for all other pixels:

$$\mathcal{I}_i(r,c) = \begin{cases} 1 & : \quad \text{pixel } r,c \text{ is contained in component } i \\ 0 & : \quad \text{otherwise} \end{cases} . \tag{6.51}$$

We will make use of the indicator function below, when we discuss computing statistics

0	0	0	0	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	X	0	0	X	X	0
0	0	0	0	X	0	0	X	X	0
0	0	0	0	X	X	X	X	X	0
0	X	X	X	X	X	X	X	X	0
0	X	X	X	X	X	X	X	X	0
0	0	0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	2	2	2	2	0
0	4	4	4	2	2	2	2	2	0
0	4	4	4	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0

(b)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	2	2	X	2	0
0	4	4	4	X	2	2	2	2	0
0	4	4	4	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0

(c)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	2	2	0
0	0	0	0	2	0	0	2	2	0
0	0	0	0	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0

(d)

Figure 6.4: The image in (a) is a simple binary image. Background pixels are denoted by *0* and object pixels are denoted by *X*. Image (b) shows the assigned labels after the first raster scan. In image (c), an *X* denotes those pixels at which an equivalence is noted during the first raster scan. Image (d) shows the final component labelled image.

Figure 6.5: The image of Figure 6.3 after connected components have been labelled.

associated with the various objects in the image.

6.5 Position and Orientation

The ultimate goal of a robotic system is to manipulate objects in the world. In order to achieve this, it is necessary to know the positions and orientations of the objects that are to be manipulated. In this section, we address the problem of determining the position and orientation of objects *in the image*. If the camera has been calibrated, it is then possible to use these image position and orientations to infer the 3D positions and orientations of the objects. In general, this problem of inferring the 3D position and orientation from image measurements can be a difficult problem; however, for many cases that are faced by industrial robots we an obtain adequate solutions. For example, when grasping parts from a conveyor belt, the depth, z, is fixed, and the perspective projection equations can be inverted if z is known.

We begin the section with a general discussion of moments, since moments will be used in the computation of both position and orientation of objects in the image.

6.5.1 Moments

Moments are functions defined on the image that can be used to summarize various aspects of the shape and size of objects in the image. The i, j moment for the k^{th} object, denoted by $m_{ij}(k)$, is defined by

$$m_{ij}(k) = \sum_{r,c} r^i c^j \mathcal{I}_k(r, c). \tag{6.52}$$

From this definition, it is evident that m_{00} is merely number of pixels in the object. The order of a moment is defined to be the sum $i + j$. The first order moments are of particular

interest when computing the centroid of an object, and they are given by

$$m_{10}(k) = \sum_{r,c} r\mathcal{I}_k(r,c), \qquad m_{01}(k) = \sum_{r,c} c\mathcal{I}_k(r,c). \tag{6.53}$$

It is often useful to compute moments with respect to the object center of mass. By doing so, we obtain characteristics that are invariant with respect to translation of the object. These moments are called *central moments*. The i,j central moment for the k^{th} object is defined by

$$C_{ij}(k) = \sum_{r,c} (r-\bar{r})^i (c-\bar{c})^j \mathcal{I}_k(r,c), \tag{6.54}$$

in which $(\bar{r},\bar{c})$ are the coordinates for the center of mass, or centroid, of the object.

6.5.2 The Centroid of an Object

It is convenient to define the position of an object to be the object's center of mass or centroid. By definition, the center of mass of an object is that point $(\bar{r},\bar{c})$ such that, if all of the object's mass were concentrated at $(\bar{r},\bar{c})$ the first moments would not change. Thus, we have

$$\sum_{r,c} \bar{r}_i \mathcal{I}_i(r,c) = \sum_{r,c} r\mathcal{I}_i(r,c) \qquad \Rightarrow \qquad \bar{r}_i = \frac{\sum_{r,c} r\mathcal{I}_i(r,c)}{\sum_{r,c} \mathcal{I}_i(r,c)} = \frac{m_{10}(i)}{m_{00}(i)} \tag{6.55}$$

$$\sum_{r,c} \bar{c}_i \mathcal{I}_i(r,c) = \sum_{r,c} c\mathcal{I}_i(r,c) \qquad \Rightarrow \qquad \bar{c}_i = \frac{\sum_{r,c} c\mathcal{I}_i(r,c)}{\sum_{r,c} \mathcal{I}_i(r,c)} = \frac{m_{01}(i)}{m_{00}(i)}. \tag{6.56}$$

Figure 6.6(a) shows the centroids for the connected components of the image of Figure 6.3.

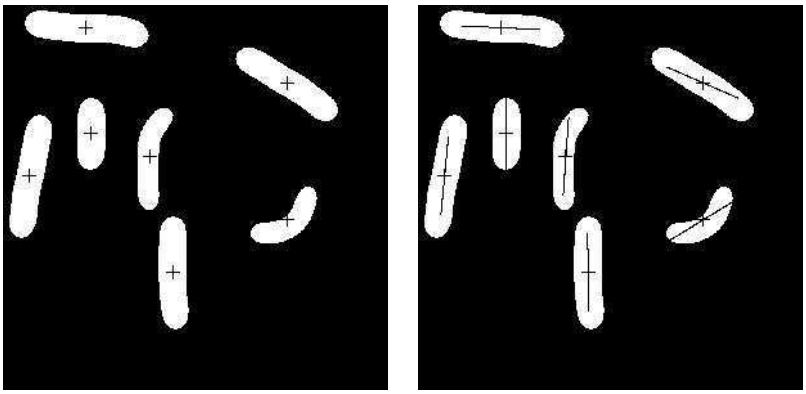
6.5.3 The Orientation of an Object

We will define the orientation of an object in the image to be the orientation of an axis that passes through the object such that the second moment of the object about that axis is minimal. This axis is merely the two-dimensional equivalent of the axis of least inertia.

For a given line in the image, the second moment of the object about that line is given by

$$\mathcal{L} = \sum_{r,c} d^2(r,c)\mathcal{I}(r,c) \tag{6.57}$$

in which $d(r,c)$ is the minimum distance from the pixel with coordinates (r,c) to the line. Our task is to minimize $\mathcal{L}$ with respect to all possible lines in the image plane. To do this, we will use the ρ,θ parameterization of lines, and compute the partial derivatives of $\mathcal{L}$ with respect to ρ and θ. We find the minimum by setting these partial derivatives to zero.



(a) (b)

Figure 6.6: (a) The image of figure 6.3 showing the centroids of each component. (b) The image of figure 6.3 showing the orientation of each component.

With the ρ, θ parameterization, a line consists of all those points x, y that satisfy

$$x \cos\theta + y \sin\theta - \rho = 0. \tag{6.58}$$

Thus, $(\cos\theta, \sin\theta)$ gives the unit normal to the line, and ρ gives the perpendicular distance to the line from the origin. Under this parameterization, the distance from the line to the point with coordinates (r, c) is given by

$$d(r, c) = r \cos\theta + c \sin\theta - \rho. \tag{6.59}$$

Thus, our task is to find

$$\mathcal{L}^\star = \min_{\rho,\theta} \sum_{r,c} (r \cos\theta + c \sin\theta - \rho)^2 \mathcal{I}(r, c) \tag{6.60}$$

We compute the partial derivative with respect to ρ as

$$\begin{aligned}
\frac{d}{d\rho}\mathcal{L} &= \frac{d}{d\rho} \sum_{r,c} (r \cos\theta + c \sin\theta - \rho)^2 \mathcal{I}(r, c) && (6.61)\\
&= -2 \sum_{r,c} (r \cos\theta + c \sin\theta - \rho) \mathcal{I}(r, c) && (6.62)\\
&= -2 \cos\theta \sum_{r,c} r\mathcal{I}(r, c) - 2 \sin\theta \sum_{r,c} c\mathcal{I}(r, c) + 2\rho \sum_{r,c} \mathcal{I}(r, c) && (6.63)\\
&= -2(\cos\theta m_{10} + \sin\theta m_{01} - \rho m_{00}) && (6.64)\\
&= -2(m_{00}\bar{r} \cos\theta + m_{00}\bar{c} \sin\theta - \rho m_{00}) && (6.65)\\
&= -2m_{00}(\bar{r} \cos\theta + \bar{c} \sin\theta - \rho). && (6.66)
\end{aligned}$$

Now, setting this to zero we obtain

$$\bar{r}\cos\theta + \bar{c}\sin\theta - \rho = 0. \tag{6.67}$$

But this is just the equation of a line that passes through the point $(\bar{r}, \bar{c})$, and therefore we conclude that the inertia is minimized by a line that passes through the center of mass. We can use this knowledge to simplify the remaining computations. In particular, define the new coordinates (r', c') as

$$r' = r - \bar{r}, \qquad c' = c - \bar{c}. \tag{6.68}$$

The line that minimizes $\mathcal{L}$ passes through the point $r' = 0, c' = 0$, and therefore its equation can be written as

$$r'\cos\theta + c'\sin\theta = 0. \tag{6.69}$$

Before computing the partial derivative of $\mathcal{L}$ (expressed in the new coordinate system) with respect to θ, it is useful to perform some simplifications.

$$\begin{aligned}
\mathcal{L} &= \sum_{r,c}(r'\cos\theta + c'\sin\theta)^2\mathcal{I}(r,c) && (6.70)\\
&= \cos^2\theta\sum_{r,c}(r')^2\mathcal{I}(r,c) + 2\cos\theta\sin\theta\sum_{r,c}(r'c')\mathcal{I}(r,c) + \sin^2\theta\sum_{r,c}(c')^2\mathcal{I}(r,c) && (6.71)\\
&= C_{20}\cos^2\theta + 2C_{11}\cos\theta\sin\theta + C_{02}\sin^2\theta && (6.72)
\end{aligned}$$

in which the C_{ij} are the central moments given in (6.54). Note that the central moments depend on neither ρ nor θ.

The final set of simplifications that we will make all rely on the double angle identities:

$$\begin{aligned}
cos^2\theta &= \frac{1}{2} + \frac{1}{2}\cos 2\theta && (6.73)\\
sin^2\theta &= \frac{1}{2} - \frac{1}{2}\cos 2\theta && (6.74)\\
cos\theta\sin\theta &= \frac{1}{2}\sin 2\theta. && (6.75)
\end{aligned}$$

Substituting these into our expression for $\mathcal{L}$ we obtain

$$\begin{aligned}
\mathcal{L} &= C_{20}(\frac{1}{2} + \frac{1}{2}\cos 2\theta) + 2C_{11}(\frac{1}{2}\sin 2\theta) + C_{02}(\frac{1}{2} - \frac{1}{2}\cos 2\theta) && (6.76)\\
&= \frac{1}{2}(C_{20} + C_{02}) + \frac{1}{2}(C_{20} - C_{02})\cos 2\theta + \frac{1}{2}C_{11}\sin 2\theta && (6.77)
\end{aligned}$$

It is now easy to compute the partial derivative with respect to θ:

$$\begin{aligned}
\frac{d}{d\theta}\mathcal{L} &= \frac{d}{d\theta}\frac{1}{2}(C_{20} + C_{02}) + \frac{1}{2}(C_{20} - C_{02})\cos 2\theta + \frac{1}{2}C_{11}\sin 2\theta && (6.78)\\
&= -(C_{20} - C_{02})\sin 2\theta + C_{11}\cos 2\theta, && (6.79)
\end{aligned}$$

and we setting this to zero we obtain

$$\tan 2\theta = \frac{C_{11}}{C_{20} - C_{02}}. \tag{6.80}$$

Figure 6.6(b) shows the orientations for the connected components of the image of Figure 6.3.

Chapter 7

PATH PLANNING AND COLLISION AVOIDANCE

In previous chapters, we have studied the geometry of robot arms, developing solutions for both the forward and inverse kinematics problems. The solutions to these problems depend only on the intrinsic geometry of the robot, and they do not reflect any constraints imposed by the workspace in which the robot operates. In particular, they do not take into account the possiblity of collision between the robot and objects in the workspace. In this chapter, we address the problem of planning collision free paths for the robot. We will assume that the initial and final configurations of the robot are specified, and that the problem is to find a collision free path for the robot that connects them.

The description of this problem is deceptively simple, yet the path planning problem is among the most difficult problems in computer science. The computational complexity of the best known complete[1] path planning algorithm grows exponentially with the number of internal degrees of freedom of the robot. For this reason, for robot systems with more than a few degrees of freedom, complete algorithms are not used in practice.

In this chapter, we treat the path planning problem as a search problem. The algorithms we describe are not guaranteed to find a solution to all problems, but they are quite effective in a wide range of practical applications. Furthermore, these algorithms are fairly easy to implement, and require only moderate computation time for most problems.

We begin in Section 7.1 by introducing the notion of *configuration space*, and describing how obstacles in the workspace can be mapped into the configuration space. We then introduce path planning methods that use artificial potential fields in Sections 7.2 and 7.3. The corresponding algorithms use gradient descent search to find a collision-free path to the goal, and, as with all gradient descent methods, these algorithms can become trapped in local minima in the potential field. Therefore, in Section 7.4 we describe how random motions can be used to escape local minima. In Section 7.5 we describe another randomized method known as the Probabilistic Roadmap (PRM) method. Finally, since each of these

[1] An algorithm is said to be complete if it finds a solution whenever one exists.

methods generates a sequence of configurations, we describe in Chapter 8 how polynomial splines can be used to generate smooth trajectories from a sequence of configurations.

As in previous chapters, we will restrict our attention in this chapter to the geometric aspects of the problem. In future chapters we will consider the problem of generating motor torques to execute such a trajectory.

7.1 The Configuration Space

In Chapter 3, we learned that the forward kinematic map can be used to determine the position and orientation of the end effector frame given the vector of joint variables. Furthermore, the A_i matrices can be used to infer the position and orientation of the reference frame for any link of the robot. Since each link of the robot is assumed to be a rigid body, the A_i matrices can therefore be used to infer the position of any point on the robot, given the values of the joint variables. In the path planning literature, a complete specification of the location of every point on the robot is referred to as a *configuration*, and the set of all possible configurations is referred to as the *configuration space*. For our purposes, the vector of joint variables, $\boldsymbol{q}$, provides a convenient representation of a configuration. We will denote the configuration space by $\mathcal{Q}$.

For a one link revolute arm, the configuration space is merely the set of orientations of the link, and thus $\mathcal{Q} = S^1$, where S^1 represents the unit circle. We could also say $\mathcal{Q} = SO(2)$. In fact, the choice of S^1 or $SO(2)$ is not particularly important, since these two are equivalent representations. In either case, we can parameterize $\mathcal{Q}$ by a single parameter, the joint angle θ_1. For the two-link planar arm, we have $\mathcal{Q} = S^1 \times S^1 = T^2$, in which T^2 represents the torus, and we can represent a configuration by $\boldsymbol{q} = (\theta_1, \theta_2)$. For a Cartesian arm, we have $\mathcal{Q} = \Re^3$, and we can represent a configuration by $\boldsymbol{q} = (d_1, d_2, d_3) = (x, y, z)$.

Although we have chosen to represent a configuration by a vector of joint variables, the notion of a configuration is more general than this. For example, as we saw in Chapter 2, for any rigid two-dimensional object, we can specify the location of every point on the object by rigidly attaching a coordinate frame to the object, and then specifying the position and orientation of this frame. Thus, for a rigid object moving in the plane we can represent a configuration by the triple $\boldsymbol{q} = (x, y, \theta)$, and the configuration space can be represented by $\mathcal{Q} = \Re^2 \times SO(2)$. Again, this is merely one possible representation of the configuration space, but it is a convenient one given the representations of position and orientation that we have learned in preceeding chapters.

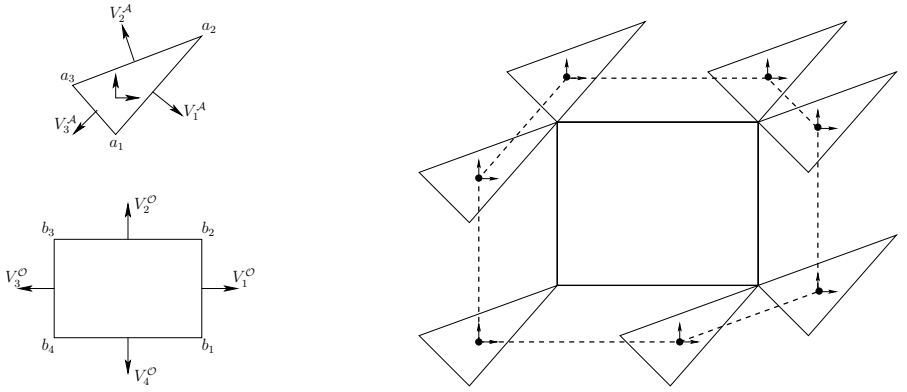
A collision occurs when the robot contacts an obstacle in the workspace. To describe collisions, we introduce some additional notation. We will denote the robot by $\mathcal{A}$, and by $\mathcal{A}(\boldsymbol{q})$ the subset of the workspace that is occupied by the robot at configuration $\boldsymbol{q}$. We denote by $\mathcal{O}_i$ the obstacles in the workspace, and by $\mathcal{W}$ the workspace (i.e., the Cartesian space in which the robot moves). To plan a collision free path, we must ensure that the robot never reaches a configuration $\boldsymbol{q}$ that causes it to make contact with an obstacle in the workspace. The set of configurations for which the robot collides with an obstacle is referred to as the configuration space obstacle, and it is defined by

$$\mathcal{QO} = \{\boldsymbol{q} \in \mathcal{Q} \mid \mathcal{A}(\boldsymbol{q}) \cap \mathcal{O} \neq \emptyset\}.$$

Here, we define $\mathcal{O} = \cup \mathcal{O}_i$. The set of collision-free configurations, referred to as the free configuration space, is then simply

$$\mathcal{Q}_{\text{free}} = \mathcal{Q} \setminus \mathcal{QO}.$$

Example 7.1 *A Rigid Body that Translates in the Plane.*



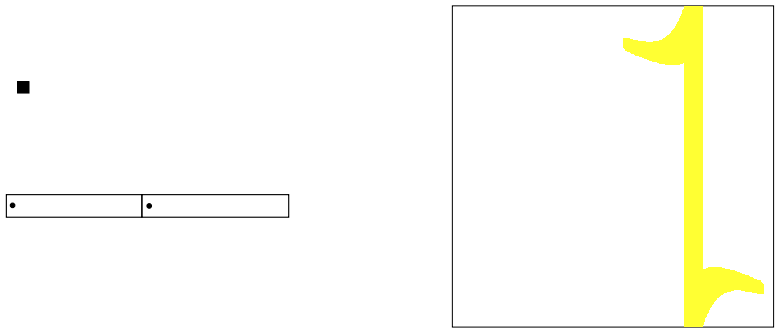
(a) (b)

Figure 7.1: (a) a rigid body, $\mathcal{A}$, in a workspace containing a single rectangular obstacle, $\mathcal{O}$, (b) illustration of the algorithm to construct $\mathcal{QO}$, with the boundary of $\mathcal{QO}$ shown as the dashed line.

Consider a simple gantry robot with two prismatic joints and forward kinematics given by $x = d_1, y = d_2$. For this case, the robot's configuration space is $\mathcal{Q} = \Re^2$, so it is particularly easy to visualize both the configuration space and the configuration space obstacle region. If there is only one obstacle in the workspace and both the robot end-effector and the obstacle are convex polygons, it is a simple matter to compute the configuration space obstacle region, $\mathcal{QO}$ (we assume here that the arm itself is positioned above the workspace, so that the only possible collisions are between the end-effector and obstacles the obstacle).

Let $V_i^{\mathcal{A}}$ denote the vector that is normal to the i^{th} edge of the robot and $V_i^{\mathcal{O}}$ denote the vector that is normal to the i^{th} edge of the obstacle. Define a_i to be the vector from the origin of the robot's coordinate frame to the i^{th} vertex of the robot and b_j to be the vector from the origin of the world coordinate frame to the j^{th} vertex of the obstacle. An example is shown in Figure 7.1(a). The vertices of $\mathcal{QO}$ can be determined as follows.

- *For each pair $V_j^{\mathcal{O}}$ and $V_{j-1}^{\mathcal{O}}$, if $V_i^{\mathcal{A}}$ points between $-V_j^{\mathcal{O}}$ and $-V_{j-1}^{\mathcal{O}}$ then add to $\mathcal{QO}$ the vertices $b_j - a_i$ and $b_j - a_{i+1}$.*



(a) (b)

Figure 7.2: (a) A two-link planar arm and a single polygonal obstacle. (b) The corresponding configuration space obstacle region.

- *For each pair $V_i^{\mathcal{A}}$ and $V_{i-1}^{\mathcal{A}}$, if $V_j^{\mathcal{O}}$ points between $-V_i^{\mathcal{A}}$ and $-V_{i-1}^{\mathcal{A}}$ then add to $\mathcal{QO}$ the vertices $b_j - a_i$ and $b_{j+1} - a_i$.*

This is illustrated in Figure 7.1(b). Note that this algorithm essentially places the robot at all positions where vertex-vertex contact between robot and obstacle are possible. The origin of the robot's local coordinate frame at each such configuration defines a vertex of $\mathcal{QO}$. The polygon defined by these vertices is $\mathcal{QO}$.

If there are multiple convex obstacles $\mathcal{O}_i$, then the configuration space obstacle region is merely the union of the obstacle regions $\mathcal{QO}_i$, for the individual obstacles. For a nonconvex obstacle, the configuration space obstacle region can be computed by first decomposing the nonconvex obstacle into convex pieces, $\mathcal{O}_i$, computing the C-space obstacle region, $\mathcal{QO}_i$ for each piece, and finally, computing the union of the $\mathcal{QO}_i$.

◇

Example 7.2 *A Two Link Planar Arm.*

For robots with revolute joints, computation of $\mathcal{QO}$ is more difficult. Consider a two-link planar arm in a workspace containing a single obstacle as shown in Figure 7.2(a). The configuration space obstacle region is illustrated in 7.2(b). The horizontal axis in 7.2(b) corresponds to θ_1, and the vertical axis to θ_2. For values of θ_1 very near $\pi/2$, the first link of the arm collides with the obstacle. Further, when the first link is near the obstacle (θ_1 near $\pi/2$), for some values of θ_2 the second link of the arm collides with the obstacle. The region $\mathcal{QO}$ shown in 7.2(b) was computed using a discrete grid on the configuration space. For each cell in the grid, a collision test was performed, and the cell was shaded when a

collision occured. Thus, this is only an approximate representation of $\mathcal{QO}$.

$\diamond$

Computing $\mathcal{QO}$ for the two-dimensional case of $\mathcal{Q} = \Re^2$ and polygonal obstacles is straightforward, but, as can be seen from the two-link planar arm example, computing $\mathcal{QO}$ becomes difficult for even moderately complex configuration spaces. In the general case (e.g., articulated arms or rigid bodies that can both translate and rotate), the problem of computing a representation of the configuration space obstacle region is intractable. One of the reasons for this complexity is that the size of the representation of the configuration space tends to grow exponentially with the number of degrees of freedom. This is easy to understand intuitively by considering the number of n-dimensional unit cubes needed to fill a space of size k. For the one dimensional case, k unit intervals will cover the space. For the two-dimensional case, k^2 squares are required. For the three-dimensional case, k^3 cubes are required, and so on. Therefore, in this chapter we will develop methods that avoid the construction of an explicit representation of $\mathcal{Q}_{\text{free}}$.

The path planning problem is to find a path from an initial configuration $\boldsymbol{q}_{\text{init}}$ to a final configuration $\boldsymbol{q}_{\text{final}}$, such that the robot does not collide with any obstacle as it traverses the path. More formally, A collision-free path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$ is a continuous map, $\tau : [0, 1] \rightarrow \mathcal{Q}_{\text{free}}$, with $\tau(0) = \boldsymbol{q}_{\text{init}}$ and $\tau(1) = \boldsymbol{q}_{\text{final}}$. We will develop path planning methods that compute a sequence of discrete configurations (set points) in the configuration space. In Chapter 8 we will show how smooth trajectories can be generated from such a sequence.

7.2 Path Planning Using Configuration Space Potential Fields

As mentioned above, it is not feasible to build an explicit representation of $\mathcal{Q}_{\text{free}}$. An alternative is to develop a search algorithm that incrementally explores $\mathcal{Q}_{\text{free}}$ while searching for a path. Such a search algorithm requires a strategy for exploring $\mathcal{Q}_{\text{free}}$, and one of the most popular is to use an *artificial potential field* to guide the search. In this section, we will introduce artificial potential field methods. Here we describe how the potential field can be constructed directly on the configuration space of the robot. However, as will become clear, computing the gradient of such a field is not feasible in general, so in Section 7.3 we will develop an alternative, in which the potential field is first constructed on the workspace, and then its effects are mapped to the configuration space.

The basic idea behind the potential field approaches is as follows. The robot is treated as a point particle in the configuration space, under the influence of an artificial potential field U. The field U is constructed so that the robot is attracted to the final configuration, $\boldsymbol{q}_{\text{final}}$, while being repelled from the boundaries of $\mathcal{QO}$. If U is constructed appropriately, there will be a single global minimum of U at $\boldsymbol{q}_{\text{final}}$, and no local minima. Unfortunately, as we will discuss below, it is often difficult to construct such a field.

In general, the field U is an additive field consisting of one component that attracts the

robot to $\boldsymbol{q}_{\text{final}}$ and a second component that repels the robot from the boundary of $\mathcal{QO}$,

$$U(\boldsymbol{q}) = U_{\text{att}}(\boldsymbol{q}) + U_{\text{rep}}(\boldsymbol{q}). \tag{7.1}$$

Given this formulation, path planning can be treated as an optimization problem, i.e., find the global minimum in U, starting from initial configuration $\boldsymbol{q}_{\text{init}}$. One of the easiest algorithms to solve this problem is gradient descent. In this case, the negative gradient of U can be considered as a force acting on the robot (in configuration space),

$$F(\boldsymbol{q}) = -\nabla U(\boldsymbol{q}) = -\nabla U_{\text{att}}(\boldsymbol{q}) - \nabla U_{\text{rep}}(\boldsymbol{q}). \tag{7.2}$$

In the remainder of this section, we will describe typical choices for the attractive and repulsive potential fields, and a gradient descent algorithm that can be used to plan paths in this field.

7.2.1 The Attractive Field

There are several criteria that the potential field U_{att} should satisfy. First, U_{att} should be monotonically increasing with distance from $\boldsymbol{q}_{\text{final}}$. The simplest choice for such a field is a field that grows linearly with the distance from $\boldsymbol{q}_{\text{final}}$, a so-called conic well potential. However, the gradient of such a field has unit magnitude everywhere but the origin, where it is zero. This can lead to stability problems, since there is a discontinuity in the attractive force at the origin. We prefer a field that is continuously differentiable, such that the attractive force decreases as the robot approaches $\boldsymbol{q}_{\text{final}}$. The simplest such field is a field that grows quadratically with the distance to $\boldsymbol{q}_{\text{final}}$. Let $\rho_f(\boldsymbol{q})$ be the Euclidean distance between $\boldsymbol{q}$ and $\boldsymbol{q}_{\text{final}}$, i.e., $\rho_f(\boldsymbol{q}) = ||\boldsymbol{q} - \boldsymbol{q}_{\text{final}}||$. Then we can define the quadratic field by

$$U_{\text{att}}(\boldsymbol{q}) = \frac{1}{2}\zeta\rho_f^2(\boldsymbol{q}), \tag{7.3}$$

in which ζ is a parameter used to scale the effects of the attractive potential. This field is sometimes referred to as a parabolic well. For $\boldsymbol{q} = (q^1, \cdots q^n)^T$, the gradient of U_{att} is given by

$$\begin{aligned}
\nabla U_{\text{att}}(\boldsymbol{q}) &= \nabla\frac{1}{2}\zeta\rho_f^2(\boldsymbol{q}) && (7.4)\\
&= \nabla\frac{1}{2}\zeta||\boldsymbol{q} - \boldsymbol{q}_{\text{final}}||^2 && (7.5)\\
&= \frac{1}{2}\zeta\nabla\sum(q^i - q^i_{\text{final}})^2 && (7.6)\\
&= \zeta(q^1 - q^1_{\text{final}}, \cdots, q^n - q^n_{\text{final}})^T && (7.7)\\
&= \zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}}). && (7.8)
\end{aligned}$$

Here, (7.7) follows since

$$\frac{\partial}{\partial q^j}\sum_i (q^i - q^i_{\text{final}})^2 = 2(q^j - q^j_{\text{final}}).$$

So, for the parabolic well, the attractve force, $F_{\text{att}}(q) = -\nabla U_{\text{att}}(\boldsymbol{q})$ is a vector directed toward $\boldsymbol{q}_{\text{final}}$ with magnitude linearly related to the distance from $\boldsymbol{q}$ to $\boldsymbol{q}_{\text{final}}$.

Note that while $F_{\text{att}}(\boldsymbol{q})$ converges linearly to zero as $\boldsymbol{q}$ approaches $\boldsymbol{q}_{\text{final}}$ (which is a desirable property), it grows without bound as $\boldsymbol{q}$ moves away from $\boldsymbol{q}_{\text{final}}$. If $\boldsymbol{q}_{\text{init}}$ is very far from $\boldsymbol{q}_{\text{final}}$, this may produce an attractive force that is too large. For this reason, we may choose to combine the quadratic and conic potentials so that the conic potential attracts the robot when it is very distant from $\boldsymbol{q}_{\text{final}}$ and the quadratic potential attracts the robot when it is near $\boldsymbol{q}_{\text{final}}$. Of course it is necessary that the gradient be defined at the boundary between the conic and quadratic portions. Such a field can be defined by

$$U_{\text{att}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\zeta\rho_f^2(\boldsymbol{q}) & : \quad \rho_f(\boldsymbol{q}) \leq d \\ \\ d\zeta\rho_f(\boldsymbol{q}) - \frac{1}{2}\zeta d^2 & : \quad \rho_f(\boldsymbol{q}) > d \end{cases} . \tag{7.9}$$

and in this case we have

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(\boldsymbol{q}) = \begin{cases} -\zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}}) & : \quad \rho_f(\boldsymbol{q}) \leq d \\ \\ -\dfrac{d\zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}})}{\rho_f(\boldsymbol{q})} & : \quad \rho_f(\boldsymbol{q}) > d \end{cases} . \tag{7.10}$$

The gradient is well defined at the boundary of the two fields since at the boundary $d = \rho_f(\boldsymbol{q})$, and the gradient of the quadratic potential is equal to the gradient of the conic potential, $F_{\text{att}}(q) = -\zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}})$.

7.2.2 The Repulsive field

There are several criteria that the repulsive field should satisfy. It should repel the robot from obstacles, never allowing the robot to collide with an obstacle, and, when the robot is far away from an obstacle, that obstacle should exert little or no influence on the motion of the robot. One way to achieve this is to define a potential that goes to infinity at obstacle boundaries, and drops to zero at a certain distance from the obstacle. If we define ρ_0 to be the distance of influence of an obstace (i.e., an obstacle will not repel the robot if the distance from the robot to the obstacle is greater that ρ_0), one potential that meets these criteria is given by

$$U_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)^2 & : \quad \rho(\boldsymbol{q}) \leq \rho_0 \\ \\ 0 & : \quad \rho(\boldsymbol{q}) > \rho_0 \end{cases} \tag{7.11}$$

in which $\rho(\boldsymbol{q})$ is the shortest distance from $\boldsymbol{q}$ to a configuration space obstacle boundary, and η is a scalar gain coefficient that determines the influence of the repulsive field. If $\mathcal{QO}$

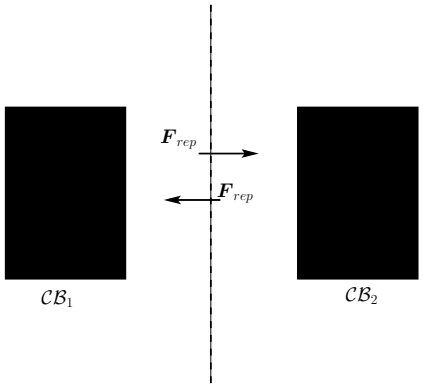


Figure 7.3: Situation in which the gradient of the repuslive potential of (7.11) is not continuous.

consists of a single convex region, the corresponding repulsive force is given by the negative gradient of the repulsive field,

$$F_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \eta\left(\dfrac{1}{\rho(\boldsymbol{q})} - \dfrac{1}{\rho_0}\right)\dfrac{1}{\rho^2(\boldsymbol{q})}\nabla\rho(\boldsymbol{q}) & : \quad \rho(\boldsymbol{q}) \leq \rho_0 \\ \\ 0 & : \quad \rho(\boldsymbol{q}) > \rho_0 \end{cases} . \tag{7.12}$$

When $\mathcal{QO}$ is convex, the gradient of the distance to the nearest obstacle is given by

$$\nabla\rho(\boldsymbol{q}) = \frac{\boldsymbol{q} - \boldsymbol{b}}{||\boldsymbol{q} - \boldsymbol{b}||}, \tag{7.13}$$

in which $\boldsymbol{b}$ is the point in the boundary of $\mathcal{QO}$ that is nearest to $\boldsymbol{q}$. The derivation of (7.12) and (7.13) are left as exercises **??**.

If $\mathcal{QO}$ is not convex, then ρ won't necessarily be differentiable everywhere, which implies discontinuity in the force vector. Figure 7.3 illustrates such a case. Here $\mathcal{QO}$ contains two rectangular obstacles. For all configurations to the left of the dashed line, the force vector points to the right, while for all configurations to the right of the dashed line the force vector points to the left. Thus, when the configuration of the robot crosses the dashed line, a discontinuity in force occurs. There are various ways to deal with this problem. The simplest of these is merely to ensure that the regions of influence of distinct obstacles do not overlap.

7.2.3 Gradient Descent Planning

Gradient descent is a well known approach for solving optimization problems. The idea is simple. Starting at the initial configuration, take a small step in the direction of the negative gradient (i.e., in the direction decreases the potential as quickly as possible). This gives a

new configuration, and the process is repeated until the final configuration is reached. More formally, we can define a gradient descent algorithm as follows.

1. $\boldsymbol{q}^0 \leftarrow \boldsymbol{q}_{\text{init}}$, $i \leftarrow 0$
2. **IF** $\boldsymbol{q}^i \neq \boldsymbol{q}_{\text{final}}$
   $\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}^i + \alpha^i \frac{F(\boldsymbol{q}^i)}{||F(\boldsymbol{q}^i)||}$
   $i \leftarrow i + 1$
   ELSE return $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$
3. **GO TO** 2

In this algorithm, the notation $\boldsymbol{q}^i$ is used to denote the value of $\boldsymbol{q}$ at the i^{th} iteration (not the i^{th} componenent of the vector $\boldsymbol{q}$) and the final path consists of the sequence of iterates $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$. The value of the scalar α^i determines the step size at the i^{th} iteration; it is multiplied by the unit vector in the direction of the resultant force. It is important that α^i be small enough that the robot is not allowed to "jump into" obstacles, while being large enough that the algorithm doesn't require excessive computation time. In motion planning problems, the choice for α^i is often made on an ad hoc or empirical basis, perhaps based on the distance to the nearest obstacle or to the goal. A number of systematic methods for choosing α^i can be found in the optimization literature [**?**]. The constants ζ and η used to define U_{att} and U_{rep} essentially arbitrate between attractive and repulsive forces. Finally, it is unlikely that we will ever exactly satisfy the condition $\boldsymbol{q}^i = \boldsymbol{q}_{\text{final}}$. For this reason, this condition is often replaced with the more forgiving condition $||\boldsymbol{q}^i - \boldsymbol{q}_{\text{final}}|| < \epsilon$, in which ϵ is chosen to be sufficiently small, based on the task requirements.

The problem that plagues all gradient descent algorithms is the possible existence of local minima in the potential field. For appropriate choice of α^i, it can be shown that the gradient descent algorithm is guaranteed to converge to a minimum in the field, but there is no guarantee that this minimum will be the global minimum. In our case, this implies that there is no guarantee that this method will find a path to $\boldsymbol{q}_{\text{final}}$. An example of this situation is shown in Figure 7.4. We will discuss ways to deal this below in Section 7.4.

One of the main difficulties with this planning approach lies in the evaluation of ρ and $\nabla\rho$. In the general case, in which both rotational and translational degrees of freedom are allowed, this becomes even more difficult. We address this general case in the next section.

7.3 Planning Using Workspace Potential Fields

As described above, in the general case, it is extremely difficult to compute an explicit representation of $\mathcal{QO}$, and thus it can be extremely difficult to compute ρ and $\nabla\rho$. In fact, in general for a curved surface there does not exist a closed form expression for the distance from a point to the surface. Thus, even if we had access to an explicit representation of $\mathcal{QO}$, it would still be difficult to compute ρ and $\nabla\rho$ in (7.12). In order to deal with these difficulties, in this section we will modify the potential field approach of Section 7.2 so that the potential function is defined on the workspace, $\mathcal{W}$, instead of the configuration space,

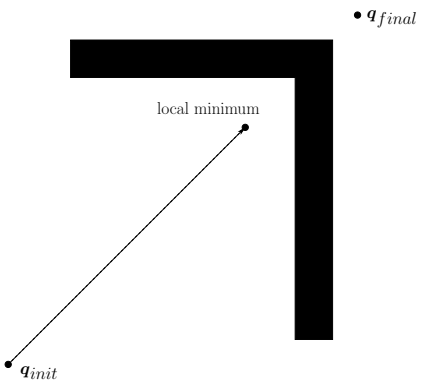


Figure 7.4: This figure illustrates the progress of a gradient descent algorithm from $\boldsymbol{q}_{\text{init}}$ to a local minimum in the field U.

$\mathcal{Q}$. Since $\mathcal{W}$ is a subset of a low dimensional space (either $\Re^2$ or $\Re^3$), it will be much easier to implement and evaluate potential functions over $\mathcal{W}$ than over $\mathcal{Q}$.

We begin by describing a method to define an appropriate potential field on the workspace. This field should have the properties that the potential is at a minimum when the robot is in its goal configuration, and the potential should grow without bound as the robot approaches an obstacle. As above, we will define a global potential field that is the sum of attractive and repulsive fields. Once we have constructed the workspace potential, we will develop the tools to map its gradient to changes in the joint variable values (i.e., we will map workspace forces to changes in configuration). Finally, we will present a gradient descent algorithm similar to the one presented above, but which can be applied to robots with more complicated kinematics.

7.3.1 Defining Workspace Potential Fields

As before, our goal in defining potential functions is to construct a field that repels the robot from obstacles, with a global minimum that corresponds to $\boldsymbol{q}_{\text{final}}$. In the configuration space, this task was conceptually simple because the robot was represented by a single point, which we treated as a point mass under the influence of a potential field. In the workspace, things are not so simple; the robot is an articulated arm with finite volume. Evaluating the effect of a potential field over the arm would involve computing an integral over the volume of the arm, and this can be quite complex (both mathematically and computationally). An alternative approach is to select a subset of points on the robot, called control points, and to define a workspace potential for each of these points. The global potential is obtained by summing the effects of the individual potential functions. Evaluating the effect a particular potential field on a single point is no different than the evaluations required in Section 7.2, but the required distance and gradient calculations are much simpler.

Let $\mathcal{A}_{\text{att}} = \{\boldsymbol{a}_1, \boldsymbol{a}_2 \cdots \boldsymbol{a}_n\}$ be a set of control points used to define the attractive potential fields. For an n-link arm, we could use the centers of mass for the n links, or the origins

for the DH frames (excluding the fixed frame 0). We denote by $\boldsymbol{a}_i(\boldsymbol{q})$ the position of the i^{th} control point when the robot is at configuration $\boldsymbol{q}$. For each $\boldsymbol{a}_i \in \mathcal{A}_{\text{att}}$ we can define an attractive potential by

$$U_{\text{att}i}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\zeta_i ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})||^2 & : \quad ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| \leq d \\ d\zeta_i ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| - \frac{1}{2}\zeta_i d^2 & : \quad ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| > d \end{cases}. \tag{7.14}$$

For the single point $\boldsymbol{a}_i$, this function is analogous the attractive potential defined in Section 7.2; it combines the conic and quadratic potentials, and reaches its global minimum when the control point reaches its goal position in the workspace. If $\mathcal{A}_{\text{att}}$ contains a sufficient number of independent control points (the origins of the DH frames, e.g.), then when all control points reach their global minimum potential value, the configuration of the arm will be $\boldsymbol{q}_{\text{final}}$.

With this potential function, the workspace force for attractive control point $\boldsymbol{a}_i$ is defined by

$$\mathcal{F}_{\text{att},i}(q) = -\nabla U_{\text{att}i}(\boldsymbol{q}) \tag{7.15}$$

$$= \begin{cases} -\zeta_i(\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})) & : \quad ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| \leq d \\ -\frac{d\zeta_i(\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}}))}{||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})||} & : \quad ||\boldsymbol{a}_i(\boldsymbol{q}) - \boldsymbol{a}_i(\boldsymbol{q}_{\text{final}})|| > d \end{cases}. \tag{7.16}$$

For the workspace repulsive potential fields, we will select a set of fixed control points on the robot $\mathcal{A}_{\text{rep}} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m\}$, and define the repulsive potential for $\boldsymbol{a}_j$ as

$$U_{\text{rep}_j}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\eta_j \left(\frac{1}{\rho(\boldsymbol{a}_j(\boldsymbol{q}))} - \frac{1}{\rho_0}\right)^2 & : \quad \rho(\boldsymbol{a}_j(\boldsymbol{q})) \leq \rho_0 \\ 0 & : \quad \rho(\boldsymbol{a}_j(\boldsymbol{q})) > \rho_0 \end{cases}, \tag{7.17}$$

in which $\rho(\boldsymbol{a}_j(\boldsymbol{q}))$ is the shortest distance between the control point $\boldsymbol{a}_j$ and any *workspace* obstacle, and ρ_0 is the workspace distance of influence in the worksoace for obstacles. The negative gradient of each U_{rep_j} corresponds to a workspace repulsive force,

$$\mathcal{F}_{\text{rep},j}(\boldsymbol{q}) = \begin{cases} \eta_j \left(\frac{1}{\rho(\boldsymbol{a}_j(\boldsymbol{q}))} - \frac{1}{\rho_0}\right) \frac{1}{\rho^2(\boldsymbol{a}_j(\boldsymbol{q}))} \nabla\rho(\boldsymbol{a}_j(\boldsymbol{q})) & : \quad \rho(\boldsymbol{a}_j(\boldsymbol{q})) \leq \rho_0 \\ 0 & : \quad \rho(\boldsymbol{a}_j(\boldsymbol{q})) > \rho_0 \end{cases}, \tag{7.18}$$

in which the notation $\nabla\rho(\boldsymbol{a}_j(\boldsymbol{q}))$ indicates the gradient $\nabla\rho(\boldsymbol{x})$ evaluated at $\boldsymbol{x} = \boldsymbol{a}_j(\boldsymbol{q})$. If $\boldsymbol{b}$ is the point on the workspace obstacle boundary that is closest to the repulsive control

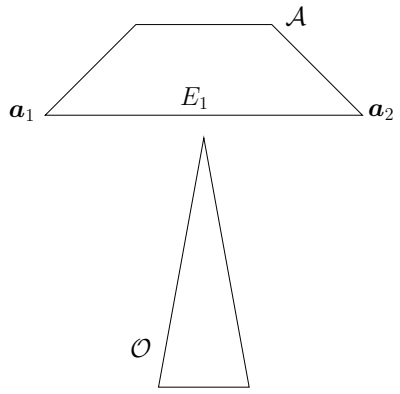


Figure 7.5: The repulsive forces exerted on the robot vertices $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ may not be sufficient to prevent a collision between edge E_1 and the obstacle.

point $\boldsymbol{a}_j$, then $\rho(\boldsymbol{a}_j(\boldsymbol{q})) = ||\boldsymbol{a}_j(\boldsymbol{q}) - \boldsymbol{b}||$, and its gradient is

$$\nabla\rho(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{a}_j(\boldsymbol{q})} = \frac{\boldsymbol{a}_j(\boldsymbol{q}) - \boldsymbol{b}}{||\boldsymbol{a}_j(\boldsymbol{q}) - \boldsymbol{b}||}, \tag{7.19}$$

i.e., the unit vector directed from $\boldsymbol{b}$ toward $\boldsymbol{a}_j(\boldsymbol{q})$.

It is important to note that this selection of repulsive control points does not guarantee that the robot cannot collide with an obstacle. Figure 7.5 shows an example where this is the case. In this figure, the repulsive control points $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are very far from the obstacle $\mathcal{O}$, and therefore the repulsive influence may not be great enough to prevent the robot edge E_1 from colliding with the obstacle. To cope with this problem, we can use a set of floating repulsive control points, $\boldsymbol{a}_{float,i}$, typically one per link of the robot arm. The floating control points are defined as points on the boundary of a link that are closest to any workspace obstacle. Obviously the choice of the $\boldsymbol{a}_{float,i}$ depends on the configuration $\boldsymbol{q}$. For the example shown in Figure 7.5, $\boldsymbol{a}_{float}$ would be located at the center of edge E_1, thus repelling the robot from the obstacle. The repulsive force acting on $\boldsymbol{a}_{float}$ is defined in the same way as for the other control points, using (7.18).

7.3.2 Mapping workspace forces to joint forces and torques

Above we have constrructed potential fields in the robot's workspace, and these fields induce artificial forces on the individual control points on the robot. In this section, we describe how these forces can be used to drive a gradient descent algorithm on the configuration space.

Suppose a force, $\mathcal{F}$ were applied to a point on the robot arm. Such a force would induce forces and torques on the robot's joints. If the joints did not resist these forces, a motion would occur. This is the key idea behind mapping artificial forces in the workspace to motions of the robot arm. Therefore, we now derive the relationship between forces applied to the robot arm, and the resulting forces and torques that are induced on the robot joints.

Let F denote the vector of joint torques (for revolute joints) and forces (for prismatic joints) induced by the workspace force. As we will describe in Chapter 9, the principle of *virtual work* can be used to derive the relationship between $\mathcal{F}$ and F. Let $(\delta x, \delta y, \delta z)^T$ be a virtual displacement in the workspace and let $\delta \boldsymbol{q}$ be a virtual displacement of the robot's joints. Then, recalling that work is the inner product of force and displacement, by applying the principle of virtual work we obtain

$$\mathcal{F} \cdot (\delta x, \delta y, \delta z)^T = F \cdot \delta \boldsymbol{q} \tag{7.20}$$

which can be written as

$$\mathcal{F}^T (\delta x, \delta y, \delta z)^T = F^T \delta \boldsymbol{q}. \tag{7.21}$$

Now, recall from Chapter 8 that

$$\begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = J \delta \boldsymbol{q},$$

in which J is the Jacobian of the forward kinematic map for linear velocity (i.e., the top three rows of the manipulator Jacobian). Substituting this into (7.20) we obtain

$$\mathcal{F}^T J \delta \boldsymbol{q} = F^T \delta \boldsymbol{q} \tag{7.22}$$

$$\tag{7.23}$$

and since this must hold for all virtual displacements $\delta \boldsymbol{q}$, we obtain

$$\mathcal{F}^T J = F^T \tag{7.24}$$

which implies that

$$J^T \mathcal{F} = F. \tag{7.25}$$

Thus we see that one can easily map forces in the workspace to joint forces and torques using (7.25).

Example 7.3 *A Force Acting on a Vertex of a Polygonal Robot.*

Consider the polygonal robot shown in Figure 7.6. The vertex $\boldsymbol{a}$ *has coordinates* $(a_x, a_y)^T$ *in the robot's local coordinate frame. Therefore, if the robot's configuration is given by* $\boldsymbol{q} = (x, y, \theta)$*, the forward kinematic map for vertex* $\boldsymbol{a}$ *(i.e., the mapping from* $\boldsymbol{q} = (x, y, \theta)$ *to the global coordinates of the vertex* $\boldsymbol{a}$*) is given by*

$$\boldsymbol{a}(x, y, \theta) = \begin{bmatrix} x + a_x \cos\theta - a_y \sin\theta \\ y + a_x \sin\theta + a_y \cos\theta \end{bmatrix}. \tag{7.26}$$

The corresponding Jacobian matrix is given by

$$J_a(x, y, \theta) = \begin{bmatrix} 1 & 0 & -a_x \sin\theta - a_y \cos\theta \\ 0 & 1 & a_x \cos\theta - a_y \sin\theta \end{bmatrix} \tag{7.27}$$

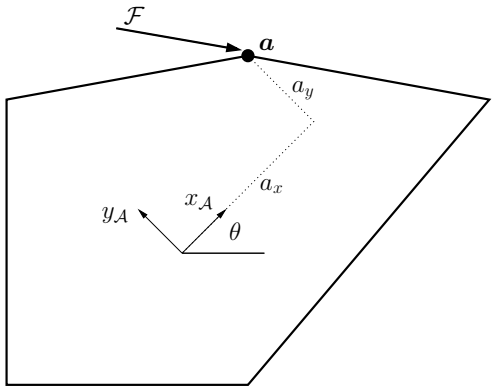


Figure 7.6: The robot $\mathcal{A}$, with coordinate frame oriented at angle θ from the world frame, and vertex $\boldsymbol{a}$ with local coordinates (a_x, a_y).

Therefore, the configuration space force is given by

$$\begin{aligned}\begin{bmatrix} F_x \\ F_y \\ F_\theta \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -a_x \sin\theta - a_y \cos\theta & a_x \cos\theta - a_y \sin\theta \end{bmatrix} \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ -\mathcal{F}_x(a_x \sin\theta - a_y \cos\theta) + \mathcal{F}_y(a_x \cos\theta - a_y \sin\theta) \end{bmatrix} \end{aligned} \tag{7.28}$$

and F_θ corresponds to the torque exerted about the origin of the robot frame.

In this simple case, one can use basic physics to arrive at the same result. In particular, recall that a force, $\mathcal{F}$, exerted at point, $\boldsymbol{a}$, produces a torque, τ, about the point $O_{\mathcal{A}}$, and this torque is given by the relationship $\tau = \boldsymbol{r} \times \mathcal{F}$, in which $\boldsymbol{r}$ is the vector from $O_{\mathcal{A}}$ to $\boldsymbol{a}$. Of course we must express all vectors relative to a common frame, and in three dimensions (since torque will be defined as a vector perpendicular to the plane in which the force acts). If we choose the world frame as our frame of reference, then we have

$$\boldsymbol{r} = \begin{bmatrix} a_x \cos\theta - a_y \sin\theta \\ a_x \sin\theta + a_y \cos\theta \\ 0 \end{bmatrix}$$

and the cross product gives

$$\begin{aligned}\tau &= \boldsymbol{r} \times \mathcal{F} \\ &= \begin{bmatrix} a_x \cos\theta - a_y \sin\theta \\ a_x \sin\theta + a_y \cos\theta \\ 0 \end{bmatrix} \times \begin{bmatrix} \mathcal{F}_x \\ \mathcal{F}_y \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ -\mathcal{F}_x(a_x \sin\theta - a_y \cos\theta) + \mathcal{F}_y(a_x \cos\theta - a_y \sin\theta) \end{bmatrix} \end{aligned} \tag{7.29}$$

Thus we see that the more general expression $J^T\mathcal{F} = F$ gives the same value for torque as the expression $\tau = \boldsymbol{r} \times \mathcal{F}$ from mechanics.

◇

Example 7.4 *Two-link Planar Arm*

Consider a two-link planar arm with the usual DH frame assignment. If we assign the control points as the origins of the DH frames (excluding the base frame), the forward kinematic equations for the arm give

$$\left[\begin{array}{cc} \boldsymbol{a}_1(\theta_1, \theta_2) & \boldsymbol{a}_2(\theta_1, \theta_2) \end{array} \right] = \left[\begin{array}{cc} l_1 \cos\theta_1 & l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 & l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{array} \right]$$

in which l_i are the link lengths (we use l_i rather than a_i to avoid confusion of link lengths and control points). For the problem of motion planning, we require only the Jacobian that maps joint velocities to linear velocities,

$$\left[\begin{array}{c} \dot{x} \\ \dot{y} \end{array} \right] = J \left[\begin{array}{c} \dot{\theta}_1 \\ \dot{\theta}_1 \end{array} \right]. \tag{7.30}$$

For the two-link arm, The Jacobian matrix for $\boldsymbol{a}_2$ is merely the Jacobian that we derived in Chapter 5:

$$J_{\boldsymbol{a}_2}(\theta_1, \theta_2) = \left[\begin{array}{cc} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{array} \right]. \tag{7.31}$$

The Jacobian matrix for $\boldsymbol{a}_1$ is similar, but takes into account that motion of joint two does not affect the velocity of $\boldsymbol{a}_1$,

$$J_{\boldsymbol{a}_1}(\theta_1, \theta_2) = \left[\begin{array}{cc} -a_1 s_1 & 0 \\ a_1 c_1 & 0 \end{array} \right]. \tag{7.32}$$

◇

The total configuration space force acting on the robot is the sum of the configuration space forces that result from all attractive and repulsive control points

$$\begin{aligned} F(\boldsymbol{q}) &= \sum_i F_{\mathrm{att}\,i}(\boldsymbol{q}) + \sum_i F_{\mathrm{rep}_i}(\boldsymbol{q}) \\ &= \sum_i J_i^T(\boldsymbol{q}) \mathcal{F}_{\mathrm{att},i}(\boldsymbol{q}) + \sum_i J_i^T(\boldsymbol{q}) \mathcal{F}_{\mathrm{rep},i}(\boldsymbol{q}) \end{aligned} \tag{7.33}$$

in which $J_i(\boldsymbol{q})$ is the Jacobian matrix for control point $\boldsymbol{a}_i$. It is essential that the addition of forces be done in the configuration space and *not* in the workspace. For example, Figure 7.7 shows a case where two workspace forces, $\mathcal{F}_1$ and $\mathcal{F}_2$, act on opposite corners of a rectang. It is easy to see that $\mathcal{F}_1 + \mathcal{F}_2 = 0$, but that the combination of these forces produces a pure torque about the center of the square.

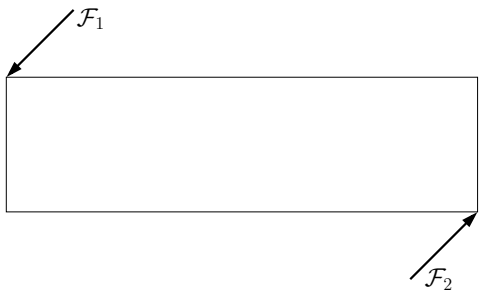


Figure 7.7: This example illustrates why forces must be mapped to the configuration space before they are added. The two forces illustrated in the figure are vectors of equal magnitude in opposite directions. Vector addition of these two forces produces zero net force, but there is a net moment induced by these forces.

Example 7.5 *Two-link planar arm revisited. Consider again the two-link planar arm. Suppose that that the workspace repulsive forces are given by* $\mathcal{F}_{\text{rep},i}(\theta_1, \theta_2) = [\mathcal{F}_{x,i}, \mathcal{F}_{y,i}]^T$. *For the two-link planar arm, the repulsive forces in the configuration space are then given by*

$$F_{\text{rep}}(\boldsymbol{q}) = \begin{bmatrix} -a_1 s_1 & a_1 c_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathcal{F}_{x,1} \\ \mathcal{F}_{y,1} \end{bmatrix} \tag{7.34}$$

$$+ \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} \\ -a_2 s_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} \mathcal{F}_{x,2} \\ \mathcal{F}_{y,2} \end{bmatrix} \tag{7.35}$$

◇

7.3.3 Motion Planning Algorithm

Having defined a configuration space force, we can use the same gradient descent method for this case as in Section 7.3. As before, there are a number of design choices that must be made.

ζ_i controls the relative influence of the attractive potential for control point $\boldsymbol{a}_i$. It is not necessary that all of the ζ_i be set to the same value. Typically, we weight one of the control points more heavily than the others, producing a "follow the leader" type of motion, in which the leader control point is quickly attracted to its final position, and the robot then reorients itself so that the other attractive control points reach their final positions.

η_j controls the relative influence of the repulsive potential for control point $\boldsymbol{a}_j$. As with the ζ_i it is not necessary that all of the η_j be set to the same value. In particular, we typically set the value of η_j to be much smaller for obstacles that are near the goal position of the robot (to avoid having these obstacles repel the robot from the goal).

ρ_0 As with the η_j, we can define a distinct ρ_0 for each obstacle. In particular, we do not want any obstacle's region of influence to include the goal position of any repulsive control point. We may also wish to assign distinct ρ_0's to the obstacles to avoid the possibility of overlapping regions of influence for distinct obstacles.

7.4 Using Random Motions to Escape Local Minima

As noted above, one problem that plagues artificial potential field methods for path planning is the existence of local minima in the potential field. In the case of articulated manipulators, the resultant field U is the sum of many attractive and repulsive fields defined over $\Re^3$. This problem has long been known in the optimization community, where probabilistic methods such as simulated annealing have been developed to cope with it. Similarly, the robot path planning community has developed what are known as *randomized methods* to deal with this and other problems. The first of these methods was developed specifically to cope with the problem of local minima in potential fields.

The first planner to use randomization to escape local minima was called RPP (for Randomized Potential Planner). The basic approach is straightforward: use gradient descent until the planner finds itself stuck in a local minimum, then use a random walk to escape the local minimum. The algorithm is a slight modification of the gradient descent algorithm of Section 7.3.

1. $\boldsymbol{q}^0 \leftarrow \boldsymbol{q}_{\text{init}}$, $i \leftarrow 0$
2. **IF** $\boldsymbol{q}^i \neq \boldsymbol{q}_{\text{final}}$
   $\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}^i + \alpha^i \frac{F(\boldsymbol{q}^i)}{\|F(\boldsymbol{q}^i)\|}$
   $i \leftarrow i + 1$
   ELSE return $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$
3. **IF** stuck in a local minimum
   execute a random walk, ending at $\boldsymbol{q}'$
   $\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}'$
4. **GO TO** 2

The two new problems that must be solved are determining when the planner is stuck in a local minimum and defining the random walk. Typically, a heuristic is used to recognize a local minimum. For example, if several successive $\boldsymbol{q}^i$ lie within a small region of the configuration space, it is likely that there is a nearby local minimum (e.g., if for some small positive ϵ we have $\|\boldsymbol{q}^i - \boldsymbol{q}^{i+1}\| < \epsilon$, $\|\boldsymbol{q}^i - \boldsymbol{q}^{i+2}\| < \epsilon$, and $\|\boldsymbol{q}^i - \boldsymbol{q}^{i+3}\| < \epsilon$ then assume $\boldsymbol{q}^i$ is near a local minimum).

Defining the random walk requires a bit more care. The original approach used in RPP is as follows. The random walk consists of t random steps. A random step from $\boldsymbol{q} = (q_1, \cdots q_n)$ is obtained by randomly adding a small fixed constant to each q_i,

$$\boldsymbol{q}_{\text{random-step}} = (q_1 \pm v_1, \cdots q_n \pm v_n)$$

with v_i a fixed small constant and the probability of adding $+v_i$ or $-v_i$ equal to 1/2 (i.e., a uniform distribution). Without loss of generality, assume that $\boldsymbol{q} = \mathbf{0}$. We can use probability theory to characterize the behavior of the random walk consisting of t random steps. In particular, the probability density function for $\boldsymbol{q}' = (q_1, \cdots, q_n)$ is given by

$$p_i(q_i, t) = \frac{1}{v_i\sqrt{2\pi t}} \exp\left(-\frac{q_i^2}{2v_i^2 t}\right) \tag{7.36}$$

which is a zero mean Gaussian density function with variance $v_i^2 t$. This is a result of the fact that the sum of a set of uniformly distributed random variables is a Gaussian random variable.[2] The variance $v_i^2 t$ essentially determines the range of the random walk. If certain characteristics of local minima (e.g., the size of the basin of attraction) are known in advance, these can be used to select the parameters v_i and t. Otherwise, they can be determined empirically, or based on weak assumptions about the potential field (the latter approach was used in the original RPP).

7.5 Probabilistic Roadmap Methods

The potential field approaches described above incrementally explore $\mathcal{Q}_{\text{free}}$, searching for a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$. At termination, these planners return a single path. Thus, if multiple path planning problems must be solved, such a planner must be applied once for each problem. An alternative approach is to construct a representation of $\mathcal{Q}_{\text{free}}$ that can be used to quickly generate paths when new path planning problems arise. This is useful, for example, when a robot operates for a prolonged period in a single workspace.

In this section, we will describe probabilistic roadmaps (PRMs), which are one-dimensional roadmaps in $\mathcal{Q}_{\text{free}}$ that can be used to quickly generate paths. Once a PRM has been constructed, the path planning problem is reduced to finding paths to connect $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the roadmap (a problem that is typically much easier than finding a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$).

A PRM is a network of simple curve segments, or arcs, that meet at nodes. Each node corresponds to a configuration. Each arc between two nodes corresponds to a collision free path between two configurations. Constructing a PRM is a conceptually straightforward process. First, a set of random configurations is generated to serve as the nodes in the network. Then, a simple, local path planner is used to generate paths that connect pairs of configurations. Finally, if the initial network consists of multiple connected components[3], it is augmented by an enhancement phase, in which new nodes and arcs are added in an attempt to connect disjoint components of the network. To solve a path planning problem,

[2] A Gaussian density function is the classical bell shaped curve. The mean indicates the center of the curve (the peak of the bell) and the variance indicates the width of the bell. The probability density function (pdf) tells how likely it is that the variable q_i will lie in a certain interval. The higher the pdf values, the more likely that q_i will lie in the corresponding interval.

[3] A connected component is a maximal subnetwork of the network such that a path exists in the subnetwork between any two nodes.

2-norm in C-space:	$\|\boldsymbol{q}' - \boldsymbol{q}\| = \left[\sum_{i=1}^{n}(q_i' - q_i)^2\right]^{\frac{1}{2}}$
∞-norm in C-space:	$\max_n \lvert q_i' - q_i\rvert$
2-norm in workspace:	$\left[\sum_{\mathrm{p}\in\mathcal{A}}\big\|\mathrm{p}(\boldsymbol{q}') - \mathrm{p}(\boldsymbol{q})\big\|^2\right]^{\frac{1}{2}}$
∞-norm in workspace:	$\max_{\mathrm{p}\in\mathcal{A}}\big\|\mathrm{p}(\boldsymbol{q}') - \mathrm{p}(\boldsymbol{q})\big\|$.

Table 7.1: Four distance functions from the literature that we have investigated

the simple, local planner is used to connect $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the roadmap, and the resulting network is searched for a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$. These four steps are illustrated in Figure 7.8. We now discuss these steps in more detail.

7.5.1 Sampling the configuration space

The simplest way to generate sample configurations is to sample the configuration space uniformly at random. Sample configurations that lie in $\mathcal{QO}$ are discarded. A simple collision checking algorithm can determine when this is the case. The disadvantage of this approach is that the number of samples it places in any particular region of $\mathcal{Q}_{\text{free}}$ is proportional to the volume of the region. Therefore, uniform sampling is unlikely to place samples in narrow passages of $\mathcal{Q}_{\text{free}}$. In the PRM literature, this is refered to as the *narrow passage problem*. It can be dealt with either by using more intelligent sampling schemes, or by using an enhancement phase during the construction of the PRM. In this section, we discuss the latter option.

7.5.2 Connecting Pairs of Configurations

Given a set of nodes that correspond to configurations, the next step in building the PRM is to determine which pairs of nodes should be connected by a simple path. The typical approach is to attempt to connect each node to it's k nearest neighbors, with k a parameter chosen by the user. Of course, to define the nearest neighbors, a distance function is required. Table 7.1 lists four distance functions that have been popular in the PRM literature. For the equations in this table, the robot has n joints, $\boldsymbol{q}$ and $\boldsymbol{q}'$ are the two configurations corresponding to different nodes in the roadmap, q_i refers to the configuration of the ith joint, and $\mathrm{p}(\boldsymbol{q})$ refers to the workspace reference point p of a set of reference points of the robot, $\mathcal{A}$, at configuration $\boldsymbol{q}$. Of these, the simplest, and perhaps most commonly used, is the 2-norm in configuraiton space.

Once pairs of neighboring nodes have been identified, a simple local planner is used to connect these nodes. Often, a straight line in configuration space is used as the candidate plan, and thus, planning the path between two nodes is reduced to collision checking along a straight line path in the configuration space. If a collision occurs on this path, it can

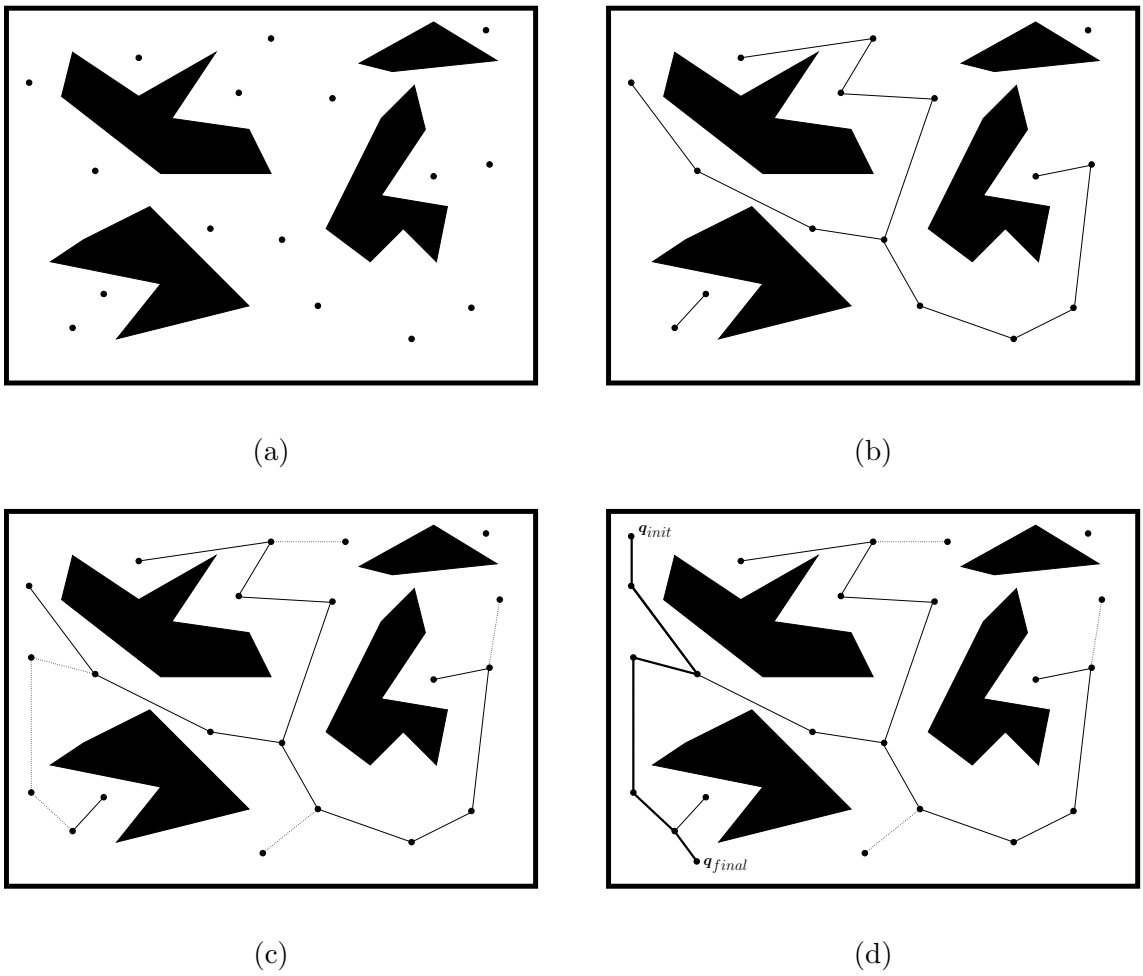


Figure 7.8: (a) A two-dimensional configuration space populated with several random samples (b) One possible PRM for the given configuration space and random samples (c) PRM after enhancement (d) path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$ found by connecting $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the roadmap and then searching the roadmap for a path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q}_{\text{final}}$.

be discarded, or a more sophisticated planner (e.g., RPP discussed above) can be used to attempt to connect the nodes.

The simplest approach to collision detection along the straight line path is to sample the path at a sufficiently fine discretization, and to check each sample for collision. This method works, provided the discretization is fine enough, but it is terribly inefficient. This is because many of the computations required to check for collision at one sample are repeated for the next sample (assuming that the robot has moved only a small amount between the two configurations). For this reason, incremental collision detection approaches have been developed. While these approaches are beyond the scope of this text, a number of collision detection software packages are available in the public domain. Most developers of robot motion planners use one of these packages, rather than implementing their own collision detection routines.

7.5.3 Enhancement

After the initial PRM has been constructed, it is likely that it will consist of multiple connected components. Often these individual components lie in large regions of $\mathcal{Q}_{\text{free}}$ that are connected by narrow passages in $\mathcal{Q}_{\text{free}}$. The goal of the enhancement process is to connect as many of these disjoint components as possible.

One approach to enhancement is to merely attempt to directly connect nodes in two disjoint components, perhaps by using a more sophisticated planner such as RPP. A common approach is to identify the largest connected component, and to attempt to connect the smaller components to it. The node in the smaller component that is closest to the larger component is typically chosen as the candidate for connection. A second approach is to choose a node randomly as candidate for connection, and to bias the random choice based on the number of neighbors of the node; a node with fewer neighbors in the network is more likely to be near a narrow passage, and should be a more likely candidate for connection.

A second approach to enhancement is to add samples more random nodes to the PRM, in the hope of finding nodes that lie in or near the narrow passages. One approach is to identify nodes that have few neighbors, and to generate sample configurations in regions around these nodes. The local planner is then used to attempt to connect these new configurations to the network.

7.5.4 Path Smoothing

After the PRM has been generated, path planning amounts to connecting $\boldsymbol{q}_{\text{init}}$ and $\boldsymbol{q}_{\text{final}}$ to the network using the local planner, and then performing path smoothing, since the resulting path will be composed of straight line segments in the configuration space. The simplest path smoothing algorithm is to select two random points on the path and try to connect them with the local planner. This process is repeated until until no significant progress is made.

7.6 Historical Perspective

The earliest work on robot planning was done in the late sixties and early seventies in a few University-based Artificial Intelligence (AI) labs [**?**, **?**, **?**]. This research dealt with high level planning using symbolic reasoning that was much in vogue at the time in the AI community. Geometry was not often explicitly considered in early robot planners, in part because it was not clear how to represent geometric constraints in a computationally plausible manner. The configuration space and its application to path planning were introduced in [**?**]. This was the first rigorous, formal treatment of the geometric path planning problem, and it initiated a surge in path planning research. The earliest work in geometric path planning developed methods to construct volumetric representations of the free configuration space. These included exact methods (e.g., [**?**]), and approximate methods (e.g., [**?**, **?**, **?**]). In the former case, the best known algorithms have exponential complexity and require exact descriptions of both the robot and its environment, while in the latter case, the size of the representation of C-space grows exponentially in the dimension of the C-space. The best known algorithm for the path planning problem, giving an upper bound on the amount of computation time required to solve the problem, appeared in [**?**]. That real robots rarely have an exact description of the environment, and a drive for faster planning systems led to the development of potential fields approaches [**?**, **?**].

By the early nineties, a great deal of research had been done on the geometric path planning problem, and this work is nicely summarized in the textbook [**?**]. This textbook helped to generate a renewed interest in the path planning problem, and it provided a common framework in which to analyze and express path planning algorithms. Soon after, the research field of *Algorithmic Robotics* was born at a small workshop in San Francisco [**?**].

In the early nineties, randomization was introduced in the robot planning community [**?**], originally to circumvent the problems with local minima in potential fields). Early randomized motion planners proved effective for a large range of problems, but sometimes required extensive computation time for some robots in certain environments [**?**]. This limitation, together with the idea that a robot will operate in the same environment for a long period of time led to the development of the probabilistic roadmap planners [**?**, **?**, **?**].

Finally, much work has been done in the area of collision detection in recent years. [**?**, **?**, **?**, **?**]. This work is primarily focused on finding efficient, incremental methods for detecting collisions between objects when one or both are moving. A number of public domain collision detection software packages are currently available on the internet.

Chapter 8

TRAJECTORY PLANNING

In chapter 7, we learned how to plan paths for robot tasks. In order to execute these plans, a few more details must be specified. For example, what should be the joint velocities and accelerations while traversing the path? These questions are addressed by a trajectory planner. The trajectory planner computes a function $q^d(t)$ that completely specifies the motion of the robot as it traverses the path.

We begin by discussing the trajectory planning problem and how it relates to path planning. We then consider the simple case of planning a trajectory between two configurations. This leads naturally to planning trajectories for paths that are specified as a sequence of configurations. We then turn our attention to the problem of planning trajectories that satisfy constraints that are specified in the Cartesian workspace of the robot (e.g., a straight line trajectory for the end effector). We end the chapter with a discussion of time scaling of trajectories.

8.1 The Trajectory Planning Problem

We begin by distinguising between a path and a trajectory. Recall from Chapter 7 that a *path* from $\boldsymbol{q}_{init}$ to $\boldsymbol{q}_{final}$ is defined as a continuous map, $\tau : [0, 1] \rightarrow \mathcal{Q}$, with $\tau(0) = \boldsymbol{q}_{init}$ and $\tau(1) = \boldsymbol{q}_{final}$. A *trajectory* is a function of time $\boldsymbol{q}(t)$ such that $\boldsymbol{q}(t_0) = \boldsymbol{q}_{init}$ and $\boldsymbol{q}(t_f) = \boldsymbol{q}_{final}$. In this case, $t_f - t_0$ represents the amount of time taken to execute the trajectory. Since the trajectory is parameterized by time, we can compute velocities and accelerations along the trajectories by differentiation. If we think of the argument to τ as a time variable, then a path is a special case of a trajectory, one that will be executed in one unit of time. In other words, in this case τ gives a complete specification of the robot's trajectory, including the time derivatives (since one need only differentiate τ to obtain these).

In general, a path planning algorithm may not actually give the map τ; it might give only a sequence of points along the path (as was the case for several of the path planning algorithms of Chapter 7). Or, it may be the case that we prefer that the duration of the robot motion not be one unit of time. Furthermore, there are other ways that the path

could be specified. In some cases, paths are specified by giving a sequence of end-effector poses, $T_6^0(k\Delta t)$. In this case, the inverse kinematic solution must be used to convert this to a sequence of joint configurations. A common way to specify paths for industrial robots is to physically lead the robot through the desired motion with a teach pendant. In this case, there is no need for calculation of the inverse kinematics. The desired motion is simply recorded as a set of joint angles (actually as a set of encoder values) and the robot can be controlled entirely in joint space. In each of these cases, the path will not serve as a trajectory for the robot.

We first consider **point to point** motion. In this case the task is to plan a trajectory from $\boldsymbol{q}(t_0)$ to $\boldsymbol{q}(t_f)$, i.e., the path is specified by its initial and final configurations. This type of motion is suitable for materials transfer tasks when the workspace is clear of obstacles and is common in so-called **teach and playback mode** where the robot is taught a sequence of moves with a teach pendant. In some cases, there may be constraints on the trajectory (e.g., if the robot must start and end with zero velocity). Nevertheless, it is easy to realize that there are infinitely many trajectories that will satisfy a finite number of constraints on the endpoints. It is common practice therefore to choose trajectories from a finitely parameterizable family, for example, polynomials of degree n, with n dependant on the number of constraints to be satisfied. This is the approach that we will take in this text..

In Section 8.2, we will discuss the trajectory planning problem for point to point motion. Once we have developed techniques for planning trajectories for point to point motion, we will turn our attention in Section 8.3 to the problem of planning trajectories for paths that are specified by a sequence of via points, such as those plans returned by the PRM planner discussed in Chapter 7. In this case, we will deal not only with constraints on initial and final configurations, but also with constraints at each via point (e.g., velocity discontinuities will not be allowed at via points). We will investigate several methods for planning these trajectories, each of which is an extension of a method in Section 8.2 to the multiple via point case.

For some purposes, such as obstacle avoidance, the path of the end-effector can be further constrained by the addition of **via points** intermediate to the initial and final configurations as illustrated in Figure 8.1. Additional constraints on the velocity or acceleration between via points, as for example in so-called **guarded motion**, shown in Figure 8.2, can be handled in the joint interpolation schemes treated in this chapter .

8.2 Trajectories for Point to Point Motion

As described above, the problem here is to find a trajectory that connects an initial to a final configuration while satisfying other specified constraints at the endpoints (e.g., velocity and/or acceleration constraints). Without loss of generality, we will, from here on, consider planning the trajectory for a single joint, since the trajectories for the remaining joints will be created independently and in exactly the same way. Thus, we will concern ourselves with the problem of determining $q(t)$, where $q(t)$ is a scalar joint variable.

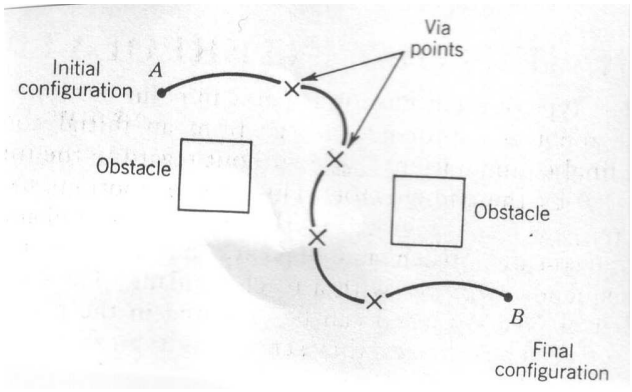


Figure 8.1: Via points to plan motion around obstacles.

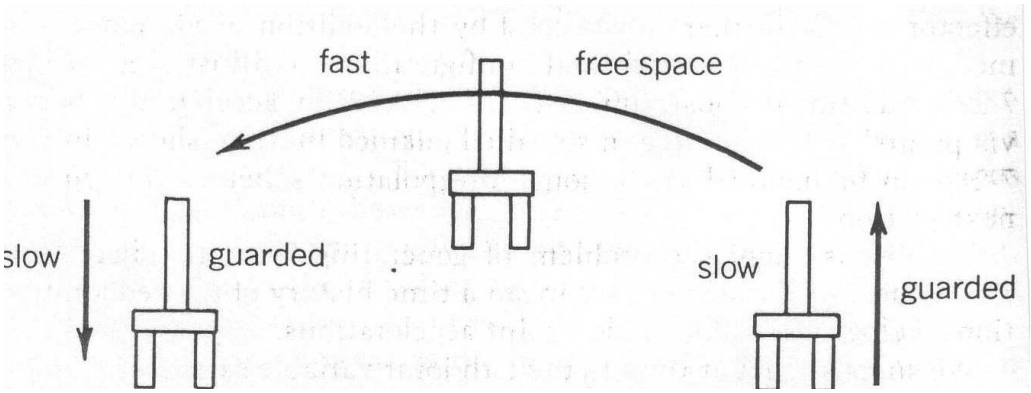


Figure 8.2: Guarded and free motions.

We suppose that at time t_0 the joint variable satisfies

$$q(t_0) = q_0 \tag{8.1}$$
$$\dot{q}(t_0) = v_0 \tag{8.2}$$

and we wish to attain the values at t_f

$$q(t_f) = q_f \tag{8.3}$$
$$\dot{q}(t_f) = v_f. \tag{8.4}$$

Figure 8.3 shows a suitable trajectory for this motion. In addition, we may wish to specify the constraints on initial and final accelerations. In this case we have two the additional equations

$$\ddot{q}(t_0) = \alpha_0 \tag{8.5}$$
$$\ddot{q}(t_f) = \alpha_f. \tag{8.6}$$

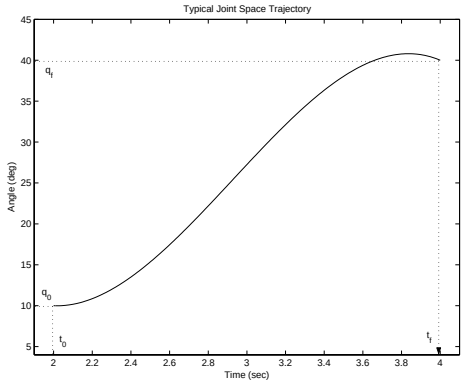


Figure 8.3: Typical Joint Space Trajectory.

8.2.1 Cubic Polynomial Trajectories

Suppose that we wish to generate a trajectory between two configurations, and that we wish to specify the start and end velocities for the trajectory. One way to generate a smooth curve such as that shown in Figure 8.3 is by a polynomial function of t. Since we have four constraints to satisfy, namely (8.1)-(8.3) we require a polynomial with four independent coefficients that can be chosen to satisfy these constraints. Thus we consider a cubic trajectory of the form

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3. \tag{8.7}$$

Then the desired velocity is automatically given as

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2. \tag{8.8}$$

Combining equations (8.7) and (8.8) with the four constraints yields four equations in four unknowns

$$\begin{aligned} q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 && (8.9)\\ v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 && (8.10)\\ q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 && (8.11)\\ v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2. && (8.12) \end{aligned}$$

These four equations can be combined into a single matrix equation

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix}. \tag{8.13}$$

It can be shown (Problem X) that the determinant of the coefficient matrix in Equation 8.13 is equal to $(t_f - t_0)^4$ and, hence, 8.13 always has a unique solution provided a nonzero time interval is allowed for the execution of the trajectory.

Example 8.1 *Writing Equation 8.13 as*

$$Ma = b \tag{8.14}$$

where M is the coefficient matrix, $a = [a_0, a_1, a_2, a_3]^T$ is the vector of coefficients of the cubic polynomial, and $b = [q_0, v_0, q_1, v_1]^T$ is the vector of initial data (initial and final positions and velocities), the Matlab script below computes the general solution as

$$a = M^{-1}b \tag{8.15}$$

$\diamond$

```
%%
%% cubic.m
%%
%% M-file to compute a cubic polynomial reference trajectory
%%
%% q0  = initial position
%% v0  = initial velocity
%% q1  = final position
%% v1  = final velocity
%% t0  = initial time
%% tf  = final time
%%
d  = input(' initial data = [q0,v0,q1,v1,t0,tf] = ')
q0 = d(1); v0= d(2); q1 = d(3); v1 = d(4); t0=d(5); tf=d(6);
%%
t = linspace(t0,tf,100*(tf-t0));
c = ones(size(t));
%%
M = [ 1 t0 t0^2 t0^3;
      0 1 2*t0 3*t0^2;
      1 tf tf^2 tf^3;
      0 1 2*tf 3*tf^2];
%%
b = [q0; v0; q1; v1];
a = inv(M)*b;
%%
% qd = reference position trajectory
```

```
% vd = reference velocity trajectory
% ad = reference acceleration trajectory
%
qd = a(1).*c + a(2).*t +a(3).*t.^2 + a(4).*t.^3;
vd = a(2).*c +2*a(3).*t +3*a(4).*t.^2;
ad = 2*a(3).*c + 6*a(4).*t;
```

Example 8.2 *As an illustrative example, we may consider the special case that the initial and final velocities are zero. Suppose we take* $t_0 = 0$ *and* $t_f = 1$ *sec, with*

$$v_0 = 0 \qquad v_f = 0. \tag{8.16}$$

Thus we want to move from the initial position q_0 *to the final position* q_f *in 1 second, starting and ending with zero velocity. From the above formula (8.13) we obtain*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ 0 \\ q_f \\ 0 \end{bmatrix}. \tag{8.17}$$

This is then equivalent to the four equations

$$\begin{aligned} a_0 &= q_0 && (8.18)\\ a_1 &= 0 && (8.19)\\ a_2 + a_3 &= q_f - q_0 && (8.20)\\ 2a_2 + 3a_3 &= 0. && (8.21) \end{aligned}$$

These latter two can be solved to yield

$$\begin{aligned} a_2 &= 3(q_f - q_0) && (8.22)\\ a_3 &= -2(q_f - q_0). && (8.23) \end{aligned}$$

The required cubic polynomial function is therefore

$$q_i(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3. \tag{8.24}$$

Figure 8.4 shows this trajectory with $q_0 = 10°$, $q_f = -20°$. *The corresponding velocity and acceleration curves are given in Figures 8.5 and 8.6.*

◇

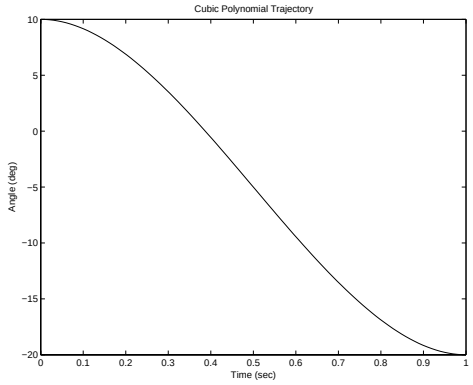


Figure 8.4: Cubic polynomial trajectory.

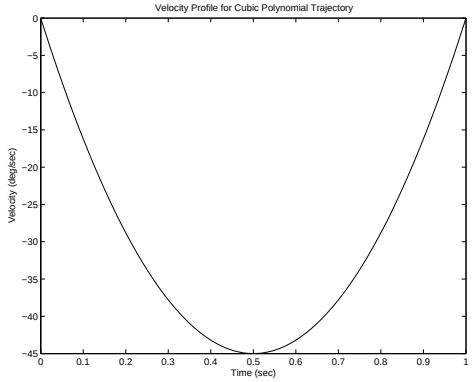


Figure 8.5: Velocity profile for cubic polynomial trajectory

8.2.2 Multiple Cubics

A sequence of moves can be planned using the above formula by using the end conditions q_f, v_f of the i-th move as initial conditions for the $i+1$-st move.

Example 8.3 *Figure 8.9 shows a 6-second move, computed in three parts using (**??**), where the trajectory begins at* 10° *and is required to reach* 40° *at 2-seconds,* 30° *at 4seconds, and* 90° *at 6-seconds, with zero velocity at 0,2,4, and 6 seconds.*
◇

8.2.3 Quintic Polynomial Trajectories

As the above example shows, planning trajectories using multiple cubic trajectories leads to continuous positions and velocities at the blend times but discontinuities in the acceleration. The derivative of acceleration is called the *Jerk*. A discontinuity in acceleration leads to an impulsive Jerk, which may excite vibrational modes in the manipulator and reduce tracking

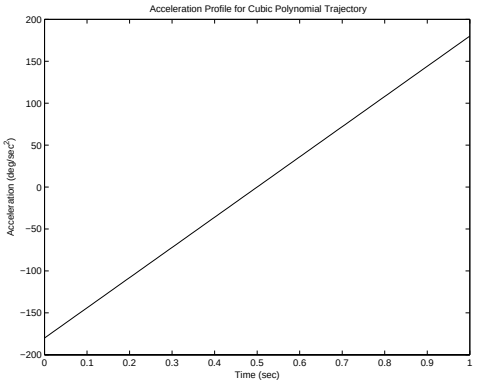


Figure 8.6: Acceleration profile for cubic polynomial trajectory.

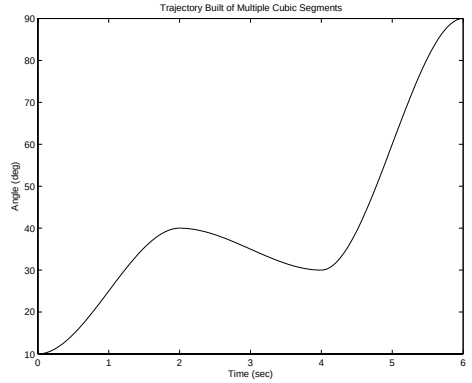


Figure 8.7: Cubic spline trajectory made from three cubic polynomials.

accuracy. For this reason, one may wish to specify constraints on the acceleration as well as on the position and velocity. In this case, we have six constraints (one each for initial and final configurations, initial and final velocities, and initial and final accelerations). Therefore we require a fifth order polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5. \tag{8.25}$$

Using (8.1) - (8.6) and taking the appropriate number of derivatives we obtain the following equations,

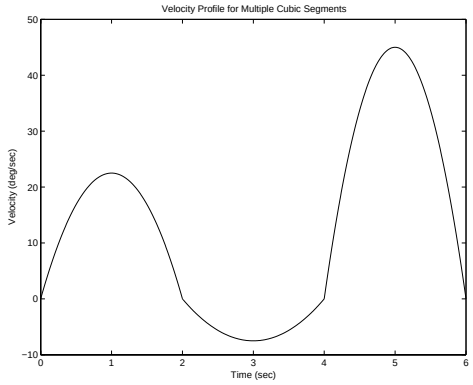


Figure 8.8: Velocity Profile for Multiple Cubic Polynomial Trajectory

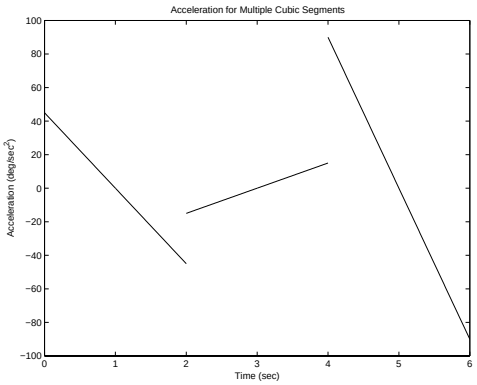


Figure 8.9: Acceleration Profile for Multiple Cubic Polynomial Trajectory

$$
\begin{aligned}
q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \\
v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \\
\alpha_0 &= 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \\
q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\
v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\
\alpha_f &= 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3
\end{aligned}
$$

which can be written as

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix}. \tag{8.26}$$

The Matlab script below gives the general solution to this equation.

```
%%
%% M-file to compute a quintic polynomial reference trajectory
%%
%% q0   = initial position
%% v0   = initial velocity
%% ac0  = initial acceleration
%% q1   = final position
%% v1   = final velocity
%% ac1  = final acceleration
%% t0   = initial time
%% tf   = final time
%%
clear
d = input(' initial data = [q0,v0,ac0,q1,v1,ac1,t0,tf] = ')
q0 = d(1); v0= d(2); ac0 = d(3); q1 = d(4);v1 = d(5);ac1=d(6);t0=d(7); tf=d(8);
%%
t = linspace(t0,tf,100*(tf-t0));
c = ones(size(t));
%%
M = [ 1 t0 t0^2 t0^3 t0^4 t0^5;
      0 1 2*t0 3*t0^2 4*t0^3 5*t0^4;
      0 0 2 6*t0 12*t0^2 20*t0^3;
      1 tf tf^2 tf^3 tf^4 tf^5;
      0 1 2*tf 3*tf^2 4*tf^3 5*tf^4;
      0 0 2 6*tf 12*tf^2 20*tf^3];
 %%
 b=[q0; v0; ac0; q1; v1; ac1];
 a = inv(M)*b;
 %%
 %% qd = position trajectory
 %% vd = velocity trajectory
 %% ad = acceleration trajectory
```

```
%%
qd = a(1).*c + a(2).*t +a(3).*t.^2 + a(4).*t.^3 +a(5).*t.^4 + a(6).*t.^5;
vd = a(2).*c +2*a(3).*t +3*a(4).*t.^2 +4*a(5).*t.^3 +5*a(6).*t.^4;
ad = 2*a(3).*c + 6*a(4).*t +12*a(5).*t.^2 +20*a(6).*t.^3;
plot(t,qd)
```

Example 8.4 *The figures below show a quintic polynomial trajectory with* $q(0) = 0$, $q(2) = 40$ *with zero initial and final velocities and accelerations.*

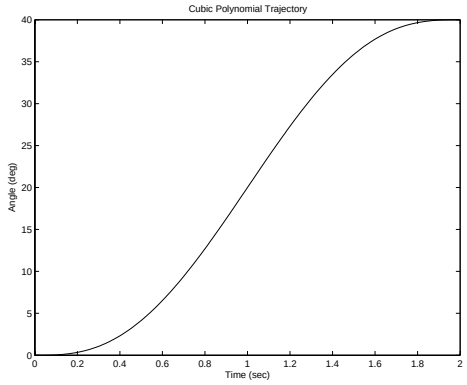


Figure 8.10: Quintic Polynomial Trajectory

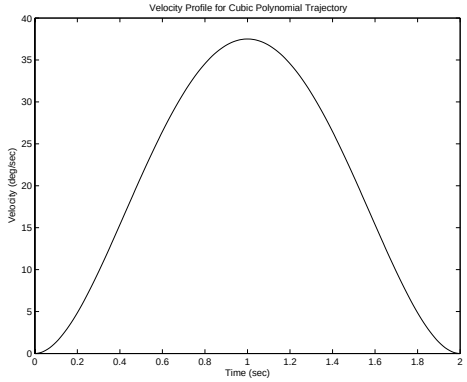


Figure 8.11: Velocity Profile for Quintic Polynomial Trajectory

◇

Example 8.5 *The next figures show the same six second trajectory as in Example 8.3 with the added constraints that the accelerations should be zero at the blend times.*

◇

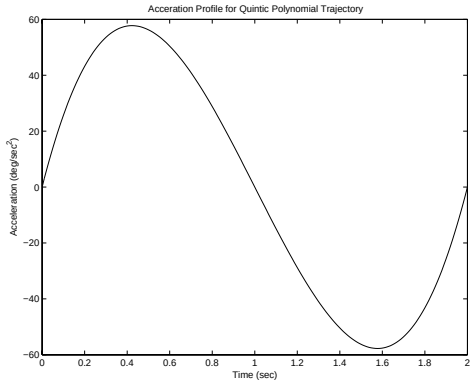


Figure 8.12: Acceleration Profile for Quintic Polynomial Trajectory

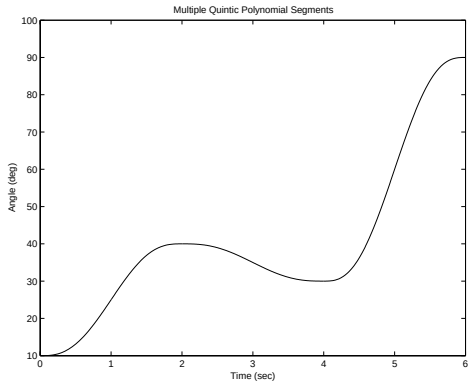


Figure 8.13: Trajectory with Multiple Quintic Segments

8.2.4 Linear Segments with Parabolic Blends (LSPB)

Another way to generate suitable joint space trajectories is by so-called **Linear Segments with Parabolic Blends** or (**LSPB**) for short. This type of trajectory is appropriate when a constant velocity is desired along a portion of the path. The LSPB trajectory is such that the velocity is initially "ramped up" to its desired value and then "ramped down" when it approaches the goal position. To achieve this we specify the desired trajectory in three parts. The first part from time t_0 to time t_b is a quadratic polynomial. This results in a linear "ramp" velocity. At time t_b, called the **blend time**, the trajectory switches to a linear function. This corresponds to a constant velocity. Finally, at time $t_f - t_b$ the trajectory switches once again, this time to a quadratic polynomial so that the velocity is linear.

We choose the blend time t_b so that the position curve is symmetric as shown in Figure 8.16. For convenience suppose that $t_0 = 0$ and $\dot{q}(t_f) = 0 = \dot{q}(0)$. Then between times 0

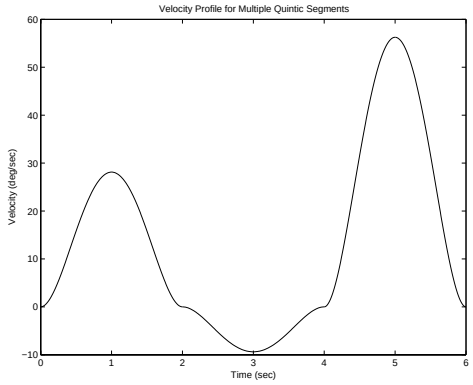


Figure 8.14: Velocity Profile for Multiple Quintic Segments

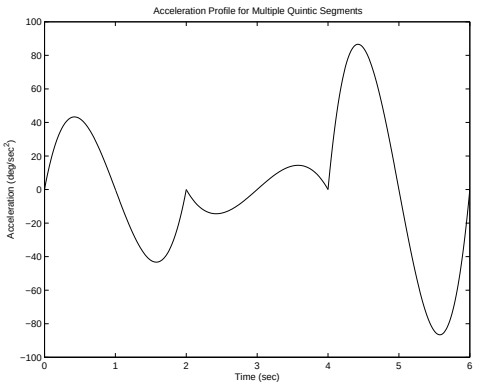


Figure 8.15: Acceleration Profile for Multiple Quintic Segments

and t_b we have

$$q(t) = a_0 + a_1 t + a_2 t^2 \tag{8.27}$$

so that the velocity is

$$\dot{q}(t) = a_1 + 2a_2 t. \tag{8.28}$$

The constraints $q_0 = 0$ and $\dot{q}(0) = 0$ imply that

$$a_0 = q_0 \tag{8.29}$$
$$a_1 = 0. \tag{8.30}$$

At time t_b we want the velocity to equal a given constant, say V. Thus, we have

$$\dot{q}(t_b) = 2a_2 t_b = V \tag{8.31}$$

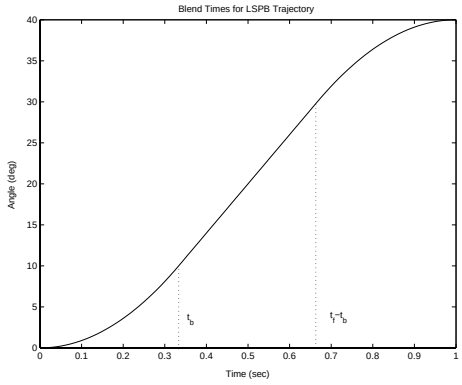


Figure 8.16: Blend times for LSPB trajectory.

which implies that

$$a_2 = \frac{V}{2t_b}. \tag{8.32}$$

Therefore the required trajectory between 0 and t_b is given as

$$\begin{aligned} q(t) &= q_0 + \frac{V}{2t_b}t^2 \qquad (8.33)\\ &= q_0 + \frac{\alpha}{2}t^2 \\ \dot{q}(t) &= \frac{V}{t_b}t = \alpha t \qquad (8.34)\\ \ddot{q} &= \frac{V}{t_b} = \alpha \qquad (8.35) \end{aligned}$$

where α denotes the acceleration.

Now, between time t_f and $t_f - t_b$, the trajectory is a linear segment (corresponding to a constant velocity V)

$$q(t) = a_0 + a_1 t = a_0 + Vt. \tag{8.36}$$

Since, by symmetry,

$$q\left(\frac{t_f}{2}\right) = \frac{q_0 + q_f}{2} \tag{8.37}$$

we have

$$\frac{q_0 + q_f}{2} = a_0 + V\frac{t_f}{2} \tag{8.38}$$

which implies that

$$a_0 = \frac{q_0 + q_f - Vt_f}{2}. \tag{8.39}$$

Since the two segments must "blend" at time t_b we require

$$q_0 + \frac{V}{2}t_b = \frac{q_0 + q_f - Vt_f}{2} + Vt_b \tag{8.40}$$

which gives upon solving for the blend time t_b

$$t_b = \frac{q_0 - q_f + Vt_f}{V}. \tag{8.41}$$

Note that we have the constraint $0 < t_b \leq \frac{t_f}{2}$. This leads to the inequality

$$\frac{q_f - q_0}{V} < t_f \leq \frac{2(q_f - q_0)}{V}. \tag{8.42}$$

To put it another way we have the inequality

$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}. \tag{8.43}$$

Thus the specified velocity must be between these limits or the motion is not possible.

The portion of the trajectory between $t_f - t_b$ and t_f is now found by symmetry considerations (Problem XX). The complete LSPB trajectory is given by

$$q(t) = \begin{cases} q_0 + \frac{a}{2}t^2 & 0 \leq t \leq t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \leq t_f - t_b \\ q_f - \frac{at_f^2}{2} + at_f t - \frac{a}{2}t^2 & t_f - t_b < t \leq t_f \end{cases}. \tag{8.44}$$

Figure 8.17 shows such an LSPB trajectory, where the maximum velocity $V = 60$. In this case $t_b = \frac{1}{3}$. The velocity and acceleration curves are given in Figures 8.18 and 8.19, respectively.

8.2.5 Minimum Time Trajectories

An important variation of this trajectory is obtained by leaving the final time t_f unspecified and seeking the "fastest" trajectory between q_0 and q_f with a given constant acceleration α, that is, the trajectory with the final time t_f a minimum. This is sometimes called a

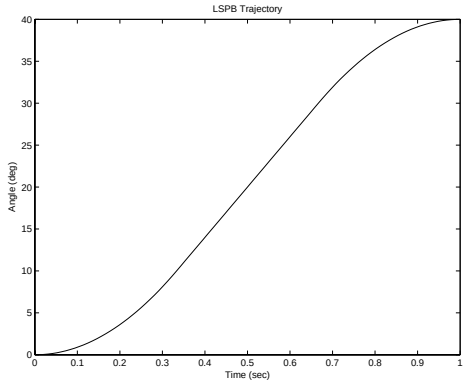


Figure 8.17: LSPB trajectory.

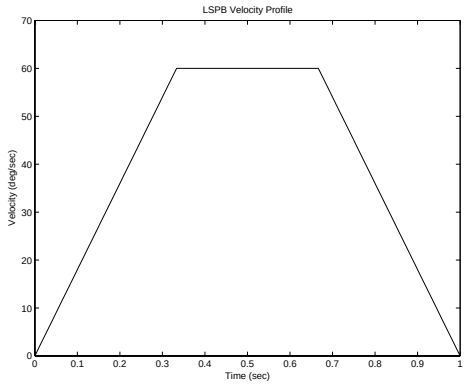


Figure 8.18: Velocity profile for LSPB trajectory.

Bang-Bang trajectory since the optimal solution is achieved with the acceleration at its maximum value $+\alpha$ until an appropriate **switching time** t_s at which time it abruptly switches to its minimum value $-\alpha$ (maximum deceleration) from t_s to t_f.

Returning to our simple example in which we assume that the trajectory begins and ends at rest, that is, with zero initial and final velocities, symmetry considerations would suggest that the switching time t_s is just $\frac{t_f}{2}$. This is indeed the case. For nonzero initial and/or final velocities, the situation is more complicated and we will not discuss it here.

If we let V_s denote the velocity at time t_s then we have

$$V_s = \alpha t_s \tag{8.45}$$

and also

$$t_s = \frac{q_0 - q_f + V_s t_f}{V_s}. \tag{8.46}$$

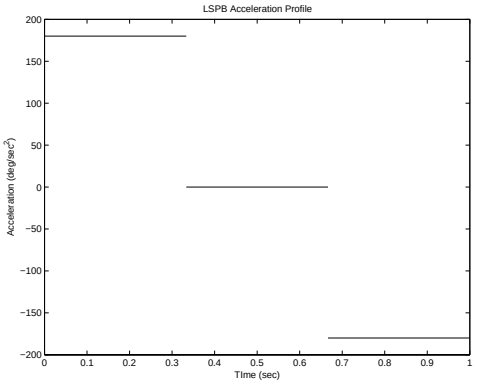


Figure 8.19: Acceleration for LSPB trajectory.

The symmetry condition $t_s = \frac{t_f}{2}$ implies that

$$V_s = \frac{q_f - q_0}{t_s}. \tag{8.47}$$

Combining these two we have the conditions

$$\frac{q_f - q_0}{t_s} = \alpha t_s \tag{8.48}$$

which implies that

$$t_s = \sqrt{q_f - q_0/2}. \tag{8.49}$$

8.3 Trajectories for Paths Specified by Via Points

NOTE: via points are also called knot points or interpolation points

path planner in chapter XXX gives a set of via points (e.g., PRM method). how do we convert this set of via points into a path???

Consider the simple of example of a path specified by three points, q_0, q_1, q_2, such that the via points are reached at times t_0, t_1 and t_2. If in addition to these three constraints we impose constraints on the initial and final velocities and accelerations, we obtain the following set of constraints,

$$\begin{aligned} q(t_0) &= q_0 \\ \dot{q}(t_0) &= v_0 \\ \ddot{q}(t_0) &= \alpha_0 \\ q(t_1) &= q_1 \\ q(t_2) &= q_2 \\ \dot{q}(t_2) &= v_2 \\ \ddot{q}(t_2) &= \alpha_2. \end{aligned}$$

which could be satisfied by generating a trajectory using the sixth order polynomial

$$q(t) = a_6 t^6 + a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t^1 + a_0. \tag{8.50}$$

One advantage to this approach is that, since $q(t)$ is continuously differentiable, we need not worry about discontinuities in either velocity or acceleration at the via point, q_1. However, to determine the coefficients for this polynomial, we must solve a linear system of dimension seven. The clear disadvantage to this approach is that as the number of via points increases, the dimension of the corresponding linear system also increases, making the method intractable when many via points are used..

An alternative to using a single high order polynomial for the entire trajectory is to use low order polynomials for trajectory segments between adjacent via points. We will denote by $Q_i(t)$ the polynomial that represents the trajectory from q_{i-1} to q_i. These polynomials sometimes refered to as interpolating polynomials or blending polynomials. With this approach, we must take care that continuity constraints (e.g., in velocity and acceleration) are satisfied at the via points, where we switch from one polynomial to another.

In this section, we will consider three types of interpolating polynomial trajectories: trajectories for which each Q_i is a cubic polynomial; trajectories for which each Q_1 and Q_3 are fourth order polynomials and Q_2 is a cubic polynomial; **NOTE: should generalize this to an n segment trajectory with fourth order polynomials for first and last segments and cubics in between.** and .... what is that last one....;

8.3.1 4-3-4 trajectories

$$q(t) = \begin{cases} Q_1(t) = & a_{14}t^4 + a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} & : & t_0 \le t < t_1 \\ Q_2(t) = & a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20} & : & t_1 \le t < t_2 \\ Q_3(t) = & a_{34}t^4 + a_{33}t^3 + a_{32}t^2 + a_{31}t + a_{30} & : & t_2 \le t < t_3 \end{cases} \tag{8.51}$$

Chapter 9

DYNAMICS

This chapter deals with the dynamics of robot manipulators. Whereas the kinematic equations describe the motion of the robot without consideration of the forces and moments producing the motion, the dynamic equations explicitly describe the relationship between force and motion. The equations of motion are important to consider in the design of robots, as well as in simulation and animation, and in the design of control algorithms. We introduce the so-called **Euler-Lagrange equations**, which describe the evolution of a mechanical system subject to **holonomic constraints** (this term is defined later on). To motivate the Euler-Lagrange approach we begin with a simple derivation of these equations from Newton's Second Law for a one-degree-of-freedom system. We then derive the Euler-Lagrange equations from the **Principle of Virtual Work** in the general case.

In order to determine the Euler-Lagrange equations in a specific situation, one has to form the **Lagrangian** of the system, which is the difference between the **kinetic energy** and the **potential energy**; we show how to do this in several commonly encountered situations. We then derive the dynamic equations of several example robotic manipulators, including a two-link cartesian robot, a two-link planar robot, and a two-link robot with remotely driven joints.

The Euler-Lagrange equations have several very important properties that can be exploited to design and analyze feedback control algorithms. Among these are explicit bounds on the inertia matrix, linearity in the inertia parameters, and the so-called skew symmetry and passivity properties. We discuss these properties in Section 9.5.

This chapter is concluded with a derivation of an alternate the formulation of the dynamical equations of a robot, known as the **Newton-Euler formulation** which is a recursive formulation of the dynamic equations that is often used for numerical calculation.

9.1 The Euler-Lagrange Equations

In this section we derive a general set of differential equations that describe the time evolution of mechanical systems subjected to holonomic constraints, when the constraint forces satisfy the principle of virtual work. These are called the **Euler-Lagrange equations** of

motion. Note that there are at least two distinct ways of deriving these equations. The method presented here is based on the method of virtual displacements; but it is also possible to derive the same equations based on Hamilton's principle of least action [**?**].

9.1.1 One Dimensional System

To motivate the subsequent derivation, we show first how the Euler-Lagrange equations can be derived from Newton's Second Law for a single degree of freedom system consisting of a particle of constant mass m, constrained to move in the y-direction, and subject to a force f and the gravitational force mg, as shown in Figure 9.1. By Newton's Second law,

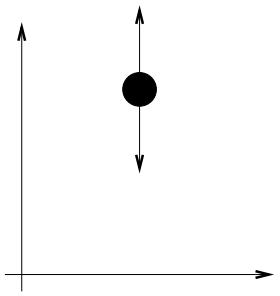


Figure 9.1: One Degree of Freedom System

$F = ma$, the equation of motion of the particle is

$$m\ddot{y} \quad = \quad f - mg \tag{9.1}$$

Notice that the left hand side of Equation (9.1) can be written as

$$m\ddot{y} = \frac{d}{dt}(m\dot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}\left(\frac{1}{2}m\dot{y}^2\right) = \frac{d}{dt}\frac{\partial \mathcal{K}}{\partial \dot{y}} \tag{9.2}$$

where $\mathcal{K} = \frac{1}{2}m\dot{y}^2$ is the **kinetic energy**. We use the partial derivative notation in the above expression to be consistent with systems considered later when the kinetic energy will be a function of several variables. Likewise we can express the gravitational force in Equation (9.1) as

$$mg \quad = \quad \frac{\partial}{\partial y}(mgy) \quad = \quad \frac{\partial \mathcal{P}}{\partial y} \tag{9.3}$$

where $\mathcal{P} = mgy$ is the **potential energy due to gravity**. If we define

$$\mathcal{L} \quad = \quad \mathcal{K} - \mathcal{P} \quad = \quad \frac{1}{2}m\dot{y}^2 - mgy \tag{9.4}$$

and note that

$$\frac{\partial \mathcal{L}}{\partial \dot{y}} \quad = \quad \frac{\partial \mathcal{K}}{\partial \dot{y}} \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial y} \quad = \quad -\frac{\partial \mathcal{P}}{\partial y}$$

then we can write Equation (9.1) as

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{y}} - \frac{\partial \mathcal{L}}{\partial y} = f. \tag{9.5}$$

The function $\mathcal{L}$, which is the difference of the kinetic and potential energy, is called the **Lagrangian** of the system, and Equation (9.5) is called the **Euler-Lagrange Equation**. The Euler-Lagrange equations provide a formulation of the dynamic equations of motion equivalent to those derived using Newton's Second Law. However, as we shall see, the Lagrangian approach is advantageous for more complex systems such as multi-link robots.

Example: 9.1 *Single-Link Manipulator*

Consider the single-link robot arm shown in Figure 9.2, consisting of a rigid link coupled

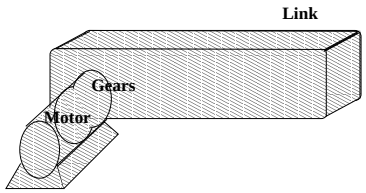


Figure 9.2: Single-Link Robot.

through a gear train to a DC-motor. Let θ_ℓ and θ_m denote the angles of the link and motor shaft, respectively. Then $\theta_m = r\theta_\ell$ where $r:1$ is the gear ratio. The algebraic relation between the link and motor shaft angles means that the system has only one degree-of-freedom and we can therefore write the equations of motion using either θ_m or θ_ℓ. In terms of θ_ℓ, the kinetic energy of the system is given by

$$\begin{aligned} K &= \frac{1}{2}J_m\dot{\theta}_m^2 + \frac{1}{2}J_\ell\dot{\theta}_\ell^2 \\ &= \frac{1}{2}(r^2J_m + J_\ell)\dot{\theta}_\ell^2 \end{aligned} \tag{9.6}$$

where J_m, J_ℓ are the rotational inertias of the motor and link, respectively. The potential energy is given as

$$P = Mg\ell(1-\cos\theta_\ell) \tag{9.7}$$

where M is the total mass of the link and ℓ is the distance from the joint axis to the link center of mass. Defining $J = r^2J_m + J_\ell$, the Lagrangian $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{2}J\dot{\theta}_\ell^2 - Mg\ell(1-\cos\theta_\ell). \tag{9.8}$$

Substituting this expression into the Euler-Lagrange equations yields the equation of motion

$$J\ddot{\theta}_\ell + Mg\ell\sin\theta_\ell = \tau_\ell. \tag{9.9}$$

The generalized force τ_ℓ consists of the motor torque $u = r\tau_m$, reflected to the link and (nonconservative) damping torques $B_m\dot{\theta}_m$, and $B_\ell, \dot{\theta}_\ell$. Reflecting motor damping to the link yields

$$\tau \quad = \quad u - B\dot{\theta}_\ell.$$

where $B = rB_m + B_\ell$. Therefore the complete expression for the dynamics of this system is

$$J\ddot{\theta}_\ell + B\dot{\theta}_\ell + Mg\ell \sin\theta_\ell \quad = \quad u. \tag{9.10}$$

In general, for any system of the type considered, an application of the Euler-Lagrange equations leads to a system of n coupled, second order nonlinear ordinary differential equations of the form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} \quad = \quad \tau_i \quad i = 1, \ldots, n \tag{9.11}$$

The order, n, of the system is determined by the number of so-called **generalized coordinates** are required to describe the evolution of the system. We shall see that the n Denavit-Hartenberg joint variables serves as a set of generalized coordinates for an n-link rigid robot.

9.1.2 The General Case

Now, consider a system consisting of k particles, with corresponding position vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k$. If these particles are free to move about without any restrictions, then it is quite an easy

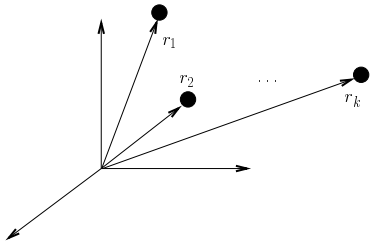


Figure 9.3: System of k particles

matter to describe their motion, by noting merely that the rate of change of the momentum of each mass equals the external force applied to it. However, if the motion of the particles is constrained in some fashion, then one must take into account not only the externally applied forces, but also the so-called constraint forces, that is, the forces needed to make the constraints hold. As a simple illustration of this, suppose the system consists of two particles, which are joined by a massless rigid wire of length ℓ. Then the two coordinates $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ must satisfy the constraint

$$\|\boldsymbol{r}_1 - \boldsymbol{r}_2\| = \ell, \quad \text{or} \quad (\boldsymbol{r}_1 - \boldsymbol{r}_2)^T(\boldsymbol{r}_1 - \boldsymbol{r}_2) = \ell^2. \tag{9.12}$$

If one applies some external forces to each particle, then the particles experience not only these external forces but also the force exerted by the wire, which is along the direction $\boldsymbol{r}_2 - \boldsymbol{r}_1$ and of appropriate magnitude. Therefore, in order to analyze the motion of the two particles, we can follow one of two options. First, we can compute, under each set of external forces, what the corresponding constraint force must be in order that the equation above continues to hold. Second, we can search for a method of analysis that does not require us to know the constraint force. Clearly, the second alternative is preferable, since it is in general quite an involved task to compute the constraint forces. The contents of this section are aimed at achieving this second objective.

First it is necessary to introduce some terminology. A constraint on the k coordinates $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k$ is called **holonomic** if it is an equality constraint of the form

$$g_i(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k) = 0, \qquad i = 1, \ldots, \ell \tag{9.13}$$

and **nonholonomic** otherwise. The constraint (9.12) imposed by connecting two particles by a massless rigid wire is a holonomic constraint. As as example of a nonholonomic constraint, consider a particle moving inside a sphere of radius p centered at the origin of the coordinate system. In this case the coordinate vector $\boldsymbol{r}$ of the particle must satisfy the constraint

$$\|\boldsymbol{r}\| \quad \leq \quad \rho. \tag{9.14}$$

Note that the motion of the particle is unconstrained so long as the particle remains away from the wall of the sphere; but when the particle comes into contact with the wall, it experiences a constraining force.

If a system is subjected to ℓ holonomic constraints, then one can think in terms of the constrained system having ℓ fewer degrees-of-freedom than the unconstrained system. In this case it may be possible to express the coordinates of the k particles in terms of n **generalized coordinates** $q_1, \ldots, q_n$. In other words, we assume that the coordinates of the various particles, subjected to the set of constraints (9.13), can be expressed in the form

$$\boldsymbol{r}_i = \boldsymbol{r}_i(q_1, \ldots, q_n), \qquad i = 1, \ldots, k \tag{9.15}$$

where $q_1, \ldots, q_n$ are all independent. In fact, the idea of generalized coordinates can be used even when there are infinitely many particles. For example, a physical rigid object such as a bar contains an infinity of particles; but since the distance between each pair of particles is fixed throughout the motion of the bar, only six coordinates are sufficient to specify completely the coordinates of any particle in the bar. In particular, one could use three position coordinates to specify the location of the center of mass of the bar, and three Euler angles to specify the orientation of the body. To keep the discussion simple, however, we assume in what follows that the number of particles is finite. Typically, generalized coordinates are positions, angles, etc. In fact, in Chapter **??** we chose to denote the joint variables by the symbols $q_1, \ldots, q_n$ precisely because these joint variables form a set of generalized coordinates for an n-link robot manipulator.

One can now speak of **virtual displacements**, which are any set, $\delta \boldsymbol{r}_1, \ldots, \delta \boldsymbol{r}_k$, of infinitesimal displacements that are consistent with the constraints. For example, consider once again the constraint (9.12) and suppose $\boldsymbol{r}_1, \boldsymbol{r}_2$ are perturbed to $\boldsymbol{r}_1 + \delta \boldsymbol{r}_1$, $\boldsymbol{r}_2 + \delta \boldsymbol{r}_2$, respectively. Then, in order that the perturbed coordinates continue to satisfy the constraint, we must have

$$(\boldsymbol{r}_1 + \delta \boldsymbol{r}_1 - \boldsymbol{r}_2 - \delta \boldsymbol{r}_2)^T (\boldsymbol{r}_1 + \delta \boldsymbol{r}_1 - \boldsymbol{r}_2 - \delta \boldsymbol{r}_2) \quad = \quad \ell^2. \tag{9.16}$$

Now let us expand the above product and take advantage of the fact that the original coordinates $\boldsymbol{r}_1, \boldsymbol{r}_2$ satisfy the constraint (9.12); let us also neglect quadratic terms in $\delta \boldsymbol{r}_1, \delta \boldsymbol{r}_2$. This shows that

$$(\boldsymbol{r}_1 - \boldsymbol{r}_2)^T (\delta \boldsymbol{r}_1 - \delta \boldsymbol{r}_2) \quad = \quad 0. \tag{9.17}$$

Thus any perturbations in the positions of the two particles must satisfy the above equation in order that the perturbed positions continue to satisfy the constraint (9.12). Any pair of infinitesimal vectors $\delta \boldsymbol{r}_1, \delta \boldsymbol{r}_2$ that satisfy (9.17) would constitute a set of virtual displacements for this problem.

Now the reason for using generalized coordinates is to avoid dealing with complicated relationships such as (9.17) above. If (9.15) holds, then one can see that the set of all virtual displacements is precisely

$$\delta \boldsymbol{r}_i \quad = \quad \sum_{j=1}^{n} \frac{\partial \boldsymbol{r}_i}{\partial q_j} \delta q_j, \quad i = 1, \ldots, k \tag{9.18}$$

where the virtual displacements $\delta q_1, \ldots, \delta q_n$ of the generalized coordinates are unconstrained (that is what makes them generalized coordinates).

Next we begin a discussion of constrained systems in equilibrium. Suppose each particle is in equilibrium. Then the net force on each particle is zero, which in turn implies that the work done by each set of virtual displacements is zero. Hence the sum of the work done by any set of virtual displacements is also zero; that is,

$$\sum_{i=1}^{k} \boldsymbol{F}_i^T \delta \boldsymbol{r}_i \quad = \quad 0 \tag{9.19}$$

where $\boldsymbol{F}_i$ is the total force on particle i. As mentioned earlier, the force $\boldsymbol{F}_i$ is the sum of two quantities, namely (i) the externally applied force $\boldsymbol{f}_i$, and (ii) the constraint force $\boldsymbol{f}_i^{(a)}$. Now suppose that the total work done by the constraint forces corresponding to any set of virtual displacements is zero, that is,

$$\sum_{i=1}^{k} (\boldsymbol{f}_i^{(a)})^T \delta \boldsymbol{r}_i \quad = \quad 0. \tag{9.20}$$

This will be true whenever the constraint force between a pair of particles is directed along the radial vector connecting the two particles (see the discussion in the next paragraph). Substituting (9.20) into (9.19) results in

$$\sum_{i=1}^{k} \boldsymbol{f}_i^T \delta \boldsymbol{r}_i \;=\; 0. \tag{9.21}$$

The beauty of this equation is that it does not involve the unknown constraint forces, but only the known external forces. This equation expresses the **principle of virtual work**, which can be stated in words as follows: The work done by external forces corresponding to any set of virtual displacements is zero. Note that the principle is not universally applicable, but requires that (9.20) hold, that is, that the constraint forces do no work. Thus, if the principle of virtual work applies, then one can analyze the dynamics of a system *without* having to evaluate the constraint forces.

It is easy to verify that the principle of virtual work applies whenever the constraint force between a pair of particles acts along the vector connecting the position coordinates of the two particles. In particular, when the constraints are of the form (9.12), the principle applies. To see this, consider once again a single constraint of the form (9.12). In this case the constraint force, if any, must be exerted by the rigid massless wire, and therefore must be directed along the radial vector connecting the two particles. In other words, the force exerted on particle 1 by the wire must be of the form

$$\boldsymbol{f}_1^{(a)} \;=\; c(\boldsymbol{r}_1 - \boldsymbol{r}_2) \tag{9.22}$$

for some constant c (which could change as the particles move about). By the law of action and reaction, the force exerted on particle 2 by the wire must be just the negative of the above, that is,

$$\boldsymbol{f}_2^{(a)} \;=\; -c(\boldsymbol{r}_1 - \boldsymbol{r}_2). \tag{9.23}$$

Now the work done by the constraint forces corresponding to a set of virtual displacements is

$$(\boldsymbol{f}_1^{(a)})^T \delta \boldsymbol{r}_1 + (\boldsymbol{f}_2^{(a)})^T \delta \boldsymbol{r}_2 \;=\; c(\boldsymbol{r}_1 - \boldsymbol{r}_2)^T (\delta \boldsymbol{r}_1 - \delta \boldsymbol{r}_2). \tag{9.24}$$

But (9.17) shows that for any set of virtual displacements, the above inner product must be zero. Thus the principle of virtual work applies in a system constrained by (9.12). The same reasoning can be applied if the system consists of several particles, which are pairwise connected by rigid massless wires of fixed lengths, in which case the system is subjected to several constraints of the form (9.12). Now, the requirement that the motion of a body be rigid can be equivalently expressed as the requirement that the distance between any pair of points on the body remain constant as the body moves, that is, as an infinity of constraints of the form (9.12). Thus the principle of virtual work applies whenever rigidity is the only constraint on the motion. There are indeed situations when this principle does not apply,

typically in the presence of magnetic fields. However, in all situations encountered in this book, we can safely assume that the principle of virtual work is valid.

In (9.21), the virtual displacements $\delta \boldsymbol{r}_i$ are not independent, so we cannot conclude from this equation that each coefficient $\boldsymbol{F}_i$ *individually* equals zero. In order to apply such reasoning, we must transform to generalized coordinates. Before doing this, we consider systems that are not necessarily in equilibrium. For such systems, **D'Alembert's principle** states that, if one introduces a fictitious additional force $-\dot{\boldsymbol{p}}_i$ on particle i for each i, where $\boldsymbol{p}_i$ is the momentum of particle i, then each particle will be in equilibrium. Thus, if one modifies (9.19) by replacing $\boldsymbol{F}_i$ by $\boldsymbol{F}_i - \dot{\boldsymbol{p}}_i$, then the resulting equation is valid for arbitrary systems. One can then remove the constraint forces as before using the principle of virtual work. This results in the equations

$$\sum_{i=1}^{k} \boldsymbol{f}_i^T \delta \boldsymbol{r}_i - \sum_{i=1}^{k} \dot{\boldsymbol{p}}_i^T \delta \boldsymbol{r}_i \;=\; 0. \tag{9.25}$$

The above equation does not mean that each coefficient of $\delta \boldsymbol{r}_i$ is zero. For this purpose, express each $\delta \boldsymbol{r}_i$ in terms of the corresponding virtual displacements of generalized coordinates, as is done in (9.18). Then the virtual work done by the forces $\boldsymbol{f}_i$ is given by

$$\sum_{i=1}^{k} \boldsymbol{f}_i^T \delta \boldsymbol{r}_i \;=\; \sum_{i=1}^{k} \sum_{j=1}^{n} \boldsymbol{f}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} \delta q_j = \sum_{j=1}^{n} \psi_j \delta q_j \tag{9.26}$$

where

$$\psi_j \;=\; \sum_{i=1}^{k} \boldsymbol{f}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} \tag{9.27}$$

is called the j-th **generalized force**. Note that ψ_j need not have dimensions of force, just as q_j need not have dimensions of length; however, $\psi_j \delta q_j$ must always have dimensions of work.

Now let us study the second summation in (9.25) Since $\boldsymbol{p}_i = m_i \dot{\boldsymbol{r}}_i$, it follows that

$$\sum_{i=1}^{k} \dot{\boldsymbol{p}}_i^T \delta \boldsymbol{r}_i \;=\; \sum_{i=1}^{k} m_i \ddot{\boldsymbol{r}}_i^T \delta \boldsymbol{r}_i = \sum_{i=1}^{k} \sum_{j=1}^{n} m_i \ddot{\boldsymbol{r}}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} \delta q_j. \tag{9.28}$$

Next, using the product rule of differentiation, we see that

$$\sum_{i=1}^{k} m_i \ddot{\boldsymbol{r}}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} \;=\; \sum_{i=1}^{k} \left\{ \frac{d}{dt} \left[m_i \dot{\boldsymbol{r}}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} \right] - m_i \dot{\boldsymbol{r}}_i^T \frac{d}{dt} \left[\frac{\partial \boldsymbol{r}_i}{\partial q_j} \right] \right\}. \tag{9.29}$$

Now differentiate (9.15) using the chain rule; this gives

$$\boldsymbol{v}_i \;=\; \dot{\boldsymbol{r}}_i = \sum_{j=1}^{n} \frac{\partial \boldsymbol{r}_i}{\partial q_j} \dot{q}_j. \tag{9.30}$$

Observe from the above equation that

$$\frac{\partial \boldsymbol{v}_i}{\partial \dot{q}_i} = \frac{\partial \boldsymbol{r}_i}{\partial q_j}. \tag{9.31}$$

Next,

$$\frac{d}{dt}\left[\frac{\partial \boldsymbol{r}_i}{\partial q_j}\right] = \sum_{\ell=1}^{n} \frac{\partial^2 \boldsymbol{r}_i}{\partial q_j \partial q_\ell}\dot{q}_\ell = \frac{\partial \boldsymbol{v}_i}{\partial q_j} \tag{9.32}$$

where the last equality follows from (9.30). Substituting from (9.31) and (9.32) into (9.29) and noting that $\dot{\boldsymbol{r}}_i = \boldsymbol{v}_i$ gives

$$\sum_{i=1}^{k} m_i \ddot{\boldsymbol{r}}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i \boldsymbol{v}_i^T \frac{\partial \boldsymbol{v}_i}{\partial \dot{q}_j}\right] - m_i \boldsymbol{v}_i^T \frac{\partial \boldsymbol{v}_i}{\partial q_j}\right\}. \tag{9.33}$$

If we define the *kinetic energy* K to be the quantity

$$K = \sum_{i=1}^{k} \frac{1}{2} m_i \boldsymbol{v}_i^T \boldsymbol{v}_i \tag{9.34}$$

then the sum above can be compactly expressed as

$$\sum_{i=1}^{k} m_i \ddot{\boldsymbol{r}}_i^T \frac{\partial \boldsymbol{r}_i}{\partial q_j} = \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j}. \tag{9.35}$$

Now, substituting from (9.35) into (9.28) shows that the second summation in (9.25) is

$$\sum_{i=1}^{k} \dot{\boldsymbol{p}}_i^T \delta \boldsymbol{r}_i = \sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j}\right\} \delta q_j. \tag{9.36}$$

Finally, combining (9.36) and (9.26) gives

$$\sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} - \psi_j \right\} \delta q_j = 0. \tag{9.37}$$

Now, since the virtual displacements δq_j are independent, we can conclude that each coefficient in (9.37) is zero, that is, that

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} = \psi_j, \quad j = 1, \ldots, n. \tag{9.38}$$

If the generalized force ψ_j is the sum of an externally applied generalized force and another one due to a potential field, then a further modification is possible. Suppose there exist functions τ_j and a potential energy function $P(q)$ such that

$$\psi_j = -\frac{\partial P}{\partial q_j} + \tau_j. \tag{9.39}$$

Then (9.38) can be written in the form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j \tag{9.40}$$

where $\mathcal{L} = K - P$ is the Lagrangian and we have recovered the **Euler-Lagrange equations of motion** as in Equation (9.11).

9.2 General Expressions for Kinetic and Potential Energy

In the previous section, we showed that the Euler-Lagrange equations can be used to derive the dynamical equations in a straightforward manner, provided one is able to express the kinetic and potential energy of the system in terms of a set of generalized coordinates. In order for this result to be useful in a practical context, it is therefore important that one be able to compute these terms readily for an n-link robotic manipulator. In this section we derive formulas for the kinetic energy and potential energy of a rigid robot using the Denavit-Hartenberg joint variables as generalized coordinates.

To begin we note that the kinetic energy of a rigid object is the sum of two terms: the translational energy obtained by concentrating the entire mass of the object at the center of mass, and the rotational kinetic energy of the body about the center of mass. Referring to Figure 9.4 we attach a coordinate frame at the center of mass (called the *body attached frame*) as shown. The kinetic energy of the rigid body is then given as

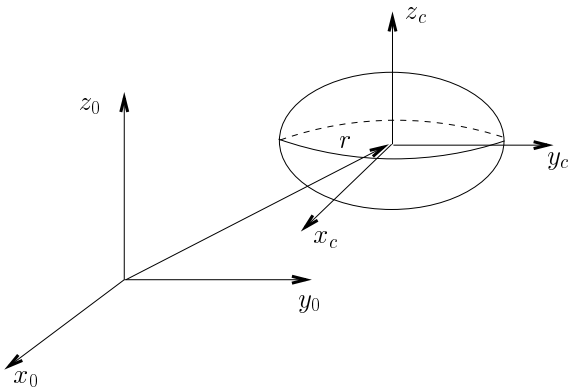


Figure 9.4: A General Rigid Body

$$\mathcal{K} = \frac{1}{2}mv^T v + \frac{1}{2}\boldsymbol{\omega}^T \mathcal{I}\boldsymbol{\omega}. \tag{9.41}$$

where m is the total mass of the object, v and ω are the linear and angular velocity vectors, respectively, and $\mathcal{I}$ is a symmetric 3×3 matrix called the **Inertia Tensor**.

9.2.1 The Inertia Tensor

It is understood that the linear and angular velocity vectors, v and ω, respectively, in the above expression for the kinetic energy are expressed in the inertial frame. In this case we know that ω is found from the skew symmetric matrix

$$S(\omega) = \dot{R}R^T \tag{9.42}$$

where R is the orientation transformation between the body attached frame and the inertial frame. It is therefore necessary to express the inertia tensor, $\mathcal{I}$, also in the inertial frame in order to compute the triple product $\omega^T \mathcal{I} \omega$. The inertia tensor relative to the inertial reference frame will depend on the configuration of the object. If we denote as I the inertia tensor expressed instead in the body attached frame, then the two matrices are related via a similarity transformation according to

$$\mathcal{I} = RIR^T \tag{9.43}$$

This is an important observation because the inertia matrix expressed in the body attached frame is a constant matrix independent of the motion of the object and easily computed. We next show how to compute this matrix explicitly.

Let the mass density of the object be represented as a function of position, $\rho(x, y, z)$. Then the inertia tensor in the body attached frame is computed as

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}. \tag{9.44}$$

where

$$\begin{aligned}
I_{xx} &= \int\int\int (y^2 + z^2)\rho(x, y, z)dx\ dy\ dz \\
I_{yy} &= \int\int\int (x^2 + z^2)\rho(x, y, z)dx\ dy\ dz \\
I_{zz} &= \int\int\int (x^2 + y^2)\rho(x, y, z)dx\ dy\ dz \\
I_{xy} = I_{yx} &= -\int\int\int xy\rho(x, y, z)dx\ dy\ dz \\
I_{xz} = I_{zx} &= -\int\int\int xz\rho(x, y, z)dx\ dy\ dz \\
I_{yz} = I_{zy} &= -\int\int\int yz\rho(x, y, z)dx\ dy\ dz
\end{aligned}$$

The integrals in the above expression are computed over the region of space occupied by the rigid body. The diagonal elements of the inertia tensor, I_{xx}, I_{yy}, I_{zz}, are called the **Principal Moments of Inertia** about the x,y,z axes, respectively. The off diagonal

terms I_{xy}, I_{xz}, etc., are called the **Cross Products of Inertia**. If the mass distribution of the body is symmetric with respect to the body attached frame then the cross products of inertia are identically zero.

Example: 9.2 *Uniform Rectangular Solid*

Consider the rectangular solid of length, a, width, b, and height, c, shown in Figure 9.5 and suppose that the density is constant, $\rho(x, y, z) = \rho$.

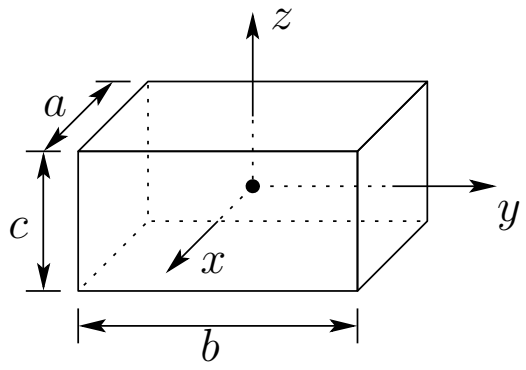


Figure 9.5: Uniform Rectangular Solid

If the body frame is attached at the geometric center of the object, then by symmetry, the cross products of inertia are all zero and it is a simple exercise to compute

$$I_{xx} = \int_{-c/2}^{c/2} \int_{-b/2}^{b/2} \int_{-a/2}^{a/2} (y^2 + z^2)\rho(x, y, z)dx\ dy\ dz = \rho \frac{abc}{12}(b^2 + c^2)$$

Likewise

$$I_{yy} = \rho \frac{abc}{12}(a^2 + c^2) \quad ; \quad I_{zz} = \rho \frac{abc}{12}(a^2 + b^2)$$

and the cross products of inertia are zero.

9.2.2 Kinetic Energy for an n-Link Robot

Now consider a manipulator consisting of n links. We have seen in Chapter 5 that the linear and angular velocities of any point on any link can be expressed in terms of the Jacobian matrix and the derivative of the joint variables. Since in our case the joint variables are indeed the generalized coordinates, it follows that, for appropriate Jacobian matrices J_{v_i} and J_{ω_i}, we have that

$$v_i = J_{v_i}(\boldsymbol{q})\dot{\boldsymbol{q}}, \qquad \boldsymbol{\omega}_i = J_{\omega_i}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{9.45}$$

Now suppose the mass of link i is m_i and that the inertia matrix of link i, evaluated around a coordinate frame parallel to frame i but whose origin is at the center of mass, equals I_i. Then from (9.41) it follows that the overall kinetic energy of the manipulator equals

$$K = \frac{1}{2}\dot{\boldsymbol{q}}^T \sum_{i=1}^{n} \left[m_i J_{v_i}(\boldsymbol{q})^T J_{v_i}(\boldsymbol{q}) + J_{\omega_i}(\boldsymbol{q})^T R_i(\boldsymbol{q}) I_i R_i(\boldsymbol{q})^T J_{\omega_i}(\boldsymbol{q}) \right] \dot{\boldsymbol{q}} \tag{9.46}$$

In other words, the kinetic energy of the manipulator is of the form

$$K = \frac{1}{2}\dot{\boldsymbol{q}}^T D(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{9.47}$$

where $D(\boldsymbol{q})$ is a symmetric positive definite matrix that is in general configuration dependent. The matrix D is called the **inertia matrix**, and in Section 9.4 we will compute this matrix for several commonly occurring manipulator configurations.

9.2.3 Potential Energy for an n-Link Robot

Now consider the potential energy term. In the case of rigid dynamics, the only source of potential energy is gravity. The potential energy of the i-th link can be computed by assuming that the mass of the entire object is concentrated at its center of mass and is given by

$$P_i = g^T r_{ci} m_i. \tag{9.48}$$

where g is vector giving the direction of gravity in the inertial frame and the vector r_{ci} gives the coordinates of the center of mass of link i. The total potential energy of the n-link robot is therefore

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} g^T r_{ci} m_i. \tag{9.49}$$

In the case that the robot contains elasticity, for example, flexible joints, then the potential energy will include terms containing the energy stored in the elastic elements.

Note that the potential energy is a function only of the generalized coordinates and not their derivatives, i.e. the potential energy depends on the configuration of the robot but not on its velocity.

9.3 Equations of Motion

In this section, we specialize the Euler-Lagrange equations derived in Section 9.1 to the special case when two conditions hold: first, the kinetic energy is a quadratic function of the vector $\dot{\boldsymbol{q}}$ of the form

$$K = \frac{1}{2} \sum_{i,j}^{n} d_{ij}(\boldsymbol{q}) \dot{q}_i \dot{q}_j := \frac{1}{2}\dot{\boldsymbol{q}}^T D(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{9.50}$$

where the $n \times n$ "inertia matrix" $D(\boldsymbol{q})$ is symmetric and positive definite for each $q \in R^n$, and second, the potential energy $P = P(\boldsymbol{q})$ is independent of $\dot{q}$. We have already remarked that robotic manipulators satisfy this condition.

The Euler-Lagrange equations for such a system can be derived as follows. Since

$$L = K - P = \frac{1}{2}\sum_{i,j} d_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j - P(\boldsymbol{q}) \tag{9.51}$$

we have that

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}\dot{q}_j \tag{9.52}$$

and

$$\begin{aligned}\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} &= \sum_i d_{kj}\ddot{q}_j + \sum_j \frac{d}{dt}d_{kj}\dot{q}_j \\ &= \sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\dot{q}_j \end{aligned} \tag{9.53}$$

Also

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial P}{\partial q_k}. \tag{9.54}$$

Thus the Euler-Lagrange equations can be written

$$\sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j - \frac{\partial P}{\partial q_k} = \tau_k. \tag{9.55}$$

By interchanging the order of summation and taking advantage of symmetry, we can show that

$$\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i}\right\}\dot{q}_i\dot{q}_j = \frac{1}{2}\sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j}\right\}\dot{q}_i\dot{q}_j. \tag{9.56}$$

Hence

$$sum_{i,j}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j = \sum_{i,j}\frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j. \tag{9.57}$$

The terms

$$c_{ijk} := \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\} \tag{9.58}$$

are known as **Christoffel symbols** (of the first kind). Note that, for a fixed k, we have $c_{ijk} = c_{jik}$, which reduces the effort involved in computing these symbols by a factor of about one half. Finally, if we define

$$\phi_k = \frac{\partial P}{\partial q_k} \tag{9.59}$$

then we can write the Euler-Lagrange equations as

$$\sum_i d_{kj}(\boldsymbol{q})\ddot{q}_j + \sum_{i,j} c_{ijk}(\boldsymbol{q})\dot{q}_i\dot{q}_j + \phi_k(\boldsymbol{q}) = \tau_k, \qquad k = 1, \ldots, n. \tag{9.60}$$

In the above equation, there are three types of terms. The first involve the second derivative of the generalized coordinates. The second are quadratic terms in the first derivatives of $\boldsymbol{q}$, where the coefficients may depend on $\boldsymbol{q}$. These are further classified into two types. Terms involving a product of the type $\dot{q}_i^2$ are called **centrifugal**, while those involving a product of the type $\dot{q}_i q_j$ where $i \neq j$ are called **Coriolis** terms. The third type of terms are those involving only $\boldsymbol{q}$ but not its derivatives. Clearly the latter arise from differentiating the potential energy. It is common to write (9.60) in matrix form as

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{9.61}$$

where the k, j-th element of the matrix $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is defined as

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i \tag{9.62}$$

$$= \sum_{i=1}^{n} \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i. \tag{9.63}$$

Let us examine an important special case, where the *inertia matrix is diagonal and independent of* $\boldsymbol{q}$. In this case it follows from (9.58) that all of the Christoffel symbols are zero, since each d_{ij} is a constant. Moreover, the quantity d_{kj} is nonzero if and only if $k = j$, so that the Equations 9.60) decouple nicely into the form

$$d_{kk}\ddot{\boldsymbol{q}} - \phi_k(\boldsymbol{q}) = \tau_k, \qquad k = 1, \ldots, n. \tag{9.64}$$

In summary, the development in this section is very general and applies to *any* mechanical system whose kinetic energy is of the form (9.50) and whose potential energy is independent of $\dot{\boldsymbol{q}}$. In the next section we apply this discussion to study specific robot configurations.

9.4 Some Common Configurations

In this section we apply the above method of analysis to several manipulator configurations and derive the corresponding equations of motion. The configurations are progressively more complex, beginning with a two-link cartesian manipulator and ending with a five-bar linkage mechanism that has a particularly simple inertia matrix.

Two-Link Cartesian Manipulator

Consider the manipulator shown in Figure 9.6, consisting of two links and two prismatic

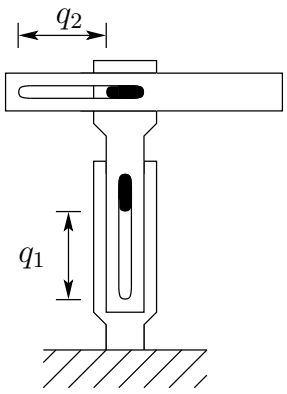


Figure 9.6: Two-link cartesian robot.

joints. Denote the masses of the two links by m_1 and m_2, respectively, and denote the displacement of the two prismatic joints by q_1 and q_2, respectively. Then it is easy to see, as mentioned in Section 9.1, that these two quantities serve as generalized coordinates for the manipulator. Since the generalized coordinates have dimensions of distance, the corresponding generalized forces have units of force. In fact, they are just the forces applied at each joint. Let us denote these by f_i, $i = 1, 2$.

Since we are using the joint variables as the generalized coordinates, we know that the kinetic energy is of the form (9.50) and that the potential energy is only a function of q_1 and q_2. Hence we can use the formulae in Section 9.3 to obtain the dynamical equations. Also, since both joints are prismatic, the angular velocity Jacobian is zero and the kinetic energy of each link consists solely of the translational term.

By (5.87) it follows that the velocity of the center of mass of link 1 is given by

$$\boldsymbol{v}_{c1} = J_{\boldsymbol{v}_{c1}} \dot{\boldsymbol{q}} \tag{9.65}$$

where

$$J_{\boldsymbol{v}_{c1}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \qquad \dot{\boldsymbol{q}} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}. \tag{9.66}$$

Similarly,

$$\boldsymbol{v}_{c2} = J_{\boldsymbol{v}_{c2}} \dot{\boldsymbol{q}} \tag{9.67}$$

where

$$J_{\boldsymbol{v}_{c2}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{9.68}$$

Hence the kinetic energy is given by

$$K = \frac{1}{2}\dot{\boldsymbol{q}}^T \left\{ m_1 J_{\boldsymbol{v}_c}^T J_{\boldsymbol{v}_{c1}} + m_2 J_{\boldsymbol{v}_{c2}}^T J_{\boldsymbol{v}_{c2}} \right\} \dot{\boldsymbol{q}}. \tag{9.69}$$

Comparing with (9.50), we see that the inertia matrix D is given simply by

$$D = \begin{bmatrix} m_1 + m_2 & 0 \\ 0 & m_2 \end{bmatrix}. \tag{9.70}$$

Next, the potential energy of link 1 is $m_1 g q_1$, while that of link 2 is $m_2 g q_1$, where g is the acceleration due to gravity. Hence the overall potential energy is

$$P = g(m_1 + m_2)q_1. \tag{9.71}$$

Now we are ready to write down the equations of motion. Since the inertia matrix is constant, all Christoffel symbols are zero. Further, the vectors ϕ_k are given by

$$\phi_1 = \frac{\partial P}{\partial q_1} = g(m_1 + m_2), \qquad \phi_2 = \frac{\partial P}{\partial q_2} = 0. \tag{9.72}$$

Substituting into (9.60) gives the dynamical equations as

$$\begin{aligned} (m_1 + m_2)\ddot{q}_1 + g(m_1 + m_2) &= f_1 \\ m_2 \ddot{q}_2 &= f_2. \end{aligned} \tag{9.73}$$

Planar Elbow Manipulator

Now consider the planar manipulator with two revolute joints shown in Figure 9.7. Let

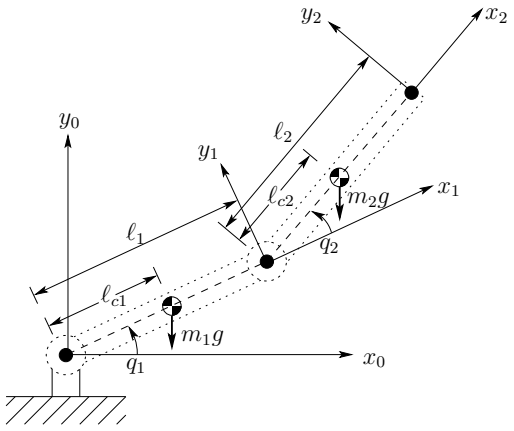


Figure 9.7: Two-link revolute joint arm.

us fix notation as follows: For $i = 1, 2$, q_i denotes the joint angle, which also serves as a

generalized coordinate; m_i denotes the mass of link i, ℓ_i denotes the length of link i; ℓ_{ci} denotes the distance from the previous joint to the center of mass of link i; and I_i denotes the moment of inertia of link i about an axis coming out of the page, passing through the center of mass of link i.

We will make effective use of the Jacobian expressions in Chapter 5 in computing the kinetic energy. Since we are using joint variables as the generalized coordinates, it follows that we can use the contents of Section 9.7. First,

$$\boldsymbol{v}_{c1} = J_{\boldsymbol{v}_{c1}} \dot{\boldsymbol{q}} \tag{9.74}$$

where, from (5.80),

$$J_{\boldsymbol{v}_{c1}} = \begin{bmatrix} -\ell_c \sin q_1 & 0 \\ \ell_{c1} \cos q_1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{9.75}$$

Similarly,

$$\boldsymbol{v}_{c2} = J_{\boldsymbol{v}_{c2}} \dot{\boldsymbol{q}} \tag{9.76}$$

where

$$J_{\boldsymbol{v}_{c2}} = \begin{bmatrix} -\ell_1 \sin q_1 - \ell_{c2} \sin(q_1 + q_2) & -\ell_{c2} \sin(q_1 + q_2) \\ \ell_1 \cos q_1 + \ell_{c2} \cos(q_1 + q_2) & \ell_{c2} \cos(q_1 + q_2) \\ 0 & 0 \end{bmatrix}. \tag{9.77}$$

Hence the translational part of the kinetic energy is

$$\frac{1}{2} m_1 \boldsymbol{v}_{c1}^T \boldsymbol{v}_{c1} + \frac{1}{2} m_2 \boldsymbol{v}_{c2}^T \boldsymbol{v}_{c2} = \frac{1}{2} \dot{\boldsymbol{q}} \left\{ m_1 J_{\boldsymbol{v}_{c1}}^T J_{\boldsymbol{v}_{c1}} + m_2 J_{\boldsymbol{v}_{c2}}^T J_{\boldsymbol{v}_{c2}} \right\} \dot{\boldsymbol{q}}. \tag{9.78}$$

Next we deal with the angular velocity terms. Because of the particularly simple nature of this manipulator, many of the potential difficulties do not arise. First, it is clear that

$$\boldsymbol{\omega}_1 = \dot{q}_1 \boldsymbol{k}, \qquad \boldsymbol{\omega}_2 = (\dot{q}_1 + \dot{q}_2) \boldsymbol{k} \tag{9.79}$$

when expressed in the base inertial frame. Moreover, since $\boldsymbol{\omega}_i$ is aligned with $\boldsymbol{k}$, the triple product $\boldsymbol{\omega}_i^T I_i \boldsymbol{\omega}_i$ reduces simply to $(I_{33})_i$ times the square of the magnitude of the angular velocity. This quantity $(I_{33})_i$ is indeed what we have labeled as I_i above. Hence the rotational kinetic energy of the overall system is

$$\frac{1}{2} \dot{\boldsymbol{q}}^T \left\{ I_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + I_2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right\} \dot{\boldsymbol{q}} \tag{9.80}$$

Now we are ready to form the inertia matrix $D(\boldsymbol{q})$. For this purpose, we merely have to add the two matrices in (9.78) and (9.80), respectively. Thus

$$D(\boldsymbol{q}) = m_1 J_{\boldsymbol{v}_{c1}}^T J_{\boldsymbol{v}_{c1}} + m_2 J_{\boldsymbol{v}_{c2}}^T J_{\boldsymbol{v}_{c2}} + \begin{bmatrix} I_1 + I_2 & I_2 \\ I_2 & I_2 \end{bmatrix}. \tag{9.81}$$

Carrying out the above multiplications and using the standard trigonometric identities $\cos^2\theta + \sin^2\theta = 1$, $\cos\alpha\cos\beta + \sin\alpha\sin\beta = \cos(\alpha - \beta)$ leads to

$$\begin{aligned} d_{11} &= m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2 + 2\ell_1\ell_{c2}^2 + 2\ell_1\ell_{c2}\cos q_2) + I_1 + I_2 \\ d_{12} &= d_{21} = m_2(\ell_{c2}^2 + \ell_1\ell_{c2}\cos q_2) + I_2 \\ d_{22} &= m_2\ell_{c2}^2 + I_2. \end{aligned} \tag{9.82}$$

Now we can compute the Christoffel symbols using the definition (9.58). This gives

$$\begin{aligned} c_{111} &= \frac{1}{2}\frac{\partial d_{11}}{\partial q_1} = 0 \\ c_{121} &= c_{211} = \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = -m_2\ell_1\ell_{c2}\sin q_2 =: h \\ c_{221} &= \frac{\partial d_{12}}{\partial q_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = h \\ c_{112} &= \frac{\partial d_{21}}{\partial q_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial q_2} = -h \\ c_{122} &= c_{212} = \frac{1}{2}\frac{\partial d_{22}}{\partial q_1} = 0 \\ c_{222} &= \frac{1}{2}\frac{\partial d_{22}}{\partial q_2} = 0. \end{aligned} \tag{9.83}$$

Next, the potential energy of the manipulator is just the sum of those of the two links. For each link, the potential energy is just its mass multiplied by the gravitational acceleration and the height of its center of mass. Thus

$$\begin{aligned} P_1 &= m_1 g\ell_{c1}\sin q_1 \\ P_2 &= m_2 g(\ell_1\sin q_1 + \ell_{c2}\sin(q_1 + q_2)) \\ P &= P_1 + P_2 = (m_1\ell_{c1} + m_2\ell_1)g\sin q_1 + m_2\ell_{c2}g\sin(q_1 + q_2). \end{aligned} \tag{9.84}$$

Hence, the functions ϕ_k defined in (9.59) become

$$\phi_1 = \frac{\partial P}{\partial q_1} = (m_1\ell_{c1} + m_2\ell_1)g\cos q_1 + m_2\ell_{c2}g\cos(q_1 + q_2) \tag{9.85}$$

$$\phi_2 = \frac{\partial P}{\partial q_2} = m_2\ell_{c2}\cos(q_1 + q_2). \tag{9.86}$$

Finally we can write down the dynamical equations of the system as in (9.60). Substituting for the various quantities in this equation and omitting zero terms leads to

$$\begin{aligned} d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{121}\dot{q}_1\dot{q}_2 + c_{211}\dot{q}_2\dot{q}_1 + c_{221}\dot{q}_2^2 + \phi_1 &= \tau_1 \\ d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{112}\dot{q}_1^2 + \phi_2 &= \tau_2. \end{aligned} \tag{9.87}$$

In this case the matrix $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is given as

$$C = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix}. \tag{9.88}$$

Planar Elbow Manipulator with Remotely Driven Link

Now we illustrate the use of Lagrangian equations in a situation where the generalized coordinates are not the joint variables defined in earlier chapters. Consider again the planar elbow manipulator, but suppose now that both joints are driven by motors mounted at the base. The first joint is turned directly by one of the motors, while the other is turned via a gearing mechanism or a timing belt (see Figure 9.8). In this case one should choose

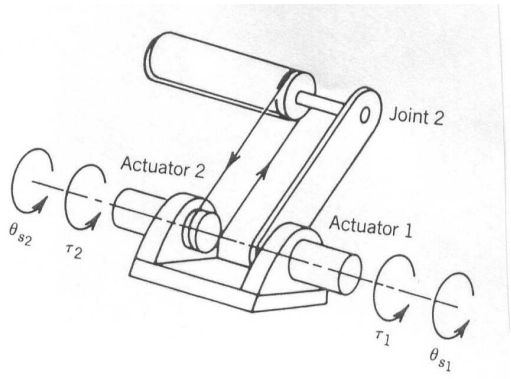


Figure 9.8: Two-link revolute joint arm with remotely driven link.

the generalized coordinates as shown in Figure 9.9, because the angle p_2 is determined by

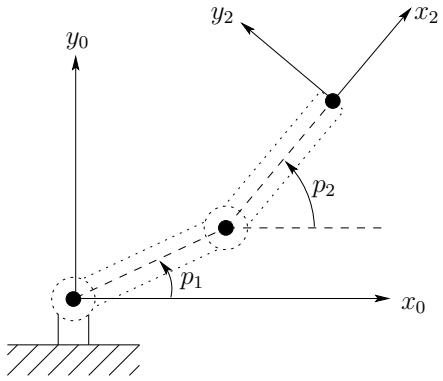


Figure 9.9: Generalized coordinates for robot of Figure 6.4.

driving motor number 2, and is not affected by the angle p_1. We will derive the dynamical equations for this configuration, and show that some simplifications will result.

Since p_1 and p_2 are not the joint angles used earlier, we cannot use the velocity Jacobians derived in Chapter 5 in order to find the kinetic energy of each link. Instead, we have to

carry out the analysis directly. It is easy to see that

$$\boldsymbol{v}_{c1} = \begin{bmatrix} -\ell_{c1}\sin p_1 \\ \ell_{c1}\cos p_1 \\ 0 \end{bmatrix} \dot{p}_1, \quad \boldsymbol{v}_{c2} = \begin{bmatrix} \ell_1 \sin p_1 & -\ell_{c2}\sin p_2 \\ \ell_1 \cos p_1 & \ell_{c2}\cos p_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} \tag{9.89}$$

$$\boldsymbol{\omega}_1 = \dot{p}_1 \boldsymbol{k}, \qquad \boldsymbol{\omega}_2 = \dot{p}_2 \boldsymbol{k}. \tag{9.90}$$

Hence the kinetic energy of the manipulator equals

$$K = \frac{1}{2}\dot{\boldsymbol{p}}^T D(\boldsymbol{p})\dot{\boldsymbol{p}} \tag{9.91}$$

where

$$D(\boldsymbol{p}) = \begin{bmatrix} m_1\ell_{c1}^2 + m_2\ell_1^2 + I_1 & m_2\ell_1\ell_{c2}\cos(p_2 - p_1) \\ m_2\ell_1\ell_{c2}\cos(p_2 - p_1) & m_2\ell_{c2}^2 + I_2. \end{bmatrix} \tag{9.92}$$

Computing the Christoffel symbols as in (9.58) gives

$$\begin{aligned} c_{111} &= \frac{1}{2}\frac{\partial d_{11}}{\partial p_1} = 0 \\ c_{121} &= c_{211} = \frac{1}{2}\frac{\partial d_{11}}{\partial p_2} = 0 \\ c_{221} &= \frac{\partial d_{12}}{\partial p_2} - \frac{1}{2}\frac{\partial d_{22}}{\partial p_1} = -m_2\ell_1\ell_{c2}\sin(p_2 - p_1) \\ c_{112} &= \frac{\partial d_{21}}{\partial p_1} - \frac{1}{2}\frac{\partial d_{11}}{\partial p_2} = m_2\ell_1\ell_{c2}\sin(p_2 - p_1) \\ c_{212} &= = c_{122} = \frac{1}{2}\frac{\partial d_{22}}{\partial p_1} = 0 \\ c_{222} &= \frac{1}{2}\frac{\partial d_{22}}{\partial p_2} = 0. \end{aligned} \tag{9.93}$$

Next, the potential energy of the manipulator, in terms of p_1 and p_2, equals

$$P = m_1 g \ell_{c1}\sin p_1 + m_2 g(\ell_1 \sin p_1 + \ell_{c2}\sin p_2). \tag{9.94}$$

Hence the gravitational generalized forces are

$$\begin{aligned} \phi_1 &= (m_1\ell_{c1} + m_2\ell_1)g\cos p_1 \\ \phi_2 &= m_2\ell_{c2}g\cos p_2. \end{aligned}$$

Finally, the equations of motion are

$$\begin{aligned} d_{11}\ddot{p}_1 + d_{12}\ddot{p}_2 + c_{221}\dot{p}_2^2 + \phi_1 &= \tau_1 \\ d_{21}\ddot{p}_1 + d_{22}\ddot{p}_2 + c_{112}\dot{p}_1^2 + \phi_2 &= \tau_2. \end{aligned} \tag{9.95}$$

Comparing (9.95) and (9.87), we see that by driving the second joint remotely from the base we have eliminated the Coriolis forces, but we still have the centrifugal forces coupling the two joints.

Five-Bar Linkage

Now consider the manipulator shown in Figure 9.10. We will show that, if the parameters

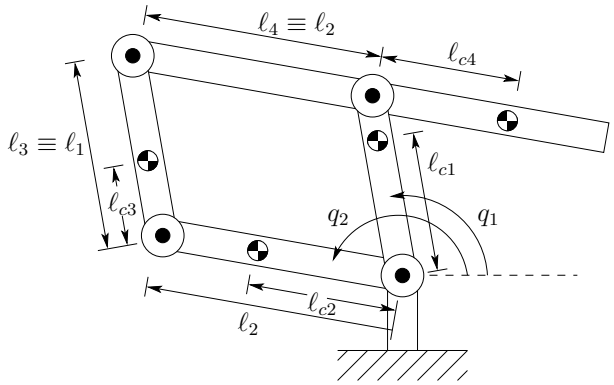


Figure 9.10: Five-bar linkage.

of the manipulator satisfy a simple relationship, then the equations of the manipulator are decoupled, so that each quantity q_1 and q_2 can be controlled independently of the other. The mechanism in Figure 9.10 is called a **five-bar linkage**. Clearly there are only four bars in the figure, but in the theory of mechanisms it is a convention to count the ground as an additional linkage, which explains the terminology. In Figure 9.10, it is assumed that the lengths of links and 3 are the same, and that the two lengths marked ℓ_2 are the same; in this way the closed path in the figure is in fact a parallelogram, which greatly simplifies the computations. Notice, however, that the quantities ℓ_{c1}, and ℓ_{c3} need not be equal. For example, even though links 1 and 3 have the same length, they need not have the same mass distribution.

It is clear from the figure that, even though there are four links being moved, there are in fact only two degrees-of-freedom, identified as q_1 and q_2. Thus, in contrast to the earlier mechanisms studied in this book, this one is a closed kinematic chain (though of a particularly simple kind). As a result, we cannot use the earlier results on Jacobian matrices, and instead have to start from scratch. As a first step we write down the coordinates of the

centers of mass of the various links as a function of the generalized coordinates. This gives

$$\begin{bmatrix} x_{c1} \\ y_{c1} \end{bmatrix} = \begin{bmatrix} \ell_{c1}\cos q_1 \\ \ell_{c1}\sin q_1 \end{bmatrix} \tag{9.96}$$

$$\begin{bmatrix} x_{c2} \\ y_{c2} \end{bmatrix} = \begin{bmatrix} \ell_{c2}\cos q_2 \\ \ell_{c2}\sin q_2 \end{bmatrix} \tag{9.97}$$

$$\begin{bmatrix} x_{c3} \\ y_{c3} \end{bmatrix} = \begin{bmatrix} \ell_{c2}\cos q_1 \\ \ell_{c2}\sin q_2 \end{bmatrix} + \begin{bmatrix} \ell_{c3}\cos q_1 \\ \ell_{c3}\sin q_1 \end{bmatrix} \tag{9.98}$$

$$\begin{aligned} \begin{bmatrix} x_{c4} \\ y_{c4} \end{bmatrix} &= \begin{bmatrix} \ell_1\cos q_1 \\ \ell_1\sin q_1 \end{bmatrix} + \begin{bmatrix} \ell_{c4}\cos(q_2-\pi) \\ \ell_{c4}\sin(q_2-\pi) \end{bmatrix} \\ &= \begin{bmatrix} \ell_1\cos q_1 \\ \ell_1\sin q_1 \end{bmatrix} - \begin{bmatrix} \ell_{c4}\cos q_2 \\ \ell_{c4}\sin q_2 \end{bmatrix}. \end{aligned} \tag{9.99}$$

Next, with the aid of these expressions, we can write down the velocities of the various centers of mass as a function of $\dot{q}_1$ and $\dot{q}_2$. For convenience we drop the third row of each of the following Jacobian matrices as it is always zero. The result is

$$\begin{aligned} \boldsymbol{v}_{c1} &= \begin{bmatrix} -\ell_{c1}\sin q_1 & 0 \\ \ell_{c1}\cos q_1 & 0 \end{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{v}_{c2} &= \begin{bmatrix} 0 & -\ell_{c2}\sin q_2 \\ 0 & \ell_{c2}\cos q_2 \end{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{v}_{c3} &= \begin{bmatrix} -\ell_{c3}\sin q_1 & -\ell_2\sin q_2 \\ \ell_{c3}\cos q_1 & \ell_2\cos q_2 \end{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{v}_{c4} &= \begin{bmatrix} -\ell_1\sin q_1 & \ell_{c4}\sin q_2 \\ \ell_1\cos q_1 & \ell_{c4}\cos q_2 \end{bmatrix} \dot{\boldsymbol{q}}. \end{aligned} \tag{9.100}$$

(9.101)

Let us define the velocity Jacobians $J_{\boldsymbol{v}_{ci}}$, $i = 1, \ldots, 4$ in the obvious fashion, that is, as the four matrices appearing in the above equations. Next, it is clear that the angular velocities of the four links are simply given by

$$\boldsymbol{\omega}_1 = \boldsymbol{\omega}_3 = \boldsymbol{q}_1\boldsymbol{k}, \boldsymbol{\omega}_2 = \boldsymbol{\omega}_4 = \dot{q}_2\boldsymbol{k}. \tag{9.102}$$

Thus the inertia matrix is given by

$$D(\boldsymbol{q}) = \sum_{i=1}^{4} m_i J_{vc}^T J_{vc} + \begin{bmatrix} I_1 + I_3 & 0 \\ 0 & I_2 + I_4 \end{bmatrix}. \tag{9.103}$$

If we now substitute from (9.100) into the above equation and use the standard trigonometric identities, when the dust settles we are left with

$$\begin{aligned} d_{11}(\boldsymbol{q}) &= m_1\ell_{c1}^2 + m_3\ell_{c3}^2 + m_4\ell_1^2 + I_1 + I_3 \\ d_{12}(\boldsymbol{q}) &= d_{21}(\boldsymbol{q}) = (m_3\ell_2\ell_{c3} - m_4\ell_1\ell_{c4})\cos(q_2 - q_1) \\ d_{22}(\boldsymbol{q}) &= m_2\ell_{c2}^2 + m_3\ell_2^2 + m_4\ell_{c4}^2 + I_2 + I_4. \end{aligned} \tag{9.104}$$

Now the thing to note is that if

$$m_3\ell_2\ell_{c3} = m_4\ell_1\ell_{c4} \tag{9.105}$$

then the inertia matrix is diagonal and constant, and as a consequence the dynamical equations will contain neither Coriolis nor centrifugal terms.

Turning now to the potential energy, we have that

$$\begin{aligned} P &= g\sum_{i=1}^{4} \boldsymbol{y}_{ci} \\ &= g\sin q_1(m_1\ell_{c1} + m_3\ell_{c3} + m_4\ell_1) \\ &= g\sin q_2(m_2\ell_{c2} + m_3\ell_2 - m_4\ell_{c4}). \end{aligned} \tag{9.106}$$

Hence

$$\begin{aligned} \phi_1 &= g\cos q_1(m_1\ell_{c1} + m_3\ell_{c3} + m_4\ell_1) \\ \phi_2 &= g\cos q_2(m_2\ell_{c2} + m_3\ell_2 - m_4\ell_{c4}). \end{aligned} \tag{9.107}$$

Notice that ϕ_1 depends only on q_1 but not on q_2 and similarly that ϕ_2 depends only on q_2 but not on q_1. Hence, if the relationship (9.105) is satisfied, then the rather complex-looking manipulator in Figure 9.10 is described by the **decoupled** set of equations

$$d_{11}\ddot{q}_1 + \phi_1(q_1) = \tau_1, \qquad d_{22}\ddot{q}_2 + \phi_2(q_2) = \tau_2. \tag{9.108}$$

This discussion helps to explain the increasing popularity of the parallelogram configuration in industrial robots (e.g., P-50). If the relationship (9.105) is satisfied, then one can adjust the two angles q_1 and q_2 independently, without worrying about interactions between the two angles. Compare this with the situation in the case of the planar elbow manipulators discussed earlier in this section.

9.5 Properties of Robot Dynamic Equations

The equations of motion for an n-link robot can be quite formidable especially if the robot contains one or more revolute joints. Fortunately, these equations contain some important structural properties which can be exploited to good advantage in particular for developing control algorithms. We will see this in subsequent chapters. Here we will discuss some of these properties, the most important of which are the so-called **skew symmetry** property and the related **passivity property**, and the **linearity in the parameters** property. For revolute joint robots, the inertia matrix also satisfies global bounds that are useful for control design.

9.5.1 The Skew Symmetry and Passivity Properties

The **Skew Symmetry** property refers to an important relationship between the inertia matrix $D(\boldsymbol{q})$ and the matrix $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ appearing in (9.61) that will be of fundamental importance for the problem of manipulator control considered in later chapters.

Proposition: 9.1 *Define the matrix $N(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{D}(\boldsymbol{q}) - 2C(\boldsymbol{q}, \dot{\boldsymbol{q}})$. Then $N(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew symmetric, that is, the components n_{jk} of N satisfy $n_{jk} = -n_{kj}$.*

Proof: Given the inertia matrix $D(\boldsymbol{q})$, the kj-th component of $\dot{D}(\boldsymbol{q})$ is given by the chain rule as

$$\dot{d}_{kj} = \sum_{i=1}^{n} \frac{\partial d_{kj}}{\partial q_i} \dot{q}_i. \tag{9.109}$$

Therefore, the kj-th component of $N = \dot{D} - 2C$ is given by

$$\begin{aligned} n_{kj} &= \dot{d}_{kj} - 2c_{kj} \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{kj}}{\partial q_i} - \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \right] \dot{q}_i \\ &= \sum_{i=1}^{n} \left[\frac{\partial d_{ij}}{\partial q_k} - \frac{\partial d_{ki}}{\partial q_j} \right] \dot{q}_i. \end{aligned} \tag{9.110}$$

Since the inertia matrix $D(\boldsymbol{q})$ is symmetric, that is, $d_{ij} = d_{ji}$, it follows from (9.110) by interchanging the indices k and j that

$$n_{jk} = -n_{kj} \tag{9.111}$$

which completes the proof.

Related to the skew symmetry property is the so-called **Passivity Property** which, in the present context, means that there exists a constant, $\beta \geq 0$, such that

$$\int_0^T \dot{q}^T(\zeta)\tau(\zeta)d\zeta \geq -\beta, \quad \forall\ T > 0. \tag{9.112}$$

The term $\dot{q}^T\tau$ has units of power hence, the expression $\int_0^T \dot{q}^T(\zeta)\tau(\zeta)d\zeta$ is the energy produced by the system over the time interval $[0, T]$. Passivity therefore means that the amount of energy produced by the system has a lower bound given by $-\beta$. The word passivity comes from circuit theory where a passive system according to the above definition is one that can be built from passive components (resistors, capacitors, inductors). Likewise a passive mechanical system can be built from masses, springs, and dampers.

To prove the passivity property, let H be the total energy of the system, i.e., the sum of the kinetic and potential energies,

$$H = \frac{1}{2}\dot{q}^T D(q)\dot{q} + P(q) \tag{9.113}$$

Then, the derivative $\dot{H}$ satisfies

$$\dot{H} = \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T \frac{\partial P}{\partial q} \tag{9.114}$$

$$= \dot{q}^T\{\tau - C(q,\dot{q}) - g(q)\} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T \frac{\partial P}{\partial q} \tag{9.115}$$

where we have substituted for $D(q)\ddot{q}$ using the equations of motion. Collecting terms and using the fact that $g(q) = \frac{\partial P}{\partial q}$ yields

$$\dot{H} = \dot{q}^T\tau + \frac{1}{2}\dot{q}^T\{\dot{D}(q) - 2C(q,\dot{q})\}\dot{q} \tag{9.116}$$

$$= \dot{q}^T\tau \tag{9.117}$$

the latter equality following from the skew-symmetry property. Integrating both sides of (9.117) with respect to time gives,

$$\int_0^T \dot{q}^T(\zeta)\tau(\zeta)d\zeta = H(T) - H(0) \geq -H(0) \tag{9.118}$$

since the total energy $H(T)$ is non–negative, and the passivity property therefore follows with $\beta = H(0)$.

9.5.2 Bounds on the Inertia Matrix

We have remarked previously that the inertia matrix for an n-link rigid robot is symmetric and positive definite. For a fixed value of the generalized coordinate q, let $0 < \lambda_1(q) \leq \ldots, \leq \lambda_n(q)$ denote the n eigenvalues of $D(q)$. These eigenvalues are positive as a consequence of the positive definiteness of $D(q)$. As a result, it can easily be shown that

$$\lambda_1(q)I_{n\times n} \leq D(q) \leq \lambda_n(q)I_{n\times n} \tag{9.119}$$

where $I_{n\times n}$ denotes the $n \times n$ identity matrix. The above inequalities are interpreted in the standard sense of matrix inequalities, namely, if A and B are $n \times n$ matrices, then $B < A$ means that the matrix $A - B$ is positive definite and $B \leq A$ means that $A - B$ is positive semi-definite.

If all of the joints are revolute then the inertia matrix contains only bounded functions of the joint variables, i.e., terms containing sine and cosine functions. As a result one can find constants λ_m and λ_M that provide uniform (independent of q) bounds in the inertia matrix

$$\lambda_m I_{n\times n} \leq D(q) \leq \lambda_M I_{n\times n} < \infty \tag{9.120}$$

9.5.3 Linearity in the Parameters

The robot equations of motion are defined in terms of certain parameters, such as link masses, moments of inertia, etc., that must be determined for each particular robot in order, for example, to simulate the equations or to tune controllers. The complexity of the dynamic equations makes the determination of these parameters a difficult task. Fortunately, the equations of motion are linear in these inertia parameters in the following sense. There exists an $n \times \ell$ function, $Y(q, \dot{q}, \ddot{q})$, which we assume is completely known, and an ℓ-dimensional vector Θ such that the Euler-Lagrange equations can be written

$$D(q) + C(q, \dot{q})\dot{q} + g(q) =: Y(q, \dot{q}, \ddot{q})\Theta = \tau \tag{9.121}$$

The function, $Y(q, \dot{q}, \ddot{q})$, is called the **Regressor** and Θ is the **Parameter** vector. The dimension of the parameter space, $\mathbf{R}^\ell$, i.e., the number of parameters needed to write the dynamics in this way, is not unique. In general, a given rigid body is described by ten parameters, namely, the total mass, the six independent entries of the inertia tensor, and the three coordinates of the center of mass. An n-link robot then has a maximum of $10n$ independent dynamics parameters. However, since the link motions are constrained and coupled by the joint interconnections, there are actually fewer than $10n$ independent parameters. Finding a minimal set of parameters that can parametrize the dynamic equations is, however, difficult in general.

Example: 9.3 *Two Link Planar Robot*

Consider the two link, revolute joint, planar robot from section 9.4 above. If we group the inertia terms appearing in Equation 9.82 as

$$\Theta_1 = m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2) + I_1 + I_2 \tag{9.122}$$
$$\Theta_2 = m_2\ell_1\ell_{c2} \tag{9.123}$$
$$\Theta_3 = m_2\ell_1\ell_{c2} \tag{9.124}$$

then we can write the inertia matrix elements as

$$d_{11} = \Theta_1 + 2\Theta_2\cos(q_2) \tag{9.125}$$
$$d_{12} = d_{21} = \Theta_3 + \Theta_2\cos(q_2) \tag{9.126}$$
$$d_{22} = \Theta_3 \tag{9.127}$$

No additional parameters are required in the Christoffel symbols as these are functions of the elements of the inertia matrix. The gravitational torques require additional parameters, in general. Setting

$$\Theta_4 = m_1\ell_{c1} + m_2\ell_1 \tag{9.128}$$
$$\Theta_5 = m_2\ell_2 \tag{9.129}$$

we can write the gravitational terms, ϕ_1 and ϕ_2 as

$$\phi_1 = \Theta_4 g\cos(q_1) + \Theta_5 g\cos(q_1+q_2) \tag{9.130}$$
$$\phi_2 = \Theta_5 \cos(q_1+q_2) \tag{9.131}$$

Substituting these into the equations of motion it is straightforward to write the dynamics in the form (9.121) where

$$Y(q,\dot{q},\ddot{q}) = \tag{9.132}$$
$$\begin{bmatrix} \ddot{q}_1 & \cos(q_2)(2\ddot{q}_1+\ddot{q}_2)+\sin(q_2)(\dot{q}_1^2-2\dot{q}_1\dot{q}_2) & \ddot{q}_2 & g\cos(q_1) & g\cos(q_1+q_2) \\ 0 & \cos(q_2)\ddot{q}_1+\sin(q_2)\dot{q}_1^2 & \ddot{q}_2 & 0 & g\cos(q_1+q_2) \end{bmatrix}$$

and the parameter vector Θ is given by

$$\Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \\ \Theta_4 \\ \Theta_5 \end{bmatrix} = \begin{bmatrix} m_1\ell_{c1}^2 + m_2(\ell_1^2+\ell_{c2}^2) + I_1 + I_2 \\ m_2\ell_1\ell_{c2} \\ m_2\ell_1\ell_{c2} \\ m_1\ell_{c1} + m_2\ell_1 \\ m_2\ell_2 \end{bmatrix} \tag{9.133}$$

Thus, we have parameterized the dynamics using a five dimensional parameter space. Note that in the absence of gravity, as in a SCARA configuration, only three parameters are needed.

9.6 Newton-Euler Formulation

In this section, we present a method for analyzing the dynamics of robot manipulators known as the **Newton-Euler formulation**. This method leads to exactly the same final answers as the Lagrangian formulation presented in earlier sections, but the route taken is quite different. In particular, in the Lagrangian formulation we treat the manipulator as a whole and perform the analysis using a Lagrangian function (the difference between the kinetic energy and the potential energy). In contrast, in the Newton-Euler formulation we treat each link of the robot in turn, and write down the equations describing its linear motion and its angular motion. Of course, since each link is coupled to other links, these equations that describe each link contain coupling forces and torques that appear also in the equations that describe neighboring links. By doing a so-called forward-backward recursion, we are able to determine all of these coupling terms and eventually to arrive at a description of the manipulator as a whole. Thus we see that the philosophy of the Newton-Euler formulation is quite different from that of the Lagrangian formulation.

At this stage the reader can justly ask whether there is a need for another formulation, and the answer is not clear. Historically, both formulations were evolved in parallel, and each was perceived as having certain advantages. For instance, it was believed at one time that the Newton-Euler formulation is better suited to recursive computation than the

Lagrangian formulation. However, the current situation is that both of the formulations are equivalent in almost all respects. Thus at present the main reason for having another method of analysis at our disposal is that it might provide different insights.

In any mechanical system one can identify a set of generalized coordinates (which we introduced in Section 9.1 and labeled $\boldsymbol{q}$) and corresponding generalized forces (also introduced in Section 9.1 and labeled τ). Analyzing the dynamics of a system means finding the relationship between $\boldsymbol{q}$ and $\boldsymbol{\tau}$. At this stage we must distinguish between two aspects: First, we might be interested in obtaining **closed-form equations** that describe the time evolution of the generalized coordinates, such as (9.88) for example. Second, we might be interested in knowing what generalized forces need to be applied in order to realize a *particular* time evolution of the generalized coordinates. The distinction is that in the latter case we only want to know what time dependent function $\tau(\cdot)$ produces a particular trajectory $\boldsymbol{q}(\cdot)$ and may not care to know the general functional relationship between the two. It is perhaps fair to say that in the former type of analysis, the Lagrangian formulation is superior while in the latter case the Newton-Euler formulation is superior. Looking ahead to topics beyond the scope of the book, if one wishes to study more advanced mechanical phenomena such as elastic deformations of the links (i.e., if one no longer assumes rigidity of the links), then the Lagrangian formulation is clearly superior.

In this section we present the general equations that describe the Newton-Euler formulation. In the next section we illustrate the method by applying it to the planar elbow manipulator studied in Section 9.4 and show that the resulting equations are the same as (9.87).

The facts of Newtonian mechanics that are pertinent to the present discussion can be stated as follows:

1. Every action has an equal and opposite reaction. Thus, if body 1 applies a force $\boldsymbol{f}$ and torque $\boldsymbol{\tau}$ to body 2, then body 2 applies a force of $-\boldsymbol{f}$ and torque of $-\boldsymbol{\tau}$ to body 1.

2. The rate of change of the linear momentum equals the total force applied to the body.

3. The rate of change of the angular momentum equals the total torque applied to the body.

Applying the second fact to the linear motion of a body yields the relationship

$$\frac{d(m\boldsymbol{v})}{dt} = \boldsymbol{f} \tag{9.134}$$

where m is the mass of the body, $\boldsymbol{v}$ is the velocity of the center of mass with respect to an inertial frame, and $\boldsymbol{f}$ is the sum of external forces applied to the body. Since in robotic applications the mass is constant as a function of time, (9.134) can be simplified to the familiar relationship

$$m\boldsymbol{a} = \boldsymbol{f} \tag{9.135}$$

where $\boldsymbol{a} = \dot{\boldsymbol{v}}$ is the acceleration of the center of mass.

Applying the third fact to the angular motion of a body gives

$$\frac{d(I_0\boldsymbol{\omega}_0)}{dt} = \boldsymbol{\tau}_0 \tag{9.136}$$

where I_0 is the moment of inertia of the body about an inertial frame whose origin is at the center of mass, $\boldsymbol{\omega}_0$ is the angular velocity of the body, and τ_0 is the sum of torques applied to the body. Now there is an essential difference between linear motion and angular motion. Whereas the mass of a body is constant in most applications, its moment of inertia with respect an inertial frame may or may not be constant. To see this, suppose we attach a frame rigidly to the body, and let I denote the inertia matrix of the body with respect to this frame. Then I remains the same irrespective of whatever motion the body executes. However, the matrix I_0 is given by

$$I_0 = RIR^T \tag{9.137}$$

where R is the rotation matrix that transforms coordinates from the body attached frame to the inertial frame. Thus there is no reason to expect that I_0 is constant as a function of time.

One possible way of overcoming this difficulty is to write the angular motion equation in terms of a frame rigidly attached to the body. This leads to

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{9.138}$$

where I is the (constant) inertia matrix of the body with respect to the body attached frame, $\boldsymbol{\omega}$ is the angular velocity, but expressed in the body attached frame, and $\boldsymbol{\tau}$ is the total torque on the body, again expressed in the body attached frame. Let us now give a derivation of (9.138) to demonstrate clearly where the term $\boldsymbol{\omega} \times (I\boldsymbol{\omega})$ comes from; note that this term is called the **gyroscopic term**.

Let R denote the orientation of the frame rigidly attached to the body w.r.t. the inertial frame; note that it could be a function of time. Then (9.137) gives the relation between I and I_0. Now by the definition of the angular velocity given in Section 2.6, we know that

$$\dot{R}R^T = S(\boldsymbol{\omega}_0). \tag{9.139}$$

In other words, the angular velocity of the body, *expressed in an inertial frame*, is given by (9.139). Of course, the same vector, expressed in the body attached frame, is given by

$$\boldsymbol{\omega}_0 = R\boldsymbol{\omega}, \boldsymbol{\omega} = R^T\boldsymbol{\omega}_0. \tag{9.140}$$

Hence the angular momentum, expressed in the inertial frame, is

$$\boldsymbol{h} = RIR^TR\boldsymbol{\omega} = RI\boldsymbol{\omega}. \tag{9.141}$$

Differentiating and noting that I is constant gives an expression for the rate of change of the angular momentum, expressed as a vector in the inertial frame:

$$\dot{\boldsymbol{h}} = \dot{R}I\boldsymbol{\omega} + RI\dot{\boldsymbol{\omega}}. \tag{9.142}$$

Now

$$S(\boldsymbol{\omega}_0) = \dot{R}R^T, \qquad \dot{R} = S(\boldsymbol{\omega})R. \tag{9.143}$$

Hence, with respect to the inertial frame,

$$\dot{\boldsymbol{h}} \;=\; S(\boldsymbol{\omega}_0)RI\boldsymbol{\omega} + RI\dot{\boldsymbol{\omega}}. \tag{9.144}$$

With respect to the frame rigidly attached to the body, the rate of change of the angular momentum is

$$\begin{aligned} R^T\dot{\boldsymbol{h}} &= R^T S(\boldsymbol{\omega}_0)RI\boldsymbol{\omega} + I\dot{\boldsymbol{\omega}} \\ &= S(R^T\boldsymbol{\omega}_0)I\boldsymbol{\omega} + I\dot{\boldsymbol{\omega}} \\ &= S(\boldsymbol{\omega})I\boldsymbol{\omega} + I\dot{\boldsymbol{\omega}} = \boldsymbol{\omega} \times (I\boldsymbol{\omega}) + I\dot{\boldsymbol{\omega}}. \end{aligned} \tag{9.145}$$

This establishes (9.138). Of course we can, if we wish, write the same equation in terms of vectors expressed in an inertial frame. But we will see shortly that there is an advantage to writing the force and moment equations with respect to a frame attached to link i, namely that a great many vectors in fact reduce to constant vectors, thus leading to significant simplifications in the equations.

Now we derive the Newton-Euler formulation of the equations of motion of an n-link manipulator. For this purpose, we first choose frames $0, \ldots, n$, where frame 0 is an inertial frame, and frame i is rigidly attached to link i for $i \geq 1$. We also introduce several vectors, *which are all expressed in frame i*. The first set of vectors pertain to the velocities and accelerations of various parts of the manipulator.

$$\begin{aligned} \boldsymbol{a}_{c,i} &= \text{the acceleration of the center of mass of link } i. \\ \boldsymbol{a}_{e,i} &= \text{the acceleration of the end of link } i \text{ (i.e., joint } i+1). \\ \boldsymbol{\omega}_i &= \text{the angular velocity of frame } i \text{ w.r.t. frame 0.} \\ \boldsymbol{\alpha}_i &= \text{the angular acceleration of frame } i \text{ w.r.t. frame 0.} \end{aligned}$$

The next several vectors pertain to forces and torques.

$$\begin{aligned} \boldsymbol{g}_i &= \text{the acceleration due to gravity (expressed in frame } i \text{).} \\ \boldsymbol{f}_i &= \text{the force exerted by link } i-1 \text{ on link } i. \\ \boldsymbol{\tau}_i &= \text{the torque exerted by link } i-1 \text{ on link } i. \\ R_i^{i+1} &= \text{the rotation matrix from frame } i+1 \text{ to frame } i. \end{aligned}$$

The final set of vectors pertain to physical features of the manipulator. *Note that each of the following vectors is constant as a function of $\boldsymbol{q}$.* In other words, each of the vectors

listed here is independent of the configuration of the manipulator.

$$\begin{aligned} m_i &= \text{the mass of link } i. \\ I_i &= \text{the inertia matrix of link } i \text{ about a frame parallel} \\ &\quad \text{to frame } i \text{ whose origin is at the center of mass of link } i. \\ \boldsymbol{r}_{i,ci} &= \text{the vector from joint } i \text{ to the center of mass of link } i. \\ \boldsymbol{r}_{i+1,ci} &= \text{the vector from joint } i+1 \text{ to the center of mass of link } i. \\ \boldsymbol{r}_{i,i+1} &= \text{the vector from joint } i \text{ to joint } i+1. \end{aligned}$$

Now consider the free body diagram shown in Figure 9.11; this shows link i together with

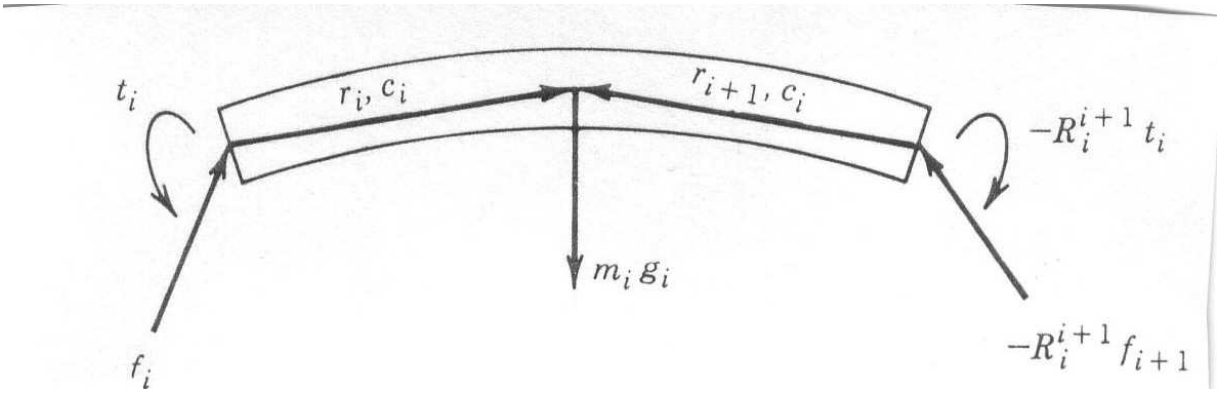


Figure 9.11: Forces and moments on link i.

all forces and torques acting on it. Let us analyze each of the forces and torques shown in the figure. First, $\boldsymbol{f}_i$ is the force applied by link $i-1$ to link i. Next, by the law of action and reaction, link $i+1$ applies a force of $-\boldsymbol{f}_{i+1}$ to link i, but this vector is expressed in frame $i+1$ according to our convention. In order to express the same vector in frame i, it is necessary to multiply it by the rotation matrix R_i^{i+1}. Similar explanations apply to the torques $\boldsymbol{\tau}_i$ and $-R_i^{i+1}\boldsymbol{\tau}_{i+1}$. The force $m_i\boldsymbol{g}_i$ is the gravitational force. Since all vectors in Figure 9.11 are expressed in frame i, the gravity vector $\boldsymbol{g}_i$ is in general a function of i.

Writing down the force balance equation for link i gives

$$\boldsymbol{f}_i - R_i^{i+1}\boldsymbol{f}_{i+1} + m_i\boldsymbol{g}_i = m_i\boldsymbol{a}_{c,i}. \tag{9.146}$$

Next we write down the moment balance equation for link i. For this purpose, it is important to note two things: First, the moment exerted by a force $\boldsymbol{f}$ about a point is given by $\boldsymbol{f} \times \boldsymbol{r}$, where $\boldsymbol{r}$ is the radial vector *from* the point where the force is applied to the point about which we are computing the moment. Second, in the moment equation below, the vector $m_i\boldsymbol{g}_i$ does not appear, since it is applied directly at the center of mass. Thus we have

$$\begin{aligned} &\boldsymbol{\tau}_i - R_i^{i+1}\boldsymbol{\tau}_{i+1} + \boldsymbol{f}_i \times \boldsymbol{r}_{i,ci} - (R_i^{i+1}\boldsymbol{f}_{i+1}) \times \boldsymbol{r}_{i+1,ci} \qquad (9.147) \\ = \;& \boldsymbol{\alpha}_i + \boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i). \end{aligned}$$

Now we present the heart of the Newton-Euler formulation, which consists of finding the vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ and $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_n$ corresponding to a given set of vectors $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$. In other words, we find the forces and torques in the manipulator that correspond to a given set of generalized coordinates and first two derivatives. This information can be used to perform either type of analysis, as described above. That is, we can either use the equations below to find the $\boldsymbol{f}$ and $\boldsymbol{\tau}$ corresponding to a **particular trajectory** $\boldsymbol{q}(\cdot)$, or else to obtain closed-form dynamical equations. The general idea is as follows: Given $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$, suppose we are somehow able to determine all of the velocities and accelerations of various parts of the manipulator, that is, all of the quantities $\boldsymbol{a}_{c,i}$, $\boldsymbol{\omega}_i$ and $\boldsymbol{\alpha}_i$. Then we can solve (9.146) and (9.147) recursively to find all the forces and torques, as follows: First, set $\boldsymbol{f}_{n+l} = 0$ and $\boldsymbol{\tau}_{n+1} = 0$. This expresses the fact that there is no link $n+1$. Then we can solve (9.146) to obtain

$$\boldsymbol{f}_i = R_i^{i+1}\boldsymbol{f}_{i+1} + m_i\boldsymbol{a}_{c,i} - m_i\boldsymbol{g}_i. \tag{9.148}$$

By successively substituting $i = n,\ n-1, \ldots, 1$ we find all forces. Similarly, we can solve (9.147) to obtain

$$\begin{aligned}\boldsymbol{\tau}_i = & \\ & R_i^{i+1}\boldsymbol{\tau}_{i+1} - \boldsymbol{f}_i \times \boldsymbol{r}_{i,ci} + (R_i^{i+1}\boldsymbol{f}_{i+1}) \times \boldsymbol{r}_{i+1,ci} + \boldsymbol{\alpha}_i + \boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i).\end{aligned} \tag{9.149}$$

By successively substituting $i = n\text{m}\ n-1, \ldots, 1$ we find all torques. Note that the above iteration is running in the direction of decreasing i.

Thus the solution is complete once we find an easily computed relation between $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ and $\boldsymbol{a}_{c,i}, \boldsymbol{\omega}_i$ and $\boldsymbol{\alpha}_i$. This can be obtained by a recursive procedure in the direction of *increasing* i. This procedure is given below, for the case of revolute j oint s; the corresponding relation ships for prismatic joints are actually easier to derive.

In order to distinguish between quantities expressed with respect to frame i and the base frame, we use a superscript (0) to denote the latter. Thus, for example, $\boldsymbol{\omega}_i$ denotes the angular velocity of frame i expressed in frame i, while $\boldsymbol{\omega}_i^{(0)}$ denotes the same quantity expressed in an inertial frame.

Now from Section 2.6 we have that

$$\boldsymbol{\omega}_i^{(0)} = \boldsymbol{\omega}_{i-1}^{(0)} + \boldsymbol{z}_{i-1}\dot{q}_i. \tag{9.150}$$

This merely expresses the fact that the angular velocity of frame i equals that of frame $i-1$ plus the added rotation from joint i. To get a relation between ω_i and $\boldsymbol{\omega}_{i-1}$, we need only express the above equation in frame i rather than the base frame, taking care to account for the fact that $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_{i-1}$ are expressed in different frames. This leads to

$$\boldsymbol{\omega}_i = (R_{i-1}^i)^T\boldsymbol{\omega}_{i-1} + \boldsymbol{b}_i\dot{q}_i \tag{9.151}$$

where

$$\boldsymbol{b}_i = (R_0^i)^T\boldsymbol{z}_{i-1} \tag{9.152}$$

is the axis of rotation of joint i expressed in frame i.

Next let us work on the angular acceleration $\boldsymbol{\alpha}_i$. It is vitally important to note here that

$$\boldsymbol{\alpha}_i = (R_0^i)^T \dot{\boldsymbol{\omega}}_i^{(0)}. \tag{9.153}$$

In other words, $\boldsymbol{\alpha}_i$ is the derivative of the angular velocity of frame i, but expressed in frame i. It is not true that $\boldsymbol{\alpha}_i = \dot{\boldsymbol{\omega}}_i$! We will encounter a similar situation with the velocity and acceleration of the center of mass. Now we see directly from (9.150) that

$$\dot{\boldsymbol{\omega}}_i^{(0)} = \dot{\boldsymbol{\omega}}_{i-1}^{(0)} + \boldsymbol{z}_{i-1}\ddot{q}_i + \boldsymbol{\omega}_i^{(0)} \times \boldsymbol{z}_{i-1}\dot{q}_i. \tag{9.154}$$

Expressing the same equation in frame i gives

$$\boldsymbol{\alpha}_i = (R_{i-1}^i)^T \boldsymbol{\alpha}_{i-1} + \boldsymbol{b}_i\ddot{q}_i + \boldsymbol{\omega}_i \times \boldsymbol{b}_i\dot{q}_i. \tag{9.155}$$

Now we come to the linear velocity and acceleration terms. Note that, in contrast to the angular velocity, the linear velocity does not appear anywhere in the dynamic equations; however, an expression for the linear velocity is needed before we can derive an expression for the linear acceleration. From Section 5.2, we get that the velocity of the center of mass of link i is given by

$$\boldsymbol{v}_{c,i}^{(0)} = \boldsymbol{v}_{e,i-1}^{(0)} + \boldsymbol{\omega}_i^{(0)} \times \boldsymbol{r}_{i,ci}^{(0)}. \tag{9.156}$$

To obtain an expression for the acceleration, we use (**??**), and note that the vector $\boldsymbol{r}_{i,ci}^{(0)}$ is constant in frame i. Thus

$$\boldsymbol{a}_{c,i}^{(0)} = \boldsymbol{a}_{e,i-1}^{(0)} \times \boldsymbol{r}_{i,ci}^{(0)} + \boldsymbol{\omega}_i^{(0)} \times (\boldsymbol{\omega}_i^{(0)} \times \boldsymbol{r}_{i,ci}^{(0)}). \tag{9.157}$$

Now

$$\boldsymbol{a}_{c,i} = (R_0^i)^T \boldsymbol{a}_{c,i}^{(0)}. \tag{9.158}$$

Let us carry out the multiplication and use the familiar property

$$R(\boldsymbol{a} \times \boldsymbol{b}) = (R\boldsymbol{a}) \times (R\boldsymbol{b}). \tag{9.159}$$

We also have to account for the fact that $\boldsymbol{a}_{e,i-1}$ is expressed in frame $i-1$ and transform it to frame i. This gives

$$\boldsymbol{a}_{c,i} = (R_{i-1}^i)^T \boldsymbol{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i,ci} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i,ci}). \tag{9.160}$$

Now to find the acceleration of the end of link i, we can use (9.160) with $r_{i,i+1}$ replacing $r_{i,ci}$. Thus

$$\boldsymbol{a}_{e,i} = (R_{i-1}^i)^T \boldsymbol{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i,i+1} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i,i+1}). \tag{9.161}$$

Now the recursive formulation is complete. We can now state the Newton-Euler formulation as follows.

1. Start with the initial conditions

$$\boldsymbol{\omega}_0 = 0, \alpha_0 = 0, \boldsymbol{a}_{c,0} = 0, \boldsymbol{a}_{e,0} = 0 \tag{9.162}$$

and solve (9.151), (9.155), (9.161) and (9.160) (in that order!) to compute $\boldsymbol{\omega}_i$, $\boldsymbol{\alpha}_i$ and $\boldsymbol{a}_{c,i}$ for i increasing from 1 to n.

2. Start with the terminal conditions

$$\boldsymbol{f}_{n+1} = 0, \qquad \boldsymbol{\tau}_{n+1} = 0 \tag{9.163}$$

and use (9.148) and (9.149) to compute $\boldsymbol{f}_i$ and $\boldsymbol{\tau}_i$ for i decreasing from n to 1.

9.7 Planar Elbow Manipulator Revisited

In this section we apply the recursive Newton-Euler formulation derived in Section 9.6 to analyze the dynamics of the planar elbow manipulator of figure 9.8, and show that the Newton-Euler method leads to the same equations as the Lagrangian method, namely (9.87).

We begin with the forward recursion to express the various velocities and accelerations in terms of q_1, q_2 and their derivatives. Note that, in this simple case, it is quite easy to see that

$$\boldsymbol{\omega}_1 = \dot{q}_1\boldsymbol{k}, \alpha_1 = \ddot{q}_1\boldsymbol{k}, \boldsymbol{\omega}_2 = (\boldsymbol{q}_1 + \boldsymbol{q}_2)\boldsymbol{k}, \alpha_2 = (\ddot{q}_1 + \ddot{q}_2)\boldsymbol{k} \tag{9.164}$$

so that there is no need to use (9.151) and (9.155). Also, the vectors that are independent of the configuration are as follows:

$$\boldsymbol{r}_{1,c1} = \ell_{c1}\boldsymbol{i}, \boldsymbol{r}_{2,c1} = (\ell_1 - \ell_{c1})\boldsymbol{i}, \boldsymbol{r}_{1,2} = \ell_1\boldsymbol{i} \tag{9.165}$$

$$\boldsymbol{r}_{2,c2} = \ell_{c2}\boldsymbol{i}, \boldsymbol{r}_{3,c2} = (\ell_2 - \ell_{c2})\boldsymbol{i}, \boldsymbol{r}_{2,3} = \ell_2\boldsymbol{i}. \tag{9.166}$$

Forward Recursion link 1

Using (9.160) with $i = 1$ and noting that $\boldsymbol{a}_{e,0} = 0$ gives

$$\begin{aligned} \boldsymbol{a}_{c,1} &= \ddot{q}_1\boldsymbol{k} \times \ell_{c1}\boldsymbol{i} + \dot{q}_1\boldsymbol{k} \times (\dot{q}_1\boldsymbol{k} \times \ell_{c1}\boldsymbol{i}) \\ &= \ell_{c1}\ddot{q}_1\boldsymbol{j} - \ell_{c1}\dot{q}_1^2\boldsymbol{i} = \begin{bmatrix} -\ell_{c1}\dot{q}_1^2 \\ \ell_c\ddot{q}_1 \\ 0 \end{bmatrix}. \end{aligned} \tag{9.167}$$

Notice how simple this computation is when we do it with respect to frame 1. Compare with the same computation in frame 0! Finally, we have

$$\boldsymbol{g}_1 = -(R_0^1)^T g\boldsymbol{j} = g \begin{bmatrix} \sin q_1 \\ -\cos q_1 \\ 0 \end{bmatrix} \tag{9.168}$$

where g is the acceleration due to gravity. At this stage we can economize a bit by not displaying the third components of these accelerations, since they are obviously always zero. Similarly, the third component of all forces will be zero while the first two components of all torques will be zero. To complete the computations for link 1, we compute the acceleration of end of link 1. Clearly, this is obtained from (9.167) by replacing ℓ_{c1} by ℓ_1. Thus

$$\boldsymbol{a}_{e,1} = \begin{bmatrix} -\ell_1\dot{q}_1^2 \\ \ell_1\ddot{q}_1 \end{bmatrix}. \tag{9.169}$$

Forward Recursion: Link 2

Once again we use (9.160) and substitute for (o2 from (9.164); this yields

$$\alpha_{c,2} = (R_1^2)^T\boldsymbol{a}_{e,1} + [(\ddot{q}_1+\ddot{q}_2)\boldsymbol{k}] \times \ell_{c2}\boldsymbol{i} + (\dot{q}_1+\dot{q}_2)\boldsymbol{k} \times [(\dot{q}_1+\dot{q}_2)\boldsymbol{k} \times \ell_{c2}\boldsymbol{i}] \tag{9.170}$$

The only quantity in the above equation which is configuration dependent is the first one. This can be computed as

$$\begin{aligned} (R_1^2)^T\boldsymbol{a}_{e,1} &= \begin{bmatrix} \cos q_2 & \sin q_2 \\ -\sin q_2 & \cos q_2 \end{bmatrix}\begin{bmatrix} -\ell_1\dot{q}_1^2 \\ \ell_1\ddot{q}_1 \end{bmatrix} \\ &= \begin{bmatrix} -\ell_1\dot{q}_1^2\cos q_2 + \ell_1\ddot{q}_1\sin\dot{q}_2 \\ \ell_1\dot{q}_1^2\sin q_2 + \ell_1\ddot{q}_1\cos q_2 \end{bmatrix}. \end{aligned} \tag{9.171}$$

Substituting into (9.170) gives

$$\boldsymbol{a}_{c,2} = \begin{bmatrix} -\ell_1\dot{q}_1^2\cos q_2 + \ell_1\ddot{q}_1\sin q_2 - \ell_{c2}(\dot{q}_1+\dot{q}_2)^2 \\ \ell_1\dot{q}_1^2\sin q_2 + \ell_1\ddot{q}_1\cos q_2 - \ell_{c2}(\ddot{q}_1+\ddot{q}_2) \end{bmatrix}. \tag{9.172}$$

The gravitational vector is

$$\boldsymbol{g}_2 = g\begin{bmatrix} \sin(q_1+q_2) \\ -\cos(q_1+q_2) \end{bmatrix}. \tag{9.173}$$

Since there are only two links, there is no need to compute $\boldsymbol{a}_{e,2}$. Hence the forward recursions are complete at this point.

Backward Recursion: Link 2

Now we carry out the backward recursion to compute the forces and joint torques. Note that, in this instance, the joint torques are the externally applied quantities, and our ultimate objective is to derive dynamical equations involving the joint torques. First we apply (9.148) with $i = 2$ and note that $\boldsymbol{f}_3 = 0$. This results in

$$\boldsymbol{f}_2 = m_2\boldsymbol{a}_{c,2} - m_2\boldsymbol{g}_2 \tag{9.174}$$

$$\boldsymbol{\tau}_2 = I_2\boldsymbol{\alpha}_2 + \boldsymbol{\omega}_2 \times (I_2\boldsymbol{\omega}_2) - \boldsymbol{f}_2 \times \ell_{c2}\boldsymbol{i}. \tag{9.175}$$

Now we can substitute for $\boldsymbol{\omega}_2, \alpha_2$ from (9.164), and for $\boldsymbol{a}_{c,2}$ from (9.172). We also note that the gyroscopic term equals zero, since both $\boldsymbol{\omega}_2$ and $I_2\boldsymbol{\omega}_2$ are aligned with $\boldsymbol{k}$. Now the cross product $\boldsymbol{f}_2 \times \ell_{c2}\boldsymbol{i}$ is clearly aligned with $\boldsymbol{k}$ and its magnitude is just the second component of $\boldsymbol{f}_2$. The final result is

$$\begin{aligned}\boldsymbol{\tau}_2 &= I_2(\ddot{q}_1 + \ddot{q}_2)\boldsymbol{k} + [m_2\ell_1\ell_{c2}\sin q_2\dot{q}_1^2 + m_2\ell_1\ell_{c2}\cos q_2\ddot{q}_1 \\ &\quad + m_2\ell_{c2}^2(\ddot{q}_1 + \ddot{q}_2) + m)2\ell_{c2}g\cos(q_1 + q_2)]\boldsymbol{k} \end{aligned} \tag{9.176}$$

Since $\boldsymbol{\tau}_2 = \tau_2\boldsymbol{k}$, we see that the above equation is the same as the second equation in (9.88).

Backward Recursion: Link 1

To complete the derivation, we apply (9.148) and (9.149) with $i = 1$. First, the force equation is

$$\boldsymbol{f}_1 = m_1\boldsymbol{a}_{c,1} + R_1^2\boldsymbol{f}_2 - m_1\boldsymbol{g}_1 \tag{9.177}$$

and the torque equation is

$$\begin{aligned}\boldsymbol{\tau}_1 &= R_1^2\boldsymbol{\tau}_2 - \boldsymbol{f}_1 \times \ell_{c,1}\boldsymbol{i} - (R_1^2\boldsymbol{f}_2) \times (\ell_1 - \ell_{c1})\boldsymbol{i} \\ &\quad + I_1\alpha_1 + \boldsymbol{\omega}_1 \times (I_1\boldsymbol{\omega}_1). \end{aligned} \tag{9.178}$$

Now we can simplify things a bit. First, $R_1^2\tau_2 = \tau_2$, since the rotation matrix does not affect the third components of vectors. Second, the gyroscopic term is the again equal to zero. Finally, when we substitute for $\boldsymbol{f}_1$ from (9.177) into (9.178), a little algebra gives

$$\begin{aligned}\boldsymbol{\tau}_1 &= \boldsymbol{\tau}_2 - m_1\boldsymbol{a}_{c,1} \times \ell_{c1}\boldsymbol{i} + m_1\boldsymbol{g}_1 \times \ell_{c1}\boldsymbol{i} - (R_1^2\boldsymbol{f}_2) \\ &\quad \times\ell_1\boldsymbol{i} + I_1\boldsymbol{i} + I_1\alpha_1. \end{aligned} \tag{9.179}$$

Once again, all these products are quite straightforward, and the only difficult calculation is that of $R_1^2\boldsymbol{f}_2$. The final result is:

$$\begin{aligned}\boldsymbol{\tau}_1 &= \boldsymbol{\tau}_2 + m_1\ell_{c1}^2 + m_1\ell_{c1}g\cos q_1 + m_2\ell_1 g\cos q_1 + I_1\ddot{q}_1 \\ &\quad + m_2\ell_1^2\ddot{q}_1 - m_1\ell_1\ell_{c2}(\dot{q}_1 + \dot{q}_2)^2\sin q_2 + m_2\ell_1\ell_{c2}(\ddot{q}_1 + \ddot{q}_2)\cos q_2. \end{aligned} \tag{9.180}$$

If we now substitute for $\boldsymbol{\tau}_1$ from (9.176) and collect terms, we will get the first equation in (9.88); the details are routine and are left to the reader.

Chapter 10

INDEPENDENT JOINT CONTROL

10.1 Introduction

The control problem for robot manipulators is the problem of determining the time history of joint inputs required to cause the end-effector to execute a commanded motion. The joint inputs may be joint forces and torques, or they may be inputs to the actuators, for example, voltage inputs to the motors, depending on the model used for controller design. The commanded motion is typically specified either as a sequence of end-effector positions and orientations, or as a continuous path.

There are many control techniques and methodologies that can be applied to the control of manipulators. The particular control method chosen as well as the manner in which it is implemented can have a significant impact on the performance of the manipulator and consequently on the range of its possible applications. For example, continuous path tracking requires a different control architecture than does point-to-point control.

In addition, the mechanical design of the manipulator itself will influence the type of control scheme needed. For example, the control problems encountered with a cartesian manipulator are fundamentally different from those encountered with an elbow type manipulator. This creates a so-called *hardware/software trade-off* between the mechanical structure of the system and the architecture/programming of the controller.

Technological improvements are continually being made in the mechanical design of robots, which in turn improves their performance potential and broadens their range of applications. Realizing this increased performance, however, requires more sophisticated approaches to control. One can draw an analogy to the aerospace industry. Early aircraft were relatively easy to fly but possessed limited performance capabilities. As performance increased with technological advances so did the problems of control to the extent that the latest vehicles, such as the space shuttle or forward swept wing fighter aircraft, cannot be flown without sophisticated computer control.

As an illustration of the effect of the mechanical design on the control problem, compare

a robot actuated by permanent magnet DC motors with gear reduction to a direct-drive robot using high-torque motors with no gear reduction. In the first case, the motor dynamics are linear and well understood and the effect of the gear reduction is largely to decouple the system by reducing the inertia coupling among the joints. However, the presence of the gears introduces friction, drive train compliance and backlash.

In the case of a direct-drive robot, the problems of backlash, friction, and compliance due to the gears are eliminated. However, the coupling among the links is now significant, and the dynamics of the motors themselves may be much more complex. The result is that in order to achieve high performance from this type of manipulator, a different set of control problems must be addressed.

In this chapter we consider the simplest type of control strategy, namely, independent joint control. In this type of control each axis of the manipulator is controlled as a single-input/single-output (SISO) system. Any coupling effects due to the motion of the other links is treated as a disturbance. We assume, in this chapter, that the reader has had an introduction to the theory of feedback control systems up to the level of say, Kuo [**?**].

The basic structure of a single-input/single-output feedback control system is shown in Figure 10.1. The design objective is to choose the compensator in such a way that the plant

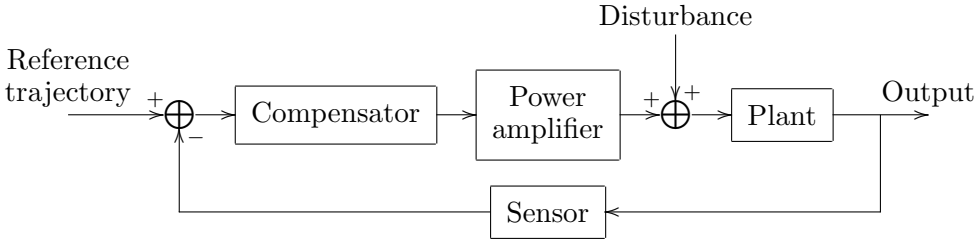


Figure 10.1: Basic structure of a feedback control system.

output "tracks" or follows a desired output, given by the reference signal. The control signal, however, is not the only input acting on the system. Disturbances, which are really inputs that we do not control, also influence the behavior of the output. Therefore, the controller must be designed, in addition, so that the effects of the disturbances on the plant output are reduced. If this is accomplished, the plant is said to "reject" the disturbances. The twin objectives of *tracking* and *disturbance rejection* are central to any control methodology.

10.2 Actuator Dynamics

In Chapter 9 we obtained the following set of differential equations describing the motion of an n degree of freedom robot (cf. Equation (9.61))

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \quad = \quad \tau \tag{10.1}$$

It is important to understand exactly what this equation represents. Equation (10.1) represents the dynamics of an interconnected chain of ideal rigid bodies, supposing that there

is a generalized force τ acting at the joints. We can assume that the k-th component τ_k of the generalized force vector τ is a torque about the joint axis z_{k-1} if joint k is revolute and is a force along the joint axis z_{k-1} if joint k is prismatic. This generalized force is produced by an actuator, which may be electric, hydraulic or pneumatic. Although (10.1) is extremely complicated for all but the simplest manipulators, it nevertheless is an idealization, and there are a number of dynamic effects that are *not* included in (10.1). For example, friction at the joints is not accounted for in these equations and may be significant for some manipulators. Also, no physical body is completely rigid. A more detailed analysis of robot dynamics would include various sources of flexibility, such as elastic deformation of bearings and gears, deflection of the links under load, and vibrations. In this section we are interested mainly in the dynamics of the actuators producing the generalized force $\boldsymbol{\tau}$. We treat only the dynamics of permanent magnet DC-motors, as these are common for use in present-day robots.

A DC-motor basically works on the principle that a current carrying conductor in a magnetic field experiences a force $\boldsymbol{F} = \boldsymbol{i} \times \boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is the magnetic flux and $\boldsymbol{i}$ is the current in the conductor. The motor itself consists of a fixed *stator* and a movable *rotor* that rotates inside the stator, as shown in Figure 10.2. If the stator produces a radial magnetic

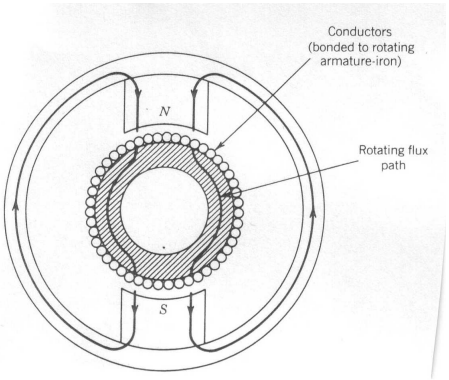


Figure 10.2: Cross-sectional view of a surface-wound permanent magnet DC motor.

flux $\boldsymbol{\phi}$ and the current in the rotor (also called the *armature*) is $\boldsymbol{i}$ then there will be a torque on the rotor causing it to rotate. The magnitude of this torque is

$$\tau_m = K_1 \phi i_a \tag{10.2}$$

where τ_m is the motor torque $(N - m)$, ϕ is the magnetic flux (webers), i_a is the armature current (amperes), and K_1 is a physical constant. In addition, whenever a conductor moves in a magnetic field, a voltage V_b is generated across its terminals that is proportional to the velocity of the conductor in the field. This voltage, called the *back emf*, will tend to oppose the current flow in the conductor.

Thus, in addition to the torque τ_m in (10.2), we have the back emf relation

$$V_b = K_2 \phi \omega_m \tag{10.3}$$

where V_b denotes the back emf (Volts), ω_m is the angular velocity of the rotor (rad/sec), and K_2 is a proportionality constant.

DC-motors can be classified according to the way in which the magnetic field is produced and the armature is designed. Here we discuss only the so-called *permanent magnet* motors whose stator consists of a permanent magnet. In this case we can take the flux, ϕ, to be a constant. The torque on the rotor is then controlled by controlling the armature current, i_a.

Consider the schematic diagram of Figure 10.3 where

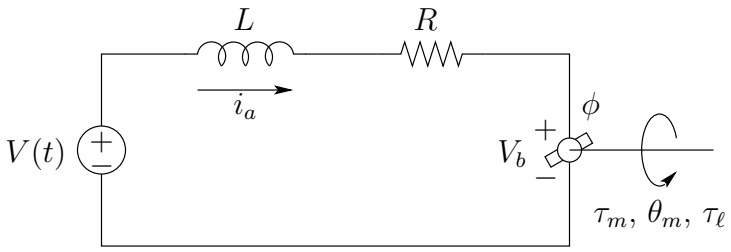


Figure 10.3: Circuit diagram for armature controlled DC motor.

$$\begin{aligned} V &= \text{armature voltage} \\ L &= \text{armature inductance} \\ R &= \text{armature resistance} \\ V_b &= \text{back emf} \\ i_a &= \text{armature current} \\ \theta_m &= \text{rotor position (radians)} \\ \tau_m &= \text{generated torque} \\ \tau_\ell &= \text{load torque} \\ \phi &= \text{magnetic flux due to stator} \end{aligned}$$

The differential equation for the armature current is then

$$L\frac{di_a}{dt} + Ri_a = V - V_b. \tag{10.4}$$

Since the flux ϕ is constant the torque developed by the motor is

$$\tau_m = K_1\phi i_a = K_m i_a \tag{10.5}$$

where K_m is the torque constant in $N-m/\text{amp}$. From (10.3) we have

$$V_b = K_2\phi\omega_m = K_b\omega_m = K_b\frac{d\theta_m}{dt} \tag{10.6}$$

where K_b is the back emf constant.

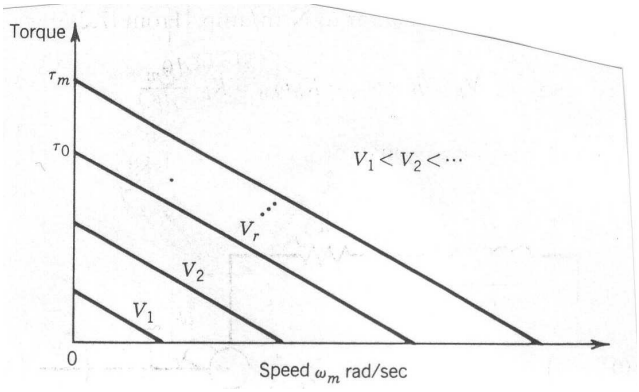


Figure 10.4: Typical torque-speed curves of a DC motor.

We can determine the torque constant of the DC motor using a set of torque-speed curves as shown in Figure 10.4 for various values of the applied voltage V. When the motor is stalled, the blocked-rotor torque at the rated voltage is denoted τ_0. Using Equation (10.4) with $V_b = 0$ and $di_a/dt = 0$ we have

$$\begin{aligned} V_r &= Ri_a \\ &= \frac{R\tau_0}{K_m}. \end{aligned} \tag{10.7}$$

Therefore the torque constant is

$$K_m = \frac{R\tau_0}{V_r}. \tag{10.8}$$

The remainder of the discussion refers to Figure 10.5 consisting of the DC-motor in series

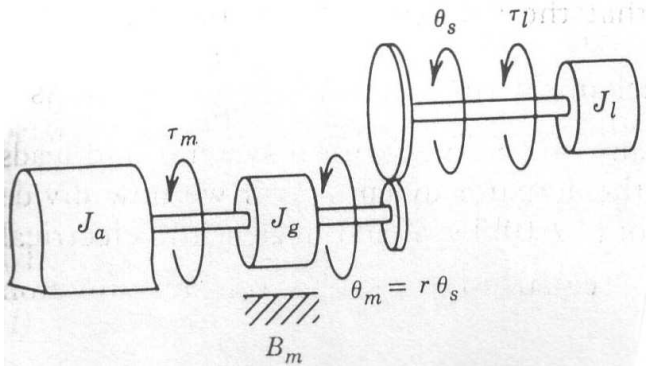


Figure 10.5: Lumped model of a single link with actuator/gear train.

with a gear train with gear ratio $r : 1$ and connected to a link of the manipulator. The gear

ratio r typically has values in the range 20 to 200 or more. Referring to Figure 10.5, we set $J_m = J_a + J_g$, the sum of the actuator and gear inertias. The equation of motion of this system is then

$$J_m \frac{d^2\theta_m}{dt^2} + B_m \frac{d\theta_m}{dt} = \tau_m - \tau_\ell / r \tag{10.9}$$

$$= K_m i_a - \tau_\ell / r \tag{10.10}$$

the latter equality coming from (10.5). In the Laplace domain the three equations (10.4), (10.6) and (10.9) may be combined and written as

$$(Ls + R) I_a(s) = V(s) - K_b s \Theta_m(s) \tag{10.11}$$

$$(J_m s^2 + B_m s)\Theta_m(s) = K_i I_a(s) - \tau_\ell / r(s). \tag{10.12}$$

The block diagram of the above system is shown in Figure 10.6. The transfer function from

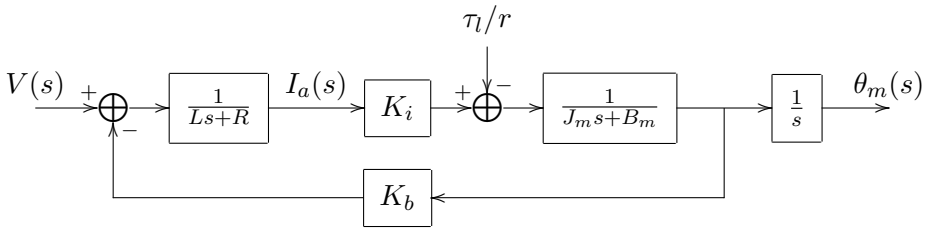


Figure 10.6: Block diagram for a DC motor system.

$V(s)$ to $\Theta_m(s)$ is then given by (with $\tau_\ell = 0$)

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m}{s\left[(Ls + R)(J_m s + B_m) + K_b K_m\right]}. \tag{10.13}$$

The transfer function from the load torque τ_ℓ / r to Θ_m is given by (with $V = 0$)

$$\frac{\Theta_m(s)}{\tau_\ell(s)} = \frac{-(Ls + R)}{s\left[(Ls + R)(J_m s + B_m) + K_b K_m\right]}. \tag{10.14}$$

Frequently it is assumed that the "electrical time constant" $\frac{L}{R}$ is much smaller than the "mechanical time constant" $\frac{J_m}{B_m}$. This is a reasonable assumption for many electromechanical systems and leads to a reduced order model of the actuator dynamics. If we now divide numerator and denominator of (10.11) by R and neglect the electrical time constant by setting $\frac{L}{R}$ equal to zero, the transfer function between Θ_m and V becomes (again, with $\tau_\ell = 0$)

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m / R}{s(J_m s + B_m + K_b K_m / R)}. \tag{10.15}$$

Similarly the transfer function between Θ_m and τ_ℓ/r is

$$\frac{\Theta_m(s)}{\tau_\ell(s)} = -\frac{1}{s(J_m(s) + B_m + K_b K_m/R)}. \tag{10.16}$$

In the time domain Equations (10.15) and (10.16) represent, by superposition, the second order differential equation

$$J_m\ddot{\theta}_m(t) + (B_m + K_b K_m/R)\dot{\theta}_m(t) = (K_m/R)V(t) - \tau_\ell(t)/r \tag{10.17}$$

The block diagram corresponding to the reduced order system (10.17) is shown in Figure 10.7.

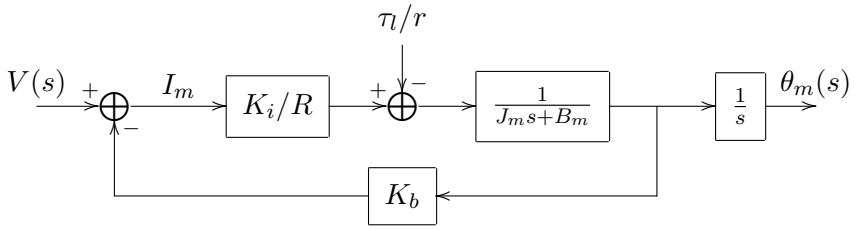


Figure 10.7: Block diagram for reduced order system.

If the output side of the gear train is directly coupled to the link, then the joint variables and the motor variables are related by

$$\theta_{m_k} = r_k q_k \quad ; \quad k = 1, \ldots, n \tag{10.18}$$

where r_k is the k-th gear ratio. Similarly, the joint torques τ_k given by (10.1) and the actuator load torques τ_{ℓ_k} are related by

$$\tau_{\ell_k} = \tau_k \quad ; \quad k = 1, \ldots, n. \tag{10.19}$$

However, in manipulators incorporating other types of drive mechanisms such as belts, pulleys, chains, etc., θ_{m_k} need not equal $r_k q_k$. In general one must incorporate into the dynamics a transformation between joint space variables and actuator variables of the form

$$q_k = f_k(\theta_{s_1}, \ldots, \theta_{s_n}) \quad ; \quad \tau_{\ell_k} = f_k(\tau_1, \ldots, \tau_n) \tag{10.20}$$

where $\theta_{s_k} = \theta_{m_k}/r_k$.

Example 7.2.1

Consider the two link planar manipulator shown in Figure 10.8, whose actuators are both located on link 1. In this case we have,

$$q_1 = \theta_{s_1} \quad ; \quad q_2 = \theta_{s_1} + \theta_{s_2}. \tag{10.21}$$

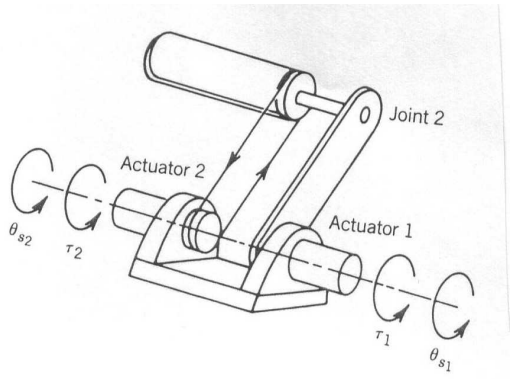


Figure 10.8: Two-link manipulator with remotely driven link.

Similarly, the joint torques τ_i and the actuator load torques τ_ℓ, are related by

$$\tau_{\ell_1} = \tau_1 \quad ; \quad \tau_{\ell_2} = \tau_1 + \tau_2. \tag{10.22}$$

The inverse transformation is then

$$\theta_{s_1} = q_1 \quad ; \quad \theta_{s_2} = q_2 - q_1 \tag{10.23}$$

and

$$\tau_1 = \tau_{\ell_1} \quad ; \quad \tau_2 = \tau_{\ell_2} - \tau_{\ell_1}. \tag{10.24}$$

10.3 Set-Point Tracking

In this section we discuss set-point tracking using a PD or PID compensator. This type of control is adequate for applications not involving very fast motion, especially in robots with large gear reduction between the actuators and the links. The analysis in this section follows typical engineering practice rather than complete mathematical rigor.

For the following discussion, assume for simplicity that

$$\begin{aligned} q_k = \theta_{s_k} &= \theta_{m_k}/r_k \text{ and} \\ \tau_{\ell_k} &= \tau_k. \end{aligned} \tag{10.25}$$

Then, for $k = 1, \ldots, n$, the equations of motion of the manipulator can be written as

$$\sum_{j=1}^{n} d_{jk}(\boldsymbol{q})\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i q_j + g_k(\boldsymbol{q}) = \tau_k \tag{10.26}$$

$$J_{m_k}\ddot{\theta}_{m_k} + (B_{m_k} + K_{b_k}K_{m_k}/R_k)\dot{\theta}_{m_k} = K_{m_k}/R_k V_k - \tau_k/r_k \tag{10.27}$$

Combining these equations yields

$$(J_{m_k} + \frac{1}{r_k^2} d_{kk}(q))\ddot{\theta}_{m_k} + (B_{m_k} + K_{b_k} K_{m_k}/R_k)\dot{\theta}_{m_k} = K_{m_k}/R_k V_k - d_k \tag{10.28}$$

where d_k is defined by

$$d_k \;:=\; \frac{1}{r_k} \sum_{j \neq k} \ddot{q}_j + \sum_{i,j} c_{ijk} \dot{q}_i \dot{q}_j + g_k. \tag{10.29}$$

Note that the coefficient of $\ddot{\theta}_{m_k}$ in the above expression is a nonlinear function of the manipulator configuration, q. However, large gear reduction, i.e. large values of r_k, mitigates the influence of this term and one often defines a constant *average*, or *effective inertia* J_{eff_k} as an approximation of the exact expression $J_{m_k} + \frac{1}{r_k^2} d_{kk}(q)$. If we further define

$$B_{eff_k} = B_{m_k} + K_{b_k} K_{m_k}/R_k \quad \text{and} \quad u_k = K_{m_k}/R_k V_k \tag{10.30}$$

we may write (10.27) as

$$J_{eff_k} \ddot{\theta}_{m_k} + B_{eff_k} \dot{\theta}_{m_k} \;=\; u_k - d_k \tag{10.31}$$

The advantage of this model is its simplicity since the motor dynamics represented by (10.27) are linear. The effect of the nonlinear coupling terms is treated as a disturbance d_k, which may be small for large gear reduction provided the velocities and accelerations of the joints are also small.

Henceforth we suppress the subscript k representing the particular joint and represent (10.31) in the Laplace domain by the block diagram of Figure 10.9. The *set-point tracking*

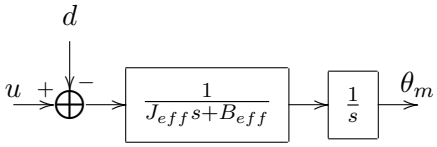


Figure 10.9: Block diagram of simplified open loop system with effective inertia and damping.

problem is now the problem of tracking a constant or step reference command θ^d.

10.3.1 PD Compensator

As a first illustration we choose a so-called PD-compensator. The resulting closed loop system is shown in Figure 10.10. The input $U(s)$ is given by

$$U(s) \;=\; K_p(\Theta^d(s) - \Theta(s)) - K_D s \Theta(s) \tag{10.32}$$

where K_p, K_D are the proportional (P) and derivative (D) gains, respectively.

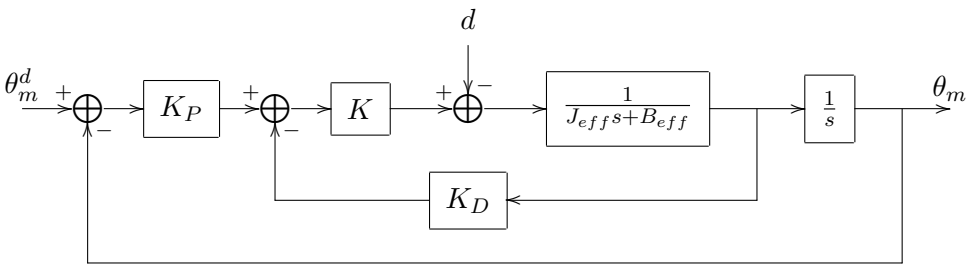


Figure 10.10: Closed loop system with PD-control.

Taking Laplace transforms of both sides of (10.31) and using the expression (10.32) for the feedback control $V(s)$, leads to the closed loop system

$$\Theta_m(s) = \frac{KK_p}{\Omega(s)}\Theta^d(s) - \frac{1}{\Omega(s)}D(s) \tag{10.33}$$

where $\Omega(s)$ is the closed loop characteristic polynomial

$$\Omega(s) = J_{eff}s^2 + (B_{eff} + KK_D)s + KK_p. \tag{10.34}$$

The closed loop system will be stable for all positive values of K_p and K_D and bounded disturbances, and the tracking error is given by

$$E(s) = \Omega^d(s) - \Theta_m(s) \tag{10.35}$$

$$= \frac{J_{eff}s^2 + (B_{eff} + KK_D)s}{\Omega(s)}\Theta^d(s) + \frac{1}{\Omega(s)}D(s). \tag{10.36}$$

For a step reference input

$$\Theta^d(s) = \frac{\Omega^d}{s} \tag{10.37}$$

and a constant disturbance

$$D(s) = \frac{D}{s} \tag{10.38}$$

it now follows directly from the final value theorem [4] that the steady state error e_{ss} satisfies

$$e_{ss} = \lim_{s\to 0} sE(s) \tag{10.39}$$

$$= \frac{-D}{KK_p}. \tag{10.40}$$

Since the magnitude D of the disturbance is proportional to the gear reduction $\frac{1}{r}$ we see that the steady state error is smaller for larger gear reduction and can be made arbitrarily

small by making the position gain K_p large, which is to be expected since the system is Type 1.

We know, of course, from (10.29) that the disturbance term $D(s)$ in (10.35) is not constant. However, in the steady state this disturbance term is just the gravitational force acting on the robot, which is constant. The above analysis therefore, while only approximate, nevertheless gives a good description of the actual steady state error using a PD compensator assuming stability of the closed loop system.

10.3.2 Performance of PD Compensators

For the PD-compensator given by (10.32) the closed loop system is second order and hence the step response is determined by the closed loop natural frequency ω and damping ratio ζ. Given a desired value for these quantities, the gains K_D and K_p can be found from the expression

$$s^2 + \frac{(B_{eff} + KK_D)}{J_{eff}}s + \frac{KK_p}{J_{eff}} \quad = \quad s^2 + 2\zeta\omega s + \omega^2 \tag{10.41}$$

as

$$K_p = \frac{\omega^2 J_{eff}}{K}, \qquad K_D = \frac{2\zeta\omega J_{eff} - B_{eff}}{K}. \tag{10.42}$$

It is customary in robotics applications to take $\zeta = 1$ so that the response is critically damped. This produces the fastest non-oscillatory response. In this context ω determines the speed of response.

Example 10.1 *Consider the second order system of Figure 10.11. The closed loop characteristic polynomial is*

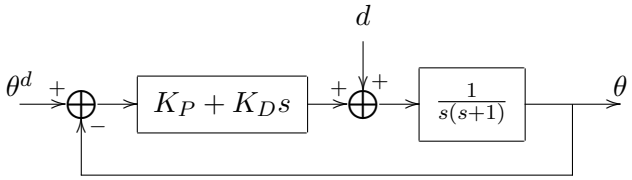


Figure 10.11: System of Example 7.3.1.

$$p(s) \quad = \quad s^2 + (1 + K_D)s + K_p. \tag{10.43}$$

Suppose $\theta^d = 10$ and there is no disturbance ($d = 0$). With $\zeta = 1$, the required PD gains for various values of ω are shown in Table 10.1. The corresponding step responses are shown in Figure 10.12.

Now suppose that there is a constant disturbance $d = 40$ acting on the system. The response of the system with the PD gains of Table 10.1 are shown in Figure 10.13. We see that the steady state error due to the disturbance is smaller for large gains as expected. ⋄

Table 10.1: .

Ω	K_P	K_D
4	16	7
8	64	15
12	144	23

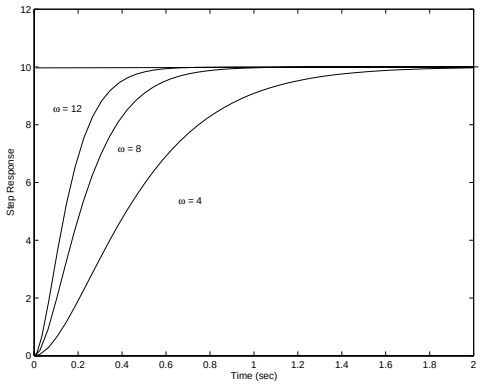


Figure 10.12: Critically damped second order step responses.

10.3.3 PID Compensator

In order to reject a constant disturbance using PD control we have seen that large gains are required. By using integral control we may achieve zero steady state error while keeping the gains small. Thus, let us add an integral term $\frac{K_I}{s}$ to the above PD compensator. This leads to the so-called PID control law, as shown in Figure 10.14. The system is now Type 2 and the PID control achieves exact steady tracking of step (and ramp) inputs while rejecting step disturbances, provided of course that the closed loop system is stable.

With the PID compensator

$$C(s) = K_p + K_D s + \frac{K_I}{s} \tag{10.44}$$

the closed loop system is now the third order system

$$\Theta_m(s) = \frac{(K_D s^2 + K_p s + K_I)}{\Omega_2(s)}\Theta^d(s) - \frac{rs}{\Omega_2(s)}D(s) \tag{10.45}$$

where

$$\Omega_2 = J_{eff}s^3 + (B_{eff} + KK_D)s^2 + KK_p s + KK_I. \tag{10.46}$$

Applying the Routh-Hurwitz criterion to this polynomial, it follows that the closed loop

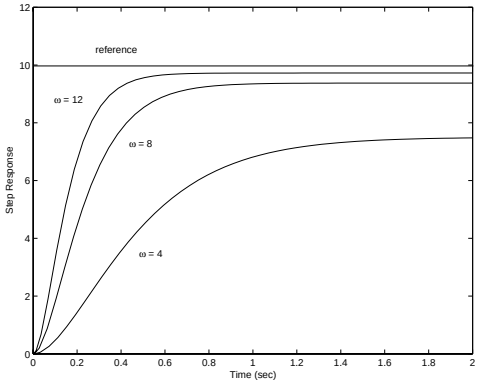


Figure 10.13: Second order system response with disturbance added.

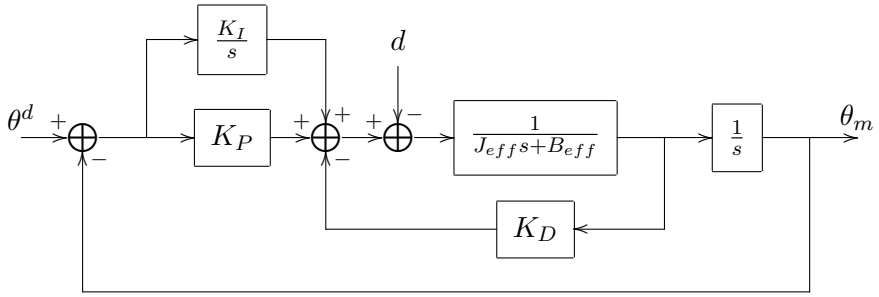


Figure 10.14: Closed loop system with PID control.

system is stable if the gains are positive, and in addition,

$$K_I \quad < \quad \frac{(B_{eff} + KK_D)K_p}{J_{eff}}. \tag{10.47}$$

Example 10.2 *To the system of Example 7.3.1 we have added a disturbance and an integral control term in the compensator. The step responses are shown in Figure 10.17. We see that the steady state error due to the disturbance is removed.* ◇

10.3.4 Saturation

In theory, one could achieve arbitrarily fast response and arbitrarily small steady state error to a constant disturbance by simply increasing the gains in the PD or PID compensator. In practice, however, there is a maximum speed of response achievable from the system. Two major factors, heretofore neglected, limit the achievable performance of the system. The first factor, *saturation*, is due to limits on the maximum torque (or current) input. Many manipulators, in fact, incorporate current limiters in the servo-system to prevent damage that might result from overdrawing current. The second effect is flexibility in the

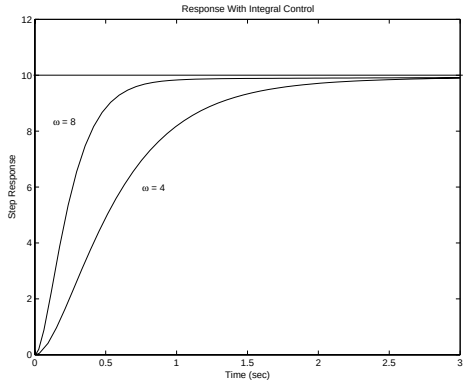


Figure 10.15: Response with integral control action.

motor shaft and/or drive train. We illustrate the effects of saturation below and drive train flexibility in section 10.5.

Example 10.3 *Consider the block diagram of Figure 10.16, where the saturation function*

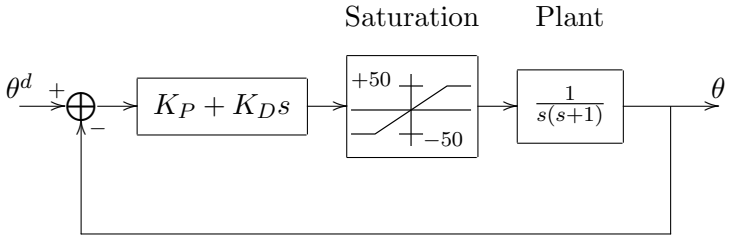


Figure 10.16: Second order system with input saturation.

represents the maximum allowable input. With PID control and saturation the response is below. ◇

The second effect to consider is the joint flexibility. Let k_r be the effective stiffness at the joint. The joint resonant frequency is then $\omega_4 = \sqrt{k_r/J_{eff}}$. It is common engineering practice to limit ω in (10.42) to no more than half of ω_r to avoid excitation of the joint resonance. We will discuss the effects of the joint flexibility in more detail in section 10.5.

These examples clearly show the limitations of PID-control when additional effects such as input saturation, disturbances, and unmodeled dynamics must be considered.

10.4 Feedforward Control and Computed Torque

In this section we introduce the notion of *feedforward control* as a method to track time varying trajectories and reject time varying disturbances.

Suppose that $r(t)$ is an arbitrary reference trajectory and consider the block diagram

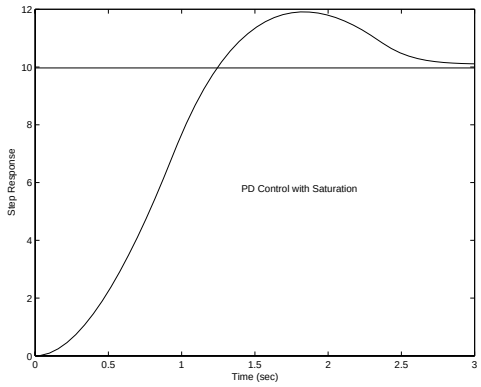


Figure 10.17: Response with Saturation and Integral Control Action

of Figure 10.18, where $G(s)$ represents the forward transfer function of a given system and

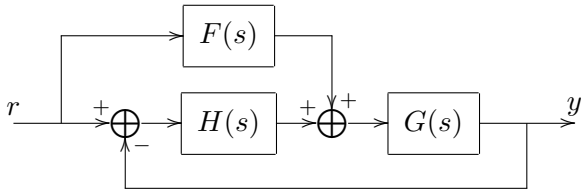


Figure 10.18: Feedforward control scheme.

$H(s)$ is the compensator transfer function. A feedforward control scheme consists of adding a feedforward path with transfer function $F(s)$ as shown.

Let each of the three transfer functions be represented as ratios of polynomials

$$G(s) = \frac{q(s)}{p(s)} \quad H(s) = \frac{c(s)}{d(s)} \quad F(s) = \frac{a(s)}{b(s)} \tag{10.48}$$

We assume that $G(s)$ is strictly proper and $H(s)$ is proper. Simple block diagram manipulation shows that the closed loop transfer function $T(s) = \frac{Y(s)}{R(s)}$ is given by (Problem 7-9)

$$T(s) \quad = \quad \frac{q(s)(c(s)b(s) + a(s)d(s))}{b(s)(p(s)d(s) + q(s)c(s))}. \tag{10.49}$$

The closed loop characteristic polynomial of the system is then $b(s)(p(s)d(s)+q(s)c(s))$. For stability of the closed loop system therefore we require that the compensator $H(s)$ and the feedforward transfer function $F(s)$ be chosen so that the polynomials $p(s)d(s)+q(s)c(s)$ and $b(s)$ are Hurwitz. This says that, in addition to stability of the closed loop system the feedforward transfer function $F(s)$ must itself be stable.

If we choose the feedforward transfer function $F(s)$ equal to $\frac{1}{G(s)'}$ the inverse of the forward plant, that is, $a(s) = p(s)$ and $b(s) = q(s)$, then the closed loop system becomes

$$q(s)(p(s)d(s) + q(s)c(s))Y(s) \quad = \quad q(s)(p(s)d(s) + q(s)c(s))R(s) \tag{10.50}$$

or, in terms of the tracking error $E(s) = R(s) - Y(s)$,

$$q(s)(p(s)d(s) + q(s)c(s))E(s) \quad = \quad 0. \tag{10.51}$$

Thus, assuming stability, the output $y(t)$ will track any reference trajectory $r(t)$. Note that we can only choose $F(s)$ in this manner provided that the numerator polynomial $q(s)$ of the forward plant is Hurwitz, that is, as long as all zeros of the forward plant are in the left half plane. Such systems are called *minimum phase*.

If there is a disturbance $D(s)$ entering the system as shown in Figure 10.19, then it is

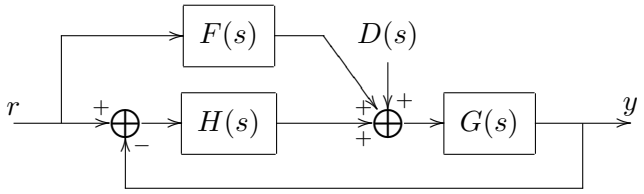


Figure 10.19: Feedforward control with disturbance.

easily shown that the tracking error $E(s)$ is given by

$$E(s) \quad = \quad \frac{q(s)d(s)}{p(s)d(s) + q(s)c(s)} D(s). \tag{10.52}$$

We have thus shown that, in the absence of disturbances the closed loop system will track *any* desired trajectory $r(t)$ provided that the closed loop system is stable. The steady state error is thus due only to the disturbance.

Let us apply this idea to the robot model of Section **??**. Suppose that $\theta^d(t)$ is an arbitrary trajectory that we wish the system to track. In this case we have from (10.31) $G(s) = \frac{K}{J_{eff}s^2 + B_{eff}s}$ together with a PD compensator $H(s) = K_p + K_D s$. The resulting system is shown in Figure 10.20. Note that $G(s)$ has no zeros at all and hence is minimum phase.

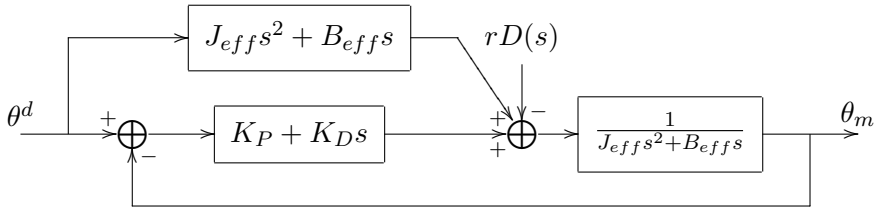


Figure 10.20: Feedforward compensator for second order system.

Note also that $G(s)^{-1}$ is not a proper rational function. However, since the derivatives of the reference trajectory θ^d are known and precomputed, the implementation of the above scheme does not require differentiation of an actual signal. It is easy to see from (10.52) that the steady state error to a step disturbance is now given by the same expression (10.39) independent of the reference trajectory. As before, a PID compensator would result in zero

steady state error to a step disturbance. In the time domain the control law of Figure 10.20 can be written as

$$V(t) = \frac{J_{eff}}{K}\ddot{\theta}^d + \frac{B_{eff}}{K}\dot{\theta}^d + K_D(\dot{\theta}^d - \dot{\theta}_m) + K_p(\theta^d - \theta_m) \quad (10.53)$$

$$= f(t) + K_D\dot{e}(t) + K_p e(t) \quad (10.54)$$

where $f(t)$ is the feedforward signal

$$f(t) = \frac{J_{eff}}{K}\ddot{\theta}^d + \frac{B_{eff}}{K}\dot{\theta}^d \quad (10.55)$$

and $e(t)$ is the tracking error $\theta^d(t) - \theta(t)$. Since the forward plant equation is

$$J_{eff}\ddot{\theta}_m + B_{eff}\dot{\theta}_m = KV(t) - rd(t)$$

the closed loop error $e(t) = \theta_m - \theta^d$ satisfies the second order differential equation

$$J_{eff}\ddot{e} + (B_{eff} + KK_D)\dot{e} + KK_p e(t) = -rd(t). \quad (10.56)$$

Remark 7.6.1

We note from (10.56) that the characteristic polynomial of the closed loop system is identical to (10.34). The system now however is written in terms of the tracking error$e(t)$. Therefore, assuming that the closed loop system is stable, the tracking error will approach zero asymptotically for any desired joint space trajectory in the absence of disturbances, that is, if $d = 0$.

Computed Torque Disturbance Cancellation

We see that the feedforward signal (10.55) results in asymptotic tracking of any trajectory in the absence of disturbances but does not otherwise improve the disturbance rejection properties of the system. However, although the term $\boldsymbol{d}(t)$ in (10.56) represents a disturbance, it is not completely unknown since $\boldsymbol{d}$ satisfies (10.29). Thus we may consider adding to the above feedforward signal, a term to anticipate the effects of the disturbance $\boldsymbol{d}(t)$. Consider the diagram of Figure 10.21. Given a desired trajectory, then we superimpose, as shown, the term

$$d^d := \sum d_{jk}(\boldsymbol{q}^d)\ddot{q}_j^d + \sum c_{ijk}(\boldsymbol{q}^d)\dot{q}_i^d\dot{q}_j^d + g_k(\boldsymbol{q}^d) \quad (10.57)$$

since d^d has units of torque, the above feedforward disturbance cancellation control is called the *method of computed torque*. The expression (10.57) thus compensates in a feedforward manner the nonlinear coupling inertial, coriolis, centripetal, and gravitational forces arising due to the motion of the manipulator. Although the difference $\Delta d := d^d - d$ is zero only in the ideal case of perfect tracking ($\theta = \theta^d$) and perfect computation of (10.57), in practice, Δd can be expected to be smaller than d and hence the computed torque has

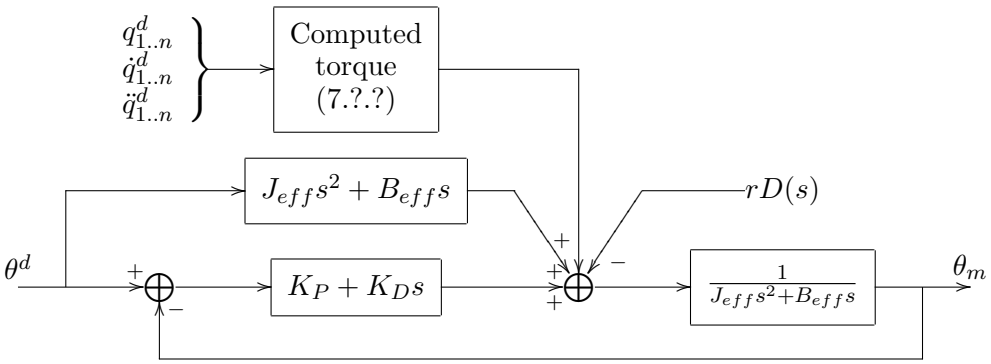


Figure 10.21: Feedforward computed torque compensation.

the advantage of reducing the effects of d. Note that the expression (10.57) is in general extremely complicated so that the computational burden involved in computing (10.57) is of major concern. In fact the problem of real-time implementation of this computed torque control has stimulated a great deal of research. The development of efficient recursive formulations of manipulator dynamics, such as the recursive Newton-Euler equations of Chapter **??**, was partly motivated by the need to compute the expression (10.57) in real-time.

Since only the values of the desired trajectory need to be known, many of these terms can be precomputed and stored off-line. Thus there is a trade-off between memory requirements and on-line computational requirements. This has led to the development of table look up schemes to implement (10.57) and also to the development of computer programs for the automatic generation and simplification of manipulator dynamic equations.

10.5 Drive Train Dynamics

In this section we discuss in more detail the problem of joint flexibility. For many manipulators, particularly those using harmonic drives[1] for torque transmission, the joint flexibility is significant. In addition to torsional flexibility in the gears, joint flexibility is caused by effects such as shaft windup, bearing deformation, and compressibility of the hydraulic fluid in hydraulic robots.

Consider the idealized situation of Figure 10.22 consisting of an actuator connected to a load through a torsional spring which represents the joint flexibility. For simplicity we take the motor torque u as input. The equations of motion are easily derived using the techniques of Chapter **??**, with generalized coordinates θ_ℓ and θ_m, the link angle, and the

[1]Harmonic drives are a type of gear mechanism that are very popular for use in robots due to their low backlash, high torque transmission and compact size. However, they also introduce unwanted friction and flexibility at the joints.

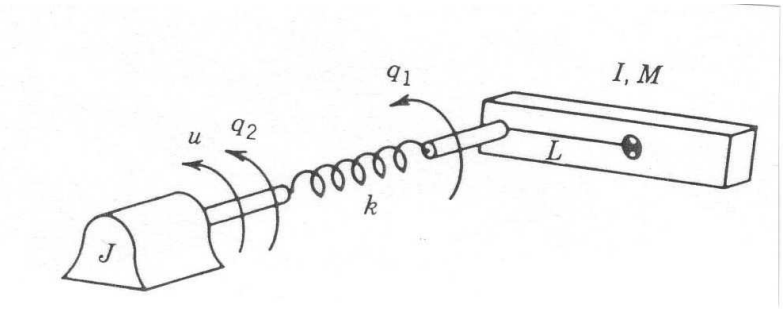


Figure 10.22: Idealized model to represent joint flexibility.

motor angle, respectively, as

$$J_\ell \ddot{\theta}_\ell + B_\ell \dot{\theta}_\ell + k(\theta_\ell - \theta_m) = 0 \tag{10.58}$$

$$J_m \ddot{\theta}_m + B_m \dot{\theta}_m - k(\theta_\ell - \theta_m) = u \tag{10.59}$$

where J_ℓ, J_m are the load and motor inertias, B_ℓ and B_m are the load and motor damping constants, and u is the input torque applied to the motor shaft. In the Laplace domain we can write this as

$$p_\ell(s)\Theta_\ell(s) = k\Theta_m(s) \tag{10.60}$$

$$p_m(s)\Theta_m(s) = k\Theta_\ell(s) + U(s) \tag{10.61}$$

where

$$p_\ell(s) = J_\ell s^2 + B_\ell s + k \tag{10.62}$$

$$p_m(s) = J_m s^2 + B_m s + k. \tag{10.63}$$

This system is represented by the block diagram of Figure 10.23.

The output to be controlled is, of course, the load angle θ_ℓ. The open loop transfer function between U and Θ_ℓ is given by

$$\frac{\Theta_\ell(s)}{U(s)} = \frac{k}{p_\ell(s)p_m(s) - k^2}. \tag{10.64}$$

The open loop characteristic polynomial is

$$J_\ell J_m s^4 + (J_\ell B_m + J_m B_\ell)s^3 + (k(J_\ell + J_m) + B_\ell B_m)s^2 + k(B_\ell + B_m)s. \tag{10.65}$$

If the damping constants B_ℓ and B_m are neglected, the open loop characteristic polynomial is

$$J_\ell J_m s^4 + k(J_\ell + J_m)s^2 \tag{10.66}$$

Figure 10.23: Block diagram for the system (10.60)-(10.61).

which has a double pole at the origin and a pair of complex conjugate poles at $s = \pm j\omega$ where $\omega^2 = k\left(\frac{1}{J_\ell} + \frac{1}{J_m}\right)$. Assuming that the open loop damping constants B_ℓ and B_m are small, then the open loop poles of the system (10.60)-(10.61) will be in the left half plane near the poles of the undamped system.

Suppose we implement a PD compensator $C(s) = K_p + K_D s$. At this point the analysis depends on whether the position/velocity sensors are placed on the motor shaft or on the load shaft, that is, whether the PD-compensator is a function of the motor variables or the load variables. If the motor variables are measured then the closed loop system is given by the block diagram of Figure 10.24. Set $K_p + K_D s = K_D(s + a)$; $a = K_p/K_D$. The root

Figure 10.24: PD-control with motor angle feedback.

locus for the closed loop system in terms of K_D is shown in Figure 10.25.

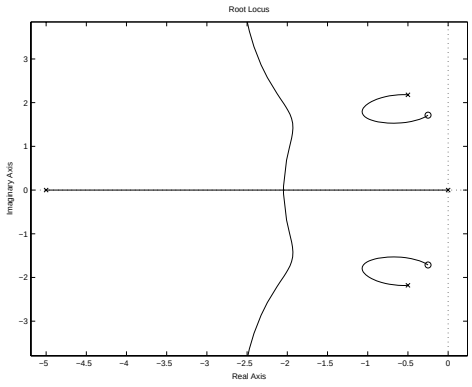


Figure 10.25: Root locus for the system of Figure 10.24.

We see that the system is stable for all values of the gain K_D but that the presence of

the open loop zeros near the $j\omega$ axis may result in poor overall performance, for example, undesirable oscillations with poor settling time. Also the poor relative stability means that disturbances and other unmodeled dynamics could render the system unstable.

If we measure instead the load angle θ_ℓ, the system with PD control is represented by the block diagram of Figure 10.26. The corresponding root locus is shown in Figure 10.27.

Figure 10.26: PD-control with load angle feedback.

In this case the system is unstable for large K_D. The critical value of K_D, that is, the value

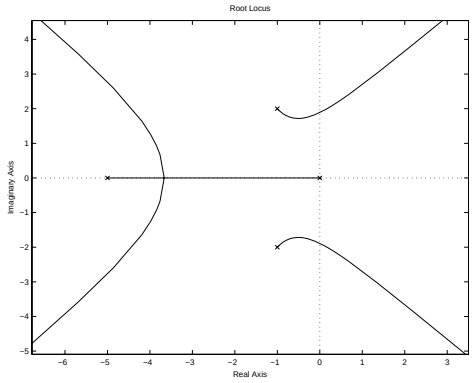


Figure 10.27: Root locus for the system of Figure 7.22.

of K_D for which the system becomes unstable, can be found from the Routh criterion. The best that one can do in this case is to limit the gain K_D so that the closed loop poles remain within the left half plane with a reasonable stability margin.

Example 10.4 *Suppose that the system (10.58)-(10.59) has the following parameters (see*

[1])

$$\begin{aligned} k &= 0.8Nm/rad & B_m &= 0.015Nms/rad \\ J_m &= 0.0004Nms^2/rad & B_\ell &= 0.0Nms/rad \\ J_\ell &= 0.0004Nm^2/rad. \end{aligned} \tag{10.67}$$

If we implement a PD controller $K_D(s+a)$ then the response of the system with motor (respectively, load) feedback is shown in Figure 10.28 (respectively, Figure 10.29). $\diamond$

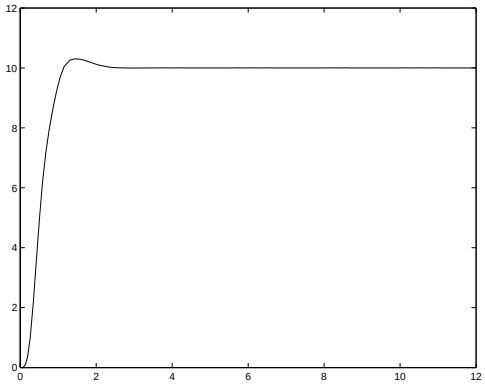


Figure 10.28: Step response – PD-control with motor angle feedback.

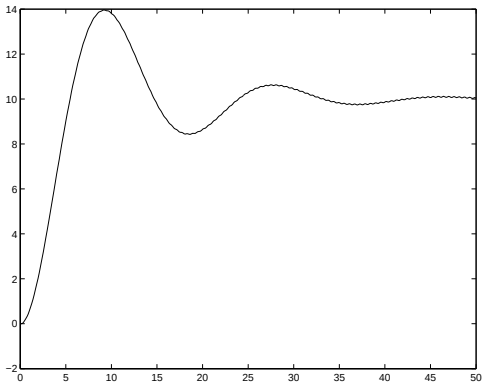


Figure 10.29: Step response – PD control with load angle feedback.

Chapter 11

MULTIVARIABLE CONTROL

11.1 Introduction

In the previous chapter we discussed techniques to derive a control law for each joint of a manipulator based on a single-input/single-output model. Coupling effects among the joints were regarded as disturbances to the individual systems. In reality, the dynamic equations of a robot manipulator form a complex, nonlinear, and multivariable system. In this chapter, therefore, we treat the robot control problem in the context of nonlinear, multivariable control. This approach allows us to provide more rigorous analysis of the performance of control systems, and also allows us to design robust and adaptive nonlinear control laws that guarantee stability and tracking of arbitrary trajectories.

Let us first reformulate the manipulator dynamic equations in a form more suitable for the discussion to follow. Recall the robot equations of motion (10.26) and (10.27)

$$\sum_{j=1}^{n} d_{jk}(\boldsymbol{q})\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i\dot{q}_j + \phi_k = \tau_k \tag{11.1}$$

$$J_{m_k}\ddot{\theta}_{m_k} + B_k\dot{\theta}_{m_k} = K_{m_k}/R_kV_k - \tau_k/r_k. \tag{11.2}$$

where $B_k = B_{m_k} + K_{b_k}K_{m_k}/R_k$. Multiplying (11.2) by r_k and using the fact that

$$\theta_{m_k} = r_kq_k \tag{11.3}$$

we write Equation (11.2) as

$$r_k^2J_m\ddot{q}_k + r_k^2B_k\dot{q}_k = r_kK_{m_k}/RV_k - \tau_k \tag{11.4}$$

Substituting (11.4) into (11.1) yields

$$r_k^2J_{m_k}\ddot{q}_k + \sum_{j-1}^{n} d_{jk}\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}\dot{q}_i\dot{q}_j + r_k^2B_k\dot{q}_k + \phi_k = r_k\frac{K_m}{R}V_k. \tag{11.5}$$

In matrix form these equations of motion can be written as

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + B\dot{\boldsymbol{q}} + g(\boldsymbol{q}) \quad = \quad \boldsymbol{u} \tag{11.6}$$

where $M(\boldsymbol{q}) = D(q) + J$ where J is a diagonal matrix with diagonal elements $r_k^2 J_{m_k}$. The vector $g(\boldsymbol{q})$ and the matrix $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are defined by (9.61) and (9.62), respectively, and the input vector $\boldsymbol{u}$ has components

$$u_k \quad = \quad r_k \frac{K_{m_k}}{R_k} V_k.$$

Note that u_k has units of torque.

Henceforth, we will take $B = 0$ for simplicity in Equation (11.6) and use this equation for all of our subsequent development. We leave it as an exercise for the reader (cf:Problem X) to show that the properties of passivity, skew-symmetry, bounds on the inertia matrix and linearity in the parameters continue to hold for the system (11.6).

11.2 PD Control Revisited

It is rather remarkable that the simple PD-control scheme for set-point control of rigid robots that we discussed in Chapter 10 can be rigorously shown to work in the general case.[1]. An independent joint PD-control scheme can be written in vector form as

$$\boldsymbol{u} \quad = \quad K_P \tilde{\boldsymbol{q}} - K_D \dot{\boldsymbol{q}} \tag{11.7}$$

where $\tilde{\boldsymbol{q}} = \boldsymbol{q}^d - \boldsymbol{q}$ is the error between the desired joint displacements $\boldsymbol{q}^d$ and the actual joint displacements $\boldsymbol{q}$, and K_P, K_D are diagonal matrices of (positive) proportional and derivative gains, respectively. We first show that, in the absence of gravity, that is, if $\boldsymbol{g}$ is zero in (11.6), the PD control law (11.7) achieves asymptotic tracking of the desired joint positions. This, in effect, reproduces the result derived previously, but is more rigorous, in the sense that the nonlinear equations of motion (11.1) are not approximated by a constant disturbance.

To show that the above control law achieves zero steady state error consider the Lyapunov function candidate

$$V \quad = \quad 1/2 \dot{\boldsymbol{q}}^T M(\boldsymbol{q}) \dot{\boldsymbol{q}} + 1/2 \tilde{\boldsymbol{q}}^T K_P \tilde{\boldsymbol{q}}. \tag{11.8}$$

The first term in (11.8) is the kinetic energy of the robot and the second term accounts for the proportional feedback $K_P \tilde{\boldsymbol{q}}$. Note that V represents the total kinetic energy that would result if the joint actuators were to be replaced by springs with stiffnesses represented by K_P and with equilibrium positions at $\boldsymbol{q}^d$. Thus V is a positive function except at the "goal" $\boldsymbol{q} = \boldsymbol{q}^d$, $\dot{\boldsymbol{q}} = 0$, at which point V is zero. The idea is to show that along any motion

[1]The reader should review the discussion on Lyapunov Stability in Appendix C.

of the robot, the function V is decreasing to zero. This will imply that the robot is moving toward the desired goal configuration.

To show this we note that, since J and $\boldsymbol{q}^d$ are constant, the time derivative of V is given by

$$\dot{V} = \dot{\boldsymbol{q}}^T M(\boldsymbol{q})\ddot{\boldsymbol{q}} + 1/2\dot{\boldsymbol{q}}^T \dot{D}(\boldsymbol{q})\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^T K_P \tilde{\boldsymbol{q}}. \tag{11.9}$$

Solving for $M(\boldsymbol{q})\ddot{\boldsymbol{q}}$ in (11.6) with $\boldsymbol{g}(\boldsymbol{q}) = 0$ and substituting the resulting expression into (11.9) yields

$$\begin{aligned} \dot{V} &= \dot{\boldsymbol{q}}^T(\boldsymbol{u} - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}) + 1/2\dot{\boldsymbol{q}}^T \dot{D}(\boldsymbol{q})\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^T K_P \tilde{\boldsymbol{q}} \\ &= \dot{\boldsymbol{q}}^T(\boldsymbol{u} - K_P\tilde{\boldsymbol{q}}) + 1/2\dot{\boldsymbol{q}}^T(\dot{D}(\boldsymbol{q}) - 2C(\boldsymbol{q}, \dot{\boldsymbol{q}}))\dot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{q}}^T(\boldsymbol{u} - K_P\tilde{\boldsymbol{q}}) \end{aligned} \tag{11.10}$$

where in the last equality we have used the fact (Theorem 6.3.1) that $\dot{D} - 2C$ is skew symmetric. Substituting the PD control law (11.7) for $\boldsymbol{u}$ into the above yields

$$\dot{V} = -\dot{\boldsymbol{q}}^T K_D \dot{\boldsymbol{q}} \leq 0. \tag{11.11}$$

The above analysis shows that V is decreasing as long as $\dot{\boldsymbol{q}}$ is not zero. This, by itself is not enough to prove the desired result since it is conceivable that the manipulator can reach a position where $\dot{\boldsymbol{q}} = 0$ but $\boldsymbol{q} \neq \boldsymbol{q}^d$. To show that this cannot happen we can use LaSalle's Theorem (Appendix C). Suppose $\dot{V} \equiv 0$. Then (11.11) implies that $\dot{\boldsymbol{q}} \equiv 0$ and hence $\ddot{\boldsymbol{q}} \equiv 0$. From the equations of motion with PD-control

$$M(q)\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = -K_P\tilde{\boldsymbol{q}} - K_D\dot{\boldsymbol{q}}$$

must then have

$$0 = -K_P\tilde{\boldsymbol{q}}$$

which implies that $\tilde{\boldsymbol{q}} = 0$, $\dot{\boldsymbol{q}} = 0$. LaSalle's Theorem then implies that the equilibrium is asymptotically stable.

In case there are gravitational terms present in (11.6) Equation (11.10) must be modified to read

$$\dot{V} = \dot{\boldsymbol{q}}^T(\boldsymbol{u} - \boldsymbol{g}(\boldsymbol{q}) - K_P\tilde{\boldsymbol{q}}). \tag{11.12}$$

The presence of the gravitational term in (11.12) means that PD control alone cannot guarantee asymptotic tracking. In practice there will be a steady state error or offset. Assuming that the closed loop system is stable the robot configuration $\boldsymbol{q}$ that is achieved will satisfy

$$K_P(\boldsymbol{q}^d - \boldsymbol{q}) = \boldsymbol{g}(\boldsymbol{q}). \tag{11.13}$$

The physical interpretation of (11.13) is that the configuration $\boldsymbol{q}$ must be such that the motor generates a steady state "holding torque" $K_P(\boldsymbol{q}^d - \boldsymbol{q})$ sufficient to balance the gravitational torque $\boldsymbol{g}(\boldsymbol{q})$. Thus we see that the steady state error can be reduced by increasing the position gain K_P.

In order to remove this steady state error we can modify the PD control law as

$$\boldsymbol{u} \;=\; K_P\tilde{\boldsymbol{q}} - K_D\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}). \tag{11.14}$$

The modified control law (11.14), in effect, cancels the effect of the gravitational terms and we achieve the same Equation (11.11) as before. The control law (11.14) requires microprocessor implementation to compute at each instant the gravitational terms $\boldsymbol{g}(\boldsymbol{q})$ from the Lagrangian equations. In the case that these terms are unknown the control law (11.14) cannot be implemented. We will say more about this and related issues later.

11.3 Inverse Dynamics

We now consider the application of more complex nonlinear control techniques for trajectory tracking of rigid manipulators. Consider again the dynamic equations of an n-link robot in matrix form from (11.6)

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \;=\; \boldsymbol{u}. \tag{11.15}$$

The idea of inverse dynamics is to seek a nonlinear feedback control law

$$u \;=\; f(\boldsymbol{q},\dot{\boldsymbol{q}},t) \tag{11.16}$$

which, when substituted into (11.15), results in a linear closed loop system. For general nonlinear systems such a control law may be quite difficult or impossible to find. In the case of the manipulator dynamic equations (11.15), however, the problem is actually easy. By inspecting (11.15) we see that if we choose the control $\boldsymbol{u}$ according to the equation

$$\boldsymbol{u} \;=\; M(\boldsymbol{q})a_q + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{q} + g(q) \tag{11.17}$$

then, since the inertia matrix M is invertible, the combined system (11.15)-(11.17) reduces to

$$\ddot{\boldsymbol{q}} \;=\; a_q \tag{11.18}$$

The term a_q represents a new input to the system which is yet to be chosen. Equation (11.18) is known as the *double integrator system* as it represents n uncoupled double integrators. The nonlinear control law (11.17) is called the *inverse dynamics control*[2] and achieves a rather remarkable result, namely that the "new" system (11.18) is linear, and

[2]We should point out that in the research literature the control law (11.17) is frequently called *computed torque* as well.

decoupled. This means that each input a_{q_k} can be designed to control a scalar linear system. Moreover, assuming that a_{q_k} is a function only of q_k and its derivatives, then a_{q_k} will affect $\boldsymbol{q}_k$ *independently* of the motion of the other links.

Since a_{q_k} can now be designed to control a linear second order system, the obvious choice is to set

$$a_q = -K_0\boldsymbol{q} - K_1\dot{\boldsymbol{q}} + \boldsymbol{r} \tag{11.19}$$

where K_0 and K_1 are diagonal matrices with diagonal elements consisting of position and velocity gains, respectively. The closed loop system is then the linear system

$$\ddot{\boldsymbol{q}} + K_1\dot{\boldsymbol{q}} + K_0\boldsymbol{q} = r. \tag{11.20}$$

Now, given a desired trajectory

$$t \rightarrow (\boldsymbol{q}^d(t), \dot{\boldsymbol{q}}^d(t)). \tag{11.21}$$

if one chooses the reference input $r(t)$ as[3]

$$\boldsymbol{r}(t) = \ddot{\boldsymbol{q}}^d(t) + K_0\boldsymbol{q}^d(t) + K_1\dot{\boldsymbol{q}}^d(t) \tag{11.22}$$

then the tracking error $e(t) = \boldsymbol{q} - \boldsymbol{q}^d$ satisfies

$$\ddot{\boldsymbol{e}}(t) + K_1\boldsymbol{e}(t) + K_0\boldsymbol{e}(t) = 0. \tag{11.23}$$

A simple choice for the gain matrices K_0 and K_1 is

$$\begin{aligned} K_0 &= \text{diag}\{\omega_1^2, \ldots, \omega_n^2\} \\ K_1 &= \text{diag}\{2\omega_1, \ldots, 2\omega_n\} \end{aligned} \tag{11.24}$$

which results in a closed loop system which is globally decoupled, with each joint response equal to the response of a critically damped linear second order system with natural frequency ω_i. As before, the natural frequency ω_i determines the speed of response of the joint, or equivalently, the rate of decay of the tracking error.

The inverse dynamics approach is extremely important as a basis for control of robot manipulators and it is worthwhile trying to see it from alternative viewpoints. We can give a second interpretation of the control law (11.17) as follows. Consider again the manipulator dynamic equations (11.15). Since $M(\boldsymbol{q})$ is invertible for $\boldsymbol{q} \in \mathbb{R}^n$ we may solve for the acceleration $\ddot{\boldsymbol{q}}$ of the manipulator as

$$\ddot{\boldsymbol{q}} = M^{-1}\{\boldsymbol{u} - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{q} - g(q)\}. \tag{11.25}$$

Suppose we were able to specify the acceleration as the input to the system. That is, suppose we had actuators capable of producing directly a commanded acceleration (rather

[3]Compare this with the feedforward expression (10.53).

than indirectly by producing a force or torque). Then the dynamics of the manipulator, which is after all a position control device, would be given as

$$\ddot{\boldsymbol{q}}(t) \quad = \quad a_q(t) \tag{11.26}$$

where $a_q(t)$ is the input acceleration vector. This is again the familiar double integrator system. Note that (11.26) is not an approximation in any sense; rather it represents the actual open loop dynamics of the system provided that the acceleration is chosen as the input. The control problem for the system (11.26) is now easy and the acceleration input a_q can be chosen as before according to (11.19).

In reality, however, such "acceleration actuators" are not available to us and we must be content with the ability to produce a generalized force (torque) u_i at each joint i. Comparing equations (11.25) and (11.26) we see that the torque $\boldsymbol{u}$ and the acceleration a_q of the manipulator are related by

$$M^{-1}\{\boldsymbol{u}(t) - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{q} - g(q)\} \quad = \quad a_q \tag{11.27}$$

By the invertibility of the inertia matrix we may solve for the input torque $\boldsymbol{u}(t)$ as

$$\boldsymbol{u} \quad = \quad M(\boldsymbol{q})a_q + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{q} + g(q) \tag{11.28}$$

which is the same as the previously derived expression (11.17). Thus the inverse dynamics can be viewed as an input transformation which transforms the problem from one of choosing torque input commands, which is difficult, to one of choosing acceleration input commands, which is easy.

Note that the implementation of this control scheme requires the computation at each sample instant of the inertia matrix $M(\boldsymbol{q})$ and the vector of Coriolis, centrifugal, and gravitational. Unlike the computed torque scheme (10.56), however, the inverse dynamics *must* be computed on-line. In other words, as a feedback control law, it cannot be precomputed off-line and stored as can the computed torque (10.57). An important issue therefore in the control system implementation is the design of the computer architecture for the above computations. As processing power continues to increase the computational issues of real-time implementation become less important. An attractive method to implement this scheme is to use a dedicated hardware interface, such as a DSP chip, to perform the required computations in real time. Such a scheme is shown in Figure 11.1.

Figure 11.1 illustrates the notion of *inner-loop/outer-loop* control. By this we mean that the computation of the nonlinear control (11.17) is performed in an inner loop, perhaps with a dedicated hardwire interface, with the vectors $\boldsymbol{q}, \dot{\boldsymbol{q}}$, and a_q as its inputs and $\boldsymbol{u}$ as output. The outer loop in the system is then the computation of the additional input term a_q. Note that the outer loop control a_q is more in line with the notion of a feedback control in the usual sense of being error driven. The design of the outer loop feedback control is in theory greatly simplified since it is designed for the plant represented by the dotted lines in Figure 11.1, which is now a linear or nearly linear system.

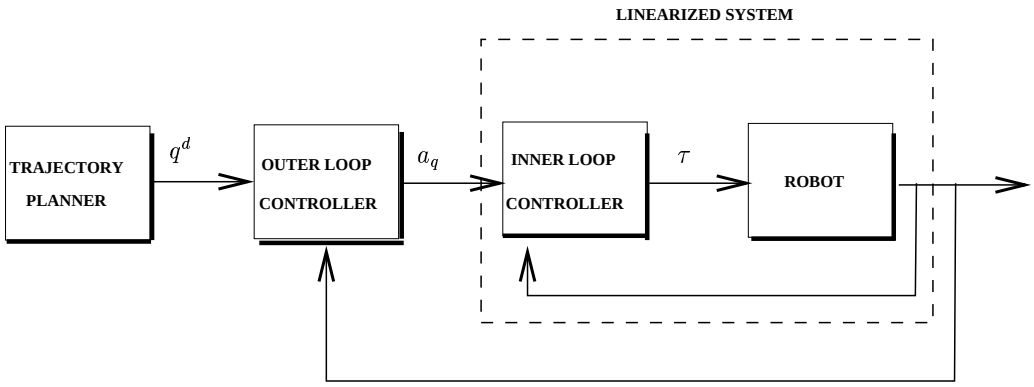


Figure 11.1: Inner loop/outer control architecture.

11.3.1 Task Space Inverse Dynamics

As an illustration of the importance of the inner loop/outer loop paradigm, we will show that tracking in task space can be achieved by modifying our choice of outer loop control $\ddot{q}$ in (11.18) while leaving the inner loop control unchanged. Let $X \in R^6$ represent the end-effector pose using any minimal representation of $SO(3)$. Since X is a function of the joint variables $q \in \mathcal{C}$ we have

$$\dot{X} = J(q)\dot{q} \tag{11.29}$$

$$\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q}. \tag{11.30}$$

where $J = J_a$ is the analytical Jacobian of section 5.8. Given the double integrator system, (11.18), in joint space we see that if a_q is chosen as

$$a_q = J^{-1}\left\{a_X - \dot{J}\dot{q}\right\} \tag{11.31}$$

the result is a double integrator system in task space coordinates

$$\ddot{X} = a_X \tag{11.32}$$

Given a task space trajectory $X^d(t)$, satisfying the same smoothness and boundedness assumptions as the joint space trajectory $q^d(t)$, we may choose a_X as

$$a_X = \ddot{X}^d + K_P(X^d - X) + K_D(\dot{X}^d - \dot{X}) \tag{11.33}$$

so that the Cartesian space tracking error, $\tilde{X} = X - X^d$, satisfies

$$\ddot{\tilde{X}} + K_D\dot{\tilde{X}} + K_P\tilde{X} = 0. \tag{11.34}$$

Therefore, a modification of the outer loop control achieves a linear and decoupled system directly in the task space coordinates, without the need to compute a joint space trajectory and without the need to modify the nonlinear inner loop control.

Note that we have used a minimal representation for the orientation of the end–effector in order to specify a trajectory $X \in \Re^6$. In general, if the end–effector coordinates are given in $SE(3)$, then the Jacobian J in the above formulation will be the geometric Jacobian J. In this case

$$V = \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \omega \end{pmatrix} = J(q)\dot{q} \tag{11.35}$$

and the outer loop control

$$a_q = J^{-1}(q)\{\begin{pmatrix} a_x \\ a_\omega \end{pmatrix} - \dot{J}(q)\dot{q}\} \tag{11.36}$$

applied to (11.18) results in the system

$$\ddot{x} = a_x \in \Re^3 \tag{11.37}$$

$$\dot{\omega} = a_\omega \in \Re^3 \tag{11.38}$$

$$\dot{R} = S(\omega)R, \quad R \in SO(3), \; S \in so(3). \tag{11.39}$$

Although, in this latter case, the dynamics have not been linearized to a double integrator, the outer loop terms a_v and a_ω may still be used to defined control laws to track end–effector trajectories in $SE(3)$.

In both cases we see that non–singularity of the Jacobian is necessary to implement the outer loop control. If the robot has more or fewer than six joints, then the Jacobians are not square. In this case, other schemes have been developed using, for example, the pseudoinverse in place of the inverse of the Jacobian. See [**?**] for details.

The inverse dynamics control approach has been proposed in a number of different guises, such as *resolved acceleration control* [**?**] and *operational space control* [**?**]. These seemingly distinct approaches have all been shown to be equivalent and may be incorporated into the general framework shown above [**?**].

11.4 Robust and Adaptive Motion Control

A drawback to the implementation of the inverse dynamics control methodology described in the previous section is the requirement that the parameters of the system be known exactly. If the parameters are not known precisely, for example, when the manipulator picks up an unknown load, then the ideal performance of the inverse dynamics controller is no longer guaranteed. This section is concerned with robust and adaptive motion control of manipulators. The goal of both robust and adaptive control to maintain performance in terms of stability, tracking error, or other specifications, despite parametric uncertainty, external disturbances, unmodeled dynamics, or other uncertainties present in the system. In distinguishing between robust control and adaptive control, we follow the commonly accepted notion that a robust controller is a fixed controller, static or dynamic, designed to satisfy performance specifications over a given range of uncertainties whereas an adaptive

controller incorporates some sort of on-line parameter estimation. This distinction is important. For example, in a repetitive motion task the tracking errors produced by a fixed robust controller would tend to be repetitive as well whereas tracking errors produced by an adaptive controller might be expected to decrease over time as the plant and/or control parameters are updated based on runtime information. At the same time, adaptive controllers that perform well in the face of parametric uncertainty may not perform well in the face of other types of uncertainty such as external disturbances or unmodeled dynamics. An understanding of the trade-offs involved is therefore important in deciding whether to employ robust or adaptive control design methods in a given situation.

Many of the fundamental theoretical problems in motion control of robot manipulators were solved during an intense period of research from about the mid-1980's until the early-1990's during which time researchers first began to exploit fundamental structural properties of manipulator dynamics such as feedback linearizability, passivity, multiple time-scale behavior, and other properties that we discuss below.

11.4.1 Robust Feedback Linearization

The feedback linearization approach relies on exact cancellation of nonlinearities in the robot equations of motion. Its practical implementation requires consideration of various sources of uncertainties such as modeling errors, unknown loads, and computation errors. Let us return to the Euler-Lagrange equations of motion

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{11.40}$$

and write the inverse dynamics control input u as

$$u = \hat{M}(q)a_q + \hat{C}(q,\dot{q})\dot{q} + \hat{g}(q) \tag{11.41}$$

where the notation $\hat{(\cdot)}$ represents the computed or nominal value of $(\cdot)$ and indicates that the theoretically exact feedback linearization cannot be achieved in practice due to the uncertainties in the system. The error or mismatch $\tilde{(\cdot)} = (\cdot) - \hat{(\cdot)}$ is a measure of one's knowledge of the system dynamics.

If we now substitute (11.41) into (11.40) we obtain, after some algebra,

$$\ddot{q} = a_q + \eta(q,\dot{q},a_q) \tag{11.42}$$

where

$$\eta \;\; = \;\; M^{-1}(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{g}) \tag{11.43}$$

is called the *Uncertainty*. We note that

$$M^{-1}\tilde{M} = M^{-1}\hat{M} - I =: E \tag{11.44}$$

and so we may decompose η as

$$\eta = Ea_q + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) \tag{11.45}$$

We note that the system (11.42) is still nonlinear and coupled due to the uncertainty $\eta(q, \dot{q}, a_q)$. Thus we have no guarantee that the outer loop control given by Equation (11.19) will satisfy desired tracking performance specifications. In this chapter we discuss several design methods to modify the outer loop control (**??**) to guarantee global convergence of the tracking error for the system (11.42).

Outer Loop Design via Lyapunov's Second Method

There are several approaches to treat the robust feedback linearization problem outlined above. We will discuss only one method, namely the so-called theory of **guaranteed stability of uncertain systems**, which is based on Lyapunov's second method. In this approach we set the outer loop control a_q as

$$\ddot{q} = \ddot{q}^d(t) + K_P(q^d - q) + K_D(\dot{q}^d - \dot{q}) + \delta a \tag{11.46}$$

In terms of the tracking error

$$e = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix} = \begin{bmatrix} q - q^d \\ \dot{q} - \dot{q}^d \end{bmatrix} \tag{11.47}$$

we may write

$$\dot{e} = Ae + B\{\delta a + \eta\} \tag{11.48}$$

where

$$A = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} \; ; \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{11.49}$$

Thus the double integrator is first stabilized by the linear feedback, $-K_P e - K_D \dot{e}$, and δa is an additional control input that must be designed to overcome the potentially destabilizing effect of the uncertainty η. The basic idea is to compute a time–varying scalar bound, $\rho(e, t) \geq 0$, on the uncertainty η, i.e.,

$$||\eta|| \leq \rho(e, t) \tag{11.50}$$

and design the additional input term δa to guarantee ultimate boundedness of the state trajectory $x(t)$ in (11.48).

Returning to our expression for the uncertainty

$$\begin{aligned} \eta &= E\ddot{q} + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) && (11.51) \\ &= E\delta a + E(\ddot{q}^d - K_P e - K_D \dot{e}) + M^{-1}(\tilde{C}\dot{q} + \tilde{g}) && (11.52) \end{aligned}$$

we assume a bound of the form

$$||\eta|| \leq \alpha||\delta a|| + \gamma_1||e|| + \gamma_2||e||^2 + \gamma_3 \tag{11.53}$$

where $\alpha = ||E|| = ||M^{-1}\hat{M} - I||$ and γ_i are nonnegative constants. Assuming for the moment that $||\delta a|| \leq \rho(e, t)$, which must then be checked a posteriori, we have

$$||\eta|| \leq \alpha\rho(e, t) + \gamma_1||e|| + \gamma_2||e||^2 + \gamma_3 =: \rho(e, t) \tag{11.54}$$

which defines ρ as

$$\rho(e,t) = \frac{1}{1-\alpha}(\gamma_1||e|| + \gamma_2||e||^2 + \gamma_3) \tag{11.55}$$

Since K_P and K_D are chosen in so that A in (11.48) is a Hurwitz matrix, we choose $Q > 0$ and let $P > 0$ be the unique symmetric positive definite matrix satisfying the Lyapunov equation,

$$A^T P + PA = -Q. \tag{11.56}$$

Defining the control δa according to

$$\delta a = \begin{cases} -\rho(e,t)\dfrac{B^T Pe}{||B^T Pe||} & ; \quad \text{if} \quad ||B^T Pe|| \neq 0 \\ \\ 0 & ; \quad \text{if} \quad ||B^T Pe|| = 0 \end{cases} \tag{11.57}$$

it follows that the Lyapunov function $V = e^T Pe$ satisfies $\dot{V} \leq 0$ along solution trajectories of the system (11.48). To show this result, we compute

$$\dot{V} \;\; = \;\; -e^T Qe + 2e^T PB\{\delta a + \eta\} \tag{11.58}$$

For simplicity, set $w = B^T Pe$ and consider the second term, $w^T\{\delta a + \eta\}$ in the above expression. If $w = 0$ this term vanishes and for $w \neq 0$, we have

$$\delta a = -\rho\frac{w}{||w||} \tag{11.59}$$

and (11.58) becomes, using the Cauchy-Schwartz inequality,

$$w^T(-\rho\frac{w}{||w||} + \eta) \;\; \leq \;\; -\rho||w|| + ||w||||\eta|| \tag{11.60}$$

$$= \;\; ||w||(-\rho + ||\eta||) \leq 0 \tag{11.61}$$

since $||\eta|| \leq \rho$ and hence

$$\dot{V} < -e^T Qe \tag{11.62}$$

and the result follows. Note that $||\delta a|| \leq \rho$ as required.

Since the above control term δa is discontinuous on the manifold defined by $B^T Pe = 0$, solution trajectories on this manifold are not well defined in the usual sense. One may define solutions in a generalized sense, the so-called **Filipov solutions** [**?**]. A detailed treatment of discontinuous control systems is beyond the scope of this text. In practice, the discontinuity in the control results in the phenomenon of *chattering*, as the control switches rapidly across the manifold $B^T Pe = 0$. One may implement a continuous approximation to the discontinuous control as

$$\delta a = \begin{cases} -\rho(e,t)\dfrac{B^T Pe}{||B^T Pe||} & ; \quad \text{if} \quad ||B^T Pe|| > \epsilon \\ \\ -\dfrac{\rho(e,t)}{\epsilon}B^T Pe & ; \quad \text{if} \quad ||B^T Pe|| \leq \epsilon \end{cases} \tag{11.63}$$

In this case, since the control signal (11.63, a solution to the system (11.48) exists and is *uniformly ultimately bounded* (u.u.b). Ssee Appendix C for the definition of uniform ultimate boundedness.

Theorem 1 *The origin of the system (11.48) is u.u.b. with respect to the set S, defined below, using the continuous control law (11.63).*

Proof: As before, choose $V(e) = e^T P e$ and compute

$$\begin{aligned} \dot{V} &= -e^T Q e + 2w^T(\delta a + \eta) && (11.64) \\ &\leq -e^T Q e + 2w^T(\delta a + \rho \frac{w}{||w||}) && (11.65) \end{aligned}$$

with $||w|| = ||B^T P e||$ as above.

For $||w|| \geq \epsilon$ the argument proceeds as above and $\dot{V} < 0$. For $||w|| \leq \epsilon$ the second term above becomes

$$\begin{aligned} & 2w^T(-\frac{\rho}{\epsilon} w + \rho \frac{w}{||w||}) \\ = \; & -2\frac{\rho}{\epsilon}||w||^2 + 2\rho||w|| \end{aligned}$$

This expression attains a maximum value of $\epsilon\frac{\rho}{2}$ when $||w|| = \frac{\epsilon}{2}$. Thus we have

$$\dot{V} = -e^T Q e + \epsilon\frac{\rho}{2} < 0 \tag{11.66}$$

provided

$$-e^T Q e > \epsilon\frac{\rho}{2} \tag{11.67}$$

Using the relationship

$$\lambda_{min}(Q)||e||^2 \leq e^T Q e \leq \lambda_{max}(Q)||e||^2 \tag{11.68}$$

where $\lambda_{min}(Q)$, $\lambda_{max}(Q)$ denote the minimum and maximum eigenvalues, respectively, of the matrix Q, we have that $\dot{V} < 0$ if

$$\lambda_{min}(Q)||e||^2 \geq \epsilon\frac{\rho}{2} \tag{11.69}$$

or, equivalently

$$||e|| \geq \left(\frac{\epsilon\rho}{2\lambda_{min}(Q)}\right)^{\frac{1}{2}} =: \delta \tag{11.70}$$

Let S_δ denote the smallest level set of V containing $B(\delta)$, the ball of radius δ and let B_r denote the smallest ball containing S_δ. Then all solutions of the closed loop system are u.u.b. with respect to $S := B_r$. The situation is shown in Figure 11.2. All trajectories will eventually enter the ball, B_r; in fact, all trajectories will reach the boundary of S_δ since $\dot{V}$ is negative definite outside of S_δ.

Figure 11.2: Uniform Ultimate Boundedness Set

11.4.2 Passivity Based Robust Control

In this section we derive an alternative robust control algorithm which exploits the passivity and linearity in the parameters of the rigid robot dynamics. This methods are qualitatively different from the previous methods which were based on feedback linearization. In the passivity based approach we modify the inner loop control as

$$u = \hat{M}(q)a + \hat{C}(q,\dot{q})v + \hat{g}(q) - Kr. \tag{11.71}$$

where v, a, and r are given as

$$\begin{aligned} v &= \dot{q}^d - \Lambda\tilde{q} \\ a &= \dot{v} = \ddot{q}^d - \Lambda\dot{\tilde{q}} \\ r &= \dot{q}^d - v = \dot{\tilde{q}} + \Lambda\tilde{q} \end{aligned}$$

with K, Λ diagonal matrices of positive gains. In terms of the linear parametrization of the robot dynamics, the control (11.71) becomes

$$u = Y(q,\dot{q},a,v)\hat{\theta} - Kr \tag{11.72}$$

and the combination of (11.71) with (11.40) yields

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y(\theta - \theta_0). \tag{11.73}$$

Note that, unlike the inverse dynamics control, the modified inner loop control (11.40) does not achieve a linear, decoupled system, even in the known parameter case $\hat{\theta} = \theta$.

In the robust passivity based approach of [**?**], the term $\hat{\theta}$ in (11.72) is chosen as

$$\hat{\theta} = \theta_0 + u \tag{11.74}$$

where θ_0 is a fixed nominal parameter vector and u is an additional control term. The system (11.73) then becomes

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y(a,v,q,\dot{q})(\tilde{\theta} + u) \tag{11.75}$$

where $\tilde{\theta} = \theta_0 - \theta$ is a constant vector and represents the parametric uncertainty in the system. If the uncertainty can be bounded by finding a nonnegative constant, $\rho \geq 0$, such that

$$\|\tilde{\theta}\| = \|\theta_0 - \theta\| \leq \rho, \tag{11.76}$$

then the additional term u can be designed according to the expression,

$$u = \begin{cases} -\rho\dfrac{Y^T r}{||Y^T r||} & ; \quad \text{if} \quad ||Y^T r|| > \epsilon \\ \\ -\dfrac{\rho}{\epsilon}Y^T r & ; \quad \text{if} \quad ||Y^T r|| \leq \epsilon \end{cases} \tag{11.77}$$

The Lyapunov function

$$V = \frac{1}{2} r^T M(q) r + \tilde{q}^T \Lambda K \tilde{q} \tag{11.78}$$

is used to show uniform ultimate boundedness of the tracking error. Calculating $\dot{V}$ yields

$$\begin{aligned} \dot{V} &= r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} && (11.79) \\ &= -r^T K r + 2\tilde{q}^T \Lambda K \dot{\tilde{q}} + \frac{1}{2} r^T (\dot{M} - 2C) r + r^T Y(\tilde{\theta} + u) && (11.80) \end{aligned}$$

Using the passivity property and the definition of r, this reduces to

$$\dot{V} = -\tilde{q}^T \Lambda^T K \Lambda \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} + r^T Y(\tilde{\theta} + u) \tag{11.81}$$

Defining $w = Y^T r$ and

$$Q = \begin{bmatrix} \Lambda^T K \Lambda & 0 \\ 0 & \Lambda K \end{bmatrix} \tag{11.82}$$

and mimicking the argument in the previous section, we have

$$\begin{aligned} \dot{V} &= -e^T Q e + w^T (u + \tilde{\theta}) && (11.83) \\ &= -e^T Q e + w^T (u + \rho \frac{w}{||w||}) && (11.84) \end{aligned}$$

Uniform ultimate boundedness of the tracking error follows with the control u from (11.77) exactly as in the proof of Theorem 1.

Comparing this approach with the approach in the section (11.4.1), we see that finding a constant bound ρ for the constant vector $\tilde{\theta}$ is much simpler than finding a time–varying bound for η in (11.43). The bound ρ in this case depends only on the inertia parameters of the manipulator, while $\rho(x,t)$ in (11.50) depends on the manipulator state vector, the reference trajectory and, in addition, requires some assumptions on the estimated inertia matrix $\hat{M}(q)$.

Example 11.1 *TO BE WORKED OUT* $\diamond$

11.4.3 Passivity Based Adaptive Control

In the adaptive approach the vector $\hat{\theta}$ in (11.73) is taken to be a time-varying estimate of the true parameter vector θ. Combining the control law (11.71) with (11.40) yields

$$M(q)\dot{r} + C(q,\dot{q})r + Kr = Y\tilde{\theta}. \tag{11.85}$$

The parameter estimate $\hat{\theta}$ may be computed using standard methods such as gradient or least squares. For example, using the gradient update law

$$\dot{\hat{\theta}} = -\Gamma^{-1} Y^T(q, \dot{q}, a, v) r \tag{11.86}$$

together with the Lyapunov function

$$V = \frac{1}{2} r^T M(q) r + e^T \Lambda K e + \frac{1}{2} \tilde{\theta}^T \Gamma \tilde{\theta} \tag{11.87}$$

results in global convergence of the tracking errors to zero and boundedness of the parameter estimates since

$$\dot{V} = -e^T \Lambda^T K \Lambda e - \dot{e}^T K \dot{e} + \tilde{\theta}^T \{\Gamma \dot{\hat{\theta}} + Y^T r\}. \tag{11.88}$$

COMPLETE THE PROOF

Example 11.2 *TO BE WORKED OUT* $\diamond$

Chapter 12

FORCE CONTROL

12.1 Introduction

Position control is adequate for tasks such as materials transfer and spot welding where the manipulator is not interacting significantly with objects in the workplace (hereafter referred to as the *environment*). However, tasks such as assembly, grinding, and deburring, which involve extensive contact with the environment, are often better handled by controlling the *forces*[1] of interaction between the manipulator and the environment directly. For example, consider an application where the manipulator is required to wash a window, or to write with a felt tip marker. In both cases a pure position control scheme is unlikely to work. Slight deviations of the end-effector from a planned trajectory would cause the manipulator either to lose contact with the surface or to press too strongly on the surface. For a highly rigid structure such as a robot, a slight position error could lead to extremely large forces of interaction with disastrous consequences (broken window, smashed pen, damaged end-effector, etc.). The above applications are typical in that they involve both force control and trajectory control. In the window washing application, for example, one clearly needs to control the forces normal to the plane of the window and position in the plane of the window.

A force control strategy is one that modifies position trajectories based on the sensed force. There are three main types of sensors for force feedback, *wrist force* sensors, *joint torque* sensors, and *tactile* or hand sensors. A wrist force sensor such as that shown in Figure 12.1 consists of an array of strain gauges and can delineate the three components of the force vector along the three axes of the sensor coordinate frame, and the three components of the torque about these axes. A joint torque sensor consists of strain gauges located on the actuator shaft. Tactile sensors are usually located on the fingers of the gripper and are useful for sensing gripping force and for shape detection. For the purposes of controlling the end-effector/environment interactions, the six-axis wrist sensor usually gives the best results and we shall henceforth assume that the manipulator is equipped with such a device.

[1]Hereafter we use *force* to mean force and/or torque and *position* to mean position and/or orientation.

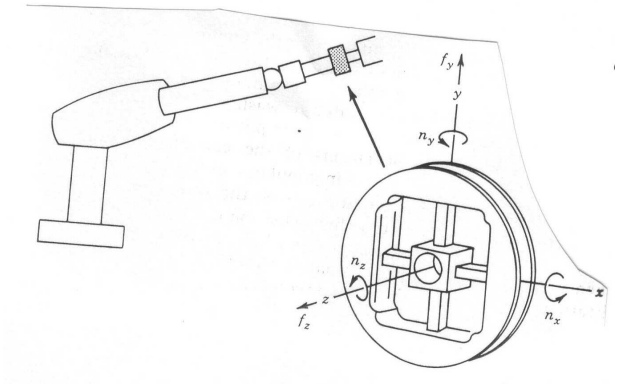


Figure 12.1: A Wrist Force Sensor.

12.2 Constrained Dynamics

Force control tasks can be thought of in terms of constraints imposed by the robot/environment interaction. A manipulator moving through free space within its workspace is unconstrained in motion and can exert no forces since there is no source of reaction force from the environment. A wrist force sensor in such a case would record only the inertial forces due to any acceleration of the end-effector. As soon as the manipulator comes in contact with the environment, say a rigid surface as shown in Figure 12.2, one or more degrees of freedom in motion may be lost since the manipulator cannot move through the environment surface. At the same time the manipulator can exert forces against the environment.

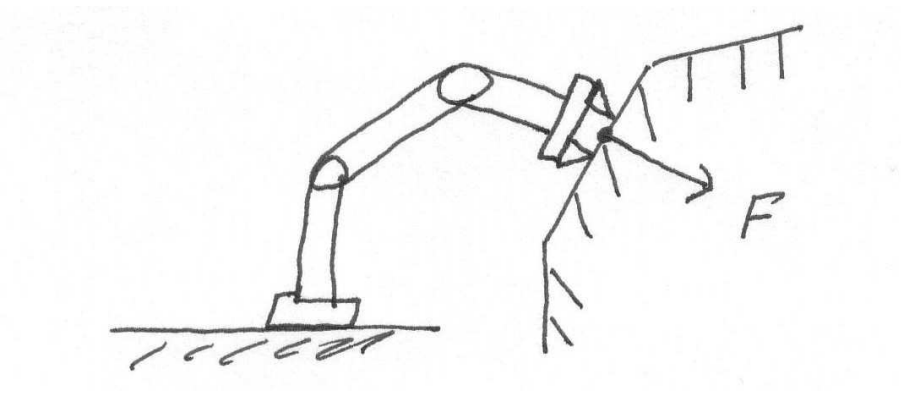


Figure 12.2: Robot End-Effector in contact with a Rigid Surface

In order to describe the robot/environment interaction, let $V = (v, \omega)^T \in \mathrm{so}(3)$ represent the instantaneous linear and angular velocity of the end-effector and let $F = (f, n)^T \in \mathrm{so}^*(3)$ represent the instantaneous force and moment. The notation $\mathrm{so}^*(3)$ refers to the dual space

of so(3).[2].

If $e_1, \ldots, e_6$ is a basis for the vector space so(3), and $f_1, \ldots, f_6$ is a basis for so*(3), we say that these basis vectors are *reciprocal* provided

$$\begin{aligned} e_i f_j &= 0 \quad \text{if } i \neq j \\ e_i f_j &= 1 \quad \text{if } i = j \end{aligned}$$

Definition 12.1 The Reciprocal Product $V \in so(3)$ *and* $F \in so^*(3)$ *is defined as*

$$V \cdot F = V^T F = v^T f + \omega^T n \tag{12.1}$$

Note that Equation (12.1) is not the same as the usual inner product since V and F lie in different vector spaces. The advantage of using reciprocal basis vectors is that the product $V^T F$ is then invariant with respect to a linear change of basis from one reciprocal coordinate system to another. This invariance property would, of course, automatically be satisfied if V and F were vectors in the same vector space. We note that expressions such as $V_1^T V_2$ or $F_1^T F_2$ for vectors V_i, F_i belonging to so(3) and so*(3), respectively, are not well defined. For example, the expression

$$V_1^T V_2 = v_1^T v_2 + \omega_1^T \omega_2 \tag{12.2}$$

is not invariant with respect to either choice of units or basis vectors in so(3). It is possible to define inner product like operations, i.e. symmetric, bilinear forms on so(3) and so*(3), which have the necessary invariance properties. These are the so-called *Klein Form*, $KL(V_1, V_2)$, and *Killing Form*, $KI(V_1, V_2)$, defined according to

$$KL(V_1, V_2) = v_1^T \omega_2 + \omega_1^T v_2 \tag{12.3}$$

$$KI(V_1, V_2) = \omega_1^T \omega_2 \tag{12.4}$$

A detailed discussion of these concepts is beyond the scope of this text. As the reader may suspect, the need for a careful treatment of these concepts is related to the geometry of $SO(3)$ as we have seen before in other contexts.

Example 12.1 *[?] Suppose that*

$$\begin{aligned} V_1 &= (1, 1, 1, 2, 2, 2)^T \\ V_2 &= (2, 2, 2, -1, -1, -1)^T \end{aligned}$$

where the linear velocity is in meters/sec and angular velocity is in radians/sec. The clearly, $V_1^T V_2 = 0$ *and so one could infer that* V_1 *and* V_2 *are orthogonal vectors in so*(3)*. However, suppose now that the linear velocity is represented in units of centimeters/sec. Then*

$$\begin{aligned} V_1 &= (1 \times 10^2, 1 \times 10^2, 1 \times 10^2, 2, 2, 2)^T \\ V_2 &= (2 \times 10^2, 2 \times 10^2, 2 \times 10^2, -1, -1, -1)^T \end{aligned}$$

[2]In more advanced texts, the vectors V and F are called *Twists* and *Wrenches* [**?**] although we will continue to refer to them simply as velocity and force.

and clearly $V_1^T V_2 \neq 0$. *Thus, usual notion of orthogonality is not meaningful in* $so(3)$. *It is easy to show that the equality* $KL(V_1, V_2) = 0$ *(respectively,* $KI(V_1, V_2) = 0$*) is independent of the units or the basis chosen to represent* V_1 *and* V_2. *For example, the condition* $KI(V_1, V_2) = 0$ *means that the axes of rotation defining* ω_1 *and* ω_2 *are orthogonal.* $\diamond$ We will utilize the reciprocal product $V^T F$ in later sections to derive force control strategies.

12.2.1 Static Force/Torque Relationships

Forces at the end-effector due to robot/environment interaction induce torques about the joint axes of the robot. Similarly, torques applied at the joint axes will produce forces at the robot/environment interface. Let τ denote the vector of joint torques induced by an end-effector force, F, and let δX represent a virtual end-effector displacement caused by the force F. Let δq represent the corresponding virtual joint displacement. These virtual displacements are related through the manipulator Jacobian $J(q)$ according to

$$\delta X = J(q)\delta q. \tag{12.5}$$

The virtual work δw of the system is

$$\delta w = F^T \delta X - \tau^T \delta q. \tag{12.6}$$

Substituting (12.5)) into (12.6) yields

$$\delta w = (F^T J - \tau^T)\delta q \tag{12.7}$$

which is equal to zero if the manipulator is in equilibrium. Since the generalized coordinates q are independent we have the equality

$$\tau = J(q)^T F. \tag{12.8}$$

In other words the end-effector forces are related to the joint torques by the *transpose* of the manipulator Jacobian according to (12.8).

Example 12.2 *Consider the two-link planar manipulator of Figure 12.3, with a force* $F = (F_x, F_y)^T$ *applied at the end of link two as shown. The Jacobian of this manipulator is given by Equation (5.93). The resulting joint torques* $\tau = (\tau_1, \tau_2)$ *are then given as*

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} & 0 & 0 & 0 & 1 \\ -a_2 s_{12} & a_2 c_{12} & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \\ n_x \\ n_y \\ n_z \end{bmatrix}. \tag{12.9}$$

$\diamond$

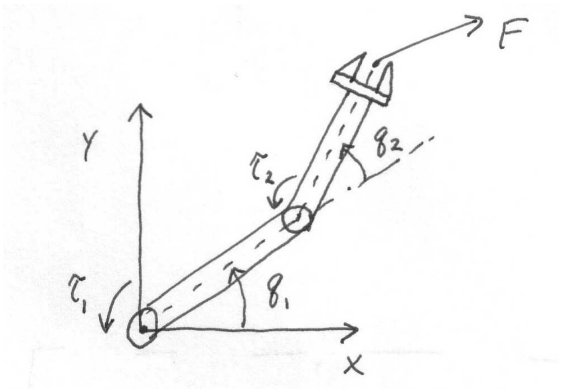


Figure 12.3: Two-link planar robot.

When the manipulator is in contact with the environment, the dynamic equations of Chapter 9 must be modified to include the reaction torque $J^T F_e$ corresponding to the end-effector force F_e. Thus the equations of motion of the manipulator in joint space are given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + J^T(q)F_e \quad = \quad u \tag{12.10}$$

12.2.2 Constraint Surfaces

Let us assume that the force F_e in Equation (12.10) is due to the contact between the end-effector and a rigid environment. Supppose the robot/environment constraint can be expressed in configuration space by $n_f < n$ algebraic equations of the form

$$\phi_q(q) = \begin{bmatrix} \phi_{q_1} \\ \vdots \\ \phi_{q_{nf}} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{12.11}$$

where $\phi_q : \mathbf{R}^n \rightarrow \mathbf{R}^n$ is a continuously differentiable function. Assume that the constraints are independent so that the Jacobian $J_{\phi_q} = \left(\dfrac{\partial \phi_{q_i}}{\partial q_j} \right)$ for $i = 1, \ldots, n_f$, $j = 1, \ldots, n$ has full rank equal to n_f

In the Lagrangian approach to the derivation of the robot dynamics, the standard way to incorporate such constraint functions is to modify the Lagrangian function as

$$\mathcal{L} = \mathcal{K} - \mathcal{P} - \sum_{j=1}^{n_f} \lambda_j \phi_{q_j}(q) \tag{12.12}$$

where $\lambda = [\lambda_1, \ldots, \lambda_{n_f}]^T$ is a vector of *Lagrange Multipliers*. The solution to the Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} \quad = \quad u_k \quad k = 1, \ldots, n \tag{12.13}$$

then results in (Problem **??**)

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + J_{\phi_q}^T(q)\lambda = u \tag{12.14}$$

Comparing Equations (12.10) and (12.14) gives

$$\begin{aligned} J^T F_e &= J_{\phi_q}^T \lambda \\ &= J^T J^{-T} J_{\phi_q}^T \lambda \\ &= J^T J_{\phi_x}^T \lambda \end{aligned} \tag{12.15}$$

where we define

$$J_{\phi_x} = J_{\phi_q} J^{-1} \tag{12.16}$$

Thus, as long as the manipulator Jacobian, $J(q)$, is invertible we may write the environment force, F_e, as

$$F_e = J_{\phi_x}^T \lambda \tag{12.17}$$

Equation (12.17) means that the reaction forces, F_e, belong to an n_f dimensional subspace spanned by the column vectors of J_{ϕ_x}. Let S_f denote a basis for this vector space. Then any constraint force, F_e, may be expressed as

$$F_e = S_f \beta \tag{12.18}$$

where $\beta = [\beta_1, \ldots, \beta_{n_f}]^T$ is a vector of dimensionless scalars. We refer to the vector space spanned by S_f as the *Force Controlled Subspace*.

Example 12.3 *Suppose the robot is constrained to follow the circular surface shown in Figure 12.4 With the coordinates as shown in the Figure, the constraint can be written as*

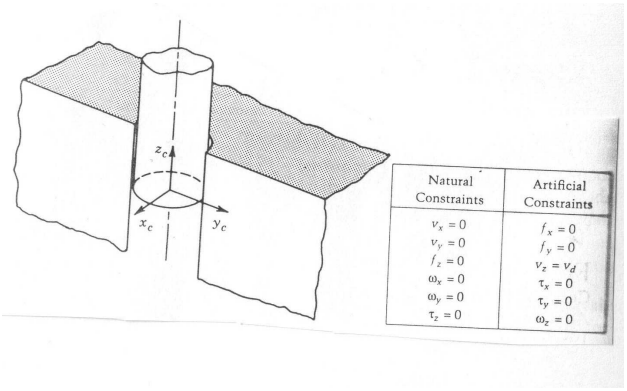


Figure 12.4: End Effector Constrainted to Follow a Circle

$$\phi_x = (x-a)^2 + (y-b)^2 - r^2 = 0 \tag{12.19}$$

or, using the forward kinematic equations,

$$\phi_q = (a_1c_1 + a_2c_{12} - a)^2 + (a_1s_1 + a_2s_{12} - b)^2 - r^2 = 0 \tag{12.20}$$

The Jacobian, J_{ϕ_x}, is therefore given by

$$J_{\phi_x} = [2(x-a), 2(y-b)] \tag{12.21}$$

*It can be verified by direct calculation (Problem **??**) that J_{ϕ_q} is given by*

$$J_{\phi_q} = [2(a_1c_1 + a_2c_{12} - a), 2(a_1s_2 + a_2s_{12} - b)] \begin{bmatrix} -a_2s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \end{bmatrix} \tag{12.22}$$

Therefore

$$J_{\phi_x} = J_{\phi_q} J \tag{12.23}$$

where J is the Jacobian of the two-link planar robot derived in Chapter 1. If we take

$$S_f = \begin{bmatrix} x-a \\ y-b \end{bmatrix} \tag{12.24}$$

then the contact force between the robot and the environment may be expressed as

$$F_e = S_f\beta = \begin{bmatrix} x-a \\ y-b \end{bmatrix} \beta \tag{12.25}$$

where β is an arbitrary scalar. Note that F_e is always directed along the line through the center of the circle, i.e. perpendicular to the tangent to the circle. $\diamond$

We may therefore assume that

$$\mathrm{so}^*(3) = \mathrm{span}(S_f) \oplus \mathrm{span}(S_f)^\perp \tag{12.26}$$

We can similarly describe the end-effector velocity subject to the constraints (12.11). With $\phi_q(q)$ identically zero we have

$$0 = \dot{\phi}_q(q) = \frac{\partial \phi_q}{\partial q}\dot{q} = J_{\phi_q}\dot{q} \tag{12.27}$$

Thus

$$0 = J_{\phi_q}\dot{q} = (J_{\phi_q}J^{-1})J\dot{q} = J_{\phi_x}V \tag{12.28}$$

where $V = (v, \omega)$ is the usual end-effector linear and angular velocity. It follows from (12.17) and (12.28) that the reciprocal product satisfies

$$V^T F_e = F_e^T V = 0 \tag{12.29}$$

Equation (12.29) is called the *Reciprocity Condition*. This means that we can define a set of $n_x = n - n_f$ basis vectors so that

$$V = S_x\gamma, \quad \text{for some } \gamma \tag{12.30}$$

Furthermore, the reciprocity condition (12.29) implies that

$$S_x^T S_f = 0 \tag{12.31}$$

12.2.3 Natural and Artificial Constraints

In this section we discuss so-called *Natural Constraints* which are defined using the reciprocity condition (12.1). We then discuss the notion of *Artificial Constraints*, which are used to define reference inputs for motion and force control tasks.

We begin by defining a so-called *Compliance Frame* $o_c x_c y_c z_c$ (also called a *Constraint Frame*) in which the task to be performed is easily described. For example in the window washing application we can define a frame at the tool with the z_c-axis along the surface normal direction. The task specification would then be expressed in terms of maintaining a constant force in the z_c direction while following a prescribed trajectory in the $x_c - y_c$ plane. Such a position constraint in the z_c direction arising from the presence of a rigid surface is a natural constraint. The force that the robot exerts against the rigid surface in the z_c direction, on the other hand, is not constrained by the environment. A desired force in the z_c direction would then be considered as an artificial constraint that must be maintained by the control system.

Figure 12.5 shows a typical task, that of inserting a peg into a hole. With respect to a compliance frame $o_c x_c y_c z_c$ as shown at the end of the peg, we may take the the standard orthonormal basis in $\Re^6$ for both so(3) and so*(3), in which case

$$V^T F = v_x f_x + v_y f_y + v_z f_z + \omega_x n_x + \omega_y n_y + \omega_z n_z \tag{12.32}$$

If we assume that the walls of the hole and the peg are perfectly rigid and there is no friction, it is easy to see that

$$\begin{array}{lll} v_x = 0 & v_y = 0 & f_z = 0 \\ \omega_x = 0 & \omega_y = 0 & n_z = 0 \end{array} \tag{12.33}$$

and thus the reciprocity condition $V^T F = 0$ is satisfied. These relationships (12.33) are termed *Natural Constraints*.

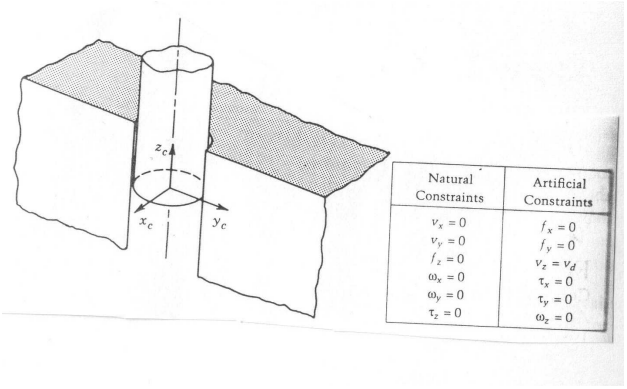


Figure 12.5: Inserting a peg into a hole.

Examining Equation (12.32) we see that the variables

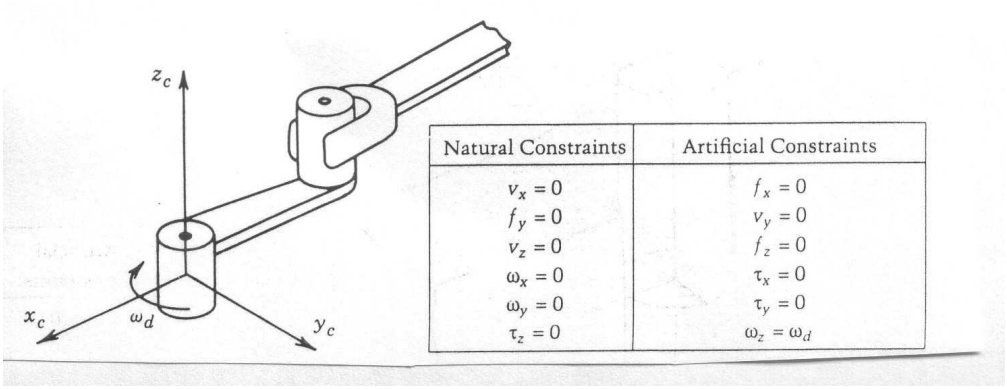
$$f_x \quad f_y \quad v_z \quad n_x \quad n_y \quad \omega_z \tag{12.34}$$

are unconstrained by the environment, i.e. given the natural constraints (12.33), the reciprocity condition $V^T F = 0$ holds for all values of the above variables (12.34). We may therefore assign reference values, called *Artificial Constraints*, for these variables that may then be enforced by the control system to carry out the task at hand. For example, in the peg-in-hole task we may define artificial constraints as

$$\begin{array}{lll} f_x = 0 & f_y = 0 & v_z = v^d \\ n_x = 0 & n_y = 0 & \omega_z = 0 \end{array} \tag{12.35}$$

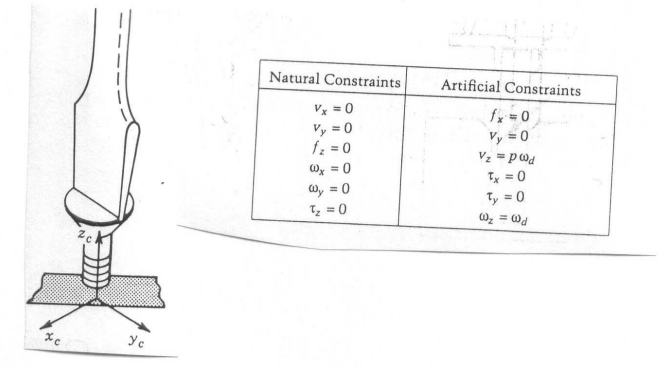
where v^d is the desired speed of insertion of the peg in the z-direction.

Figures 12.6 and 12.7 show natural and artificial constraints for two additional tasks, that of turning a crank and and turning a screw, respectively.



Natural Constraints	Artificial Constraints
$v_x = 0$	$f_x = 0$
$f_y = 0$	$v_y = 0$
$v_z = 0$	$f_z = 0$
$\omega_x = 0$	$\tau_x = 0$
$\omega_y = 0$	$\tau_y = 0$
$\tau_z = 0$	$\omega_z = \omega_d$

Figure 12.6: Turning a crank



Natural Constraints	Artificial Constraints
$v_x = 0$	$f_x = 0$
$v_y = 0$	$v_y = 0$
$f_z = 0$	$v_z = p\,\omega_d$
$\omega_x = 0$	$\tau_x = 0$
$\omega_y = 0$	$\tau_y = 0$
$\tau_z = 0$	$\omega_z = \omega_d$

Figure 12.7: Turning a screw.

12.3 Network Models and Impedance

The reciprocity condition $V^T F = 0$ means that the forces of constraint do no work in directions compatible with motion constraints and holds under the ideal conditions of no friction and perfect rigidity of both the robot and environment. In practice, compliance and friction present at the robot/environment interface will alter the strict separation between motion constraints and force constraints. For example, consider the situation in Figure 12.8. Since the environment deforms in response to a force there is clearly both motion and force

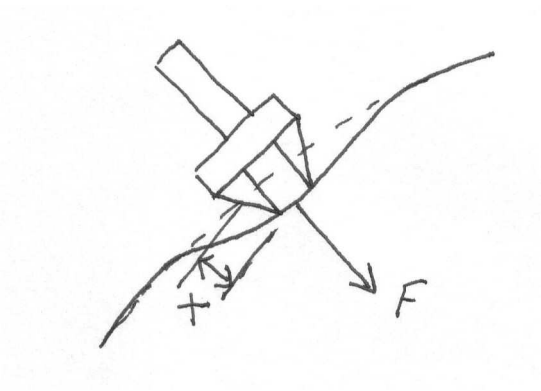


Figure 12.8: Compliant Environment

in the same direction, i.e. normal to the surface. Thus the product $V(t)F(t)$ along this direction will not be zero. Let k represent the stiffness of the surface so that $F = kx$. Then

$$\int_0^t V(u)F(u)du = \int_0^t \dot{x}(u)kx(u)du = k\int_0^t \frac{d}{du}\frac{1}{2}kx^2(u)du = \frac{1}{2}k(x^2(t) - x^2(0)) \quad (12.36)$$

is the change of the potential energy. The environment stiffness, k, determines the amount of force needed to produce a given motion. The higher the value of k the more the environment "impedes" the motion of the end-effector.

In this section we introduce the notion of *Mechanical Impedance* which captures the relation between force and motion. We introduce so-called *Network Models*, which are particularly useful for modeling the robot/environment interaction. We model the robot and environment as *One Port Networks* as shown in Figure 12.9. The dynamics of the robot and environment, respectively, determine the relation between the *Port Variables*, V_r, F_r, and V_e, F_e, respectively. F_r, F_e are known as *Effort* or *Across* variables while V_r, V_e are known as *Flow* or *Through* variables. In a mechanical system, such as a robot, force and velocity are the effort and flow variables while in an electrical system, voltage and current are the effort and flow variables, respectively. With this description, the "product" of the port variables, $V^T F$, represents instantaneous power and the integral of this product

$$\int_0^t V^T(\sigma)F(\sigma)d\sigma$$

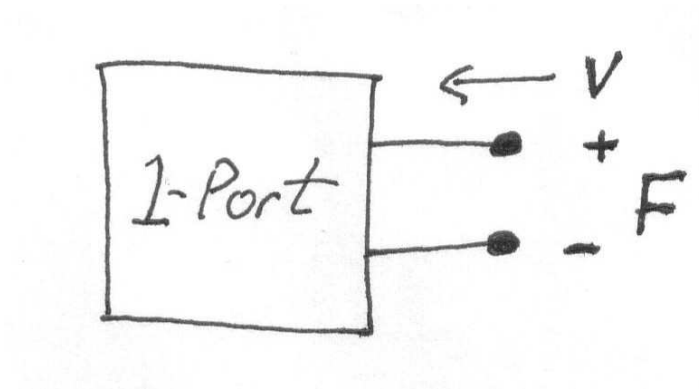


Figure 12.9: One-Port Networks

is the *Energy* dissipated by the Network over the time interval $[0, t]$.

The robot and the environment are then coupled through their interaction ports, as shown in Figure 12.10, which describes the energy exchange between the robot and the environment.

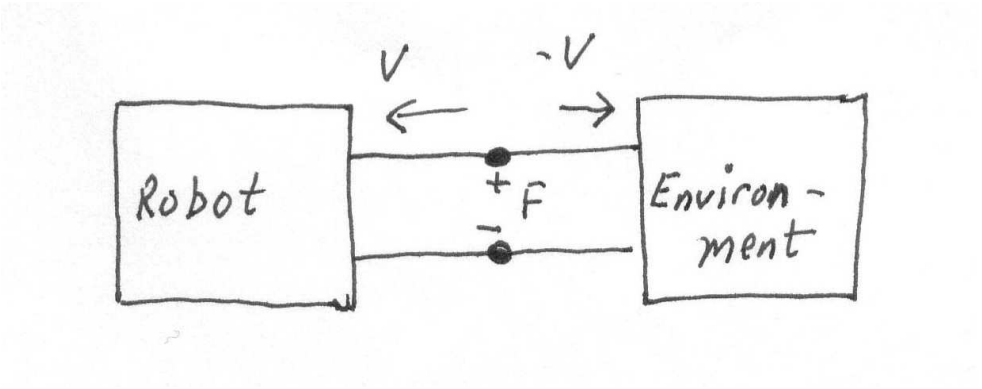


Figure 12.10: Robot/Environment Interaction

12.3.1 Impedance Operators

The relationship between the effort and flow variables may be described in terms of an *Impedance Operator*. For linear, time invariant systems, we may utilize the s-domain or Laplace domain to define the Impedance.

Definition 12.2 *Given the one-port network 12.9 the Impedance, $Z(s)$ is defined as the ratio of the Laplace Transform of the effort to the Laplace Transform of the flow, i.e.*

$$Z(s) = \frac{F(s)}{V(s)} \tag{12.37}$$

Example 12.4 *Suppose a mass-spring-damper system is described by the differential equation*

$$M\ddot{x} + B\dot{x} + Kx = F \tag{12.38}$$

Taking Laplace Transforms of both sides (assuming zero initial conditions) it follows that

$$Z(s) = F(s)/V(s) = Ms + B + K/s \tag{12.39}$$

◇

12.3.2 Classification of Impedance Operators

Definition 12.3 *An impedance $Z(s)$ in the Laplace variable s is said to be*

1. *Inertial if and only if $|Z(0)| = 0$*
2. *Resistive if and only if $|Z(0)| = B$ for some constant $0 < B < \infty$*
3. *Capacitive if and only if $|Z(0)| = \infty$*

Thus we classify impedance operators based on their low frequency or *DC-gain*, which will prove useful in the steady state analysis to follow.

Example 12.5 *Figure 12.11 shows examples of environment types. Figure 12.11(a) shows a mass on a frictionless surface. The impedance is $Z(s) = Ms$, which is inertial. Figure 12.11(b) shows a mass moving in a viscous medium with resistance B. Then $Z(s) = Ms + B$, which is resistive. Figure 12.11(c) shows a linear spring with stiffness K. Then $Z(s) = K/s$, which is capacitive.* ◇

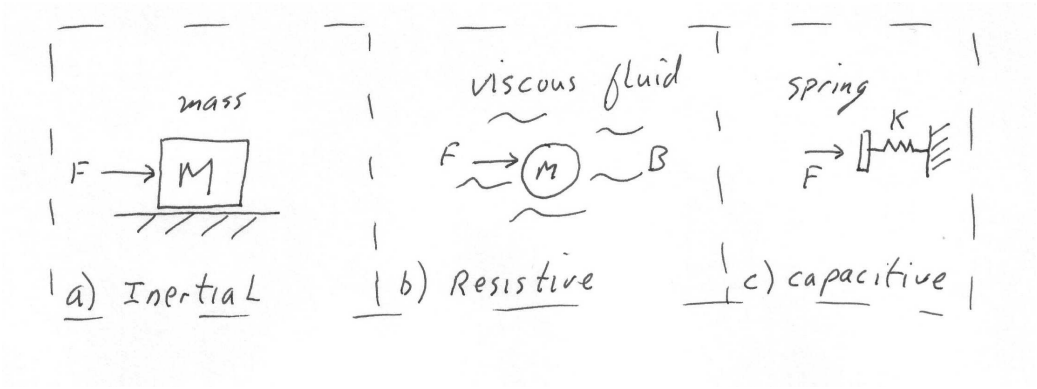


Figure 12.11: Inertial, Resistive, and Capacitive Environment Examples

12.3.3 Thévenin and Norton Equivalents

In linear circuit theory it is common to use so-called *Thévenin* and *Norton* equivalent circuits for analysis and design. It is easy to show that any one-port network consisting of passive elements (resistors, capacitors, inductors) and active current or voltage sources can be represented either as an impedance, $Z(s)$, in series with an effort source (Thévenin Equivalent) or as an impedance, $Z(s)$, in parallel with a flow source (Norton Equivalent). The independent sources, F_s and V_s may be used to represent reference signal generators

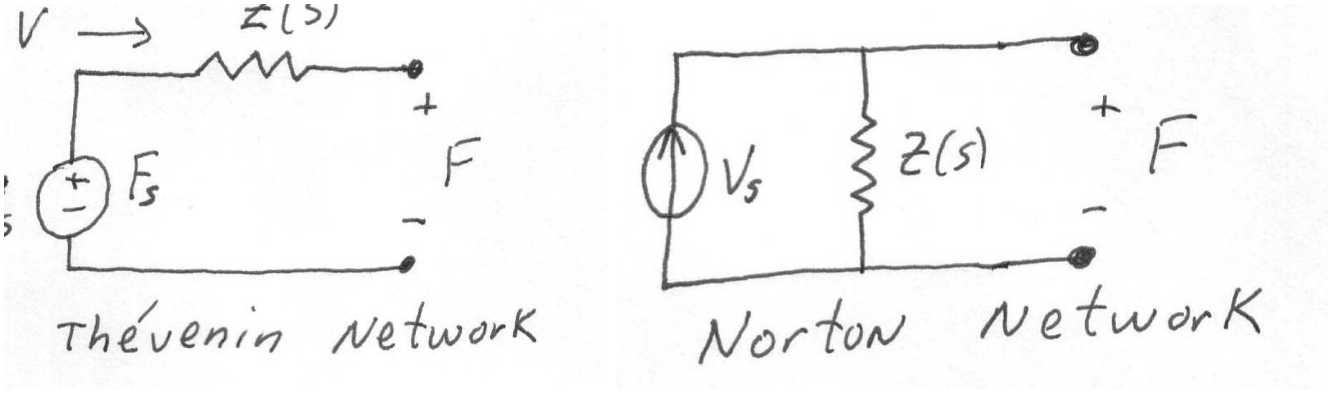


Figure 12.12: Thévenin and Norton Equivalent Networks

for force and velocity, respectively, or they may represent external disturbances.

12.4 Force Control Strategies

Let us consider a modified inverse dynamics control law of the form

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q) + J^T(q)a_f \tag{12.40}$$

where a_q and a_f are outer look controls with units of acceleration and force, respectively. Using the relationship between joint space and task space variables derived in Chapter 11

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{12.41}$$

$$a_x = J(q)a_q + \dot{J}(q)\dot{q} \tag{12.42}$$

we substitute (12.40)-(12.42) into (12.10) to obtain

$$\ddot{x} = a_x + W(q)(F_e - a_f) \tag{12.43}$$

where $W(q) = J(q)M^{-1}(q)J^T(q)$ is called the *Mobility Tensor*. There is often a conceptual advantage to separate the position and force control terms by assuming that a_x is a function only of position and velocity and a_f is a function only of force [**?**]. However, for simplicity,

we shall take $a_f = F_e$ to cancel the environment force, F_e and thus recover the task space double integrator system

$$\ddot{x} = a_x \tag{12.44}$$

and we will assume that any additional force feedback terms are included in the outer loop term a_x. This entails no loss of generality as long as the Jacobian (hence $W(q)$) is invertible. This will become clear in the sequel.

12.4.1 Impedance Control

In this section we discuss the notion of *Impedance Control*. We begin with an example that illustrates in a simple way the effect of force feedback

Example 12.6 *Consider the one-dimensional system in Figure 12.13 consisting of a mass,*

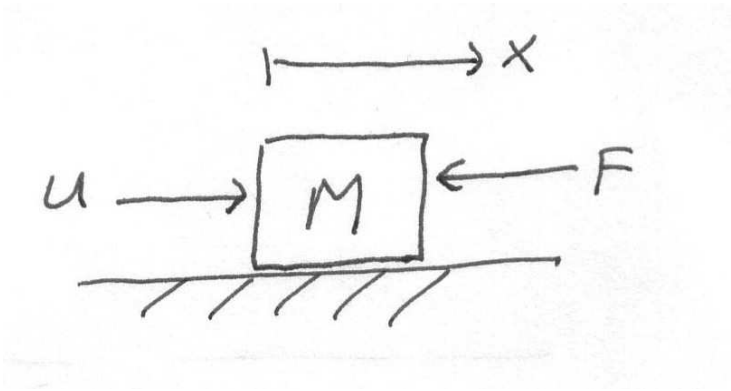


Figure 12.13: One Dimensional System

M, on a frictionless surface subject to an environmental force F and control input u. The equation of motion of the system is

$$M\ddot{x} = u - F \tag{12.45}$$

With $u = 0$, the object "appears to the environment" as a pure inertia with mass M. Suppose the control input u is chosen as a force feedback term $u = -mF$. Then the closed loop system is

$$M\ddot{x} = -(1+m)F \implies \frac{M}{1+m}\ddot{x} = -F \tag{12.46}$$

Hence, the object now appears to the environment as an inertia with mass $\frac{M}{1+m}$. Thus the force feedback has the effect of changing the apparent inertia *of the system.* $\diamond$

The idea behind Impedance Control is to regulate the mechanical impedance, i.e., the apparent inertia, damping, and stiffness, through force feedback as in the above example.

For example, in a grinding operation, it may be useful to reduce the apparent stiffness of the end-effector normal to the part so that excessively large normal forces are avoided.

We may formulate Impedance Control within our standard inner loop/outer loop control architecture by specifying the outer loop term, a_x, in Equation (12.44). Let $x^d(t)$ be a reference trajectory defined in task space coordinates and let M_d, B_d, K_d, be 6×6 matrices specifying desired inertia, damping, and stiffness, respectively. Let $e(t) = x(t) - x^d(t)$ be the tracking error in task space and set

$$a_x = \ddot{x}^d - M_d^{-1}(B_d\dot{e} + K_d e + F) \tag{12.47}$$

where F is the measured environmental force. Substituting (12.47) into (12.44) yields the closed loop system

$$M_d\ddot{e} + B_d\dot{e} + K_d e = -F \tag{12.48}$$

which results in desired impedance properties of the end-effector. Note that for $F = 0$ tracking of the reference trajectory, $x^d(t)$, is achieved, whereas for nonzero environmental force, tracking is not necessarily achieved. We will address this difficulty in the next section.

12.4.2 Hybrid Impedance Control

In this section we introduce the notion of *Hybrid Impedance Control* following the treatment of [**?**]. We again take as our starting point the linear, decoupled system (12.44). The impedance control formulation in the previous section is independent of the environment dynamics. It is reasonable to expect that stronger results may be obtained by incorporating a model of the environment dynamics into the design. For example, we will see below that one may regulate both position and impedance simultaneously which is not possible with the pure impedance control law (12.47).

We consider a one-dimensional system representing one component of the outer loop system (12.44)

$$\ddot{x}_i = a_{x_i} \tag{12.49}$$

and we henceforth drop the subscript, i, for simplicity. We assume that the impedance of the environment in this direction, Z_e is fixed and known, a priori. The impedance of the robot, Z_r, is of course determined by the control input. The Hybrid Impedance Control design proceeds as follows based on the classification of the environment impedance into inertial, resistive, or capacitive impedances:

1. If the environment impedance, $Z_e(s)$, is capacitive, use a Norton network representation. Otherwise, use a Thévenin network representation[3].

2. Represent the robot impedance as the *Dual* to the environment impedance. Thévenin and Norton networks are considered dual to one another.

3. In the case that the environment impedance is capacitive we have the robot/environment interconnection as shown in Figure 12.14 where the environment one-port is the Nor-

[3]In fact, for a resistive environment, either representation may be used

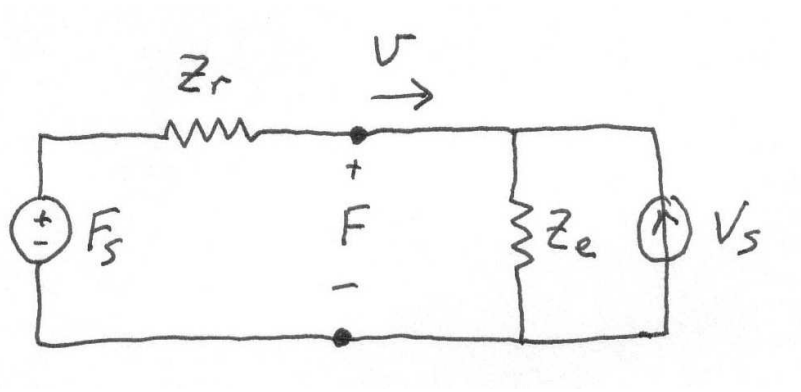


Figure 12.14: Capacitive Environment Case

ton network and the robot on-port is the Thévenin network. Suppose that $V_s = 0$, i.e. there are no environmental disturbances, and that F_s represents a reference force. From the circuit diagram it is straightforward to show that

$$\frac{F}{F_s} = \frac{Z_e(s)}{Z_e(s) + Z_r(s)} \tag{12.50}$$

Then the steady state force error, e_{ss}, to a step reference force, $F_s = \frac{F^d}{s}$ is given by the *Final Value Theorem* as

$$e_{ss} = \frac{-Z_r(0)}{Z_r(0) + Z_e(0)} = 0 \tag{12.51}$$

since $Z_e(0) = \infty$ (capacitive environment) and $Z_r \neq 0$ (non-capacitive robot).

The implications of the above calculation are that we can track a constant force reference value, while simultaneously specifying a given impedance, Z_r, for the robot.

In order to realize this result we need to design outer loop control term a_x in (12.49) using only position, velocity, and force feedback. This imposes a practical limitation on the the achievable robot impedance functions, Z_r.

Suppose Z_r^{-1} has relative degree one. This means that

$$Z_r(s) = M_c s + Z_{rem}(s) \tag{12.52}$$

where $Z_{rem}(s)$ is a proper rational function. If we now choose

$$a_x = -\frac{1}{M_c} Z_{rem}\dot{x} + \frac{1}{m_c}(F_s - F) \tag{12.53}$$

Substituting this into the double integrator system $\ddot{x} = a_x$ yields

$$Z_r(s)\dot{x} = F_s - F \tag{12.54}$$

Thus we have shown that, for a capacitive environment, force feedback can be used to regulate contact force and specify a desired robot impedance.

4. In the case that the environment impedance is inertial we have the robot/environment interconnection as shown in Figure 12.15 where the environment one-port is a Thévenin

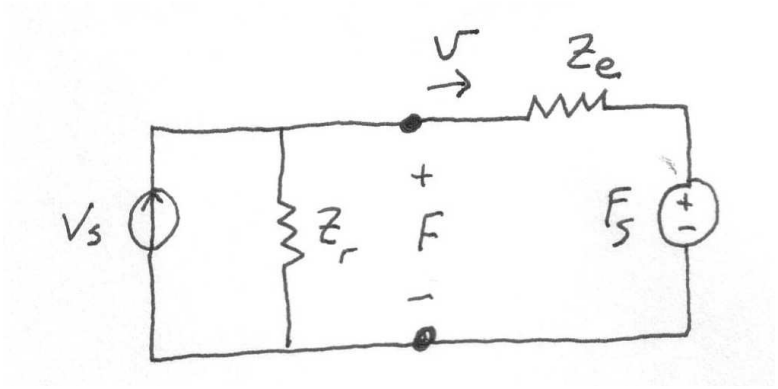


Figure 12.15: Inertial Environment Case

network and the robot on-port is a Norton network. Suppose that $F_s = 0$, and that V_s represents a reference velocity. From the circuit diagram it is straightforward to show that

$$\frac{V}{V_s} = \frac{Z_r(s)}{Z_e(s) + Z_r(s)} \tag{12.55}$$

Then the steady state force error, e_{ss}, to a step reference velocity command, $V_s = \frac{V^d}{s}$ is given by the *Final Value Theorem* as

$$e_{ss} = \frac{-Z_e(0)}{Z_r(0) + Z_e(0)} = 0 \tag{12.56}$$

since $Z_e(0) = 0$ (inertial environment) and $Z_r \neq 0$ (non-inertial robot).

To achieve this non-inertia robot impedance we take, as before,

$$Z_r(s) = M_c s + Z_{rem}(s) \tag{12.57}$$

and set

$$a_x = \ddot{x}^d + \frac{1}{M_c} Z_{rem}(\dot{x}^d - \dot{x}) + \frac{1}{M_c} F \tag{12.58}$$

Then, substituting this into the double integrator equation, $\ddot{x} = a_x$, yields

$$Z_r(s)(\dot{x}^d - x) = F \tag{12.59}$$

Thus we have shown that, for a capacitive environment, position control can be used to regulate a motion reference and specify a desired robot impedance.

Chapter 13

FEEDBACK LINEARIZATION

13.1 Introduction

In this chapter we discuss the notion of *feedback linearization of nonlinear systems*. This approach generalizes the concept of inverse dynamics of rigid manipulators discussed in Chapter 11. The basic idea of feedback linearization is to construct a nonlinear control law as a so-called *inner loop control* which, in the ideal case, exactly linearizes the nonlinear system after a suitable state space change of coordinates. The designer can then design a second stage or *outer loop control* in the new coordinates to satisfy the traditional control design specifications such as tracking, disturbance rejection, and so forth.

In the case of rigid manipulators the inverse dynamics control of Chapter 11 and the feedback linearizing control are the same. However, as we shall see, the full power of the feedback linearization technique for manipulator control becomes apparent if one includes in the dynamic description of the manipulator the transmission dynamics, such as elasticity resulting from shaft windup, gear elasticity, etc.

To introduce the idea of feedback linearization consider the following simple system,

$$\dot{x}_1 = a\sin(x_2) \tag{13.1}$$
$$\dot{x}_2 = -x_1^2 + u. \tag{13.2}$$

Note that we cannot simply choose u in the above system to cancel the nonlinear term $a\sin(x_2)$. However, if we first change variables by setting

$$y_1 = x_1 \tag{13.3}$$
$$y_2 = a\sin(x_2) = \dot{x}_1 \tag{13.4}$$

then, by the chain rule, y_1 and y_2 satisfy

$$\dot{y}_1 = y_2 \tag{13.5}$$
$$\dot{y}_2 = a\cos(x_2)(-x_1^2 + u). \tag{13.6}$$

We see that the nonlinearities can now be cancelled by the input

$$u = \frac{1}{a\cos(x_2)}v + x_1^2 \tag{13.7}$$

which result in the linear system in the (y_1, y_2) coordinates

$$\dot{y}_1 = y_2 \tag{13.8}$$
$$\dot{y}_2 = v. \tag{13.9}$$

The term v has the interpretation of an outer loop control and can be designed to place the poles of the second order linear system (13.6) in the coordinates (y_1, y_2). For example the outer loop control

$$v = -k_1 y_1 - k_2 y_2 \tag{13.10}$$

applied to (13.6) results in the closed loop system

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -k_1 y_1 - k_2 y_2 \end{aligned} \tag{13.11}$$

which has characteristic polynomial

$$p(s) = s^2 + k_2 s + k_1 \tag{13.12}$$

and hence the closed loop poles of the system with respect to the coordinates (y_1, y_2) are completely specified by the choice of k_1 and k_2. Figure 13.1 illustrates the inner loop/outer

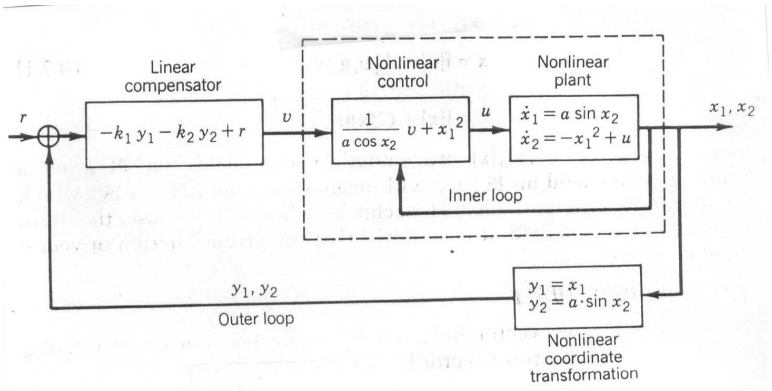


Figure 13.1: Architecture of feedback linearization controller.

loop implementation of the above control strategy. The response in the y variables is easy to determine. The corresponding response of the system in the original coordinates (x_1, x_2) can be found by inverting the transformation (13.2), in this case

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= \sin^{-1}(y_2/a) \qquad -a < y_2 < +a. \end{aligned} \tag{13.13}$$

This example illustrates several important features of feedback linearization. The first thing to note is the local nature of the result. We see from (13.3) and (13.4) that the transformation and the control make sense only in the region $-\infty < x_1 < \infty$, $-\frac{\pi}{2} < x_2 < \frac{\pi}{2}$. Second, in order to control the linear system (13.6), the coordinates (y_1, y_2) must be available for feedback. This can be accomplished by measuring them directly if they are physically meaningful variables, or by computing them from the measured (x_1, x_2) coordinates using the transformation (13.2). In the latter case the parameter a must be known precisely.

In Section 13.3 give necessary and sufficient conditions under which a general singl-input nonlinear system can be transformed into a linear system in the above fashion, using a nonlinear change of variables and nonlinear feedback as in the above example.

13.2 Background: The Frobenius Theorem

In this section we give some background from differential geometry that is necessary to understand the feedback linearization results to follow. In recent years an impressive volume of literature has emerged in the area of differential geometric methods for nonlinear systems, treating not only feedback linearization but also other problems such as disturbance decoupling, estimation, etc. The reader is referred to [**?**] for a comprehensive treatment of the subject. Most of the results in this area are intended to give abstract, coordinate-free descriptions of various geometric properties of nonlinear systems and as such are difficult for the non-mathematician to follow. It is our intent here to give only that portion of the theory that finds an immediate application to the manipulator control problem, and even then to give only the simplest versions of the results.

We restrict our attention here to single-input nonlinear systems of the form

$$\dot{x} = f(x) + g(x)u \tag{13.14}$$

where $f(x), g(x)$ are smooth vector fields on $\mathbb{R}^n$. By a *smooth vector field on* $\mathbb{R}^n$ we will mean a function $f : \mathbb{R}^n \to \mathbb{R}^n$ which is infinitely differentiable. Henceforth, whenever we use the term function or vector field, it is assumed that the given function or vector field is smooth.

Definition 13.1 *Let f and g be two vector fields on $\mathbb{R}^n$. The* Lie Bracket *of f and g, denoted by $[f, g]$, is a vector field defined by*

$$[f, g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g \tag{13.15}$$

where $\frac{\partial g}{\partial x}$ (respectively, $\frac{\partial f}{\partial x}$) denotes the $n \times n$ Jacobian matrix whose ij-th entry is $\frac{\partial g_i}{\partial x_j}$ (respectively, $\frac{\partial f_i}{\partial x_j}$).

Example 13.1 *Suppose that vector fields $f(x)$ and $g(x)$ on $\mathbb{R}^3$ are given as*

$$f(x) = \begin{bmatrix} x_2 \\ \sin x_1 \\ x_3^2 + x_1 \end{bmatrix} \qquad g(x) = \begin{bmatrix} 0 \\ x_2^2 \\ 0 \end{bmatrix}.$$

Then the vector field $[f, g]$ is computed according to (13.15) as

$$\begin{aligned} [f,g] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2x_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ \sin x_1 \\ x_1 + x_3^2 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ \cos x_1 & 0 & 0 \\ 1 & 0 & 2x_3 \end{bmatrix} \begin{bmatrix} 0 \\ x_2^2 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -x_2^2 \\ 2x_2 \sin s_1 \\ -2x_3 \end{bmatrix}. \end{aligned}$$

◇

We also denote $[f, g]$ as $ad_f(g)$ and define $ad_f^k(g)$ inductively by

$$ad_f^k(g) = [f, ad_f^{k-1}(g)] \tag{13.16}$$

with $ad_f^0(g) = g$.

Definition 13.2 *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a smooth vector field on $\mathbb{R}^n$ and let $h : \mathbb{R}^n \to \mathbb{R}$ be a scalar function. The* Lie Derivative *of h, with respect to f, denoted $L_f h$, is defined as*

$$L_f h = \frac{\partial h}{\partial x} f(x) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} f_i(x) \tag{13.17}$$

The Lie derivative is simply the directional derivative of h in the direction of $f(x)$, equivalently the inner product of the gradient of h and f. We denote by $L_f^2 h$ the Lie Derivative of $L_f h$ with respect to f, i.e.

$$L_f^2 h = L_f(L_f h) \tag{13.18}$$

In general we define

$$L_f^k h = L_f(L_f^{k-1} h) \quad \text{for } k = 1, \ldots, n \tag{13.19}$$

with $L_f^0 h = h$.

The following technical lemma gives an important relationship between the Lie Bracket and Lie derivative and is crucial to the subsequent development.

Lemma 13.1 *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a scalar function and f and g be vector fields on $\mathbb{R}^n$. Then we have the following identity*

$$L_{[f,g]} h = L_f L_g h - L_g L_f h \tag{13.20}$$

Proof: Expand Equation (13.20) in terms of the coordinates $x_1, \ldots, x_n$ and equate both sides. The i-th component $[f,g]_i$ of the vector field $[f,g]$ is given as

$$[f,g]_i = \sum_{j=1}^{n} \frac{\partial g_i}{\partial x_j} f_j - \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} g_j.$$

Therefore, the left-hand side of (13.20) is

$$\begin{aligned} L_{[f,g]}h &= \sum_{i=1}^{n} \frac{\partial h}{\partial x_i}[f,g]_i \\ &= \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} \left(\sum_{j=1}^{n} \frac{\partial g_i}{\partial x_j} f_j - \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} g_j \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial h}{\partial x_i} \left(\frac{\partial g_i}{\partial x_j} f_j - \frac{\partial f_i}{\partial x_j} g_j \right). \end{aligned}$$

If the right hand side of (13.20) is expanded similarly it can be shown, with a little algebraic manipulation, that the two sides are equal. The details are left as an exercise (Problem 12-1).

In order to discuss the idea of feedback linearization we first present a basic result in differential geometry known as the *Frobenius Theorem*. The Frobenius Theorem can be thought of as an existence theorem for solutions to certain systems of first order partial differential equations. Although a rigorous proof of this theorem is beyond the scope of this text, we can gain an intuitive understanding of it by considering the following system of partial differential equations

$$\frac{\partial z}{\partial x} = f(x,y,z) \tag{13.21}$$

$$\frac{\partial z}{\partial y} = g(x,y,z) \tag{13.22}$$

In this example there are two partial differential equations in a single unknown z. A solution to (13.21)-(13.22) is a function $z = \phi(x,y)$ satisfying

$$\frac{\partial \phi}{\partial x} = f(x,y,\phi(x,y)) \tag{13.23}$$

$$\frac{\partial \phi}{\partial y} = g(x,y,\phi(x,y)) \tag{13.24}$$

We can think of the function $z = \phi(x,y)$ as defining a surface in $\mathbb{R}^3$ as in Figure 13.2. The function $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ defined by

$$\Phi(x,y) = (x,y,\phi(x,y)) \tag{13.25}$$

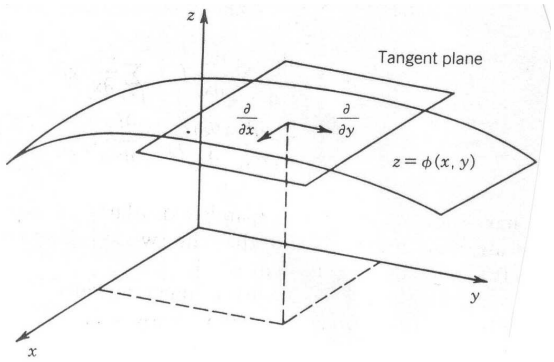


Figure 13.2: Integral manifold in R^3.

then characterizes both the surface and the solution to the system of Equations (13.21). At each point (x, y) the tangent plane to the surface is spanned by two vectors found by taking partial derivatives of Φ in the x and y directions, respectively, that is, by

$$X_1 = (1, 0, f(x, y, \phi(x, y)))^T \tag{13.26}$$
$$X_2 = (0, 1, g(x, y, \phi(x, y)))^T. \tag{13.27}$$

The vector fields X_1 and X_2 are linearly independent and span a two dimensional subspace at each point. Notice that X_1 and X_2 are completely specified by the system of Equations (13.21). Geometrically, one can now think of the problem of solving the system of first order partial differential Equations (13.21) as the problem of finding a surface in $\mathbb{R}^3$ whose tangent space at each point is spanned by the vector fields X_1 and X_2. Such a surface, if it can be found, is called an *integral manifold* for the system (13.21). If such an integral manifold exists then the set of vector fields, equivalently, the system of partial differential equations, is called *completely integrable*.

Let us reformulate this problem in yet another way. Suppose that $z = \phi(x, y)$ is a solution of (13.21). Then it is a simple computation (Problem 12-2) to check that the function

$$h(x, y, z) = z - \phi(x, y) \tag{13.28}$$

satisfies the system of partial differential equations

$$L_{X_1} h = 0 \tag{13.29}$$
$$L_{X_2} h = 0 \tag{13.30}$$

Conversely, suppose a scalar function h can be found satisfying (13.29)-(13.30), and suppose that we can solve the equation

$$h(x, y, z) = 0 \tag{13.31}$$

for z, as $z = \phi(x, y)$.[1] Then it can be shown that ϕ satisfies (13.21). (Problem 12-3) Hence, complete integrability of the set of vector fields (X_1, X_2) is equivalent to the existence of h satisfying (13.29). With the preceding discussion as background we state the following

Definition 13.3 *A linearly independent set of vector fields $\{X_1, \ldots, X_m\}$ on $\mathbb{R}^n$ is said to be* completely integrable *if and only if there are $n - m$ linearly independent functions $h_1, \ldots, h_{n-m}$ satisfying the system of partial differential equations*

$$L_{X_i} h_j = 0 \quad \text{for } 1 \leq i \leq n \ ; \ 1 \leq j \leq m \tag{13.32}$$

Definition 13.4 *A linearly independent set of vector fields $\{X_1, \ldots, X_m\}$ is said to be* involutive *if and only if there are scalar functions $\alpha_{ijk} : \mathbb{R}^n \to \mathbb{R}$ such that*

$$[X_i, X_j] \ = \ \sum_{k=1}^{m} \alpha_{ijk} X_k \text{ for all } i, j, k. \tag{13.33}$$

Involutivity simply means that if one forms the Lie Bracket of any pair of vector fields from the set $\{X_1, \ldots, X_m\}$ then the resulting vector field can be expressed as a linear combination of the original vector fields $X_1, \ldots, X_m$. Note that the coefficients in this linear combination are allowed to be smooth functions on $\mathbb{R}^n$. In the simple case of (13.21) one can show that if there is a solution $z = \phi(x, y)$ of (13.21) then involutivity of the set $\{X_1, X_2\}$ defined by (13.32) is equivalent to interchangeability of the order of partial derivatives of ϕ, that is, $\frac{\partial^2 \phi}{\partial x \partial y} = \frac{\partial^2 \phi}{\partial y \partial x}$. The Frobenius Theorem, stated next, gives the conditions for the existence of a solution to the system of partial differential Equations (13.32).

Theorem 2 *Let $\{X_1, \ldots, X_m\}$ be a set of vector fields that are linearly independent at each point. Then the set of vector fields is completely integrable if and only if it is involutive.*

Proof: See, for example, Boothby [**?**].

13.3 Single-Input Systems

The idea of feedback linearization is easiest to understand in the context of single-input systems. In this section we derive the feedback linearization result of Su [**?**] for single-input nonlinear systems. As an illustration we apply this result to the control of a single-link manipulator with joint elasticity.

Definition 13.5 *A single-input nonlinear system*

$$\dot{x} \ = \ f(x) + g(x)u \tag{13.34}$$

[1] The so-called *Implicit Function Theorem* states that (13.31) can be solved for z as long as $\frac{\partial h}{\partial z} \neq 0$.

where $f(x)$ and $g(x)$ are smooth vector fields on $\mathbb{R}^n$, $f(0) = 0$, and $u \in \mathbb{R}$, is said to be feedback linearizable *if there exists a region U in $\mathbb{R}^n$ containing the origin, a diffeomorphism*[2] *$T : U \to \mathbb{R}^n$, and nonlinear feedback*

$$u = \alpha(x) + \beta(x)v \tag{13.35}$$

with $\beta(x) \neq 0$ on U such that the transformed variables

$$y = T(x) \tag{13.36}$$

satisfy the system of equations

$$\dot{y} = Ay + bv \tag{13.37}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & & & 0 \\ 0 & 0 & 1 & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & 1 \\ 0 & 0 & \cdot & \cdot & 0 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}. \tag{13.38}$$

Remark 13.1 *The nonlinear transformation (13.36) and the nonlinear control law (13.35), when applied to the nonlinear system (13.34), result in a linear controllable system (13.37). The diffeomorphism $T(x)$ can be thought of as a nonlinear change of coordinates in the state space. The idea of feedback linearization is then that if one first changes to the coordinate system $y = T(x)$, then there exists a nonlinear control law to cancel the nonlinearities in the system. The feedback linearization is said to be* global *if the region U is all of $\mathbb{R}^n$.*

We next derive necessary and sufficient conditions on the vector fields f and g in (13.34) for the existence of such a transformation. Let us set

$$y = T(x) \tag{13.39}$$

and see what conditions the transformation $T(x)$ must satisfy. Differentiating both sides of (13.39) with respect to time yields

$$\dot{y} = \frac{\partial T}{\partial x}\dot{x} \tag{13.40}$$

where $\dfrac{\partial T}{\partial x}$ is the Jacobian matrix of the transformation $T(x)$. Using (13.34) and (13.37), Equation (13.40) can be written as

$$\frac{\partial T}{\partial x}(f(x) + g(x)u) = Ay + bv. \tag{13.41}$$

[2] A *diffeomorphism* is simply a differentiable function whose inverse exists and is also differentiable. We shall assume both the function and its inverse to be infinitely differentiable. Such functions are customarily referred to as C^∞ *diffeomorphisms.*

In component form with

$$T = \begin{bmatrix} T_1 \\ \cdot \\ \cdot \\ \cdot \\ T_n \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 & 0 & & & 0 \\ 0 & 0 & 1 & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & 1 \\ 0 & 0 & \cdot & \cdot & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \tag{13.42}$$

we see that the first equation in (13.41) is

$$\frac{\partial T_1}{\partial x_1}\dot{x}_1 + \cdots + \frac{\partial T_1}{\partial x_n}\dot{x}_n = T_2 \tag{13.43}$$

which can be written compactly as

$$L_f T_1 + L_g T_1 u = T_2 \tag{13.44}$$

Similarly, the other components of T satisfy

$$\begin{aligned} L_f T_2 + L_g T_2 u &= T_3 \\ &\cdot \\ &\cdot \\ &\cdot \\ L_f T_n + L_g T_n u &= v. \end{aligned} \tag{13.45}$$

Since we assume that $T_1, \ldots, T_n$ are independent of u while v is not independent of u we conclude from (13.45) that

$$L_g T_1 = L_g T_2 = \cdots = L_g T_{n-1} = 0 \tag{13.46}$$

$$L_g T_n \neq 0 \tag{13.47}$$

This leads to the system of partial differential equations

$$L_f T_i = T_{i+1} \quad i = 1, \ldots n-1 \tag{13.48}$$

together with

$$L_f T_n + L_g T_n u = v \tag{13.49}$$

Using Lemma 13.1 and the conditions (13.46) and (13.47) we can derive a system of partial differential equations in terms of T_1 alone as follows. Using $h = T_1$ in Lemma 13.1 we have

$$L_{[f,g]} T_1 = L_f L_g T_1 - L_g L_f T_1 = 0 - L_g T_2 = 0 \tag{13.50}$$

Thus we have shown

$$L_{[f,g]}T_1 = 0 \tag{13.51}$$

By proceeding inductively it can be shown (Problem 12-4) that

$$L_{ad_f^k g}T_1 = 0 \quad k = 0, 1, \ldots n-2 \tag{13.52}$$

$$L_{ad_f^{n-1} g}T_1 \neq 0 \tag{13.53}$$

If we can find T_1 satisfying the system of partial differential Equations (13.52), then $T_2, \ldots, T_n$ are found inductively from (13.48) and the control input u is found from

$$L_f T_n + L_g T_n u = v \tag{13.54}$$

as

$$u = \frac{1}{L_g T_n}(v - L_f T_n) \tag{13.55}$$

We have thus reduced the problem to solving the system (13.52) for T_1. When does such a solution exist? First note that the vector fields g, $ad_f(g), \ldots, ad_f^{n-1}(g)$ must be linearly independent. If not, that is, if for some index i

$$ad_f^i(g) = \sum_{k=0}^{i-1} \alpha_k ad_f^k(g) \tag{13.56}$$

then $ad_f^{n-1}(g)$ would be a linear combination of g, $ad_f(g), \ldots, ad_f^{n-2}(g)$ and (13.53) could not hold. Now by the Frobenius Theorem (13.52) has a solution if and only if the set of vector fields $\{g, ad_f(g), \ldots, ad_f^{n-2}(g)\}$ is involutive. Putting this together we have shown the following.

Theorem 3 *The nonlinear system*

$$\dot{x} = f(x) + g(x)u \tag{13.57}$$

with $f(x)$, $g(x)$ smooth vector fields, and $f(0) = 0$ is feedback linearizable if and only if there exists a region U containing the origin in $\mathbb{R}^n$ in which the following conditions hold:

1. *The vector fields $\{g, ad_f(g), \ldots, ad_f^{n-1}(g)\}$ are linearly independent in U.*

2. *The set $\{g, ad_f(g), \ldots, ad_f^{n-2}(g)\}$ is involutive in U.*

Example 13.2 *Consider the single link manipulator with flexible joint shown in Figure 13.3. Ignoring damping for simplicity, the equations of motion are*

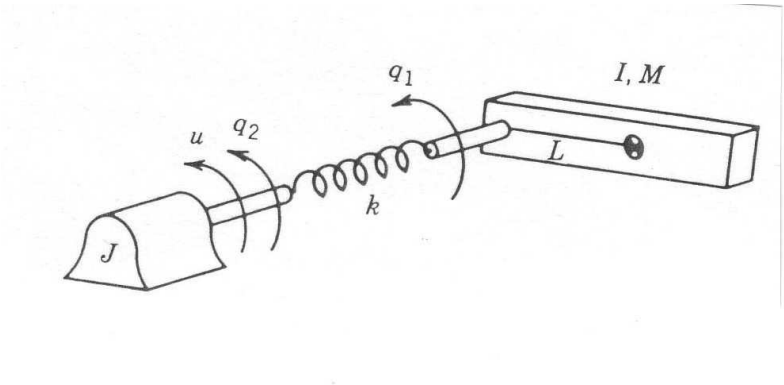


Figure 13.3: Single-Link Flexible Joint Robot

$$
\begin{aligned}
I\ddot{q}_1 + Mgl\sin(q_1) + k(q_1 - q_2) &= 0 && (13.58)\\
J\ddot{q}_2 + k(q_2 - q_1) &= u && (13.59)
\end{aligned}
$$

Note that since the nonlinearity enters into the first equation the control u cannot simply be chosen to cancel it as in the case of the rigid manipulator equations.

In state space we set

$$
\begin{aligned}
x_1 = q_1 \qquad & x_2 = \dot{q}_1 \\
x_3 = q_2 \qquad & x_4 = \dot{q}_2
\end{aligned} \tag{13.60}
$$

and write the system (13.58) as

$$
\begin{aligned}
\dot{x}_1 &= x_2 && (13.61)\\
\dot{x}_2 &= -\frac{MgL}{I}\sin(x_1) - \frac{k}{I}(x_1 - x_3)\\
\dot{x}_3 &= x_4\\
\dot{x}_4 &= \frac{k}{J}(x_1 - x_3) + \frac{1}{J}u.
\end{aligned}
$$

The system is thus of the form (13.34) with

$$
f(x) = \begin{bmatrix} x_2 \\ -\frac{MgL}{I}\sin(x_1) - \frac{k}{I}(x_1 - x_3) \\ x_4 \\ \frac{k}{J}(x_1 - x_3) \end{bmatrix} \qquad g(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J} \end{bmatrix}. \tag{13.62}
$$

Therefore $n = 4$ and the necessary and sufficient conditions for feedback linearization of this system are that

$$
rank\left[g, ad_f(g), ad_f^2(g), ad_f^3(g)\right] = 4 \tag{13.63}
$$

and that the set

$$\{g, ad_f(g), ad_f^2(g)\} \tag{13.64}$$

be involutive. Performing the indicated calculations it is easy to check that (Problem 12-7)

$$[g, ad_f(g), ad_f^2(g), ad_f^3(g)] = \begin{bmatrix} 0 & 0 & 0 & \frac{k}{IJ} \\ 0 & 0 & \frac{k}{IJ} & 0 \\ 0 & \frac{1}{J} & 0 & -\frac{k}{J^2} \\ \frac{1}{J} & 0 & -\frac{k}{J^2} & 0 \end{bmatrix} \tag{13.65}$$

which has rank 4 for $k > 0$, $I, J < \infty$. Also, since the vector fields $\{g, ad_f(g), ad_f^2(g)\}$ are constant, they form an involutive set. To see this it suffices to note that the Lie Bracket of two constant vector fields is zero. Hence the Lie Bracket of any two members of the set of vector fields in (13.64) is zero which is trivially a linear combination of the vector fields themselves. It follows that the system (13.58) is feedback linearizable. The new coordinates

$$y_i = T_i \qquad i = 1, \ldots, 4 \tag{13.66}$$

are found from the conditions (13.52), with $n = 4$, that is

$$\begin{aligned} L_g T_1 &= 0 \\ L_{[f,g]} T_1 &= 0 \\ L_{ad_f^2 g} T_1 &= 0 \\ L_{ad_f^3 g} T_1 &= 0 \end{aligned}$$

Carrying out the above calculations leads to the system of equations (Problem 12-8)

$$\frac{\partial T_1}{\partial x_2} = 0 \; ; \; \frac{\partial T_1}{\partial x_3} = 0 \; ; \; \frac{\partial T_1}{\partial x_4} = 0 \tag{13.67}$$

and

$$\frac{\partial T_1}{\partial x_1} \neq 0. \tag{13.68}$$

From this we see that the function T_1 should be a function of x_1 alone. Therefore, we take the simplest solution

$$y_1 = T_1 = x_1 \tag{13.69}$$

and compute from (13.48) (Problem 12-9)

$$y_2 = T_2 = L_f T_1 = x_2 \tag{13.70}$$

$$y_3 = T_3 = L_f T_2 = -\frac{MgL}{I} \sin(x_1) - \frac{k}{I}(x_1 - x_3) \tag{13.71}$$

$$y_4 = T_4 = L_f T_3 = -\frac{MgL}{I} \cos(x_1) - \frac{K}{I}(x_2 - x_4). \tag{13.72}$$

The feedback linearizing control input u is found from the condition

$$u = \frac{1}{L_g T_4}(v - L_f T_4) \tag{13.73}$$

as (Problem 12-10)

$$u = \frac{IJ}{k}(v - a(x)) = \beta(x)v + \alpha(x) \tag{13.74}$$

where

$$\begin{aligned} a(x) := & \frac{MgL}{I}\sin(x_1)\left(x_2^2 + \frac{MgL}{I}\cos(x_1) + \frac{k}{I}\right) \\ & + \frac{k}{I}(x_1 - x_3)\left(\frac{k}{I} + \frac{k}{J} + \frac{MgL}{I}\cos(x_1)\right). \end{aligned} \tag{13.75}$$

◇

Therefore in the coordinates $y_1, \ldots, y_4$ with the control law (13.74) the system becomes

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= y_3 \\ \dot{y}_3 &= y_4 \\ \dot{y}_4 &= v \end{aligned} \tag{13.76}$$

or, in matrix form,

$$\dot{y} = Ay + bv \tag{13.77}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{13.78}$$

Remark 13.2 *The above feedback linearization is actually global. In order to see this we need only compute the inverse of the change of variables (13.69)-(13.72). Inspecting (13.69)(13.72) we see that*

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 \\ x_3 &= y_1 + \frac{I}{k}\left(y_3 + \frac{MgL}{I}\sin(y_1)\right) \\ x_4 &= y_2 + \frac{I}{k}\left(y_4 + \frac{MgL}{I}\cos(y_1)y_2\right). \end{aligned} \tag{13.79}$$

(13.80)

The inverse transformation is well defined and differentiable everywhere and, hence, the feedback linearization for the system (13.58) holds globally. The transformed variables $y_1, \ldots, y_4$ *are themselves physically meaningful. We see that*

$$\begin{aligned} y_1 = x_1 &= \textit{link position} \\ y_2 = x_2 &= \textit{link velocity} \\ y_3 = \dot{y}_2 &= \textit{link acceleration} \\ y_4 = \dot{y}_3 &= \textit{link jerk.} \end{aligned} \tag{13.81}$$

Since the motion trajectory of the link is typically specified in terms of these quantities they are natural variables to use for feedback.

Example 13.3 *One way to execute a step change in the link position while keeping the manipulator motion smooth would be to require a constant jerk during the motion. This can be accomplished by a cubic polynomial trajectory using the methods of Chapter 8. Therefore, let us specify a trajectory*

$$\theta_\ell^d(t) = y_1^d = a_1 + a_2 t + a_3 t^2 + a_4 t^3 \tag{13.82}$$

so that

$$\begin{aligned} y_2^d &= \dot{y}_1^d = a_2 + 2a_3 t + 3a_4 t^2 \\ y_3^d &= \dot{y}_2^d = 2a_3 + 6a_4 t \\ y_4^d &= \dot{y}_3^d = 6a_4. \end{aligned}$$

Then a linear control law that tracks this trajectory and that is essentially equivalent to the feedforward/feedback scheme of Chapter 11 is given by

$$v = \dot{y}_4^d - k_1(y_1 - y_1^d) - k_2(y_2 - y_2^d) - k_3(y_3 - y_3^d) - k_4(y_4 - y_4^d) \tag{13.83}$$

Applying this control law to the fourth order linear system (13.74) we see that the tracking error $e(t) = y_1 - y_1^d$ *satisfies the fourth order linear equation*

$$\frac{d^4 e}{dt^4} + k_4 \frac{d^3 e}{dt^3} + k_3 \frac{d^2 e}{dt^2} + k_2 \frac{de}{dt} + k_1 e = 0 \tag{13.84}$$

and, hence, the error dynamics are completely determined by the choice of gains $k_1, \ldots, k_4$*.* ◇

Notice that the feedback control law (13.83) is stated in terms of the variables $y_1, \ldots, y_4$. Thus, it is important to consider how these variables are to be determined so that they may be used for feedback in case they cannot be measured directly.

Although the first two variables, representing the link position and velocity, are easy to measure, the remaining variables, representing link acceleration and jerk, are difficult

to measure with any degree of accuracy using present technology. One could measure the original variables $x_1, \ldots, x_4$ which represent the motor and link positions and velocities, and compute $y_1, \ldots, y_4$ using the transformation Equations (13.69)-(13.72). In this case the parameters appearing in the transformation equations would have to be known precisely. Another, and perhaps more promising, approach is to construct a dynamic observer to estimate the state variables $y_1, \ldots, y_4$.

13.4 Feedback Linearization for N-Link Robots

In the general case of an n-link manipulator the dynamic equations represent a multi-input nonlinear system. The conditions for feedback linearization of multi-input systems are more difficult to state, but the conceptual idea is the same as the single-input case. That is, one seeks a coordinate systems in which the nonlinearities can be exactly canceled by one or more of the inputs. In the multi-input system we can also decouple the system, that is, linearize the system in such a way that the resulting linear system is composed of subsystems, each of which is affected by only a single one of the outer loop control inputs. Since we are concerned only with the application of these ideas to manipulator control we will not need the most general results in multi-input feedback linearization. Instead, we will use the physical insight gained by our detailed derivation of this result in the single-link case to derive a feedback linearizing control both for n-link rigid manipulators and for n-link manipulators with elastic joints directly.

Example 13.4 *We will first verify what we have stated previously, namely that for an n-link rigid manipulator the feedback linearizing control is identical to the inverse dynamics control of Chapter 11. To see this, consider the rigid equations of motion (11.6), which we write in state space as*

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -M(x_1)^{-1}(C(x_1, x_2)x_2 + g(x_1)) + M(x_1)^{-1}u \end{aligned} \tag{13.85}$$

with $X_1 = q$; $X_2 = \dot{q}$. In this case a feedback linearizing control is found by simply inspecting (13.85) as

$$u = M(x_1)v + C(x_1, x_2)x_2 + g(x_1) \tag{13.86}$$

Substituting (13.86) into (13.85) yields

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= v. \end{aligned} \tag{13.87}$$

Equation (13.97) represents a set of n-second order systems of the form

$$\begin{aligned} \dot{x}_{1i} &= x_{2i} \\ \dot{x}_{2i} &= v_i, \quad i = 1, \ldots, n. \end{aligned} \tag{13.88}$$

Comparing (13.86) with (11.17) we see indeed that the feedback linearizing control for a rigid manipulator is precisely the inverse dynamics control of Chapter 11. $\diamond$

Example 13.5 *If the joint flexibility is included in the dynamic description of an n-link robot the equations of motion can be written as*

$$D(q_1)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) = 0 \tag{13.89}$$
$$J\ddot{q}_2 - K(q_1 - q_2) = u. \tag{13.90}$$

In state space, which is now $\mathbb{R}^{4n}$, we define state variables in block form

$$\begin{aligned} \dot{x}_1 &= q_1 \qquad x_2 = \dot{q}_1 \\ \dot{x}_3 &= q_2 \qquad x_4 = \dot{q}_2. \end{aligned} \tag{13.91}$$

Then from (13.89)-(13.90) we have:

$$\dot{x}_1 = x_2 \tag{13.92}$$
$$\dot{x}_2 = -D(x_1)^{-1}\{h(x_1,x_2) + K(x_1 - x_3)\} \tag{13.93}$$
$$\dot{x}_3 = x_4 \tag{13.94}$$
$$\dot{x}_4 = J^{-1}K(x_1 - x_3) + J^{-1}u. \tag{13.95}$$

where we define $h(x_1,x_2) = C(x_1,x_2)x_2 + g(x_1)$ for simplicity. This system is then of the form

$$\dot{x} = f(x) + G(x)u. \tag{13.96}$$

In the single-link case we saw that the appropriate state variables with which to define the system so that it could be linearized by nonlinear feedback were the link position, velocity, acceleration, and jerk. Following the single-input example, then, we can attempt to do the same thing in the multi-link case and derive a feedback linearizing transformation blockwise as follows: Set

$$y_1 = T_1(x_1) := x_1 \tag{13.97}$$
$$y_2 = T_2(x) := \dot{y}_1 = \dot{x}_2 \tag{13.98}$$
$$y_3 = T_3(x) := \dot{y}_2 = \dot{x}_2 \tag{13.99}$$
$$= -D(x_1)^{-1}\{h(x_1,x_2) + K(x_1 - x_3)\}$$
$$y_4 = T_4(x) := \dot{y}_3 \tag{13.100}$$
$$= -\frac{d}{dt}[D(x_1)^{-1}]\{h(x_1,x_2) + K(x_1 - x_3)\} - D(x_1)^{-1}\left\{\frac{\partial h}{\partial x_1}x_2 + \frac{\partial h}{\partial x_2}[-D(x_1)^{-1}(h(x_1,x_2) + K(x_1 - x_3))] + K(x_2 - x_4)\right\}$$
$$:= a_4(x_1,x_2,x_3) + D(x_1)^{-1}Kx_4$$

where for simplicity we define the function a_4 to be everything in the definition of y_4 except the last term, which is $D^{-1}Kx_4$. Note that x_4 appears only in this last term so that a_4 depends only on x_1, x_2, x_3.

As in the single-link case, the above mapping is a global diffeomorphism. Its inverse can be found by inspection to be

$$\begin{aligned} x_1 &= y_1 && (13.101)\\ x_2 &= y_2 && (13.102)\\ x_3 &= y_1 + K^{-1}(D(y_1)y_3 + h(y_1, y_2)) && (13.103)\\ x_4 &= K^{-1}D(y_1)(y_4 - a_4(y_1, y_2, y_3)). && (13.104) \end{aligned}$$

The linearizing control law can now be found from the condition

$$\dot{y}_4 = v \qquad (13.105)$$

where v is a new control input. Computing $\dot{y}_4$ from (13.100) and suppressing function arguments for brevity yields

$$\begin{aligned} v = \tfrac{\partial a_4}{\partial x_1}x_2 - \tfrac{\partial a_4}{\partial x_2}D^{-1}(h + K(x_1 - x_3)) \qquad (13.106)\\ + \tfrac{\partial a_4}{\partial x_3}x_4 + \tfrac{d}{dt}[D^{-1}]Kx_4 + D^{-1}K(J^{-1}K(x_1 - x_3) + J^{-1}u)\\ =: a(x) + b(x)u \end{aligned}$$

where $a(x)$ denotes all the terms in (13.106) but the last term, which involves the input u, and $b(x) := D^{-1}(x)KJ^{-1}$.

Solving the above expression for u yields

$$\begin{aligned} u &= b(x)^{-1}(v - a(x)) && (13.107)\\ &=: alpha(x) + \beta(x)v && (13.108) \end{aligned}$$

where $\beta(x) = JK^{-1}D(x)$ and $alpha(x) = -b(x)^{-1}a(x)$.

With the nonlinear change of coordinates (13.97)-(13.100) and nonlinear feedback (13.107) the transformed system now has the linear block form

$$\begin{aligned} \dot{y} &= \begin{bmatrix} 0 & I & 0 & 0\\ 0 & 0 & I & 0\\ 0 & 0 & 0 & I\\ 0 & 0 & 0 & 0 \end{bmatrix} y + \begin{bmatrix} 0\\ 0\\ 0\\ I \end{bmatrix} v && (13.109)\\ &=: Ay + Bv \end{aligned}$$

where $I = n \times n$ identity matrix, $0 = n \times n$ zero matrix, $y^T = (y_1^T, y_2^T, y_3^T, y_4^T) \in \mathbb{R}^{4n}$, and $v \in \mathbb{R}^n$. The system (13.109) represents a set of n decoupled quadruple integrators. The outer loop design can now proceed as before, because not only is the system linearized, but it consists of n subsystems each identical to the fourth order system (13.76). $\diamond$