3

Inverse Kinematics

Bill Goodwine
University of Notre Dame

3.1 Introduction

This chapter presents results related to the *inverse kinematics problem* for robotic manipulators. As presented elsewhere, the forward kinematics problem of a manipulator is to determine the configuration (position and orientation) of the end effector of the manipulator as a function of the manipulator's joint angles. The inverse problem of that, i.e., determining the joint angles given a desired end effector configuration, is the inverse kinematics problem and the subject of this chapter. This chapter will outline and provide examples for two main categories of approaches to this problem; namely, closed-form analytical methods and numerical approaches.

The main difficulty of the inverse kinematics problem in general is that for some desired end effector configuration, there may be no solutions, there may be a unique solution, or there may be multiple solutions. The advantage of a numerical approach is that it is relatively easy to implement. As illustrated subsequently, however, one drawback is that the method only leads to one solution for one set of starting values for what is fundamentally an iterative method. Also, if no solutions exist, a numerical approach will simply fail to converge, so care must be taken to distinguish between an attempted solution that will never converge and one that is simply slow to converge. The advantage of analytical approaches is that all solutions can be found and if no solutions exist, it will be evident from the computations. The disadvantage is that they are generally algebraically cumbersome and involve many steps and computations. Also, closed form solutions only exist for certain categories of manipulators, but fortunately, the kinematics associated with the most common manipulators generally seem to belong to the class of solvable systems.

3.2 Preliminaries

This section will elaborate upon the nature of the inherent difficulties associated with the inverse kinematics problem and also provide a summary of the nomenclature and notation used in this chapter. The first part of this section provides simple examples illustrating the fact that a various number of solutions may exist

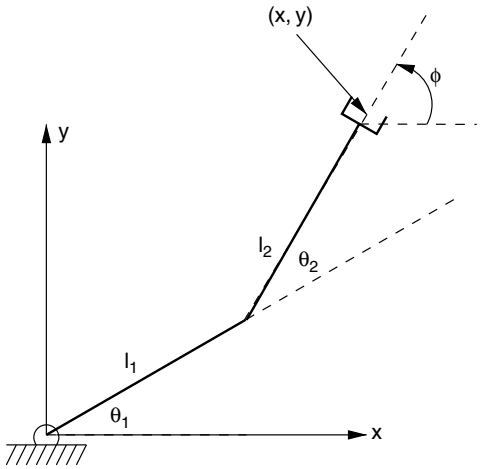

FIGURE 3.1 Simple two link manipulator.

for a given desired end effector configuration of a robot. The second section provides a summary of the notation and nomenclature used subsequently in this chapter.

3.2.1 Existence and Uniqueness of Solutions

Consider the very simple planar two link robot illustrated in Figure 3.1. Assume that the first link has a length of l_1, the second link has a length of l_2 and that θ_1 and θ_2 denote the angles of links one and two, respectively, as illustrated in the figure. The variables x, y, and ϕ denote the position and orientation of the end effector. Since this planar robot has only two joints, only two variables may be specified for the desired end effector configuration. For the purposes of this example, the (x, y) location of the end effector is utilized; however, any two variables, i.e., (x, y), (x, ϕ), or (y, ϕ) may be specified.

For simplicity, if we assume that $l_1 = l_2$, then Figure 3.2 illustrates a configuration in which two inverse kinematic solutions exist, which is the case when $(x, y) = (l_1, l_2)$. The two solutions are obviously, $(\theta_1, \theta_2) = (0°, 90°)$ and $(\theta_1, \theta_2) = (90°, 0°)$. Figure 3.3 illustrates a configuration in which an infinite number of kinematic solutions exist, which is the case where $(x, y) = (0, 0)$. In this case, the manipulator is "folded back" upon itself, which is typically physically unrealizable, but certainly is mathematically feasible. Since this configuration will allow the robot to rotate arbitrarily about the origin, there are an infinite number of configurations that place the end effector at the origin. Figure 3.4 illustrates a configuration in which only one inverse kinematics solution exists, which is the case when $(x, y) = (l_1 + l_2, 0)$. Finally, Figure 3.5 illustrates a case where no inverse kinematic solutions exist, which is the case when $(x, y) = (l_1 + l_2 + 1, 0)$.

While the preceding example was very simple, it illustrates the fundamental point that the inverse kinematics problem is complicated by the fact that there may be zero, one, multiple, or an infinite number of

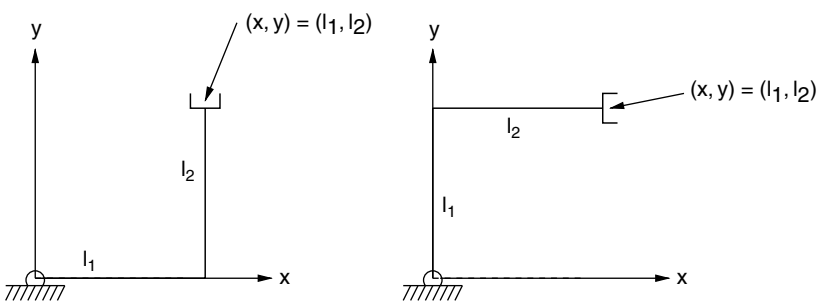

FIGURE 3.2 A configuration with two inverse kinematic solutions.

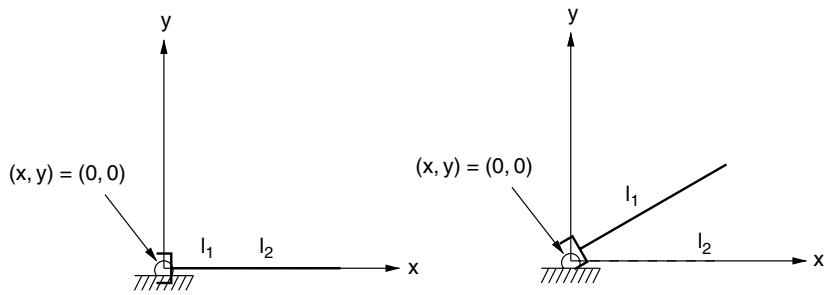

FIGURE 3.3 A configuration with an infinite number of inverse kinematic solutions.

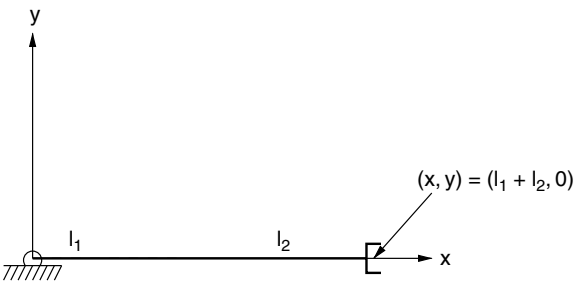

FIGURE 3.4 A configuration with one inverse kinematic solution.

solutions to the problem for a specified configuration. This phenomenon extends to the more complicated kinematics of six degree of freedom manipulators as well. As will become apparent in Section 3.3, certain assumptions must be made regarding the kinematics of the manipulator to make the inverse kinematics problem more feasible.

3.2.2 Notation and Nomenclature

This chapter will utilize the popular notation from Craig [1]. In particular:

${}^{A}P$	is a position vector, P, referenced to the coordinate frame A;
$\hat{X}_A, \hat{Y}_A, \hat{Z}_A$	are coordinate axes for frame A;
${}^{B}\hat{X}_A, {}^{B}\hat{Y}_A, {}^{B}\hat{Z}_A$	are coordinate axes for frame A expressed in frame B;
${}^{A}_{B}R$	is the rotation matrix describing the orientation of frame B relative to frame A;

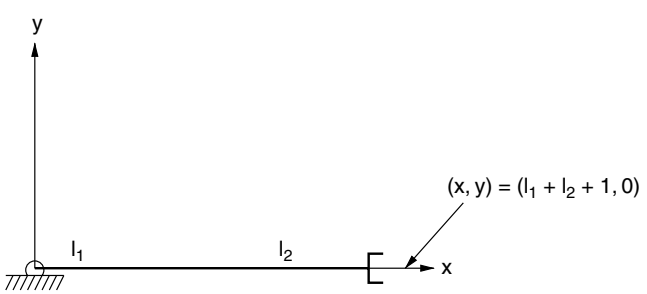

FIGURE 3.5 A configuration with no inverse kinematic solutions.

${}^{A}P_{BORG}$	is the origin of frame B expressed in coordinate of frame A;
${}^{A}_{B}T$	is the homogeneous transformation relating frame B to frame A; and,
$J(\theta)$	is the Jacobian which maps joint angle velocities to the rigid body velocity of the nth coordinate frame.

Also, unless otherwise indicated, this chapter will assume that coordinate frames are assigned to axes of the manipulator in accordance with the Denavit-Hartenberg [2] procedure presented in Craig [1]. This particular frame assignment procedure is implicit in some of the equations that are part of the algorithms presented. In particular, frames i and $i+1$ are affixed to the manipulator in accordance with the following rules:

1. At the point of intersection between the joint axes i and $i+1$, or the point where the common perpendicular between axes i and $i+1$ intersects axis i, assign the link frame origin for frame i.
2. Assign $\hat{Z}_i$ to point along the ith joint axis.
3. Assign $\hat{X}_i$ to point along the common perpendicular with axis $i+1$, or if axes i and $i+1$ intersect, normal to the plane defined by the axes i and $i+1$.
4. Assign frame 0 to match frame 1 when the first joint variable is zero, and for the nth frame (the last frame), the origin and $\hat{X}_n$ axis can be assigned arbitrarily.

If this procedure is followed, then the following *link parameters* are well-defined:

α_i	is the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$;
a_i	is the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$;
d_i	is the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along Z_i; and,
θ_i	is the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$.

A detailed, but straightforward derivation shows that

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

3.3 Analytical Approaches

This section outlines various analytical solution techniques that lead to closed form solutions to the inverse kinematics problem. This section is not completely comprehensive because specific manipulators may have kinematic features that allow for unique approaches. However, the primary procedures are outlined. First, Section 3.3.1 outlines the general approach of decoupling the manipulator kinematics so that the inverse kinematics problem can be decomposed into a set of subproblems. Section 3.3.2 presents the so-called "Pieper's solution," which is applicable to six degree of freedom manipulators in which the last three axes are rotational axes which mutually intersect. This approach essentially decomposes the inverse kinematics problem into two subproblems, which are solved separately. Finally, Section 3.3.4 outlines two other relatively recently developed althernative approaches.

3.3.1 Reduction of Inverse Kinematics to Subproblems

The basic approach of many analytical approaches is to decompose the complete inverse kinematics problem into a series of decoupled subproblems. This approach will mirror that presented in [6], but will be presented in a manner consistent with the Denavit-Hartenberg approach rather than the product of exponentials approach in [6]. First, two relatively simple motivational example problems will be presented, followed by the characterization of some more general results.

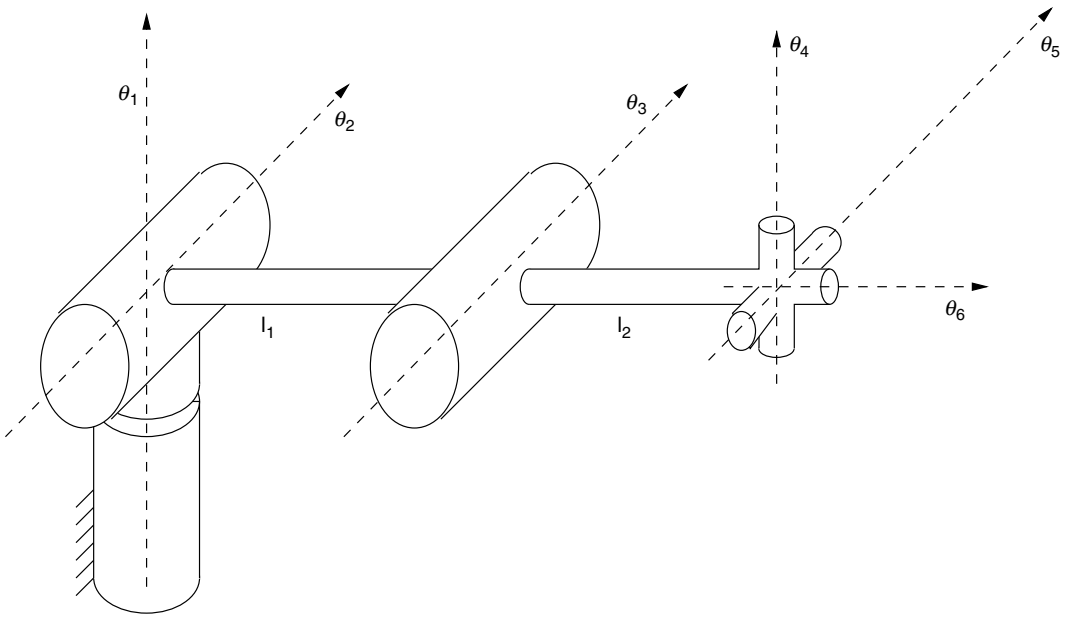

FIGURE 3.6 Elbow manipulator.

3.3.1.1 Inverse Kinematics for Two Examples *via* Subproblems

Consider the schematic illustration of the "Elbow Manipulator" in Figure 3.6. The link frame attachments are illustrated in Figure 3.7. With respect to the elbow manipulator, we can make the following observations:

- If ${}^0_6T_{des}$ is specified for the manipulator in Figure 3.6, generally two values for θ_3 may be determined since the location of the common origin of frames 4, 5, and 6 is given by ${}^0_6T_{des}$ and the distance from

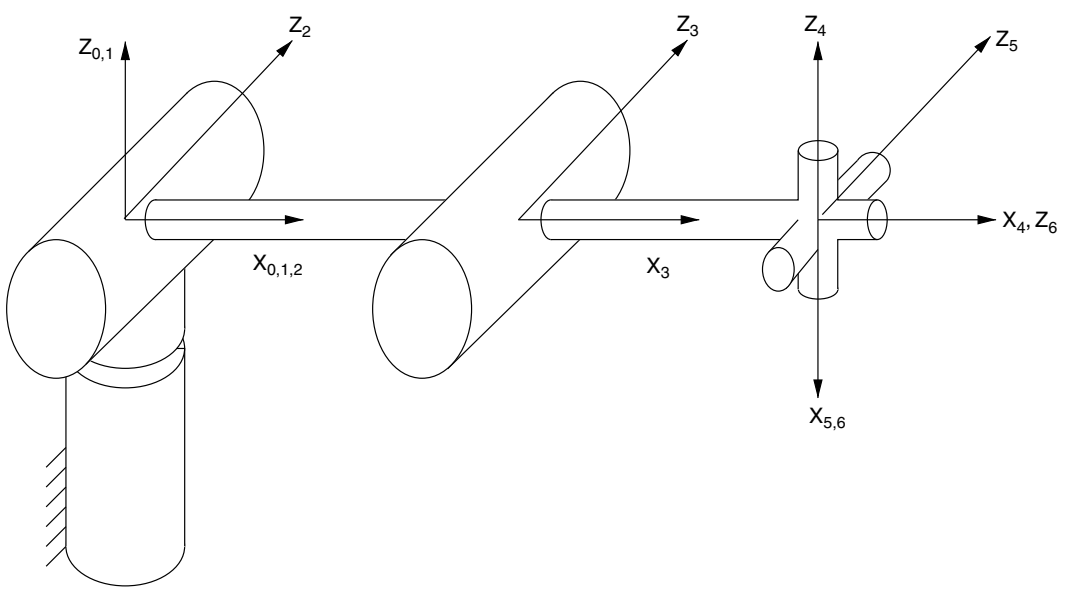

FIGURE 3.7 Link frame attachments for the elbow manipulator.

the common origin of each of the 0, 1, and 2 frames to the origins of any of the 4, 5, and 6 frames is only affected by θ_3.

- Once θ_3 is determined, the height of the origins of frames 4, 5, and 6, i.e., ${}^0P_{iORG}$, $i = 4, 5, 6$, is only affected by θ_2. Again, generally two values of θ_2 can be determined.
- For each pair of (θ_2, θ_3) values, the x and y components of ${}^0P_{iORG}$, $i = 4, 5, 6$, determines one unique θ_1 value.
- Once θ_1, θ_2, and θ_3 are known, ${}^0_3T$ can be computed. Using this, ${}^3_6T_{des}$ can be computed from

$$ {}^3_6T_{des} = {}^0_3T^{-1}\,{}^0_6T_{des} $$

- Since axes 4, 5, and 6 intersect, they will share a common origin. Therefore,

$$ a_4 = a_5 = d_5 = d_6 = 0 $$

Hence,

$$ {}^3_6T_{des} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & c_6s_4 - c_4c_5s_6 & c_4s_5 & l_2 \\ c_6s_5 & -s_5s_6 & -c_5 & 0 \\ c_5c_6s_4 + c_4s_6 & c_4c_6 - c_5s_4s_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

where c_i and s_i are shorthand for $\cos\theta_i$ and $\sin\theta_i$ respectively. Hence, two values for θ_5 can be computed from the third element of the second row.

- Once θ_5 is computed, one value of θ_4 can be computed from the first and third elements of the third column.
- Finally, two remaining elements of ${}^3_6T_{des}$, such as the first two elements of the second row, can be used to compute θ_6.

Generically, this procedure utilized the two following subproblems:

1. Determining a rotation that produced a specified distance. In the example, θ_3 determined the distance from the origins of the 1, 2, and 3 frames to the origins of the 4, 5, and 6 frames and subsequently, θ_2 was determined by the height of the origins of the 4, 5, and 6 frames.
2. Determining a rotation about a single axis that specified a particular point to be located in a desired position. In the example, θ_1 was determined in this manner.

Other subproblems are possible as well, such as determining two rotations, which, concatenated together, specify a particular point to be located in a particular position. These concepts are presented with full mathematical rigor in [6] and are further elaborated in [8].

As an additional example, consider the variation on the Stanford manipulator illustrated in Figure 3.8. The assumed frame configurations are illustrated in Figure 3.9. By analogous reasoning, we can compute the inverse kinematic solutions using the following procedure:

- Determine d_3 (the prismatic joint variable) from the distance between the location of ${}^0P_{6ORG}$ and ${}^0P_{0ORG}$.
- Determine θ_2 from the height of ${}^0P_{6ORG}$.
- Determine θ_1 from the location of ${}^0P_{6ORG}$.
- Determine θ_4, θ_5, and θ_6 in the same manner as for the elbow manipulator.

The presentation of these subproblems is intended to be a motivational conceptual introduction rather than a complete exposition. As is clear from the examples, for some manipulators, the inverse kinematics problem may be solved using only one or two of the subproblems. In contrast, some inverse kinematics problems cannot be solved in this manner. The following section presents Pieper's solution, which is a more mathematically complete solution technique, but one based fundamentally on such subproblems.

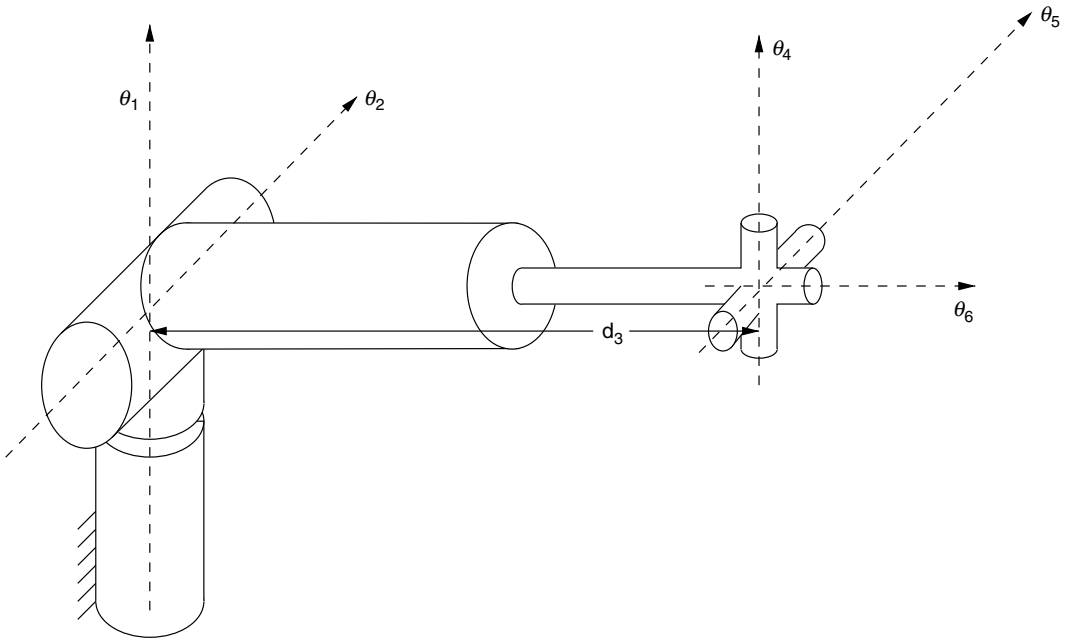

FIGURE 3.8 Variation of Stanford manipulator.

3.3.2 Pieper's Solution

The general formulation of Pieper's solution will closely parallel that presented in Craig [1], but more computational details will be provided. As mentioned previously, the approach presented only applied to manipulators in which the last three axes intersect and where all six joints are revolute. Fortunately,

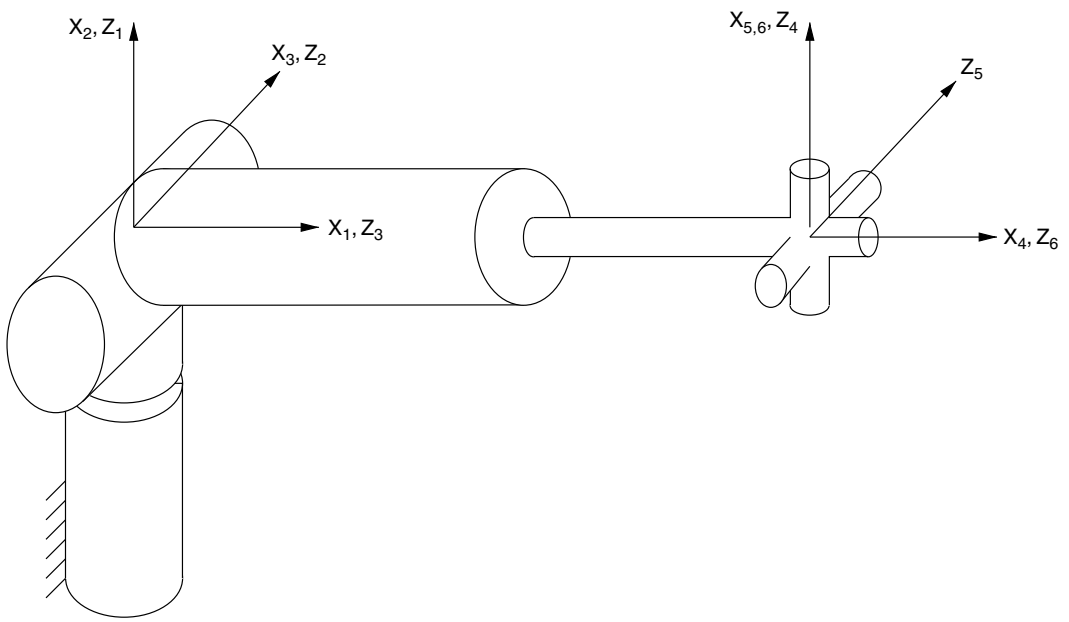

FIGURE 3.9 Link frame attachments for the variation of Stanford manipulator.

however, this happens to be the case when the manipulator is equipped with a three-axis spherical wrist for joints four through six.

Assuming a six degree of freedom manipulator and assuming that axes four, five, and six intersect, then the point of intersection will be the origins of frames 4, 5, and 6. Thus, the problem of solving for θ_1, θ_2, and θ_3 simplifies to a three-link position problem, since θ_4, θ_5, and θ_6 do not affect the position of their common origins, ${}^0P_{4ORG} = {}^0P_{5ORG} = {}^0P_{6ORG}$.

Recall that the first three elements in the fourth column of Equation (3.1) give the position of the origin of frame i expressed in frame $i-1$. Thus, from ${}^3_4T$,

$$ {}^3P_{4ORG} = \begin{bmatrix} a_3 \\ -\sin\alpha_3 d_4 \\ \cos\alpha_3 d_4 \end{bmatrix} $$

Expressing this in the 0 frame,

$$ {}^0P_{4ORG} = {}^0_1T\,{}^1_2T\,{}^2_3T \begin{bmatrix} a_3 \\ -\sin\alpha_3 d_4 \\ \cos\alpha_3 d_4 \\ 1 \end{bmatrix} $$

Since

$$ {}^2_3T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3\cos\alpha_2 & \cos\theta_3\cos\alpha_2 & -\sin\alpha_2 & -\sin\alpha_2 d_3 \\ \sin\theta_3\sin\alpha_2 & \cos\theta_3\sin\alpha_2 & \cos\alpha_2 & \cos\alpha_2 d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

we can define three functions that are a function only of the joint angle θ_3,

$$ \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_3(\theta_3) \\ 1 \end{bmatrix} = {}^2_3T \begin{bmatrix} a_3 \\ -\sin\alpha_3 d_4 \\ \cos\alpha_3 d_4 \\ 1 \end{bmatrix} $$

where

$$ \begin{aligned} f_1(\theta_3) &= a_2\cos\theta_3 + d_4\sin\alpha_3\sin\theta_3 + a_2 \\ f_2(\theta_3) &= a_3\sin\theta_3\cos\alpha_2 - d_4\sin\alpha_3\cos\alpha_2\cos\theta_3 - d_4\sin\alpha_2\cos\alpha_3 - d_3\sin\alpha_2 \\ f_3(\theta_3) &= a_3\sin\alpha_2\sin\theta_3 - d_4\sin\alpha_3\sin\alpha_2\cos\theta_3 + d4\cos\alpha_2\cos\alpha_3 + d_3\cos\alpha_2 \end{aligned} $$

and thus

$$ {}^0P_{4ORG} = {}^0_1T\,{}^1_2T \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_2(\theta_3) \\ 1 \end{bmatrix} \tag{3.3} $$

If frame 0 is specified in accordance with step four of the Denavit-Hartenberg frame assignment outlined in Section 3.2.2, then $\hat{Z}_0$ will be parallel to $\hat{Z}_1$, and hence $\alpha_0 = 0$. Also, since the origins of frames 0 and 1 will be coincident, $a_0 = 0$ and $d_1 = 0$. Thus, substituting these values into ${}^0_1T$ and expanding Equation (3.3),

we have

$$
{}^0P_{4ORG} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\times \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_1 \\ \sin\theta_2\cos\alpha_1 & \cos\theta_i\cos\alpha_1 & -\sin\alpha_1 & -\sin\alpha_1 d_i \\ \sin\theta_2\sin\alpha_1 & \cos\theta_2\sin\alpha_1 & \cos\alpha_1 & \cos\alpha_1 d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_3(\theta_3) \\ 1 \end{bmatrix}
$$

Since the second matrix is only a function of θ_2, we can write

$$
{}^0P_{4ORG} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_1(\theta_1,\theta_2) \\ g_2(\theta_1,\theta_2) \\ g_3(\theta_1,\theta_2) \\ 1 \end{bmatrix} \tag{3.4}
$$

where

$$
g_1(\theta_2,\theta_3) = \cos\theta_2 f_1(\theta_3) - \sin\theta_2 f_2(\theta_3) + a_1 \tag{3.5}
$$

$$
g_2(\theta_2,\theta_3) = \sin\theta_2\cos\alpha_1 f_1(\theta_3) + \cos\theta_2\cos\alpha_1 f_2(\theta_3) - \sin\alpha_1 f_3 - d_2\sin\alpha_1 \tag{3.6}
$$

$$
g_3(\theta_2,\theta_3) = \sin\theta_2\sin\alpha_1 f_1(\theta_3) + \cos\theta_2\sin\alpha_1 f_2(\theta_3) + \cos\alpha_1 f_3(\theta_3) + d_2\cos\alpha_1. \tag{3.7}
$$

Hence, multiplying Equation (3.4),

$$
{}^0P_{4ORG} = \begin{bmatrix} \cos\theta_1 g_1(\theta_2,\theta_3) - \sin\theta_1 g_2(\theta_2,\theta_3) \\ \sin\theta_1 g_1(\theta_2,\theta_3) + \cos\theta_1 g_2(\theta_2,\theta_3) \\ g_3(\theta_2,\theta_3) \\ 1 \end{bmatrix} \tag{3.8}
$$

It is critical to note that the "height" (more specifically, the z coordinate of the center of the spherical wrist expressed in frame 0) is the third element of the vector in Equation (3.8) and is independent of θ_1. Specifically,

$$
z = \sin\theta_2\sin\alpha_1 f_1(\theta_3) + \cos\theta_2\sin\alpha_1 f_2(\theta_3) + \cos\alpha_1 f_3(\theta_3) + d_2\cos\alpha_1 \tag{3.9}
$$

Furthermore, note that the distance from the origin of the 0 and 1 frames to the center of the spherical wrist will also be independent of θ_1. The square of this distance, denoted by r^2 is simply the sum of the squares of the first three elements of the vector in Equation (3.8); namely,

$$
\begin{aligned} r^2 &= g_1^2(\theta_2,\theta_3) + g_2^2(\theta_2,\theta_3) + g_3^2(\theta_2,\theta_3) \\ &= f_1^2(\theta_3) + f_2^2(\theta_3) + f_3^2(\theta_3) + a_1^2 + d_2^2 + 2d_2 f_3(\theta_3) \\ &\quad + 2a_1(\cos\theta_2 f_1(\theta_3) - \sin\theta_2 f_2(\theta_3)) \end{aligned} \tag{3.10}
$$

We now consider three cases, the first two of which simplify the problem considerably, and the third of which is the most general approach.

3.3.2.1 Simplifying Case Number 1: $a_1 = 0$

Note that if $a_1 = 0$ (this will be the case when axes 1 and 2 intersect), then from Equation (3.10), the distance from the origins of the 0 and 1 frames to the center of the spherical wrist (which is the origin of

frames 4, 5, and 6) is a function of θ_3 only; namely,

$$r^2 = f_1^2(\theta_3) + f_2^2(\theta_3) + f_3^2(\theta_3) + a_1^2 + d_2^2 + 2d_2 f_3(\theta_3) \tag{3.11}$$

Since it is much simpler in a numerical example, the details of the expansion of this expression will be explored in the specific examples subsequently; however, at this point note that since the f_i's contain trigonometric function of θ_3, there will typically be two values of θ_3 that satisfy Equation (3.11). Thus, given a desired configuration,

$${}^0_6T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the distance from the origins of frames 0 and 1 to the origins of frames 4, 5, and 6 is simply given by

$$r^2 = t_{14}^2 + t_{24}^2 + t_{34}^2 = f_1^2(\theta_3) + f_2^2(\theta_3) + f_3^2(\theta_3) + a_1^2 + d_2^2 + 2d_2 f_3(\theta_3) \tag{3.12}$$

Once one or more values of θ_3 that satisfy Equation (3.12) are determined, then the height of the center of the spherical wrist in the 0 frame is given by

$$t_{34} = g_3(\theta_2, \theta_3) \tag{3.13}$$

Since one or more values of θ_3 are known, Equation (3.13) will yield one value for θ_2 for each value of θ_3. Finally, returning to Equation (3.8), one value of θ_1 can be computed for each pair of (θ_2, θ_3) which have already been determined.

Finding a solution for joints 4, 5, and 6 is much more straightforward. First note that ${}^3_6R$ is determined by

$${}^0_6R = {}^0_3R^3_6R \Longrightarrow {}^3_6R = {}^0_3R^T{}^0_6R$$

where ${}^0_6R$ is specified by the desired configuration, and ${}^0_3R$ can be computed since $(\theta_1, \theta_2, \theta_3)$ have already been computed. This was outlined previously in Equation (3.2) and the corresponding text.

3.3.2.2 Simplifying Case Number 2: $\alpha_1 = 0$

Note that if $\alpha_1 = 0$, then, by Equation (3.5), the height of the spherical wrist center in the 0 frame will be

$$\begin{aligned} g_3(\theta_2, \theta_3) &= \sin\theta_2 \sin\alpha_1 f_1(\theta_3) + \cos\theta_2 \sin\alpha_1 f_2(\theta_3) + \cos\alpha_1 f_3(\theta_3) + d_2 \cos\alpha_1 \\ g_3(\theta_3) &= f_3(\theta_3) + d_2, \end{aligned}$$

so typically, two values can be determined for θ_3. Then Equation (3.10), which represents the distance from the origin of the 0 and 1 frames to the spherical wrist center, is used to determine one value for θ_2. Finally, returning to Equation (3.8) and considering the first two equations expressed in the system, one value of θ_1 can be computed for each pair of (θ_2, θ_3) which have already been determined.

3.3.2.3 General Case when $a_1 \neq 0$ and $\alpha_1 \neq 0$

This case is slightly more difficult and less intuitive, but it is possible to combine Equation (3.7) and Equation (3.11) to eliminate the θ_2 dependence and obtain a fourth degree equation in θ_3. For a few more details regarding this more complicated case, the reader is referred to [1].

TABLE 3.1 Example PUMA 560 Denavit-Hartenberg Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	0	0	θ_2
3	0	2 ft	0.5 ft	θ_3
4	-90°	0.1666 ft	2 ft	θ_4
5	90°	0	0	θ_5
6	90°	0	0	θ_6

3.3.3 Example

Consider a PUMA 560 manipulator with

$$ {}^0_6T_{des} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 1 \\ 0 & -1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

and assume the Denavit-Hartenberg parameters are as listed in Table 3.1. Note that $a_1 = 0$, so the procedure to follow is that outlined in Section 3.3.2.1.

First, substituting the link parameter values and the value for r^2 (the desired distance from the origin of frame 0 to the origin of frame 6) into Equation (3.11) and rearranging gives

$$ -8 \sin\theta_3 + 0.667 \cos\theta_3 = -5.2778 \tag{3.14} $$

Using the two trigonometric identities

$$ \sin\theta = \frac{2\tan\frac{\theta}{2}}{1+\tan^2\left(\frac{\theta}{2}\right)} \quad \text{and} \quad \cos\theta = \frac{1-\tan^2\left(\frac{\theta}{2}\right)}{1+\tan^2\left(\frac{\theta}{2}\right)} \tag{3.15} $$

and substituting into Equation (3.14) and rearranging gives

$$ -4.611\tan^2\left(\frac{\theta_3}{2}\right) + 16\tan\left(\frac{\theta_3}{2}\right) - 5.944 = 0 \tag{3.16} $$

Using the quadratic formula gives

$$ \tan\frac{\theta_3}{2} = \frac{-16 \pm \sqrt{(16)^2 - 4(-4.611)(-5.944)}}{2(-4.611)} \tag{3.17} $$

$$ \Longrightarrow \theta_3 = 45.87^\circ, 143.66^\circ \tag{3.18} $$

Considering the $\theta_3 = 45.87^\circ$ solution, substituting the link parameter values and the z-coordinate for ${}^0P_{6ORG}$ into Equation (3.8) and rearranging gives

$$ 0.6805\sin\theta_2 + 1.5122\cos\theta_2 = 1 $$

$$ \Longrightarrow \theta_2 = -28.68^\circ \quad \text{or} \quad 77.14^\circ $$

Considering the $\theta_3 = 143.66^\circ$ solution, substituting the link parameter values and the z-coordinate for ${}^0P_{6ORG}$ into Equation (3.8) and rearranging gives

$$ -0.6805\sin\theta_2 + 1.5122\cos\theta_2 = -1 $$

$$ \Longrightarrow \theta_2 = 102.85^\circ \quad \text{or} \quad -151.31^\circ $$

At this point, we have four combination of (θ_2, θ_3) solutions; namely

$$(\theta_2, \theta_3) = (-28.68^\circ, 45.87^\circ)$$
$$(\theta_2, \theta_3) = (77.14^\circ, 45.87^\circ)$$
$$(\theta_2, \theta_3) = (102.85^\circ, 143.66^\circ)$$
$$(\theta_2, \theta_3) = (-151.31^\circ, 143.66^\circ)$$

Now, considering the x and y elements of ${}^0P_{6ORG}$ in Equation (3.8) and substituting a pair of values for (θ_2, θ_3) into $g_1(\theta_2, \theta_3)$ and $g_1(\theta_2, \theta_3)$ in the first two lines and rearranging we have

$$\begin{bmatrix} g_1(\theta_2,\theta_3) & -g_2(\theta_2,\theta_3) \\ g_2(\theta_2,\theta_3) & g_1(\theta_2,\theta_3) \end{bmatrix} \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} = \begin{bmatrix} g_1(\theta_2,\theta_3) & -g_2(\theta_2,\theta_3) \\ g_2(\theta_2,\theta_3) & g_1(\theta_2,\theta_3) \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Substituting the $(\theta_2, \theta_3) = (-151.31^\circ, 143.66^\circ)$ solution for the g_i's gives

$$\begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} = \begin{bmatrix} -1.3231 & -0.5 \\ 0.5 & -1.3231 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.4114 \\ -0.9113 \end{bmatrix} \Longrightarrow \theta_1 = 245.7^\circ$$

Substituting the four pairs of (θ_2, θ_3) into the g_i's gives the following four sets of solutions including θ_1:

1. $(\theta_1, \theta_2, \theta_3) = (24.3^\circ, -28.7^\circ, 45.9^\circ)$
2. $(\theta_1, \theta_2, \theta_3) = (24.3^\circ, 102.9^\circ, 143.7^\circ)$
3. $(\theta_1, \theta_2, \theta_3) = (-114.3^\circ, 77.14^\circ, 45.9^\circ)$
4. $(\theta_1, \theta_2, \theta_3) = (-114.3^\circ, -151.32^\circ, 143.7^\circ)$

Now that θ_1, θ_2, and θ_3 are known, ${}^0_3T$ can be computed. Using this, ${}^3_6T_{des}$ can be computed from

$${}^3_6T_{des} = {}^0_3T^{-1}\,{}^0_6T_{des}$$

Since axes 4, 5, and 6 intersect, they will share a common origin. Therefore,

$$a_4 = a_5 = d_5 = d_6 = 0$$

Hence,

$${}^3_6T_{des} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & c_6s_4 - c_4c_5s_6 & c_4s_5 & l_2 \\ c_6s_5 & -s_5s_6 & -c_5 & 0 \\ c_5c_6s_4 + c_4s_6 & c_4c_6 - c_5s_4s_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

where c_i and s_i are shorthand for $\cos\theta_i$ and $\sin\theta_i$ respectively. Hence, two values for θ_5 can be computed from the third element of the second row.

Considering only the case where $(\theta_1, \theta_2, \theta_3) = (-114.29^\circ, -151.31^\circ, 143.65^\circ)$,

$${}^3_6T_{des} = \begin{bmatrix} 0.38256 & 0.90331 & -0.194112 & 2. \\ -0.662034 & 0.121451 & -0.739\,568 & 24. \\ -0.644484 & 0.411438 & 0.644484 & -2.922\,0.0 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

TABLE 3.2 Complete List of Solutions for the PUMA 560 Example

θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
-114.29°	-151.31°	143.65°	-106.76°	-137.69°	10.39°
-114.29°	-151.31°	143.65°	73.23°	137.69°	-169.60°
-114.29°	77.14°	45.86°	-123.98°	-51.00°	-100.47°
-114.29°	77.14°	45.86°	56.01°	51.00°	79.52°
24.29°	-28.68°	45.86°	-144.42°	149.99°	-165.93°
24.29°	-28.68°	45.86°	35.57°	-149.99°	14.06°
24.29°	102.85°	143.65°	-143.39°	29.20°	129.34°
24.29°	102.85°	143.65°	36.60°	-29.20°	-50.65°

using the third element of the second row, $\theta_5 = \pm 137.7^\circ$. Using the first and second elements of the second row, and following the procedure presented in Equation (3.14) through Equation (3.17), if $\theta_5 = 137.7^\circ$, then $\theta_6 = -169.6^\circ$ and using the first and third elements of the third column, $\theta_4 = 73.23^\circ$.

Following this procedure for the other combinations of $(\theta_1, \theta_2, \theta_3)$ produces the complete set of solutions presented in Table 3.2.

3.3.4 Other Approaches

This section very briefly outlines two other approaches and provides references for the interested reader.

3.3.4.1 Dialytical Elimination

One of the most general approaches is to reduce the inverse kinematics problem to a system of polynomial equations through rather elaborate manipulation and then use results from elimination theory from algebraic geometry to solve the equations. The procedure has the benefit of being very general. A consequence of this fact, however, is that it therefore does not exploit any specific geometric kinematic properties of the manipulator, since such specific properties naturally are limited to a subclass of manipulators. Details can be found in [5, 9].

3.3.4.2 Zero Reference Position Method

This approach is similar in procedure to Pieper's method, but is distinct in that only a single, base coordinate system is utilized in the computations. In particular, in the case where the last three link frames share a common origin, the position of the wrist center can be used to determine θ_1, θ_2, and θ_3. This is in contrast to Pieper's method which uses the distance from the origin of the wrist frame to the origin of the 0 frame to determine θ_3. An interested reader is referred to [4] for a complete exposition.

3.4 Numerical Techniques

This section presents an outline of the inverse kinematics problem utilizing numerical techniques. This approach is rather straightforward in that it utilizes the well-known and simple Newton's root finding technique. The two main, somewhat minor, complications are that generally, for a six degree of freedom manipulator, the desired configuration is specified as a homogeneous transformation, which contains 12 elements, but the total number of degrees of freedom is only six. Also, Jacobian singularities are also potentially problematics. As mentioned previously, however, one drawback is that for a given set of initial values for Newton's method, the algorithm converges to only one solution when perhaps multiple solutions may exist. The rest of this section assumes that a six degree of freedom manipulator is under consideration; however, the results presented are easily extended to more or fewer degrees of freedom systems.

3.4.1 Newton's Method

Newton's method is directed toward finding a solution to the system of equations

$$\begin{aligned} f_1(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) &= a_1 \\ f_2(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) &= a_2 \\ f_3(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) &= a_3 \\ f_4(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) &= a_4 \\ f_5(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) &= a_5 \\ f_6(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) &= a_6 \end{aligned} \tag{3.20}$$

which may be nonlinear. In matrix form, these equations can be expressed as

$$\mathbf{f}(\theta) = \mathbf{a} \quad \text{or} \quad \mathbf{f}(\theta) - \mathbf{a} = 0 \tag{3.21}$$

where $\mathbf{f}$, θ, and $\mathbf{a}$ are vectors in $\mathbb{R}^6$. Newton's method is an iterative technique to find the roots of Equation (3.21), which is given concisely by

$$\theta_{n+1} = \theta_n - \mathbf{J}^{-1}(\theta_n)\mathbf{f}(\theta_n) \tag{3.22}$$

where

$$\mathbf{J}(\theta_n) = \left[\frac{\partial f_i(\theta)}{\partial \theta_j}\right] = \begin{bmatrix} \frac{\partial f_1(\theta)}{\partial \theta_1} & \frac{\partial f_1(\theta)}{\partial \theta_2} & \frac{\partial f_1(\theta)}{\partial \theta_3} & \cdots & \frac{\partial f_1(\theta)}{\partial \theta_{n-1}} & \frac{\partial f_1(\theta)}{\partial \theta_n} \\ \frac{\partial f_2(\theta)}{\partial \theta_1} & \frac{f_2(\theta)}{\theta_2} & & \cdots & & \frac{\partial f_2(\theta)}{\partial \theta_n} \\ \vdots & \vdots & & \ddots & & \vdots \\ \frac{\partial f_n(\theta)}{\partial \theta_1} & \frac{f_n(\theta)}{\theta_2} & & \cdots & & \frac{\partial f_n(\theta)}{\partial \theta_n} \end{bmatrix}$$

Two theorems regarding the conditions for convergence of Newton's method are presented in Appendix A.

3.4.2 Inverse Kinematics Solution Using Newton's Method

This section elaborates upon the application of Newton's method to the inverse kinematics problem. Two approaches are presented. The first approach considers a six degree of freedom system utilizing a 6×6 Jacobian by choosing from the 12 elements of the desired T, 6 elements. The second approach is to consider all 12 equations relating the 12 components of ${}^0_6T_{des}$ which results in an overdetermined system which must then utilize a pseudo-inverse in implementing Newton's method.

For the six by six approach, the three position elements and three *independent* elements of the rotation submatrix matrix of ${}^0_6T_{des}$. These six elements, as a function of the joint variables, set equal to the corresponding values of ${}^0_6T_{des}$ then constitute all the elements of Equation (3.20). Let

$${}^0_6T(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = {}^0_6T_\theta$$

denote the forward kinematics as a function of the joint variables, and let ${}^0_6T_\theta(a,b)$ denote the element of ${}^0_6T_\theta$ that is in the ath row and bth column. Similarly, let ${}^0_6T_{des}(a,b)$ denote the element of ${}^0_6T_{des}$ that is in the ath row and bth column.

Then the six equations in Equation (3.20) may be, for example

$$
\begin{aligned}
{}^0_6T_\theta(1,4) &= {}^0_6T_{des}(1,4)\\
{}^0_6T_\theta(1,5) &= {}^0_6T_{des}(1,5)\\
{}^0_6T_\theta(1,6) &= {}^0_6T_{des}(1,6)\\
{}^0_6T_\theta(2,3) &= {}^0_6T_{des}(2,3)\\
{}^0_6T_\theta(3,3) &= {}^0_6T_{des}(3,3)\\
{}^0_6T_\theta(3,2) &= {}^0_6T_{des}(3,2).
\end{aligned}
$$

One must take care, however, that the last three equations are actually independent. For example, in the case of

$$
{}^0_6T_{des} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 2\\ 0 & -1 & 0 & 1\\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & -1\\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

these six equation will **not** be independent because both ${}^0_6T_{des}(2,3)$ and ${}^0_6T_{des}(3,2)$ are equal to zero and the interation may converge to

$$
T = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 2\\ 0 & -1 & 0 & 1\\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & -1\\ 0 & 0 & 0 & 1 \end{bmatrix} \neq \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 2\\ 0 & -1 & 0 & 1\\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & -1\\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

A more robust approach is to construct a system of 12 equations and six unknowns; in particular,

$$
\begin{aligned}
{}^0_6T_\theta(1,4) &= {}^0_6T_{des}(1,4)\\
{}^0_6T_\theta(1,5) &= {}^0_6T_{des}(1,5)\\
{}^0_6T_\theta(1,6) &= {}^0_6T_{des}(1,6)\\
{}^0_6T_\theta(2,3) &= {}^0_6T_{des}(2,3)\\
{}^0_6T_\theta(3,3) &= {}^0_6T_{des}(3,3)\\
{}^0_6T_\theta(3,2) &= {}^0_6T_{des}(3,2)\\
{}^0_6T_\theta(1,1) &= {}^0_6T_{des}(1,1)\\
{}^0_6T_\theta(1,2) &= {}^0_6T_{des}(1,2)\\
{}^0_6T_\theta(1,3) &= {}^0_6T_{des}(1,3)\\
{}^0_6T_\theta(2,1) &= {}^0_6T_{des}(2,1)\\
{}^0_6T_\theta(2,2) &= {}^0_6T_{des}(2,2)\\
{}^0_6T_\theta(3,1) &= {}^0_6T_{des}(3,1),
\end{aligned}
$$

which are all 12 elements of the position vector (three elements) plus the entire rotation matrix (nine elements). In this case, however, the Jacobian will not be invertible since it will not be square; however, the standard pseudo-inverse which minimizes the norm of the error of the solution of an overdetermined

system can be utilized. In this case

$$\mathbf{J}(\theta_n) = \left[\frac{\partial f_i(\theta)}{\partial \theta_j}\right] = \begin{bmatrix} \frac{\partial f_1(\theta)}{\partial \theta_1} & \frac{\partial f_1(\theta)}{\partial \theta_2} & \frac{\partial f_1(\theta)}{\partial \theta_3} & \cdots & \frac{\partial f_1(\theta)}{\partial \theta_5} & \frac{\partial f_1(\theta)}{\partial \theta_6} \\ \frac{\partial f_2(\theta)}{\partial \theta_1} & \frac{f_2(\theta)}{\partial \theta_2} & & \cdots & & \frac{\partial f_2(\theta)}{\partial \theta_6} \\ \vdots & \vdots & & \ddots & & \vdots \\ \frac{\partial f_{12}(\theta)}{\partial \theta_1} & \frac{f_{12}(\theta)}{\partial \theta_2} & & \cdots & & \frac{\partial f_1 2(\theta)}{\partial \theta_6} \end{bmatrix}$$

which clearly is not square with six columns and 12 rows and, hence, not invertible. The analog of Equation (3.22) which utilizes a pseudo-inverse is

$$\theta_{n+1} = \theta_n - (\mathbf{J}^{\mathbf{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathbf{T}}(\theta_n)\mathbf{f}(\theta_n) \tag{3.23}$$

Appendix B presents C code that implements Newton's method for a six degree of freedom manipulator. Note, that the values for each are constrained such that $-\pi \leq \theta_i \leq \pi$ by adding or subtracting π from the value of θ_i if the value of θ_i becomes less than $-\pi$ or greater than π, respectively. Finally, implementing the program for the same PUMA 560 example from Section 3.3.3, the following represents the iterative evolution of the θ values for a typical program run where the initial conditions were picked randomly:

Iteration	theta1	theta2	theta3	theta4	theta5	theta6
0	-136.23	-139.17	88.74	102.89	137.92	-23.59
1	-2.20	-27.61	178.80	34.10	14.86	21.61
2	44.32	69.98	51.25	-38.03	97.29	20.31
3	69.22	-32.48	114.17	-51.25	109.17	-68.76
4	120.32	56.05	29.90	-11.28	-3.81	-80.29
5	63.89	-20.66	91.33	-115.55	33.74	4.29
6	-70.59	32.21	169.15	-87.01	135.05	161.33
7	-50.98	108.14	54.72	-143.86	141.17	107.12
8	-88.86	53.58	81.08	-156.97	135.54	54.00
9	-173.40	127.80	38.88	-174.21	4.49	-27.35
1	-130.68	72.18	81.86	-104.16	14.69	55.74
11	-76.12	15.76	58.71	106.37	-27.14	-40.86
12	164.40	-155.70	58.08	13.01	43.00	-176.37
13	-18.18	-153.18	148.99	74.92	92.84	-32.89
14	-47.83	-160.55	99.49	45.87	112.59	25.23
15	-99.11	16.87	21.47	-177.35	-30.07	-115.73
16	-75.66	14.52	84.43	-132.83	-9.84	-122.40
17	-162.83	83.18	34.29	100.41	113.24	-93.66
18	-121.64	63.91	57.62	77.15	151.31	-97.00
19	-110.76	77.23	47.40	54.07	160.48	-117.69
20	-114.32	77.03	45.94	47.39	158.26	-129.10
21	-114.30	77.14	45.87	17.15	154.29	-165.67
22	-114.30	77.14	45.87	-12.91	134.74	-28.24
23	-114.30	77.14	45.87	35.38	140.57	26.89
24	-114.30	77.14	45.87	88.80	101.19	89.46
25	-114.30	77.14	45.87	57.85	63.60	69.39
26	-114.30	77.14	45.87	57.38	51.07	78.59
27	-114.30	77.14	45.87	56.02	51.00	79.53
28	-114.30	77.14	45.87	56.02	51.01	79.53
29	-114.30	77.14	45.87	56.02	51.01	79.53

3.5 Conclusions

This chapter outlined various approaches to the robotic manipulator inverse kinematics problem. The foundation for many analytical approaches was presented in Section 3.3.1, and one particular method, namely, Pieper's solution, was presented in complete detail. Numerical approaches were also presented with particular emphasis on Newton's iteration method utilizing a pseudo-inverse approach so that convergence to all twelve desired elements of ${}^0_6T_{des}$ is achieved. Sample C code is presented in Appendix B.

Appendix A: Theorems Relating to Newton's Method

Two theorems regarding convergence of Newton's method are presented here. See [4] for proofs. Since Newton's method is

$$\theta_{n+1} = \theta_n - \mathbf{J}^{-1}(\theta_n)\mathbf{f}(\theta_n)$$

if $\mathbf{g}(\theta) = \theta - \mathbf{J}^{-1}(\theta)\mathbf{f}(\theta)$, convergence occurs when

$$\theta = \mathbf{g}(\theta) \tag{3.24}$$

Theorem A.1 *Let Equation (3.24) have a root $\theta = \alpha$ and $\theta \in \mathbb{R}^n$. Let ρ be the interval*

$$\|\theta - \alpha\| < \rho \tag{3.25}$$

in which the components, $g_i(\theta)$ have continuous first partial derivatives that satisfy

$$\left|\frac{\partial g_i(\theta)}{\partial \theta_j}\right| \leq \frac{\lambda}{n}, \quad \lambda < 1$$

Then

1. *For any θ_0 satisfying Equation (3.25), all iterates θ_n of Equation (3.22) also satisfy Equation (3.25).*
2. *For any θ_0 satisfying Equation (3.25), all iterates θ_n of Equation (3.22) converge to the root α of Equation (3.24) which is unique in Equation (3.25).*

This theorem basically ensures convergence if the initial value, θ_0 is close enough to the root α, where "close enough" means that $\|\theta_0 - \alpha\| < \rho$. A more constructive approach, which gives a test to ensure that θ_0 is "close enough" to α is given by Theorem A.3. First, define the vector and matrix norms as follows:

Definition A.2 Let $\theta \in \mathbb{R}^n$. Define

$$\|\theta\| = \max_i |\theta_i|$$

Let $\mathbf{J} \in \mathbb{R}^{n\times n}$. Define

$$\|\mathbf{J}\| = \max_i \left\{ \sum_j = 1^m \left| j_{ij} \right| \right\}$$

Note that other definitions of vector and matrix norms are possible as well [7].

Theorem A.3 *Let $\theta_0 \in \mathbb{R}^n$ be such that*

$$\|\mathbf{J}(\theta_0)\| \leq a$$

let

$$\|\theta_1 - \theta_0\| \leq b$$

and let

$$\sum_{k=1}^{n} \left| \frac{\partial^2 f_i(\theta)}{\partial\theta_j \partial\theta_k} \right| \leq \frac{c}{n}$$

for all $\theta \in \|\theta - \theta\| \leq 2b$, *where* $i, j \in \{1, \ldots, n\}$. *If*

$$abc \leq \frac{1}{2}$$

then

1. θ_i *defined by Equation (3.22) are uniquely defined as*

$$\|\theta_n - \theta_0\| \leq 2b$$

   and
2. *the iterates converge to some vector,* α *for which* $\mathbf{g}(\alpha) = \mathbf{0}$ *and*

$$\|\theta_n - \theta_0\| \leq \frac{2b}{2^n}$$

Appendix B: Implementation of Newton's Method

This appendix presents C code that implements Newton's method for a six degree of freedom manipulator. It assumes that all the joints are revolute, i.e., the joint angles, θ_i, are the variables to be determined. It has been written not with the goal of complete robustness or efficiency, but rather for a (hopefully) optimal combination of *readability,* robustness, and efficiency, with emphasis on readability. Most of the variables that one may need to tweak are contained in the header file.

The main file is "inversekinematics.c" which utilizes Newton's method to numerically find a solution to the inverse kinematics problem. This file reads the Denavit-Hartenberg parameters for the manipulator from the file "dh.dat," reads the desired configuration for the sixth frame from the file "Tdes.dat," reads the initial values from the file "theta.dat." The other files are:

- "computejacobian.c" which numerically approximates the Jacobian by individually varying the joint angles and computing a finite approximation of each partial derivative;
- "forwardkinematics.c" which computes the forward homogeneous transformation matrix using the Denavit-Hartenberg parameters (including the joint angles, θ_i);
- "homogeneoustransformation.c" which multiplies the six forward transformation matrices to determine the overall homogeneous transformation;
- "matrixinverse.c" which inverts a matrix;
- "matrixproduct.c" which multiplies two matrices;
- "dh.dat" which contains the Denavit-Hartenberg parameters α_{i-1}, a_{i-1}, and d_i;
- "theta.dat" which contains the initial values for the last Denavit-Hartenberg parameter, θ_i; and,
- "inversekinematics.h" which is a header file for the various C files.

On a Unix machine, move all the files to the same directory and compile the program, by typing

```
> gcc *.c -lm
```

at a command prompt. To execute the program type

```
./a.out
```

at a command prompt.

To compile these programs on a PC running Windows using Microsoft Visual C++, it must be incorporated into a project (and essentially embedded into a C++ project) to be compiled.

B.1 File "inversekinematics.c"

```
/* This file is inversekinematics.c
 *
 * This program numerically solves the inverse kinematics problem for
 * an n degree of freedom robotic manipulator.
 *
 * It reads the Denavit-Hartenberg parameters stored in a file named
 * "dh.dat". The format of "dh.dat" is:
 *
 * alpha_0   a_0   d_1
 * alpha_1   a_1   d_2
 * ...
 *
 * The number of degrees of freedom is determined by the number of
 * rows in "dh.dat". For this program, it is assumed that the number
 * of degrees of freedom is six.
 *
 * This program reads the desired configuration from the file
 * "Tdes.dat" and stores it in the matrix Tdes[][].
 *
 * This program reads the initial values for Newton's iteration from
 * the file "theta.dat" and stores them in the array theta[]. The
 * initial values are in degrees.
 *
 * The convergence criterion is that the sum of the squares of the
 * change in joint variables between two successive iterations is less
 * that EPS, which is set in "inversekinematics.h".
 *
 * This program assumes that the d_i are fixed and the joint variables
 * are the theta_i.
 *
 * This program assumes a 6-by-6 Jacobian, where n is the number of
 * degrees of freedom for the system. The six elements utilized in
 * the homogeneous transformation are the first three elements of the
 * fourth column (the position of the origin of the sixth frame and
 * elements (2,3), (3,3), and (3,2) of the rotation submatrix of the
 * homogeneous transformation).
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */

#include "inversekinematics.h"

int main() {
```

```
double **J,**T, **Tdes,**Jinv, *f;
double *alpha,*a,*d,*theta;
double sum;
int i,j,k,l,n;
FILE *fp;

/* Allocate memory for the first line of DH parameters. */
alpha = malloc(sizeof(double));
a = malloc(sizeof(double));
d = malloc(2*sizeof(double));
theta = malloc(2*sizeof(double));

n = 1;
fp = fopen("dh.dat","r");

/* Read the DH parameters and reallocate memory accordingly.
   After reading all the data, n will be the number of DOF+1. */
while(fscanf(fp,"%lf %lf %lf",&alpha[n-1],&a[n-1],&d[n]) != EOF) {
  n++;
  alpha = realloc(alpha,n*sizeof(double));
  a = realloc(a,n*sizeof(double));
  d = realloc(d,(n+1)*sizeof(double));
  theta = realloc(theta,(n+1)*sizeof(double));
}
fclose(fp);

if(n-1 != N) {
  printf("Warning, this code is only written for 6 DOF manipulators!\n");
  exit(1);
}

/* Allocate memory for the actual homogeneous transformation and
   the desired final transformation. */
T=(double **) malloc((unsigned) 4*sizeof(double*));
Tdes=(double **) malloc((unsigned) 4*sizeof(double*));
for(i=0;i<4;i++) {
  T[i]=(double *) malloc((unsigned) 4*sizeof(double));
  Tdes[i] = (double *) malloc((unsigned) 4*sizeof(double));
}


/* Read the desired configuration. */
fp = fopen("Tdes.dat","r");
for(i=0;i<4;i++)
  fscanf(fp,"%lf%lf%lf%lf",&Tdes[i][0],&Tdes[i][1],
             &Tdes[i][2],&Tdes[i][3]);
fclose(fp);

printf("Desired T = \n");
for(i=0;i<4;i++) {
  for(j=0;j<4;j++) {
```

```
    printf("%f\t",Tdes[i][j]);
  }
  printf("\n");
}

/* Allocate memory for the Jacobian, its inverse and a homogeneous
   transformation matrix. */

f=(double *) malloc((unsigned) 2*N*sizeof(double));

J=(double **) malloc((unsigned) 2*N*sizeof(double*));
for(i=0;i<2*N;i++) {
  J[i]=(double *) malloc((unsigned) N*sizeof(double));
}

Jinv=(double **) malloc((unsigned) N*sizeof(double*));
for(i=0;i<N;i++) {
  Jinv[i]=(double *) malloc((unsigned) 2*N*sizeof(double));
}

/* Read the starting values for the iteration. */
fp = fopen("theta.dat","r");
for(i=1;i<=N;i++)
  fscanf(fp,"%lf",&theta[i]);

/* Change the angle values from degrees to radians. */
for(i=1;i<=N;i++)
  theta[i] *= M_PI/180.0;

for(i=1;i<=N;i++) {
  theta[i] = 2.0*(double)rand()/(double)RAND_MAX*M_PI - M_PI;
  printf("%.2f\n",theta[i]*180/M_PI);
}

/* Begin the iteration. The variable i is the number of iterations.
   MAX_ITERATIONS is set in the file "inversekinematics.h". */
i = 0;

while(i<MAX_ITERATIONS) {
  T = forwardkinematics(alpha,a,d,theta,T);
  J = computejacobian(alpha,a,d,theta,T,J);

  Jinv = pseudoinverse(J,Jinv);

  f[0] = T[0][3]-Tdes[0][3];
  f[1] = T[1][3]-Tdes[1][3];
  f[2] = T[2][3]-Tdes[2][3];
  f[3] = T[1][2]-Tdes[1][2];
  f[4] = T[2][2]-Tdes[2][2];
  f[5] = T[2][1]-Tdes[2][1];
  f[6] = T[0][0]-Tdes[0][0];
```

```
    f[7] = T[0][1]-Tdes[0][1];
    f[8] = T[0][2]-Tdes[0][2];
    f[9] = T[1][0]-Tdes[1][0];
    f[10] = T[1][1]-Tdes[1][1];
    f[11] = T[2][0]-Tdes[2][0];

    for(k=0;k<N;k++) {
      for(l=0;l<2*N;l++) {
        theta[k+1] -= Jinv[k][l]*f[l];
      }
      while(fabs(theta[k+1]) > M_PI)
        theta[k+1] -= fabs(theta[k+1])/theta[k+1]*M_PI;
    }

    printf("%d\t",i);
    for(k=0;k<6;k++)
      printf("%.2f\t",theta[k+1]*180/M_PI);
    printf("\n");

    sum = 0.0;
    for(k=0;k<2*N;k++) {
      sum += pow(f[k],2.0);
    }

    if(sum < EPS)
      break;
    i++;
  }

  printf("Iteration ended in %d iterations.\n",i);
  if(i>=MAX_ITERATIONS) {
    printf("Warning!\n");
    printf("The system failed to converge in %d iterations.\n",MAX_ITERATION
    printf("The solution is suspect!\n");
    exit(1);
  }


  printf("Final T = \n");
  for(i=0;i<4;i++) {
    for(j=0;j<4;j++) {
      printf("%f\t",T[i][j]);
    }
    printf("\n");
  }
  printf("Converged ${\\bf \\theta}:$\n");
  for(i=1;i<n;i++)
    printf("& $%.2f^\\circ$\n",theta[i]*180/M_PI);

  return(0);
}
```

B.2 File "computejacobian.c"

```
/* This file is "computejacobian.c".
 *
 * It constructs an approximate * Jacobain by numerically
 * approximating the partial derivatives with * respect to each joint
 * variable. It returns a pointer to a * six-by-six Jacobain.
 *
 * The magnitude of the perturbations is determined by the
 * PERTURBATION, which is set in "inversekinematics.h".
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */


#include "inversekinematics.h"

double** computejacobian(double* alpha,double* a, double* d,
 double* theta, double** T, double** J) {
  double **T1, **T2;
  int i;

  /* Allocate memories for the perturbed homogeneous transformations.
   * T1 is the forward perturbation and T2 is the backward perturbation.
   */
  T1=(double **) malloc((unsigned) (4)*sizeof(double*));
  T2=(double **) malloc((unsigned) (4)*sizeof(double*));
  for(i=0;i<4;i++) {
    T1[i]=(double *) malloc((unsigned) (4)*sizeof(double));
    T2[i]=(double *) malloc((unsigned) (4)*sizeof(double));
  }

    for(i=1;i<=N;i++) {

      /* Compute the actual homogeneous transformation. */
      T = forwardkinematics(alpha,a,d,theta,T);
      theta[i] += PERTURBATION;

      /* Compute the forward perturbation. */
      T1 = forwardkinematics(alpha,a,d,theta,T1);
      theta[i] -= 2.0*PERTURBATION;

      /* Compute the backward perturbation. */
      T2 = forwardkinematics(alpha,a,d,theta,T2);
      theta[i] += PERTURBATION;

      /* Let the Jacobain elements be the average of the forward
       * and backward perturbations.
       */
      J[0][i-1] = ((T1[0][3]-T[0][3])/(PERTURBATION) +
   (T2[0][3]-T[0][3])/(-PERTURBATION))/2.0;
```

```
      J[1][i-1] = ((T1[1][3]-T[1][3])/(PERTURBATION) +
   (T2[1][3]-T[1][3])/(-PERTURBATION))/2.0;

      J[2][i-1] = ((T1[2][3]-T[2][3])/(PERTURBATION) +
   (T2[2][3]-T[2][3])/(-PERTURBATION))/2.0;

      J[3][i-1] = ((T1[1][2]-T[1][2])/(PERTURBATION) +
   (T2[1][2]-T[1][2])/(-PERTURBATION))/2.0;

      J[4][i-1] = ((T1[2][2]-T[2][2])/(PERTURBATION) +
   (T2[2][2]-T[2][2])/(-PERTURBATION))/2.0;

      J[5][i-1] = ((T1[2][1]-T[2][1])/(PERTURBATION) +
   (T2[2][1]-T[2][1])/(-PERTURBATION))/2.0;

      J[6][i-1] = ((T1[0][0]-T[0][0])/(PERTURBATION) +
   (T2[0][0]-T[0][0])/(-PERTURBATION))/2.0;

      J[7][i-1] = ((T1[0][1]-T[0][1])/(PERTURBATION) +
   (T2[0][1]-T[0][1])/(-PERTURBATION))/2.0;

      J[8][i-1] = ((T1[0][2]-T[0][2])/(PERTURBATION) +
   (T2[0][2]-T[0][2])/(-PERTURBATION))/2.0;

      J[9][i-1] = ((T1[1][0]-T[1][0])/(PERTURBATION) +
   (T2[1][0]-T[1][0])/(-PERTURBATION))/2.0;

      J[10][i-1] = ((T1[1][1]-T[1][1])/(PERTURBATION) +
    (T2[1][1]-T[1][1])/(-PERTURBATION))/2.0;

      J[11][i-1] = ((T1[2][0]-T[2][0])/(PERTURBATION) +
    (T2[2][0]-T[2][0])/(-PERTURBATION))/2.0;
    }

    free(T1);
    free(T2);
    return J;
}
```

B.3 File "forwardkinematics.c"

```
/* This file is "forwardkinematics.c".
 *
 * It computes the homogeneous transformation as a function of the
 * Denavit-Hartenberg parameters as presented in Craig, Introduction
 * to Robotics Mechanics and Control, Second Ed.
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */
```

```
#include "inversekinematics.h"

double** forwardkinematics(double *alpha, double *a, double *d,
   double *theta, double **T) {

  double **T1,**T2;
  int i,j,k;

  /* Allocate memory for two additional homogeneous transformations
   * which are necessary to multiply all six transformations.
   */
  T1=(double **) malloc((unsigned) 4*sizeof(double*));
  T2=(double **) malloc((unsigned) 4*sizeof(double*));
  for(i=0;i<4;i++) {
    T1[i]=(double *) malloc((unsigned) 4*sizeof(double));
    T2[i]=(double *) malloc((unsigned) 4*sizeof(double));
  }


  T1 = homogeneoustransformation(alpha[0],a[0],d[1],theta[1],T1);

  /* This loop multiplies all six transformations. The final
   * homogeneous transformation is stored in T and a pointer is
   * returned to T.
   */
  for(i=2;i<=N;i++) {
    T2 = homogeneoustransformation(alpha[i-1],a[i-1],d[i],theta[i],T2);

    T = matrixproduct(T1,T2,T,4);

    for(j=0;j<4;j++)
      for(k=0;k<4;k++)
        T1[j][k] = T[j][k];
  }

  free(T1);
  free(T2);

  return T;
}
```

B.4 File "homogeneoustransformation.c"

```
/* This file is "homogeneoustransformation.c".
 *
 * This function computes a homogeneous transformation as a function
 * of the Denavit-Hartenberg parameters as presented in Craig,
 * Introduction to Robotics Mechanics and Control, Second Ed.
 *
 * The homogeneous transformation is stored in T and a pointer to T is
 * returned.
```

```
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */

#include "inversekinematics.h"

double** homogeneoustransformation(double alpha, double a, double d,
       double theta, double **T) {

  T[0][0] = cos(theta);
  T[0][1] = -sin(theta);
  T[0][2] = 0;
  T[0][3] = a;

  T[1][0] = sin(theta)*cos(alpha*M_PI/180.0);
  T[1][1] = cos(theta)*cos(alpha*M_PI/180.0);
  T[1][2] = -sin(alpha*M_PI/180.0);
  T[1][3] = -d*sin(alpha*M_PI/180.0);

  T[2][0] = sin(theta)*sin(alpha*M_PI/180.0);
  T[2][1] = cos(theta)*sin(alpha*M_PI/180.0);
  T[2][2] = cos(alpha*M_PI/180.0);
  T[2][3] = d*cos(alpha*M_PI/180.0);

  T[3][0] = 0;
  T[3][1] = 0;
  T[3][2] = 0;
  T[3][3] = 1;

  return T;
}
```

B.5 File "matrixinverse.c"

```
/* This file is "matrixinverse.c".
 *
 * This program computes the inverse of a matrix using Gauss-Jordan
 * elimination. Row shifting is only utilized if a diagonal element
 * of the original matrix to be inverted has a magnitude less than
 * DIAGONAL_EPS, which is set in "inversekinematics.h".
 *
 * The inverse matrix is stored in y[][], and a pointer to y is
 * returned.
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */

#include "inversekinematics.h"
```

```
double **matrixinverse(double **a, double **y, int n) {
  double temp,coef;
  double max;
  int max_row;
  int i,j,k;

  /* Initialize y[][] to be the identity element. */
  for(i=0;i<n;i++) {
    for(j=0;j<n;j++) {
      if(i==j)
        y[i][j] = 1;
      else
        y[i][j] = 0;
    }
  }

  /* Gauss-Jordan elimination with selective initial pivoting */

  /* Check the magnitude of the diagonal elements, and if one is less
   *  than DIAGONAL_EPS, then search for an element lower in the same
   *  column with a larger magnitude.
   */
  for(i=0;i<n;i++) {
    if(fabs(a[i][i]) < DIAGONAL_EPS) {
      max = a[i][i];
      max_row = i;
      for(j=i;j<n;j++) {
        if(fabs(a[j][i]) > max) {
          max = fabs(a[j][i]);
          max_row = j;
        }
      }

      if(max < DIAGONAL_EPS) {
        printf("Ill-conditioned matrix encountered. Exiting...\n");
        exit(1);
      }

      /* This loop switches rows if needed. */
      for(k=0;k<n;k++) {
        temp = a[max_row][k];
        a[max_row][k] = a[i][k];
        a[i][k] = temp;

        temp = y[max_row][k];
        y[max_row][k] = y[i][k];
        y[i][k] = temp;
      }
    }
  }
```

```
  /* This is the forward reduction. */
  for(i=0;i<n;i++) {

    coef = a[i][i];
    for(j=n-1;j>=0;j--) {
      y[i][j] /= coef;
      a[i][j] /= coef;
    }

    for(k=i+1;k<n;k++) {
      coef = a[k][i]/a[i][i];
      for(j=n-1;j>=0;j--) {
        y[k][j] -= coef*y[i][j];
        a[k][j] -= coef*a[i][j];
      }
    }
  }

  /* This is the back substitution. */
  for(i=n-1;i>=0;i--) {

    for(k=i-1;k>=0;k--) {
      coef = a[k][i]/a[i][i];
      for(j=0;j<n;j++) {
        y[k][j] -= coef*y[i][j];
        a[k][j] -= coef*a[i][j];
      }
    }
  }

  return y;
}
```

B.6 File "matrixproduct.c"

```
/* This file is "matrix product.c".
 *
 * This file multiples a and b (both square, n by n matrices) and
 * returns a pointer to the matrix c.
 *
 * Copyright (C) 2003 Bill Goodwine.
 *
 */

double** matrixproduct(double **a,double **b,double **c,int n) {
  int i,j,k;

  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      c[i][j] = 0.0;
```

```
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      for(k=0;k<n;k++)
        c[i][j] += a[i][k]*b[k][j];

  return c;

}
```

B.7 File "inversekinematics.h"

```
/*
 * Copyright (C) 2003 Bill Goodwine.
 */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define MAX_ITERATIONS 1000
#define EPS 0.0000000001
#define PERTURBATION 0.001
#define DIAGONAL_EPS 0.0001
#define N 6

double** forwardkinematics(double *alpha, double *a, double *d,
   double *theta, double **T);

double** matrixinverse(double **J, double **Jinv, int n);

double** pseudoinverse(double **J, double **Jinv);

double** computejacobian(double* alpha, double* a, double* d,
 double* theta, double** T, double** J);

double** matrixproduct(double **a,double **b, double **c, int n);

double** homogeneoustransformation(double alpha, double a, double d,
   double theta, double **T);
```

B.8 File "dh.dat"

```
0 0 0
-90 0 0
0 24 6
-90 2 24
90 0 0
-90 0 0
```

B.9 File "Tdes.dat"

```
-0.707106 0 0.707106 12
0 -1 0 12
0.707106 0 0.707106 -12
0 0 0 1
```

B.10 File "theta.dat"

```
24.0
103.0
145.0
215.0
29.2
30.0
```

References

[1] Craig, J.J. (1989). *Introduction to Robotics Mechanics and Control*. Addison-Wesley, Reading, MA.

[2] Denavit, J. and Hartenberg, R.S. (1955). A kinematic notation for lower-pair mechanisms based on matrices, *J. Appl. Mech.,* 215–221.

[3] Murray, R.M., Zexiang, L.I., and Sastry, S.S. (1994). *A Mathematical Introduction to Robotic Manipulation,* CRC Press, Boca Raton, FL.

[4] Gupta, K.C. (1997). *Mechanics and Control of Robots*. Springer-Verlag, Heidelberg.

[5] Paden, B. and Sastry, S. (1988). Optimal kinematic design of 6 manipulators. *Int. J. Robotics Res.,* 7(2), 43–61.

[6] Lee, H.Y. and Liang, C.G. (1988). A new vector theory for the analysis of spatial mechanisms. *Mechanisms and Machine Theory,* 23(3), 209–217.

[7] Raghavan, M. (1990). Manipulator kinematics. In Roger Brockett (ed.), *Robotics: Proceedings of Symposia in Applied Mathematics,* Vol. 41, American Mathematical Society, 21–48.

[8] Isaacson, E. and Keller, H.B. (1966). *Analysis of Numerical Methods*. Wiley, New York.

[9] Naylor, A.W. and Sell, G.R. (1982). *Linear Operator Theory in Engineering and Science*. Springer-Verlag, Heidelberg.