Numerical Inverse Kinematics for a Six-Degree-of-Freedom Manipulator

by

William H. Cordle, III

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Approved:

[illegible]

C. F. Reinholtz, Chairman

Harley H Cudney

H. H. Cudney

[illegible]

H. H. Robertshaw

May 6, 1993

Blacksburg, Virginia

c.2

LD
5655
V855
1993
C673
c.2

Numerical Inverse Kinematics for a Six-Degree-of-Freedom Manipulator

Abstract

This work bridges the gap between theory and practice. The development of general inverse kinematic solution techniques is new, hence few detailed applications of these methods exist. Before methods such as these were available, most commercial manipulators were designed to be geometrically simple, yielding 4th or lower degree governing equations. With the further development and application of these techniques, industry will be capable of implementing more complex manipulators for highly specialized tasks.

A general inverse kinematic analysis technique is applied to an industrial manipulator designed for the inspection of nuclear reactor vessels. The analysis is performed by solving the 16th degree univariate displacement polynomial of the general six-degree-of-freedom arm using an equivalent seven-degree-of-freedom closed-loop spatial chain. All possible combinations of joint angles for a given hand position and orientation are obtained. A region in which the manipulator has the maximum number of solutions is used as a numerical example. The inverse kinematic analysis was programmed in C, which is included in Appendix D.

Acknowledgements

First and foremost, I would like to thank Dr. Charles F. Reinholtz, without whose support my transition from Aerospace to Mechanical Engineering graduate programs would not have been possible. He has provided an atmosphere in which all of us in the Robotics and Mechanisms Group could reach our full potential. I feel very fortunate to have been able to work in such an environment. Recognition must also go to Dr. Harley H. Cudney and Dr. Harry H. Robertshaw for serving on my graduate committee and offering their unique insights regarding this research.

I would like to thank Dr. Hongyou Li* for sharing his vast research experience in and his enthusiasm for the study of manipulator inverse kinematics.

The financial support of the Special Products and Inspection Services Division of Babcock and Wilcox Nuclear Services is gratefully acknowledged.

This work is dedicated to my parents, Leonard and Marna, for their untiring support and encouragement.

* Hongyou Li is referred to as Hong-You Lee in the body of this thesis because that is the name under which he has published.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

This work is in response to a request by Babcock and Wilcox Nuclear Service Company (BWNS) for a kinematic analysis to be performed on a new manipulator. The design of this manipulator was a cooperative effort between the Robotics and Mechanisms Group at Virginia Tech and the Special Products and Inspection Services (SPIS) Division of BWNS. The purpose of the new manipulator is to supplement and eventually replace the current nuclear reactor vessel inspection manipulator, ARIS (Automated Reactor Inspection System). The design of the ARIS manipulator is based upon a polar crane concept with a four-degree-of-freedom (DOF), all-revolute (R), serial arm attached to the boom assembly. ARIS has performed well in service, but the size and complexity inherent to its design make installation and removal of the manipulator a time consuming process. Its weight of approximately sixteen tons and size requiring several tractor trailers for transportation, necessitate assembly before and during installation into the vessel and calibration thereafter. The process of assembly and calibration takes approximately three days before vessel inspections may begin. Disassembly and loading for transport requires about the same amount of time.

The URSULA (Ultrasonic Reactor Scanner Un-Like ARIS) manipulator is a design concept that greatly reduces total vessel inspection time by decreasing the amount of assembly and disassembly required on-site. The URSULA design consists of a 6 DOF, all-revolute (6R) arm secured to a mobile platform. The platform has suction pads that attach to the side of the vessel while the arm, fitted with an ultrasonic transducer array, scans welds in its workspace. After scanning in that area is complete, the tool is replaced by a suction cup from the tool rack on the platform. Once the suction tool is fixed to the vessel, the platform is released and moved to another position for further scanning. Thus, the arm is used for both locomotion and inspection. The relatively small size of URSULA facilitates rapid assembly and calibration. These characteristics also allow for the possibility of two URSULA systems running concurrently, reducing vessel inspection times further. Figure 1.1 shows a photograph of a scale model of the URSULA arm, while Figure 1.2 shows the URSULA system installed in a reactor vessel.

Both forward and inverse kinematic analysises of the URSULA arm must be performed. The forward kinematic analysis uses the manipulator joint angles to find the Cartesian coordinate position and orientation of the manipulator hand or tool plate. This analysis

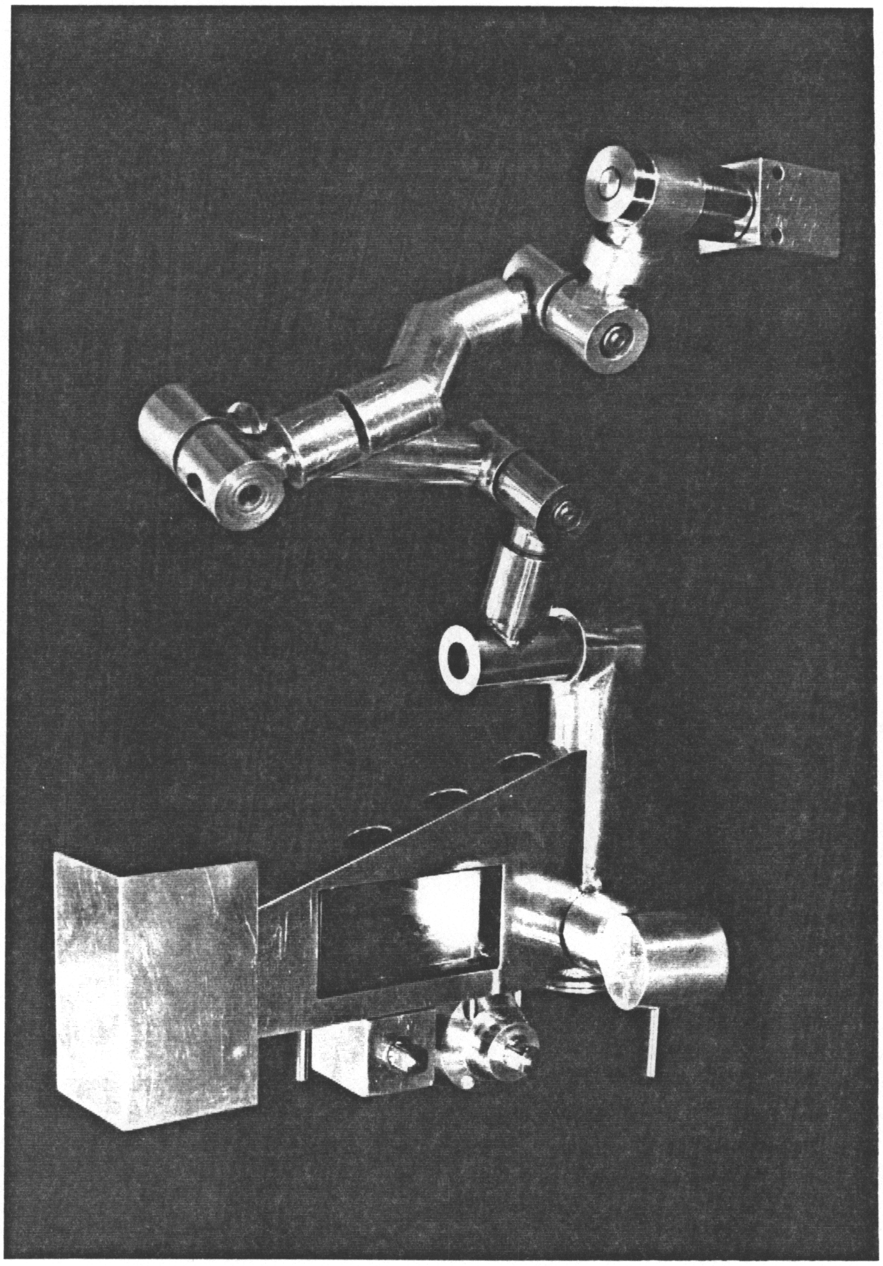

Figure 1.1 Scale Model of the URSULA Arm

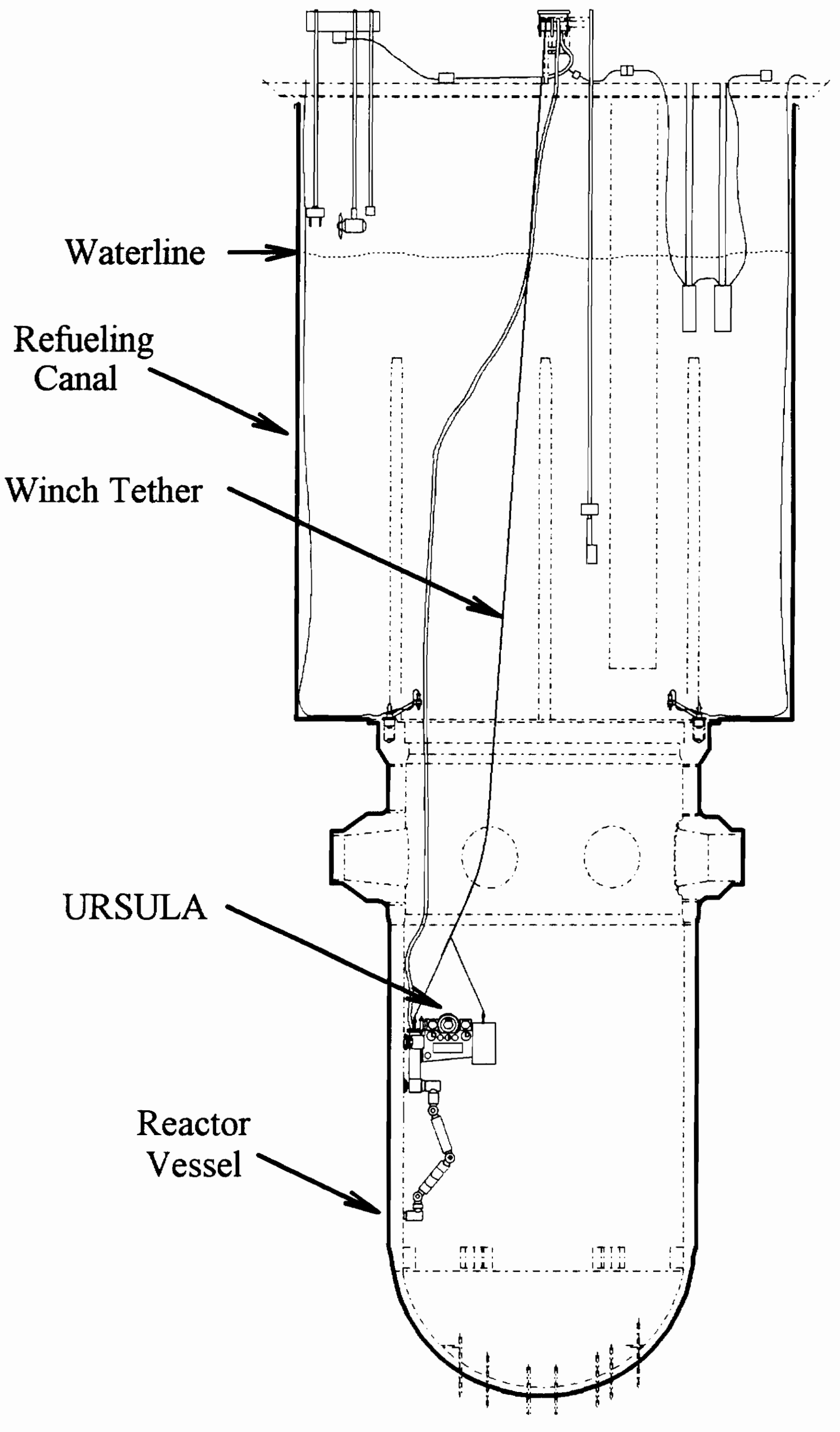

Figure 1.2 URSULA in a Reactor Vessel

requires a full understanding of the manipulator geometry and of mathematical spatial descriptions. The forward analysis also introduces important concepts and nomenclature needed for a full understanding of the inverse kinematic analysis.

1.2 Introduction to Manipulator Inverse Kinematics

The inverse kinematics of a manipulator uses a desired hand position and orientation to find a set of manipulator joint angles. The set of joint angles for a given position and orientation is not unique. In fact, for a general 6R arm, there can be up to sixteen sets of joint angles that will result in a single hand position and orientation [Lee and Liang, 1988; Raghavan and Roth, 1990; Kholi and Osvatic, 1992].

6 DOF manipulators can be separated into two classes by their inverse kinematics. In this paper, the term "closed-form" refers to manipulators whose inverse kinematics can be performed without any iteration. The URSULA manipulator does not have a closed-form solution; and, as a result, iterations are required in order to solve the inverse kinematics. For the six-degree-of-freedom class of manipulators without a closed-form solution, there are primarily two categories of methods for the formulation of the inverse kinematics, numeric and algebraic.

The numeric inverse kinematic techniques for a 6 DOF manipulator typically consist of developing a system of six nonlinear equations in sines and cosines of the unknown joint parameters. Then a method such as Newton-Raphson is used to solve the system [Thomopoulos and Tam, 1991; Goldenberg and Lawrence, 1985]. Although these methods can be efficient, only a single set of joint angles is obtained, convergence is not guaranteed, and an initial guess for the solution is required.

If obstacle avoidance is a requirement, it is often necessary to know more than one set of joint angle solutions for a given end-effector position and orientation. The algebraic methods provide a means for computing all possible joint angle solutions. These methods utilize vector mathematics to develop a univariate polynomial in one of the joint variables. This polynomial can be solved for all possible roots by one of many techniques not requiring initial guesses, such as Laguerre's Method [Press, Flannery, et al., 1992] or the Jenkin's Traub Method [Ralston and Rabinowitz, 1978]. Once the solutions are found to the joint variable in the polynomial, the remaining joint variable solutions are found in closed form. A handful of researchers, such as Albala, Angeles, Duffy, Lee, Liang, Roth, and Ragavan, is responsible for the development of these algebraic methods. Although this work concentrates primarily on methods proposed by Hong-You Lee and Chong-Gao

Liang [1988], it is important to delineate the work of other researchers that have contributed to the study of manipulator inverse kinematics.

1.3 Previous Work

Pieper [1968] is believed to be the first researcher to have proposed iterative numerical methods for the inverse kinematics solution for 6 DOF manipulators of general geometry. He showed that if a naive elimination strategy is used to develop the univariate characteristic polynomial, the polynomial would be of 524,288th degree.

In 1969 Pieper and Roth proposed closed-form solutions for 6 DOF manipulators with special geometries. Specifically, manipulators with the following characteristics were shown to have closed form solutions:

1. Three axes intersecting at a point
2. All axes perpendicular to their neighbors and three sets of pairs of intersecting axes
3. All axes perpendicular to their neighbors, no offsets, and two alternate pairs of axes intersecting.

They also showed that the analysis of a 6R, open-loop manipulator is equivalent to the displacement analysis of a 7R, single-loop spatial mechanism.

Roth, Rastegar, and Scheinman [1973] presented a non-constructive proof using arguments from synthetic geometry to show that the general 6R manipulator has at most 32 distinct solutions for a desired hand position and orientation.

In 1979 Albala and Angeles were able to find a 12x12 determinant in the tangent of the half-angle of one of the joint variables. They were unable to expand the determinant symbolically, but conjectured that the degree of the polynomial would be no more than 48.

Duffy [1980] showed that 6 DOF manipulators with three parallel axes have a closed-form solution. He also devised a method of notation used prevalently in the analysis of mechanisms and robotic manipulators.

In 1980 Duffy and Crane used spherical geometry and dialytic elimination techniques to obtain a 32nd degree univariate polynomial in the tangent of the half-angle of one of the joint variables. Although they obtained proper inverse solutions, false roots were also obtained.

Albala and Pessen [1983] showed that 6R manipulators with all link lengths equal, each axis perpendicular to its neighbors, and no offsets has a fourth degree characteristic polynomial and thus a closed-form inverse kinematic solution.

Tsai and Morgan [1985] were the first to suggest that the 6R manipulator has sixteen solutions at most. They showed this using homotopy continuation techniques on manipulators of varying geometries.

Lee and Liang [1988] derived a 16th degree univariate polynomial in the tangent of the half-angle of one of the joint variables. This was the first complete solution to this problem and verified the belief of Tsai and Morgan that there were at most sixteen solutions for a 6 DOF general manipulator. Lee and Liang derived fourteen linearly independent equations using a new vector theory based on the work of Duffy [1980]. Through dialytic elimination, the fourteen equations are reduced to the univariate polynomial in the tangent of the half-angle of a joint variable. Using the same method, they have also analyzed manipulators with prismatic (P) joints [1987].

In 1990 Ragavan and Roth [1990a] using 4x4 transformation matrices, derived a set of fourteen linearly independent equations. Using linear algebra and dialytic elimination, they obtained a 24th degree univariate polynomial. A 16th degree polynomial is obtained by dividing out two known spurious roots of multiplicity four. This method was applied to manipulators containing prismatic joints as well as all-revolute manipulators [Raghavan and Roth, 1990b,c; Kholi and Osvatic, 1992].

Lee and Reinholtz [1992] showed how the methods proposed by Lee [1990] could be used to find the univariate polynomial of minimum degree from the fewest closure equations for any 6 DOF serial-chain manipulator. For example, the 16th degree polynomial for a manipulator with four revolute joints and a cylindric joint (4RC) can be derived from only six closure equations.

2. URSULA Forward Kinematics

2.1 Spatial Descriptions

In order for a robot to manipulate an object in space, a method for describing the position and orientation of that object with respect to the manipulator is required. Once a coordinate frame is established, the position of an object in space with respect to that reference frame can be described using a 3×1 position vector, $\vec{P}$, as shown in Figure 2.1.

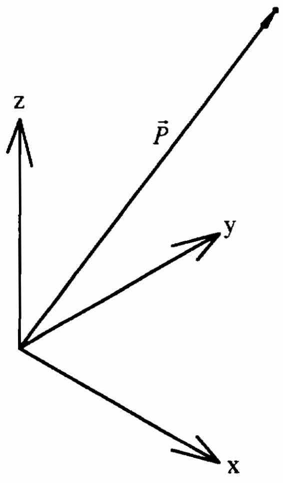

Figure 2.1 Position Vector, $\vec{P}$

where

$$\vec{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2.1}$$

A coordinate system must be attached to an object if its orientation relative to a given frame is to be described. If the coordinate system attached to the object is referred to as frame $\{2\}$ and the fixed frame is referred to as frame $\{1\}$ (Fig 2.2), the orientation of frame $\{2\}$ with respect to frame $\{1\}$ is given by the rotation matrix, ${}^1\mathbf{R}_2$.

$${}^1\mathbf{R}_2 = \begin{bmatrix} \hat{x}_2 \cdot \hat{x}_1 & \hat{y}_2 \cdot \hat{x}_1 & \hat{z}_2 \cdot \hat{x}_1 \\ \hat{x}_2 \cdot \hat{y}_1 & \hat{y}_2 \cdot \hat{y}_1 & \hat{z}_2 \cdot \hat{y}_1 \\ \hat{x}_2 \cdot \hat{z}_1 & \hat{y}_2 \cdot \hat{z}_1 & \hat{z}_2 \cdot \hat{z}_1 \end{bmatrix} \tag{2.2}$$

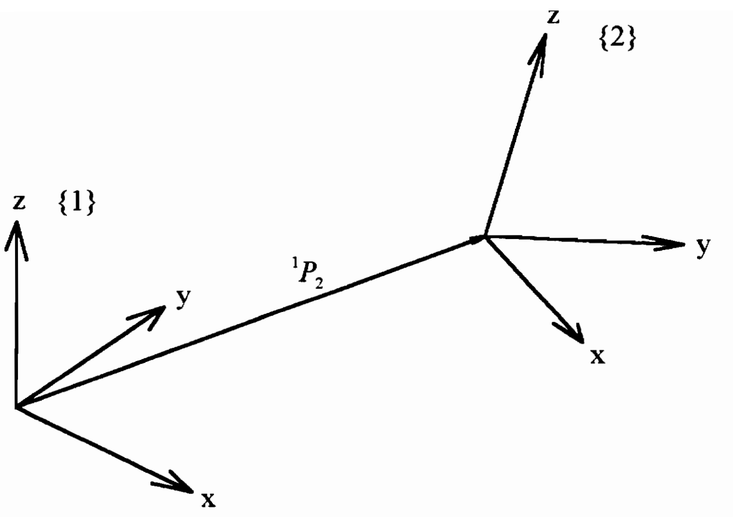

Figure 2.2 Two Coordinate Frames

The symbol $(\cdot)$ is used to denote the scalar or dot product of two vectors. If the position vector, ${}^1\vec{P}_2$, locates the origin of frame {2} with respect to frame {1}, a transformation matrix relating the position and orientation of {2} with respect to {1} may be constructed as follows [Craig, 1989]:

$$ {}^1\mathbf{T}_2 = \begin{bmatrix} {}^1\mathbf{R}_2 & {}^1\vec{P}_2 \\ \{0,0,0\} & 1 \end{bmatrix} \tag{2.3}$$

The advantage of this description of one frame with respect to another is that compound transformations can be calculated by multiplying the respective transformations together. If frame {2} is known with respect to frame {1} and frame {3} is known with respect to frame {2}, then frame {3} is known with respect to frame {1} through the following relation:

$$ {}^1\mathbf{T}_3 = {}^1\mathbf{T}_2 \, {}^2\mathbf{T}_3 \tag{2.4}$$

If the transformation giving reference frame {2} with respect to frame {1} is known, the transformation relating frame {1} to frame {2} is found by inverting the known transformation as follows [Craig, 1989]:

$$ {}^2\mathbf{T}_1 = {}^1\mathbf{T}_2^{-1} = \begin{bmatrix} {}^1\mathbf{R}_2^{\mathrm{T}} & -{}^1\mathbf{R}_2^{\mathrm{T}} \, {}^1\vec{P}_2 \\ \{0,0,0\} & 1 \end{bmatrix} \tag{2.5}$$

2.2 Forward Kinematics

The task of determining the position and orientation of the end-effector, when a set of manipulator joint angles is known, is called forward kinematic analysis. By assigning a coordinate frame to each link of the manipulator, transformation matrices can be used to calculate the position and orientation of the end-effector with respect to a fixed reference frame, typically attached to the base of the manipulator. The first step in solving the forward kinematics is to assign the coordinate frames to the manipulator.

2.2.1 Assigning Coordinate Frames to the Manipulator

Figure 2.3 shows the assignment of link frames to the URSULA manipulator. The manipulator joints are numbered 1 through 6, while the coordinate frames are numbered 1 through 7. The coordinate frames are centered at each joint and are considered fixed to the previous link. Therefore, coordinate frame 1 is centered at joint one, but is fixed to the base of the manipulator. Coordinate frame 7 is attached to the tool plate. The following guidelines should be used when assigning coordinate frames to links:

- $\hat{z}_i$ is along joint axis i
- $\hat{x}_i$ is along the common normal between $\hat{z}_{i-1}$ and $\hat{z}_i$
- $\hat{y}_i$ is defined by $\hat{z}_i \times \hat{x}_i$
- the origin of frame i is located at the intersection of $\hat{z}_i$ and $\hat{x}_i$

Coordinate frames are attached to the links so that only two rotations, one about the $\hat{x}_i$ and one about the $\hat{z}_i$ axis, are needed to fully describe the orientation of one coordinate frame with respect to another. The amount of rotation about the $\hat{x}_i$ axis is a fixed property of the link, while the rotation about the $\hat{z}_i$ axis is due to the rotation of joint i. The translation between frames is described along the rotated link.

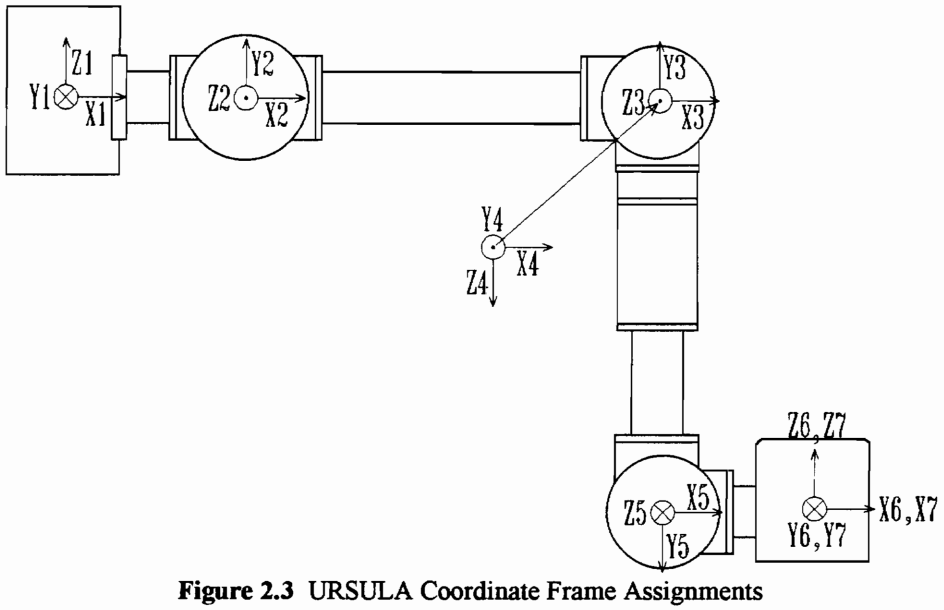

Figure 2.3 URSULA Coordinate Frame Assignments
(URSULA shown in the zero position)

2.2.2 Defining Manipulator Geometry

In order to fully understand the general transforms used to calculate the forward kinematics, certain manipulator link parameters must be defined.

$a_{i(i+1)}$:	the distance between $\hat{z}_i$ and $\hat{z}_{i+1}$ along $\hat{x}_i$ (link length)
S_i:	the distance between $\hat{x}_i$ and $\hat{x}_{i+1}$ along the $\hat{z}_i$ axis (link offset)
θ_i:	the angle between $\hat{x}_i$ and $\hat{x}_{i+1}$ about the $\hat{z}_i$ axis (joint angle)
$\alpha_{i(i+1)}$:	the angle between $\hat{z}_i$ and $\hat{z}_{i+1}$ about the $\hat{x}_i$ axis (link twist)

These values are known as Denavit-Hartenberg link parameters. Table 2.1 lists the geometry of URSULA in terms of the link parameters shown in Figure 2.4.

Table 2.1 URSULA Link Parameters

i	$a_{i(i+1)}$	S_i	$\alpha_{i(i+1)}$	θ_i
1	14.0	0	$\pi/2$	θ_1
2	31.125	0	0	θ_2
3	0	0	$\pi/2$	θ_3
4	0	31.125	$\pi/2$	θ_4
5	11.5	0	$\pi/2$	θ_5
6	0	0	0	θ_6

Note that all parameters are constant except for the six joint angles, θ_i.

2.2.3 Forward Transformation Matrices

As stated previously, the orientation of frame i+1 with respect to i may be expressed in terms of a rotation about $\hat{z}_i$ and a rotation about $\hat{x}_i$. This may be written as the multiplication of two rotation matrices as follows:

$$\mathbf{M}_i = \begin{bmatrix} c_i & -s_i & 0 \\ s_i & c_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{M}_{i(i+1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{i(i+1)} & -s_{i(i+1)} \\ 0 & s_{i(i+1)} & c_{i(i+1)} \end{bmatrix} \tag{2.6}$$

where

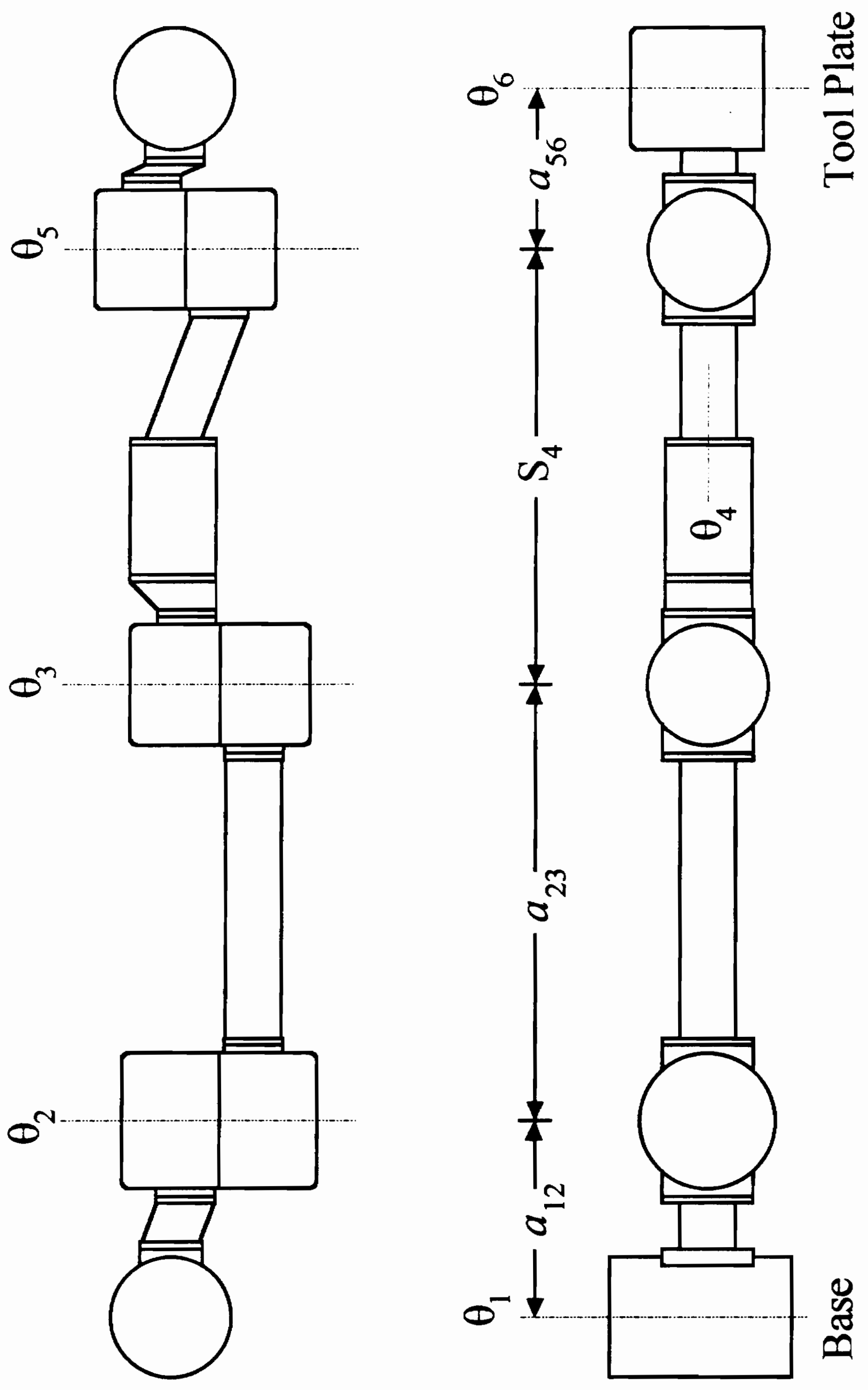

Figure 2.4 URSULA Denavit-Hartenberg Link Parameters

$$c_i = \cos(\theta_i) \quad c_{i(i+1)} = \cos\left(\alpha_{i(i+1)}\right)$$
$$s_i = \sin(\theta_i) \quad s_{i(i+1)} = \sin\left(\alpha_{i(i+1)}\right) \tag{2.7}$$

The rotation matrix $\mathbf{M}_{i(i+1)}$ is constant because it is a function of link twist only, while $\mathbf{M}_i$ is a function of the unknown joint angles. The position of frame i+1 with respect to i is a function of link geometry and joint angle θ_i. This may be written in the following form:

$$\mathbf{M}_i \begin{pmatrix} a_{i(i+1)} \\ 0 \\ S_i \end{pmatrix} \tag{2.8}$$

Now that the position and orientation of frame i+1 with respect to i is known, the forward transformation relating any two adjacent links is known.

$$^{i}\mathbf{T}_{i+1} = \begin{bmatrix} \mathbf{M}_i\,\mathbf{M}_{i+1} & \mathbf{M}_i \begin{pmatrix} a_{i(i+1)} \\ 0 \\ S_i \end{pmatrix} \\ \{0,0,0\} & 1 \end{bmatrix} \tag{2.9}$$

In its most general form, the transformation is written as follows [McKerrow, 1991]:

$$^{i}\mathbf{T}_{i+1} = \begin{bmatrix} c_i & -s_i c_{i(i+1)} & s_i s_{i(i+1)} & a_{i(i+1)} c_i \\ s_i & c_i c_{i(i+1)} & -c_i s_{i(i+1)} & a_{i(i+1)} s_i \\ 0 & s_{i(i+1)} & c_{i(i+1)} & S_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

The transformation matrices obtained by using equation (2.10) in conjunction with the link parameters given in Table 2.1 follow:

$$^{1}\mathbf{T}_2 = \begin{bmatrix} c_1 & 0 & s_1 & a_{12}c_1 \\ s_1 & 0 & -c_1 & a_{12}s_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{2}\mathbf{T}_3 = \begin{bmatrix} c_2 & -s_2 & 0 & a_{23}c_2 \\ s_2 & c_2 & 0 & a_{23}s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{3}\mathbf{T}_4 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}\mathbf{T}_5 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & S_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{5}\mathbf{T}_6 = \begin{bmatrix} c_5 & 0 & s_5 & a_{56}c_5 \\ s_5 & 0 & -c_5 & a_{56}s_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^{6}\mathbf{T}_7 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

If the six joint angles are known, the position and orientation of the tool plate with respect to the base of the manipulator can be calculated by simply multiplying the link transforms.

$$ {}^{1}\mathbf{T}_{7} = {}^{1}\mathbf{T}_{2}\,{}^{2}\mathbf{T}_{3}\,{}^{3}\mathbf{T}_{4}\,{}^{4}\mathbf{T}_{5}\,{}^{5}\mathbf{T}_{6}\,{}^{6}\mathbf{T}_{7} \qquad (2.12) $$

The forward kinematic analysis for the URSULA arm is now complete. Given a set of joint angles, $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$, and θ_6, the position and orientation of any point on the manipulator can be determined. For this analysis, the position and orientation of the tool plate are desired. Although computing the forward kinematics is not necessary for calculating the inverse kinematics, it provides a means for validating inverse kinematic results.

3. URSULA Inverse Kinematics

3.1 Loop Closure

In order to introduce the application of loop closure to the analysis of open serial-chain manipulators, a 3R planar manipulator will be analyzed as a 4R closed-loop mechanism. If the links are replaced by vectors, $\vec{r}_i$, and the position of the hand is denoted by, $\vec{R}_H$, as indicated in Figure 3.1, the vector loop closure equation may be written as follows:

$$\vec{R} = \vec{r}_1 + \vec{r}_2 + \vec{r}_3 - \vec{R}_H = \vec{0} \qquad (3.1)$$

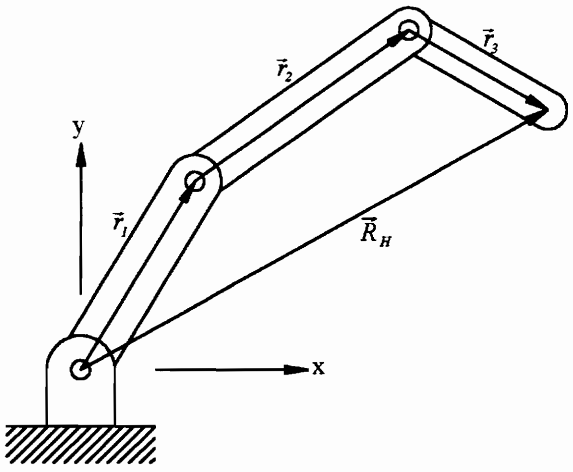

Figure 3.1 Three-Degree-of-Freedom All-Revolute Manipulator

Vector loop equation (3.1) may be written in terms of unit vectors and magnitudes as follows:

$$\vec{R} = r_1\hat{r}_1 + r_2\hat{r}_2 + r_3\hat{r}_3 - \vec{R}_H = \vec{0} \qquad (3.2)$$

All real solutions to this equation also satisfy the positional constraints of the physical system. While there may be complex solutions to this equation, these solutions are not physically realizable. By differentiating $\vec{R}$ with respect to time, it is possible to perform velocity and acceleration analyses as well. The loop closure analysis of this mechanism is identical to that of the four-bar linkage [Mabie and Reinholtz, 1987].

In a similar manner, the 6R manipulator can be modeled as a 7R closed-loop spatial mechanism. The displacement analysis of the closed-loop mechanism is equivalent to the inverse kinematic analysis of the 6R manipulator [Lee and Liang, 1988].

3.2 Assigning Vectors to the Manipulator

In order to develop the loop closure equation, vectors and coordinate frames must be assigned to the manipulator. Coordinate frames are assigned to the manipulator in the same manner as in that of the forward kinematic analysis, while for convenience, the unit vectors are defined with the following convention:

$$\hat{a}_{i(i+1)} = \hat{x}_i,$$
$$\hat{s}_i = \hat{z}_i,$$
$$\hat{b}_i = \hat{y}_i = \hat{s}_i \times \hat{a}_{(i-1)i},$$

while $\vec{R}_H$ is the desired hand position vector, and $\hat{a}_{67}$ and $\hat{s}_7$ describe the desired hand orientation. Figure 3.2 shows the vector assignments and Denavit-Hartenberg parameters for a general 6R manipulator.

The elements of the hand position vector and rotation matrix are defined as follows:

$$\vec{R}_H = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{3.3}$$

$$\begin{bmatrix} \hat{a}_{67} & \hat{b}_6 & \hat{s}_7 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.4}$$

The resulting loop closure equation for a general 6 DOF manipulator is the following:

$$\begin{aligned} \vec{R} = S_1\hat{s}_1 + a_{12}\hat{a}_{12} + S_2\hat{s}_2 + a_{23}\hat{a}_{23} + S_3\hat{s}_3 + a_{34}\hat{a}_{34} + S_4\hat{s}_4 + \\ a_{45}\hat{a}_{45} + S_5\hat{s}_5 + a_{56}\hat{a}_{56} + S_6\hat{s}_6 + a_{67}\hat{a}_{67} - \vec{R}_H = \vec{0} \end{aligned} \tag{3.5}$$

The loop closure equation for URSULA is obtained by substituting the link parameters for URSULA into the general loop closure equation.

$$\vec{R} = a_{12}\hat{a}_{12} + a_{23}\hat{a}_{23} + S_4\hat{s}_4 + a_{56}\hat{a}_{56} - \vec{R}_H = \vec{0} \tag{3.6}$$

3.3 The Three Closure Equations

Lee and Liang [1988] showed that the 16th degree univariate polynomial for the closed-loop 7R mechanism of general geometry could be derived from 14 scalar closure equations. The fourteen closure equations consist of scalar, triple scalar, and quadruple scalar products of two subchains in the vector loop equation. The subchains result from

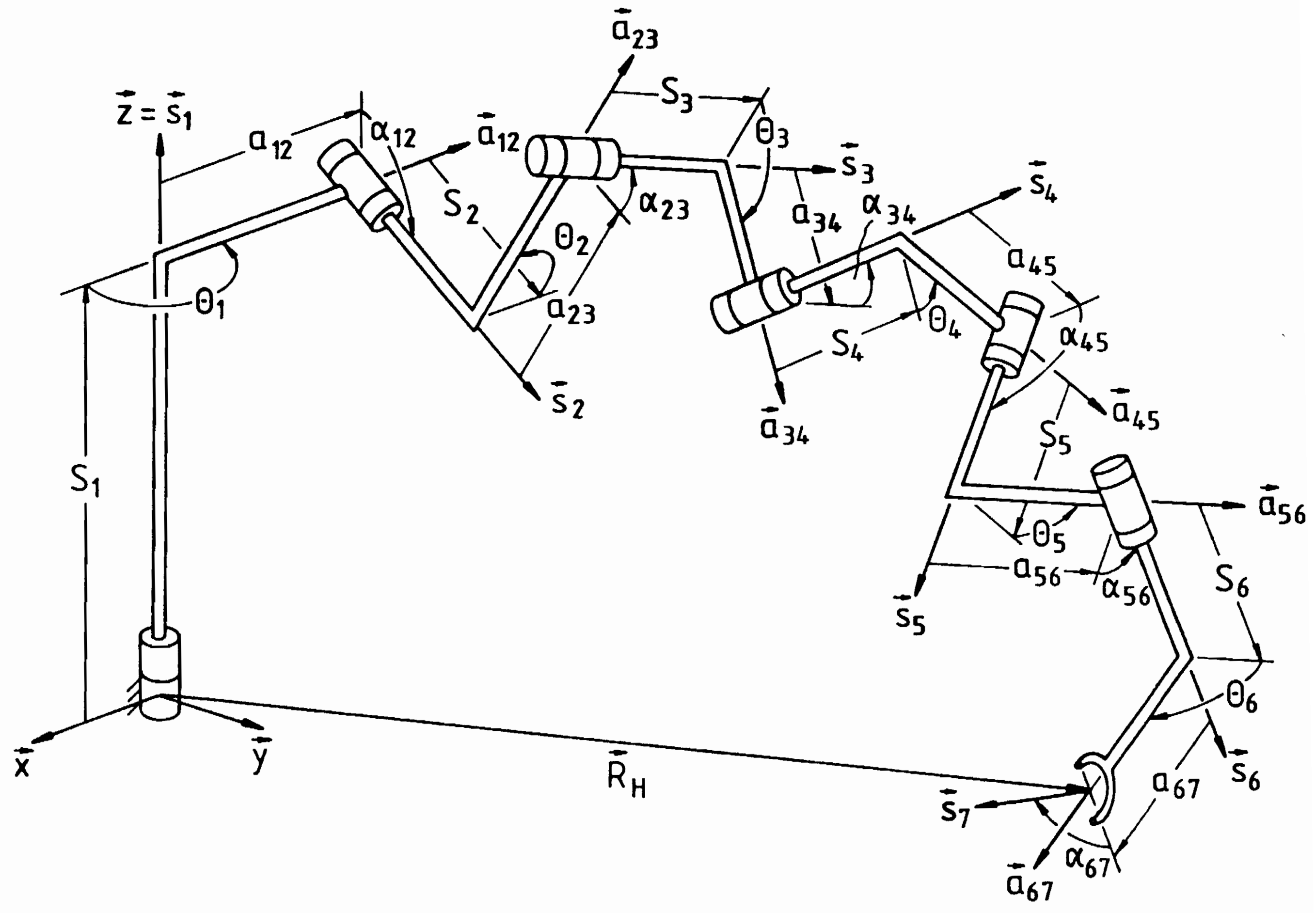

Figure 3.2 Vector Assignments and Denavit-Hartenberg Parameters for a General 6R Manipulator

the disconnection of two joints in the manipulator. The vector loop equation may be rewritten as the difference of the two subchains.

$$\vec{R} = \vec{R}_B - \vec{R}_A = \vec{0} \tag{3.7}$$

If a general 6 DOF manipulator is disconnected at joints *i* and *j*, the resulting 14 scalar equations are as follows:

$$\left[\hat{s}_i \cdot \hat{a}_{(j-1)j}\right]_A = \left[\hat{s}_i \cdot \hat{a}_{(j-1)j}\right]_B \tag{3.8}$$

$$\left[\hat{s}_i \cdot \hat{b}_j\right]_A = \left[\hat{s}_i \cdot \hat{b}_j\right]_B \tag{3.9}$$

$$\left[\hat{s}_i \cdot \hat{s}_j\right]_A = \left[\hat{s}_i \cdot \hat{s}_j\right]_B \tag{3.10}$$

$$\left[\vec{R} \times \hat{s}_i \cdot \hat{a}_{(j-1)j}\right]_A = \left[\vec{R} \times \hat{s}_i \cdot \hat{a}_{(j-1)j}\right]_B \tag{3.11}$$

$$\left[\vec{R} \times \hat{s}_i \cdot \hat{b}_j\right]_A = \left[\vec{R} \times \hat{s}_i \cdot \hat{b}_j\right]_B \tag{3.12}$$

$$\left[\vec{R} \times \hat{s}_i \cdot \hat{s}_j\right]_A = \left[\vec{R} \times \hat{s}_i \cdot \hat{s}_j\right]_B \tag{3.13}$$

$$\left[\vec{R} \cdot \hat{a}_{(j-1)j}\right]_A = \left[\vec{R} \cdot \hat{a}_{(j-1)j}\right]_B \tag{3.14}$$

$$\left[\vec{R} \cdot \hat{b}_j\right]_A = \left[\vec{R} \cdot \hat{b}_j\right]_B \tag{3.15}$$

$$\left[\vec{R} \cdot \hat{s}_j\right]_A = \left[\vec{R} \cdot \hat{s}_j\right]_B \tag{3.16}$$

$$\left[\vec{R} \cdot \hat{s}_i\right]_A = \left[\vec{R} \cdot \hat{s}_i\right]_B \tag{3.17}$$

$$\left[\vec{R} \cdot \vec{R}\right]_A = \left[\vec{R} \cdot \vec{R}\right]_B \tag{3.18}$$

$$\left[\left(\tfrac{1}{2}(\vec{R} \cdot \vec{R})\hat{s}_i - (\vec{R} \cdot \hat{s}_i)\vec{R}\right) \cdot \hat{a}_{(j-1)j}\right]_A = \left[\left(\tfrac{1}{2}(\vec{R} \cdot \vec{R})\hat{s}_i - (\vec{R} \cdot \hat{s}_i)\vec{R}\right) \cdot \hat{a}_{(j-1)j}\right]_B \tag{3.19}$$

$$\left[\left(\tfrac{1}{2}(\vec{R} \cdot \vec{R})\hat{s}_i - (\vec{R} \cdot \hat{s}_i)\vec{R}\right) \cdot \hat{b}_j\right]_A = \left[\left(\tfrac{1}{2}(\vec{R} \cdot \vec{R})\hat{s}_i - (\vec{R} \cdot \hat{s}_i)\vec{R}\right) \cdot \hat{b}_j\right]_B \tag{3.20}$$

$$\left[\left(\tfrac{1}{2}(\vec{R} \cdot \vec{R})\hat{s}_i - (\vec{R} \cdot \hat{s}_i)\vec{R}\right) \cdot \hat{s}_j\right]_A = \left[\left(\tfrac{1}{2}(\vec{R} \cdot \vec{R})\hat{s}_i - (\vec{R} \cdot \hat{s}_i)\vec{R}\right) \cdot \hat{s}_j\right]_B \tag{3.21}$$

where

$$\vec{R}_A = -\left(S_i\hat{s}_i + \hat{a}_{i(i+1)}a_{i(i+1)} + S_{i+1}\hat{s}_{i+1} + \hat{a}_{(i+1)(i+2)}a_{(i+1)(i+2)} + \cdots + S_{j-1}\hat{s}_{j-1} + \hat{a}_{(j-1)j}a_{(j-1)j}\right) \tag{3.22}$$

$$\vec{R}_B = \left(S_j\hat{s}_j + \hat{a}_{j(j+1)}a_{j(j+1)} + S_{j+1}\hat{s}_{j+1} + \hat{a}_{(j+1)(j+2)}a_{(j+1)(j+2)} + \cdots - \vec{R}_H + S_{i-1}\hat{s}_{i-1} + \hat{a}_{(i-1)i}a_{(i-1)i}\right) \tag{3.23}$$

$$\vec{R}_A = \vec{R}_B \tag{3.24}$$

Figure 3.3 shows the general 6R manipulator disconnected at joints 3 and 6, resulting in subchain A $(P_A Q_A)$ and subchain B $(P_B Q_B)$.

For the general 6 DOF manipulator, all 14 equations would need to be evaluated in order to calculate the univariate polynomial of 16th degree. URSULA is considered to have special geometry, because the first four joints operate in a single plane. This characteristic is taken into account when choosing at which joints to disconnect the manipulator loop equation. If joint 1 and joint 4 are chosen as the disconnection points, only three of the fourteen equations are required [Lee and Reinholtz, 1992]. Equations (3.13), (3.17), and (3.18) will be referred to as the closure equations and are sufficient to determine the univariate polynomial for the URSULA manipulator. The three closure equations are written as follows:

$$\left[\vec{R}\times\hat{s}_1\cdot\hat{s}_4\right]_A = \left[\vec{R}\times\hat{s}_1\cdot\hat{s}_4\right]_B \tag{3.25}$$

$$\left[\vec{R}\cdot\hat{s}_1\right]_A = \left[\vec{R}\cdot\hat{s}_1\right]_B \tag{3.26}$$

$$\left[\vec{R}\cdot\vec{R}\right]_A = \left[\vec{R}\cdot\vec{R}\right]_B \tag{3.27}$$

where

$$\vec{R}_A = -\left(a_{12}\hat{a}_{12} + a_{23}\hat{a}_{23}\right) \tag{3.28}$$

$$\vec{R}_B = \left(S_4\hat{s}_4 + a_{56}\hat{a}_{56} - \vec{R}_H\right) \tag{3.29}$$

Before the closure equations can be evaluated, methods for calculating the scalar and triple scalar products must be introduced.

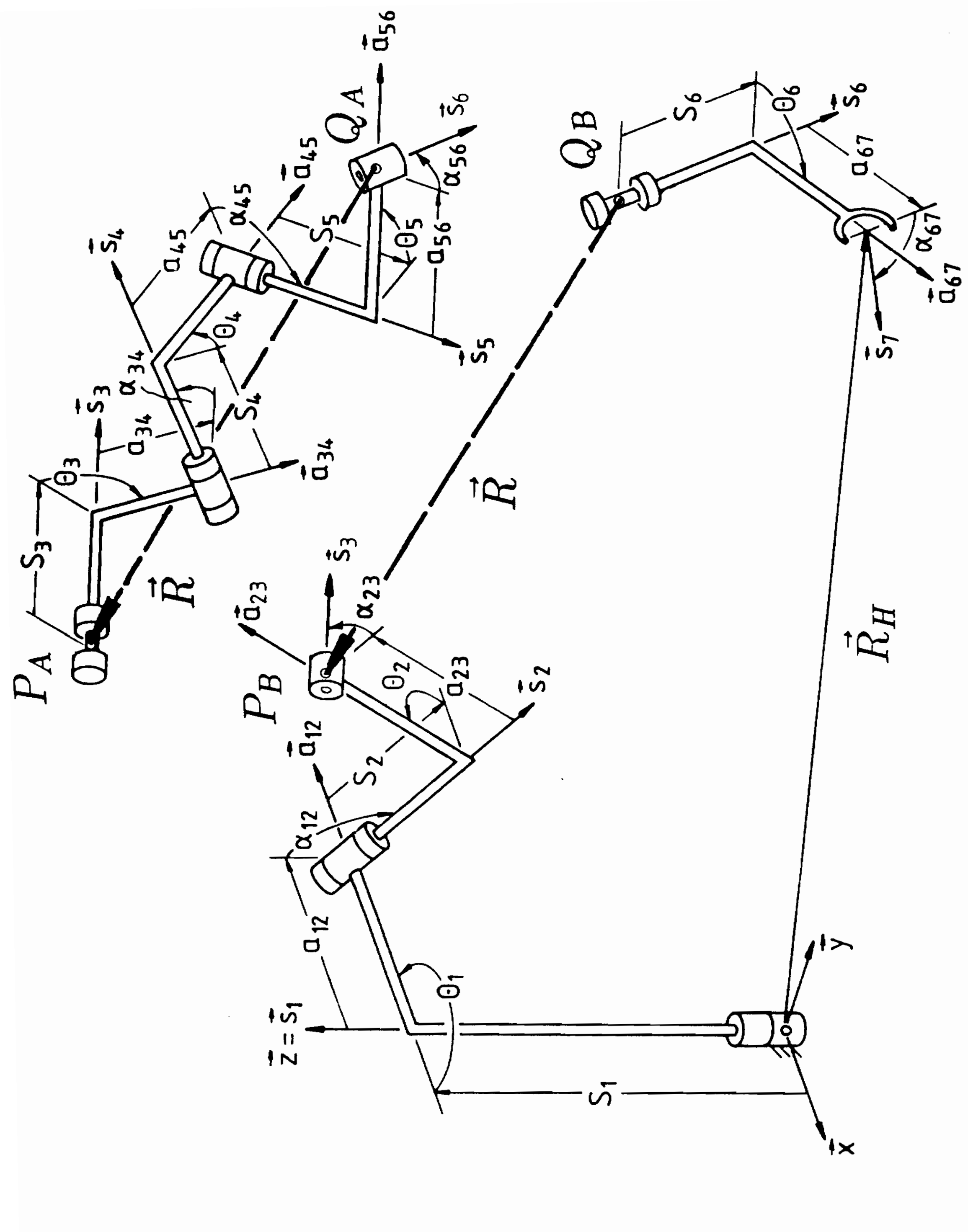

Figure 3.3 Subchains A and B for a General 6R Manipulator

3.3.1 Scalar and Triple Scalar Products

To help visualize the calculation of the scalar and triple scalar products, the closed-loop equation for the URSULA manipulator can be expressed as shown below.

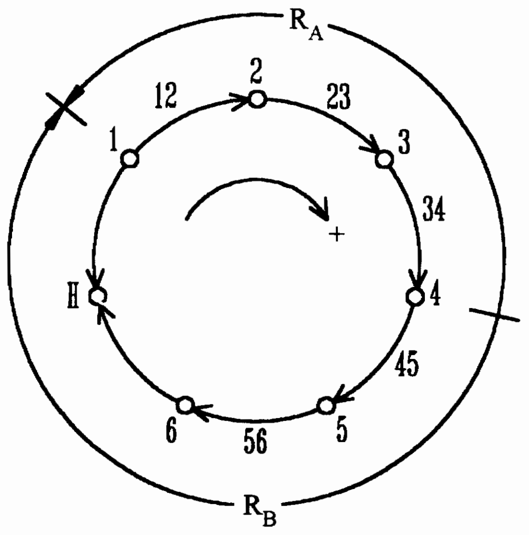

Figure 3.4 Visualization of Vector Loop Closure Equation for URSULA

The points along the closed loop represent the positions of the joints and the hand, while the clockwise arrow designates the convention for positive evaluation. Subchains A and B are also depicted.

3.3.1.1 Scalar Products

The vector scalar product or dot product of any two unit vectors in a chain is a function of the included joint angles only. The resulting multivariate polynomial is linear in the sines and cosines of the included joint angles. The vector scalar product of two vectors may be written

$$\vec{v}_1 \cdot \vec{v}_2 = \vec{v}_2 \cdot \vec{v}_1 = a_x b_x + a_y b_y + a_z b_z \tag{3.30}$$

where

$$\begin{aligned} \vec{v}_1 &= a_x \hat{i} + a_y \hat{j} + a_z \hat{k} \\ \vec{v}_2 &= b_x \hat{i} + b_y \hat{j} + b_z \hat{k} \end{aligned} \tag{3.31}$$

In order for equation (3.30) to be correct, both vectors must be expressed in the same coordinate frame. One vector may be rotated into the coordinate frame of the other, or both vectors could be rotated into a new reference frame before calculation of the scalar product. The method adopted here rotates the vector farthest in the chain into the frame of the other vector. For example, to calculate the dot product $\hat{s}_1 \cdot \hat{a}_{34}$ in subchain A, the vector $\hat{a}_{34}$ would be rotated into the coordinate frame attached to $\hat{s}_1$ before the product was calculated. This is done as follows:

$$[\hat{s}_1 \cdot \hat{a}_{34}]_A = \hat{s}_1 \cdot (\mathbf{M}_1\mathbf{M}_{12}\mathbf{M}_2\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\hat{a}_{34}) \tag{3.32}$$

where the rotation matrices are defined in equation (2.6). The sequence of rotation matrices required to successfully evaluate this dot product may be obtained by using Figure 3.4. Since the dot product of $\hat{s}_1$ and $\hat{a}_{34}$ in subchain A is desired, rotation matrices 1 through 34 are needed.

The rotation matrices $\mathbf{M}_1$ and $\mathbf{M}_{34}$ may be eliminated in the above sequence because they are rotations about the $\hat{s}_1$ and $\hat{a}_{34}$ axes, respectfully, and will have no effect on the result. Since $\hat{a}_{34}$ represents the $\hat{x}$ axis in frame $\{3\}$ and $\hat{s}_1$ represents the $\hat{z}$ axis in frame $\{1\}$, Equation (3.32) may be rewritten as

$$[\hat{s}_1 \cdot \hat{a}_{34}]_A = \hat{z}\mathbf{M}_{12}\mathbf{M}_2\mathbf{M}_{23}\mathbf{M}_3\hat{x} \tag{3.33}$$

where

$$\hat{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ and } \hat{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Calculating the scalar product in this fashion is defined as positive evaluation because the order in which the product was performed is the clockwise convention defined in Figure 3.4. The scalar product could have just as easily been evaluated as follows with the same result.

$$\begin{aligned} [\hat{s}_1 \cdot \hat{a}_{34}]_A &= [\hat{a}_{34} \cdot \hat{s}_1]_A \\ &= \hat{a}_{34}(\mathbf{M}_3^{-1}\mathbf{M}_{23}^{-1}\mathbf{M}_2^{-1}\mathbf{M}_{12}^{-1}\hat{s}_1) \\ &= \hat{x}\mathbf{M}_3^{-1}\mathbf{M}_{23}^{-1}\mathbf{M}_2^{-1}\mathbf{M}_{12}^{-1}\hat{z} \end{aligned} \tag{3.34}$$

All three closure equations will require the evaluation of vector scalar products.

3.3.1.2 Triple Scalar Products

In order to calculate the vector triple scalar product, the vectors involved must be expressed in the same reference frame. Once this is done, the triple scalar product is calculated as follows:

$$\vec{v}_1 \times \vec{v}_2 \cdot \vec{v}_3 = \vec{v}_1 \cdot \vec{v}_2 \times \vec{v}_3 = \vec{v}_2 \cdot \vec{v}_3 \times \vec{v}_1 = \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix} \tag{3.35}$$

where

$$\begin{aligned} \vec{v}_1 &= a_x\hat{i} + a_y\hat{j} + a_z\hat{k} \\ \vec{v}_2 &= b_x\hat{i} + b_y\hat{j} + b_z\hat{k} \\ \vec{v}_3 &= c_x\hat{i} + c_y\hat{j} + c_z\hat{k} \end{aligned} \tag{3.36}$$

The reference frame in which the triple scalar product is taken must be chosen with care if the resultant multivariate polynomial is to be free of nonlinear terms in sines and cosines of the included joint angles. If the coordinate system shares an axis with the unit vector in the center of the subchain, the result will be free of any product terms $c_i s_i$, $c_i^{\,2}$, and $s_i^{\,2}$ [Lee and Reinholtz, 1992].

The scalar triple product $[\hat{s}_1 \cdot \hat{a}_{45} \times \hat{s}_6]_B$ will be evaluated as an example. From Figure 3.4 it is obvious that $\hat{s}_6$ lies in the center of subchain B bounded by $\hat{s}_1$ and $\hat{a}_{45}$, so the vector scalar product will be performed in this reference frame. Using the method for evaluating the scalar product presented in the previous section, unit vectors $\hat{s}_1$ and $\hat{a}_{45}$ will be rotated into the coordinate frame $\left(\hat{a}_{56} \quad \hat{b}_6 \quad \hat{s}_6\right)$ as follows:

$$\hat{s}_1 \colon \mathbf{M}_H \hat{z} \tag{3.37}$$

and

$$\hat{a}_{45} \colon \mathbf{M}_{56}^{-1}\mathbf{M}_5^{-1}\hat{x} \tag{3.38}$$

The triple scalar product may then be calculated as

$$[\hat{s}_1 \cdot \hat{a}_{45} \times \hat{s}_6]_B = \left|\mathbf{M}_H\hat{z} \quad \mathbf{M}_{56}^{-1}\mathbf{M}_5^{-1}\hat{x} \quad \hat{z}\right| \tag{3.39}$$

Although scalar and triple scalar products can be calculated by hand, mathematical software capable of symbolic manipulation provides a means for evaluating these expressions accurately and more readily. All symbolic manipulation performed in this research was done using Mathematica. Example Mathematica input files are listed in Appendix C.

3.3.2 The First Closure Equation

Vector scalar products are used to evaluate equations (3.26) and (3.27). Evaluating the left-hand side of equation (3.26) results in the following

$$\begin{aligned}\left[\vec{R}\cdot\hat{s}_1\right]_A &= \left[-\left(a_{12}\hat{a}_{12}+a_{23}\hat{a}_{23}\right)\cdot\hat{s}_1\right]_A\\ &= \left[-a_{12}\hat{s}_1\cdot\hat{a}_{12}-a_{23}\hat{s}_1\cdot\hat{a}_{23}\right]_A\\ &= \left[-a_{23}\hat{s}_1\cdot\hat{a}_{23}\right]_A\\ &= -a_{23}\hat{z}^T\mathbf{M}_{12}\mathbf{M}_2\hat{x}\\ &= -a_{23}s_2\end{aligned} \tag{3.40}$$

while the right hand side is written as

$$\begin{aligned}\left[\vec{R}\cdot\hat{s}_1\right]_B &= \left[\left(S_4\hat{s}_4+a_{56}\hat{a}_{56}-\vec{R}_H\right)\cdot\hat{s}_1\right]_B\\ &= \left[S_4\hat{s}_4\cdot\hat{s}_1+a_{56}\hat{a}_{56}\cdot\hat{s}_1-\vec{R}_H\cdot\hat{s}_1\right]_B\\ &= S_4\hat{z}^T\mathbf{M}_{45}\mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\hat{z}+a_{56}\hat{x}^T\mathbf{M}_6\mathbf{M}_H^{-1}\hat{z}-\vec{R}_H\hat{z}\\ &= -S_4\left[s_5\left(r_{31}c_6-r_{32}s_6\right)-r_{33}c_5\right]-a_{56}\left(r_{31}c_6-r_{32}s_6\right)-r_z\end{aligned} \tag{3.41}$$

Equation (3.26) may now be written as

$$-a_{23}s_2 = -S_4\left[s_5\left(r_{31}c_6-r_{32}s_6\right)-r_{33}c_5\right]-a_{56}\left(r_{31}c_6-r_{32}s_6\right)+r_z \tag{3.42}$$

3.3.3 The Second Closure Equation

Equation (3.27) is evaluated in the same manner as equation (3.26). Expanding the left- and right-hand sides of equation (3.27) results in the following:

$$\begin{aligned}\left[\vec{R}\cdot\vec{R}\right]_A &= \left[\left(a_{12}\hat{a}_{12}+a_{23}\hat{a}_{23}\right)\cdot\left(a_{12}\hat{a}_{12}+a_{23}\hat{a}_{23}\right)\right]_A\\ &= \left[a_{12}{}^2+a_{23}{}^2+2a_{12}a_{23}\left(\hat{a}_{12}\cdot\hat{a}_{23}\right)\right]_A\\ &= a_{12}{}^2+a_{23}{}^2+2a_{12}a_{23}\hat{x}^T\mathbf{M}_2\hat{x}\\ &= a_{12}{}^2+a_{23}{}^2+2a_{12}a_{23}c_2\end{aligned} \tag{3.43}$$

$$
\begin{aligned}
&\left[\vec{R}\cdot\vec{R}\right]_B && (3.44)\\
&=\left[\left(S_4\hat{s}_4+a_{56}\hat{a}_{56}-\vec{R}_H\right)\cdot\left(S_4\hat{s}_4+a_{56}\hat{a}_{56}-\vec{R}_H\right)\right]_B\\
&=S_4{}^2+a_{56}{}^2+r_x^2+r_y^2+r_z^2+2\left[S_4a_{56}\left(\hat{s}_4\cdot\hat{a}_{56}\right)-S_4\left(\hat{s}_4\cdot\vec{R}_H\right)-a_{56}\left(\hat{a}_{56}\cdot\vec{R}_H\right)\right]\\
&=S_4{}^2+a_{56}{}^2+r_x^2+r_y^2+r_z^2+2\left(S_4a_{56}\hat{z}^T\mathbf{M}_{45}\mathbf{M}_5\hat{x}-S_4\hat{z}^T\mathbf{M}_{45}\mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\vec{R}_H-a_{56}\hat{x}^T\mathbf{M}_6\mathbf{M}_H^{-1}\vec{R}_H\right)\\
&=S_4{}^2+a_{56}{}^2+r_x^2+r_y^2+r_z^2+2\left\{S_4a_{56}s_5-S_4\left[r_x\left(s_5\left(r_{11}c_6-r_{12}s_6\right)-r_{13}c_5\right)+r_y\left(s_5\left(r_{21}c_6-r_{22}s_6\right)-r_{23}c_5\right)\right.\right.\\
&\left.\left.+r_z\left(s_5\left(r_{31}c_6-r_{32}s\right)_6-r_{33}c_5\right)\right]-a_{56}\left[r_x\left(r_{11}c_6-r_{12}s_6\right)+r_y\left(r_{21}c_6-r_{22}s_6\right)+r_z\left(r_{31}c_6-r_{32}s_6\right)\right]\right\}
\end{aligned}
$$

After moving $a_{12}^2+a_{23}^2$ to the right-hand side, equation (3.27) is now written as

$$
\begin{aligned}
a_{12}a_{23}c_2=&\frac{1}{2}\left(-a_{12}{}^2-a_{23}{}^2+S_4{}^2+a_{56}{}^2+r_x^2+r_y^2+r_z^2\right)+S_4a_{56}s_5\\
&-S_4\left[r_x\left(s_5\left(r_{11}c_6-r_{12}s_6\right)-r_{13}c_5\right)+r_y\left(s_5\left(r_{21}c_6-r_{22}s_6\right)-r_{23}c_5\right)\right.\\
&\left.+r_z\left(s_5\left(r_{31}c_6-r_{32}s_6\right)-r_{33}c_5\right)\right]-a_{56}\left[r_x\left(r_{11}c_6-r_{12}s_6\right)\right.\\
&\left.+r_y\left(r_{21}c_6-r_{22}s_6\right)+r_z\left(r_{31}c_6-r_{32}s_6\right)\right]
\end{aligned}
\qquad (3.45)
$$

3.3.4 The Third Closure Equation

Developing equation (3.25) requires the evaluation of triple scalar products. If the geometry of the manipulator is taken into account, evaluating the left-hand side of equation (3.25) is trivial. Written out, the left-hand side of equation (3.25) is

$$\left[\vec{R}\times\hat{s}_1\cdot\hat{s}_4\right]_A=\left[-\left(a_{12}\hat{a}_{12}+a_{23}\hat{a}_{23}\right)\times\hat{s}_1\cdot\hat{s}_4\right]_A \qquad (3.46)$$

It is clear that unit vectors $\hat{a}_{12}$, $\hat{a}_{23}$, and $\hat{s}_4$ lie in the same plane. Therefore,

$$\left[\vec{R}\times\hat{s}_1\cdot\hat{s}_4\right]_A=0 \qquad (3.47)$$

The right-hand side of equation (3.25) is written as

$$
\begin{aligned}
\left[\vec{R}\times\hat{s}_1\cdot\hat{s}_4\right]_B&=\left[\left(S_4\hat{s}_4+a_{56}\hat{a}_{56}-\vec{R}_H\right)\times\hat{s}_1\cdot\hat{s}_4\right]_B\\
&=\left[\left(a_{56}\hat{a}_{56}-\vec{R}_H\right)\times\hat{s}_1\cdot\hat{s}_4\right]_B
\end{aligned}
\qquad (3.48)
$$

Before calculating the first triple scalar product in this equation $\hat{a}_{56}\times\hat{s}_1\cdot\hat{s}_4$ a coordinate frame for calculations must be chosen. It is clear that the unit vector $\hat{a}_{56}$ is in the center of subchain B, and as a result the vectors will be evaluated in reference to $\left(\hat{a}_{56}\;\;\hat{b}_6\;\;\hat{s}_6\right)$. Expanding $\left[\hat{a}_{56}\times\hat{s}_1\cdot\hat{s}_4\right]_B$ results in the following:

$$\begin{aligned}\left[\hat{a}_{56} \times \hat{s}_1 \cdot \hat{s}_4\right]_B &= \left|\hat{x} \quad \mathbf{M}_6\mathbf{M}_H^{-1}\hat{z} \quad \mathbf{M}_{56}^{-1}\mathbf{M}_5^{-1}\mathbf{M}_{45}^{-1}\hat{z}\right| \\ &= -c_5\left(r_{32}c_6 + r_{31}s_6\right)\end{aligned} \tag{3.49}$$

The second triple scalar product in the right-hand side of equation (3.27) is calculated in the same manner. In this instance the first two vectors, namely $\vec{R}_H$ and $\hat{s}_1$, are both expressed in the first coordinate frame. Therefore this will also be the frame of reference for the triple scalar product. $\left[\vec{R}_H \times \hat{s}_1 \cdot \hat{s}_4\right]_B$ is calculated as follows:

$$\begin{aligned}\left[\vec{R}_H \times \hat{s}_1 \cdot \hat{s}_4\right]_B &= \left|\vec{R}_H \quad \hat{z} \quad \mathbf{M}_H\mathbf{M}_6^{-1}\mathbf{M}_{56}^{-1}\mathbf{M}_5^{-1}\mathbf{M}_{45}^{-1}\hat{z}\right| \\ &= r_x\left(s_5\left(r_{21}c_6 - r_{22}s_6\right) - r_{23}c_5\right) + r_y\left(s_5\left(-r_{11}c_6 + r_{12}s_6\right) + r_{13}c_5\right)\end{aligned} \tag{3.50}$$

Equation (3.25) may now be rewritten as the following:

$$0 = -a_{56}\left(c_5\left(r_{32}c_6 + r_{31}s_6r_x\right)\right) + \left(s_5\left(r_{21}c_6 - r_{22}s_6\right) - r_{23}c_5\right) + r_y\left(s_5\left(-r_{11}c_6 + r_{12}s_6\right) + r_{13}c_5\right) \tag{3.51}$$

Closure equations (3.42), (3.45), and (3.51) now contain all the information needed to perform a full kinematic analysis of the URSULA manipulator. These three equations are multivariate polynomials in sines and cosines of joint angles θ_2, θ_5, and θ_6. It is desired to have the resultant polynomial be a function of a single joint variable, θ_6 for example. In order to do this, the other two joint angles, θ_2 and θ_5, must be eliminated.

3.4 Solving for Manipulator Joint Variables

3.4.1 The 16th Degree Univariate Polynomial

3.4.1.1 Eliminating θ_2 from the Closure Equations

In order to eliminate θ_2, closure equations (3.42) and (3.45) can be combined. If both sides of equation (3.42) are multiplied by $-a_{12}$, equations (3.42) and (3.45) can be squared and added, thus eliminating θ_2. The resulting multivariate polynomial is strictly a function of θ_5 and θ_6.

$$\begin{aligned}&\Big\langle -a_{12}^2 - a_{23}^2 + S_4^2 + a_{56}^2 + r_x^2 + r_y^2 + r_z^2 + 2\Big\{S_4a_{56}s_5 - S_4\Big[r_x\left(s_5\left(r_{11}c_6 - r_{12}s_6\right) - r_{13}c_5\right) + r_y\left(s_5\left(r_{21}c_6 - r_{22}s_6\right) - r_{23}c_5\right) \\ &\quad + r_z\left(s_5\left(r_{31}c_6 - r_{32}s_6\right) - r_{33}c_5\right)\Big] - a_{56}\left[r_x\left(r_{11}c_6 - r_{12}s_6\right) + r_y\left(r_{21}c_6 - r_{22}s_6\right) + r_z\left(r_{31}c_6 - r_{32}s_6\right)\right]\Big\}\Big\rangle^2 \\ &\quad + \left[2a_{12}S_4\left(s_5\left(r_{31}c_6 - r_{32}s_6\right) - r_{33}c_5\right) + 2a_{12}a_{56}\left(r_{31}c_6 - r_{32}s_6\right) - 2a_{12}r_z\right]^2 - \left(2a_{12}a_{23}\right)^2 = 0\end{aligned} \tag{3.52}$$

3.4.1.2 Eliminating θ_5 from the Closure Equations

The following manipulations parallel those performed by Lee, Lin, and Duffy [1992] for an in-parallel platform mechanism. Equations (3.51) and (3.52) are functions of θ_5 and θ_6 alone. At this point, the following tangent of the half-angle substitution is made:

$$c_i = \frac{1-x_i^2}{1+x_i^2} \qquad s_i = \frac{2x_i}{1+x_i^2} \tag{3.53}$$

where

$$x_i = \tan\left(\frac{\theta_i}{2}\right) \tag{3.54}$$

After this substitution and simplification, it is readily apparent that equations (3.51) and (3.52) become multivariate polynomials in x_5 and x_6. Equation (3.51) is now a second degree polynomial in x_5, where the coefficients are themselves second degree polynomials in x_6. Equation (3.52) is a fourth degree polynomial in x_5, where the coefficients of this polynomial are fourth degree polynomials in x_6.

$$A_2x_5^2 + A_1x_5 + A_0 = 0 \tag{3.55}$$

$$B_4x_5^4 + B_3x_5^3 + B_2x_5^2 + B_1x_5 + B_0 = 0 \tag{3.56}$$

where

$$A_i = \text{second degree polynomials in } x_6$$

$$B_i = \text{fourth degree polynomials in } x_6$$

Since this system is homogeneous, it is possible to create additional equations by multiplying the existing equations by powers of x_5. Using Sylvester's Dialytic Method (Appendix A), equation (3.55) is multiplied by x_5^3, x_5^2, x_5, and 1, while equation (3.56) is multiplied by x_5 and 1. The following equations in addition to (3.55) and (3.56) result:

$$A_2x_5^3 + A_1x_5^2 + A_0x_5 = 0 \tag{3.57}$$

$$A_2x_5^4 + A_1x_5^3 + A_0x_5^2 = 0 \tag{3.58}$$

$$A_2x_5^5 + A_1x_5^4 + A_0x_5^3 = 0 \tag{3.59}$$

$$B_4x_5^5 + B_3x_5^4 + B_2x_5^3 + B_1x_5^2 + B_0x_5 = 0 \tag{3.60}$$

Equations (3.55-3.60) may now be written in the following matrix form:

$$\begin{bmatrix} A_2 & A_1 & A_0 & 0 & 0 & 0 \\ 0 & A_2 & A_1 & A_0 & 0 & 0 \\ 0 & 0 & A_2 & A_1 & A_0 & 0 \\ 0 & 0 & 0 & A_2 & A_1 & A_0 \\ B_4 & B_3 & B_2 & B_1 & B_0 & 0 \\ 0 & B_4 & B_3 & B_2 & B_1 & B_0 \end{bmatrix} \begin{bmatrix} x_5^5 \\ x_5^4 \\ x_5^3 \\ x_5^2 \\ x_5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.61}$$

Since there is no trivial solution to the above system $(1 \neq 0)$, the coefficient matrix cannot be inverted [Anton, 1987]. Since the coefficient matrix is singular, its determinant is equal to zero. The determinant of the coefficient matrix in equation (3.61) is free of x_5 and is in the following form:

$$\begin{aligned} \text{Det} = {} & A_2^4B_0^2 - A_1A_2^3B_0B_1 + A_0A_2^3B_1^2 + A_1^2A_2^2B_0B_2 - 2A_0A_2^3B_0B_2 - A_0A_1A_2^2B_1B_2 + A_0^2A_2^2B_2^2 \\ & -A_1^3A_2B_0B_3 + 3A_0A_1A_2^2B_0B_3 + A_0A_1^2A_2B_1B_3 - 2A_0^2A_2^2B_1B_3 - A_0^2A_1A_2B_2B_3 + A_0^3A_2B_3^2 \\ & +A_1^4B_0B_4 - 4A_0A_1^2A_2B_0B_4 + 2A_0^2A_2^2B_0B_4 - A_0A_1^3B_1B_4 + 3A_0^2A_1A_2B_1B_4 + A_0^2A_1^2B_2B_4 \\ & -2A_0^3A_2B_2B_4 - A_0^3A_1B_3B_4 + A_0^4B_4^2 \end{aligned} \tag{3.62}$$

which can be expressed as follows:

$$\sum_{i=0}^{16} e_i x_6^i = 0 \tag{3.63}$$

This is the desired 16th degree univariate polynomial in x_6 with constant coefficients e_i.

3.4.2 Solving for θ_6

Once a desired hand position and orientation is specified, all e_i coefficients of equation (3.61) can be calculated. The roots of this polynomial then give all possible solutions of x_6 required to reach the desired pose. Laguerre's Method was used for computing the roots of this polynomial. A detailed discussion of this root finding method can be found in Appendix B. The resulting joint angle is found as follows:

$$\theta_6 = 2\text{Atan}(x_6) \tag{3.64}$$

Any complex values resulting from rooting equation (3.63) correspond to solutions that are kinematically unrealizable for the manipulator. If all roots are complex, no kinematically realizable manipulator configurations exist for the desired hand position and orientation. If one or more real solutions for θ_6 exist, the solution for the remaining joint angles may be solved in closed form, requiring no iteration.

3.4.3 Solving for θ_5

Now that x_6 is defined, so too are the coefficients of equations (3.61). This system may now be used to solve for x_5. Equation (3.61) may be rewritten in the following form:

$$\begin{bmatrix} A_2 & A_1 & A_0 & 0 & 0 \\ 0 & A_2 & A_1 & A_0 & 0 \\ 0 & 0 & A_2 & A_1 & A_0 \\ 0 & 0 & 0 & A_2 & A_1 \\ B_4 & B_3 & B_2 & B_1 & B_0 \\ 0 & B_4 & B_3 & B_2 & B_1 \end{bmatrix} \begin{bmatrix} x_5^5 \\ x_5^4 \\ x_5^3 \\ x_5^2 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -A_0 \\ 0 \\ -B_0 \end{bmatrix} \tag{3.65}$$

Because there are more equations than unknowns, the system in equation (3.65) is overconstrained. Since this series of equations was developed from a reduced set of equations, it is possible to eliminate two equations without affecting the result. Selecting the equations to eliminate should be done so as to minimize the size of the resulting symbolic solution. Because the solution of this system will require the inversion of the coefficient matrix, the equations with the greatest number of coefficients in the coefficient matrix should be eliminated, namely the first and fifth equations. The resulting system is

$$\begin{bmatrix} A_2 & A_1 & A_0 & 0 \\ 0 & A_2 & A_1 & A_0 \\ 0 & 0 & A_2 & A_1 \\ B_4 & B_3 & B_2 & B_1 \end{bmatrix} \begin{bmatrix} x_5^4 \\ x_5^3 \\ x_5^2 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -A_0 \\ -B_0 \end{bmatrix} \tag{3.66}$$

The solution for x_5 is now found by solving this system and taking the fifth element of the solution vector, which results in

$$x_5 = \frac{-A_2^3 B_0 + A_0 A_2^2 B_2 - A_0 A_1 A_2 B_3 + A_0 A_1^2 B_4 - A_0^2 A_2 B_4}{A_2^3 B_1 - A_1 A_2^2 B_2 + A_1^2 A_2 B_3 - A_0 A_2^2 B_3 - A_1^3 B_4 + 2 A_0 A_1 A_2 B_4} \tag{3.67}$$

Again, the tangent of the half-angle relation is used to compute the resulting joint angle

$$\theta_5 = 2\text{Atan}(x_5) \tag{3.68}$$

3.4.4 Solving for θ_2

Since joint angles θ_5 and θ_6 are now defined, it is possible to solve directly for s_2 and c_2 using equations (3.42) and (3.45) respectively. The result is shown below.

$$s_2 = \frac{-S_4\{s_5(r_{31}c_6 - r_{32}s_6) - r_{33}c_5\} - a_{56}(r_{31}c_6 - r_{32}s_6) + r_z}{-a_{23}} \tag{3.69}$$

$$c_2 = \Big\langle -a_{12}{}^2 - a_{23}{}^2 + S_4{}^2 + a_{56}{}^2 + r_x^2 + r_y^2 + r_z^2 + 2\big\{S_4 a_{56} s_5$$
$$-S_4\big[r_x\big(s_5(r_{11}c_6 - r_{12}s_6) - r_{13}c_5\big) + r_y\big(s_5(r_{21}c_6 - r_{22}s_6) - r_{23}c_5\big)$$
$$+r_z\big(s_5(r_{31}c_6 - r_{32}s_6) - r_{33}c_5\big)\big] - a_{56}\big[r_x(r_{11}c_6 - r_{12}s_6) \tag{3.70}$$
$$+r_y(r_{21}c_6 - r_{22}s_6) + r_z(r_{31}c_6 - r_{32}s_6)\big]\big\}\Big\rangle \Big/ 2a_{12}a_{23}$$

In order to maintain the proper quadrant for the solution of θ_2, the Atan2 function is used.

$$\theta_2 = \text{Atan2}(s_2, c_2) \tag{3.71}$$

3.4.5 Solving for θ_3

Using the same loop subchains, $\vec{R}_A$ and $\vec{R}_B$, used in the evaluation of the three closure equations, (3.25), (3.26), and (3.27), expressions including the sine and cosine of θ_3 can be found. These subchains can be used because the disconnection of the manipulator at joints 1 and 4 excludes those joint angles from any subsequent calculations involving equations (3.10), (3.13), (3.16-3.18), and (3.21). In other words, results obtained from these equations will be free of unknown joint variables θ_1 and θ_4. Loop equations (3.10) and (3.16) will be used to find θ_3.

$$[\hat{s}_1 \cdot \hat{s}_4]_A = [\hat{s}_1 \cdot \hat{s}_4]_B, \tag{3.72}$$

$$\big[\vec{R} \cdot \hat{s}_4\big]_A = \big[\vec{R} \cdot \hat{s}_4\big]_B \tag{3.73}$$

Expanding equations (3.72) and (3.73) in vector notation

$$\hat{z}^T \mathbf{M}_{12}\mathbf{M}_2\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\hat{z} = \hat{z}^T \mathbf{M}_{45}\mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\hat{z} \tag{3.74}$$

$$-a_{12}\hat{x}^T\mathbf{M}_2\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\hat{z} - a_{23}\hat{x}^T\mathbf{M}_3\mathbf{M}_{34}\hat{z} = S_4 + a_{56}\hat{z}^T\mathbf{M}_{45}\mathbf{M}_5\hat{x} - \hat{z}^T\mathbf{M}_{45}\mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\vec{R}_H \tag{3.75}$$

results in

$$-c_2c_3 + s_2s_3 = -r_{33}c_5 + s_5(r_{31}c_6 - r_{32}s_6) \tag{3.76}$$

$$-a_{23}s_3 - a_{12}(c_3s_2 + c_2s_3) = S_4 + a_{56}s_5 - r_x\{-r_{13}c_5 + s_5(r_{11}c_6 - r_{12}s_6)\}$$
$$-r_y\{-r_{23}c_5 + s_5(r_{21}c_6 - r_{22}s_6)\} \tag{3.77}$$
$$-r_z\{-r_{33}c_5 + s_5(r_{31}c_6 - r_{32}s_6)\}$$

These two equations are linear in terms of sines and cosines of θ_3 and can thus be solved for directly. The results follow:

$$s_3 = -a_{12}s_2\{r_{33}c_5 - s_5(r_{31}c_6 - r_{32}s_6)\} - c_2\{S_4 + a_{56}s_5 + c_5(r_{13}r_x + r_{23}r_y + r_{33}r_z)$$
$$-s_5c_6(r_{11}r_x + r_{21}r_y + r_{31}r_z) + s_5s_6(r_{12}r_x + r_{22}r_y + r_{32}r_z)\}\Big/(a_{23}c_2 + a_{12}) \tag{3.78}$$

$$c_3 = \langle (a_{23} + a_{12}c_2)\{r_{33}c_5 - s_5(r_{31}c_6 - r_{32}s_6)\} - s_2\{S_4 + a_{56}s_5$$
$$+c_5(r_{13}r_x + r_{23}r_y + r_{33}r_z) - s_5c_6(r_{11}r_x + r_{21}r_y + r_{31}r_z) \tag{3.79}$$
$$+s_5s_6(r_{12}r_x + r_{22}r_y + r_{32}r_z)\}\rangle\Big/(a_{23}c_2 + a_{12})$$

The Atan2 function is again used to resolve the proper angle.

$$\theta_3 = \text{Atan2}(s_3, c_3) \tag{3.80}$$

3.4.6 Solving for θ_4

Two different subchains need to be used in solving for θ_4. Subchains $\vec{R}_A$ and $\vec{R}_B$ result from disconnecting the manipulator at joints 1 and 4, thus eliminating any dependence on θ_1 and θ_4 in the resulting equations. To ensure dependence on θ_4, the manipulator will be disconnected at joints 1 and 5. The following figure is used to visualize this loop closure.

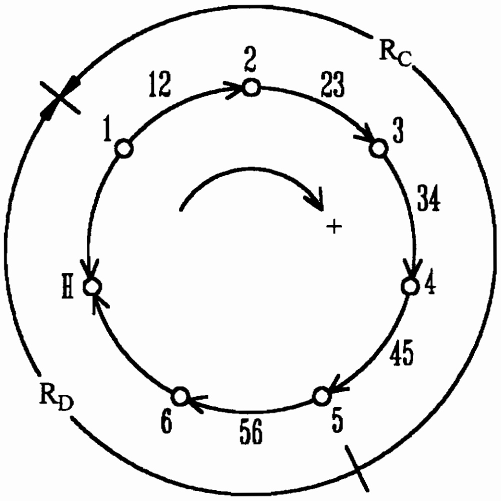

Figure 3.5 Loop Closure Diagram for Solution of θ_4

The resulting subchains are

$$\vec{R}_C = -(a_{12}\hat{a}_{12} + a_{23}\hat{a}_{23} + S_4\hat{s}_4) \tag{3.81}$$

and

$$\vec{R}_D = (a_{56}\hat{a}_{56} - \vec{R}_H) \tag{3.82}$$

The following two loop closure equations, which coincide with equations (3.8) and (3.10), are used to calculate the sine and cosine of the fourth joint angle.

$$(\hat{s}_1 \cdot \hat{a}_{45})_C = (\hat{s}_1 \cdot \hat{a}_{45})_D \tag{3.83}$$

$$(\hat{s}_1 \cdot \hat{s}_5)_C = (\hat{s}_1 \cdot \hat{s}_5)_D \tag{3.84}$$

Equations (3.83) and (3.84) may be written

$$\hat{z}^T \mathbf{M}_{12}\mathbf{M}_2\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\mathbf{M}_4\hat{x} = \hat{x}^T \mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\hat{z} \tag{3.85}$$

$$\hat{z}^T \mathbf{M}_{12}\mathbf{M}_2\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\mathbf{M}_4\mathbf{M}_{45}\hat{z} = \hat{z}^T \mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\hat{z} \tag{3.86}$$

Expanding the above equations results in

$$c_4(c_3s_2 + c_2s_3) = c_5(r_{31}c_6 - r_{32}s_6) + r_{33}s_5 \tag{3.87}$$

$$s_4(c_3s_2 + c_2s_3) = c_6r_{32} + s_6r_{31} \tag{3.88}$$

and solving for c_4 and s_4 results in

$$c_4 = \frac{c_5(c_6r_{31} - s_6r_{32}) + s_5r_{33}}{c_3s_2 + c_2s_3} \tag{3.89}$$

and

$$s_4 = \frac{c_6r_{32} + s_6r_{31}}{c_3s_2 + c_2s_3} \tag{3.90}$$

Again, the Atan2 function is used to find proper solution to θ_4.

$$\theta_4 = \text{Atan2}(s_4, c_4) \tag{3.91}$$

3.4.7 Solving for θ_1

The manipulator will be disassembled at joints 2 and 5 to find the single remaining joint angle θ_1, as shown in the following figure:

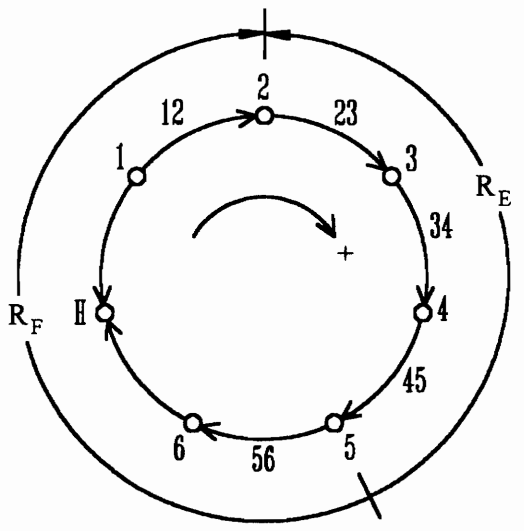

Figure 3.6 Loop Closure Diagram for Solution of θ_1

The resulting subchains are

$$\vec{R}_E = -\left(a_{23}\hat{a}_{23} + S_4\hat{s}_4\right) \tag{3.92}$$

and

$$\vec{R}_F = \left(a_{56}\hat{a}_{56} - \vec{R}_H + a_{12}\hat{a}_{12}\right) \tag{3.93}$$

Loop equations (3.8) and (3.17) are used to obtain equations with linear dependencies on sines and cosines of joint angle 1.

$$\left(\hat{s}_2 \cdot \hat{a}_{45}\right)_E = \left(\hat{s}_2 \cdot \hat{a}_{45}\right)_F \tag{3.94}$$

$$\left(\hat{s}_2 \cdot \hat{b}_5\right)_E = \left(\hat{s}_2 \cdot \hat{b}_5\right)_F \tag{3.95}$$

Equations (3.94) and (3.95) may be written as follows:

$$\hat{z}^T\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\mathbf{M}_4\hat{x} = \hat{x}^T\mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\mathbf{M}_1\mathbf{M}_{12}\hat{z} \tag{3.96}$$

$$\hat{z}^T\mathbf{M}_{23}\mathbf{M}_3\mathbf{M}_{34}\mathbf{M}_4\mathbf{M}_{45}\hat{y} = \hat{y}^T\mathbf{M}_5\mathbf{M}_{56}\mathbf{M}_6\mathbf{M}_H^{-1}\mathbf{M}_1\mathbf{M}_{12}\hat{z} \tag{3.97}$$

Fully expanding equations (3.96) and (3.97) results in

$$s_4 = s_1\left\{c_5\left(r_{11}c_6 - r_{12}s_6\right) + r_{13}s_5\right\} - c_1\left\{c_5\left(r_{21}c_6 - r_{22}s_6\right) + r_{23}s_5\right\} \tag{3.98}$$

and

$$0 = s_1\{s_5(c_6 r_{11} - s_6 r_{12}) - c_5 r_{13}\} - c_1\{s_5(c_6 r_{21} - s_6 r_{22}) - c_5 r_{23}\} \tag{3.99}$$

The solution for s_1 and c_1 follows:

$$s_1 = \frac{s_4(c_5 r_{23} - s_5(c_6 r_{21} - s_6 r_{22}))}{c_6(r_{11}r_{23} - r_{13}r_{21}) + s_6(r_{13}r_{22} - r_{12}r_{23})} \tag{3.100}$$

and

$$c_1 = \frac{s_4(c_5 r_{13} - s_5(c_6 r_{11} - s_6 r_{12}))}{c_6(r_{11}r_{23} - r_{13}r_{21}) + s_6(r_{13}r_{22} - r_{12}r_{23})} \tag{3.101}$$

Again, the Atan2 function is used to find proper solution to θ_1.

$$\theta_1 = \text{Atan2}(s_1, c_1) \tag{3.102}$$

3.5 Locomotion

Since the URSULA arm will be used for locomotion within the vessel as well as for scanning vessel welds, the inverse kinematics must be able to calculate joint angles for the case when the manipulator base and hand switch roles. Presently, the inverse kinematics takes a hand position and orientation with respect to the manipulator base as input for calculating joint angle solution sets. This may be expressed in terms of transformation matrices as

$${}^1\mathbf{T}_7 = \begin{bmatrix} {}^1\mathbf{R}_7 & \vec{R}_H \\ 0 & 1 \end{bmatrix} \tag{3.103}$$

where

$${}^1\mathbf{R}_7 = \begin{bmatrix} \hat{a}_{67} & \hat{b}_7 & \hat{s}_7 \end{bmatrix} \tag{3.104}$$

When the hand is equipped with the suction tool and is secured to the vessel, it is desired to move the base to a new position and orientation with respect to the hand. The position and orientation of the base with respect to the hand is written as ${}^7\mathbf{T}_1$. If this transformation matrix is first inverted, it can be used as input for the inverse kinematics without any modification to the code. Equation (2.5) shows that

$${}^7\mathbf{T}_1^{-1} = {}^1\mathbf{T}_7 \tag{3.105}$$

which is the required input for the inverse kinematics. Because the hand is fixed, the joint angle solutions will yield the desired base position and orientation.

4. Numerical Example

The inverse kinematics for URSULA was coded in Mathematica as well as in C. Source code for both is included in the Appendix. Solving for all sixteen sets of joint angle solutions for a given hand position and orientation using the C code (included in Appendix D), executes at 31 Hz on a 486/DX50 personal computer as measured by Borland's Turbo Profiler 2.0.

As previously stated, for a general six-degree-of-freedom manipulator, there are at most sixteen sets of solutions for a given hand position and orientation. Once link lengths and joint configurations are specified, not all six-degree-of-freedom manipulators will have sixteen solutions. However, sixteen solutions have been found in several regions of the URSULA workspace. The following numerical example enumerates one such case. Given the desired hand position

$$\bar{R}_H = \begin{bmatrix} 13.0 \\ 0.0 \\ -4.0 \end{bmatrix} \tag{4.1}$$

and orientation

$$\hat{a}_{67} = \begin{bmatrix} -0.3594733385 \\ -0.8686187185 \\ 0.3409991800 \end{bmatrix} \qquad \hat{b}_7 = \hat{s}_7 \times \hat{a}_{67} \qquad \hat{s}_7 = \begin{bmatrix} 0.68198091541 \\ 0.48779240062e-2 \\ 0.73135370161 \end{bmatrix} \tag{4.2}$$

sixteen real solutions result from the inverse kinematics. The solutions were calculated in the C program called INVERSE.C and verified in the Mathematica input file INVERSEM.M. These solutions are listed in Table 4.1.

Figure 4.2 shows URSULA in all sixteen configurations, verifying the results given in Table 4.1. The URSULA model depicted in the figure is kinematically correct, but joint sizes and link thicknesses have been reduced for clarity. Figure 4.1 shows URSULA with all joint angles set equal to zero.

Table 4.1 Sixteen Sets of Solutions for URSULA

Set	θ_1 (deg)	θ_2 (deg)	θ_3 (deg)	θ_4 (deg)	θ_5 (deg)	θ_6 (deg)
1	179.9033	96.0738	-125.5060	179.6375	72.4323	-119.7389
2	178.3324	-119.4352	-54.4854	-177.6416	-143.0747	-120.3661
3	49.0985	68.2942	-96.9311	85.0196	30.9449	-75.5673
4	44.0534	35.5585	-83.0657	113.6261	30.9145	-37.8810
5	31.6256	-134.5337	-107.3798	-136.3586	-149.1931	-3.2239
6	25.1578	-121.6020	-72.4324	-33.9849	-149.2853	-108.5355
7	13.4626	-121.5989	-71.5189	-17.7544	-149.6610	-114.1758
8	3.4120	-135.0063	-115.8500	-175.6314	-152.0355	53.9432
9	0.6153	77.1860	-108.7958	0.7095	11.3910	-119.4547
10	0.4350	19.0655	-64.0631	179.5080	1.9976	59.4899
11	-34.5620	-133.9532	-105.3544	131.3194	-148.6354	131.2579
12	-37.1823	-121.7763	-74.6084	53.1200	-148.6594	-139.3203
13	-44.3160	37.0708	-83.6684	-112.4619	31.2875	160.1019
14	-49.0064	67.2129	-96.3342	-86.5005	31.2595	-165.4183
15	-179.3568	-108.3464	-16.3858	0.7447	-167.6619	59.1022
16	-179.8985	145.3491	-163.7133	-0.2395	-61.3641	60.1107

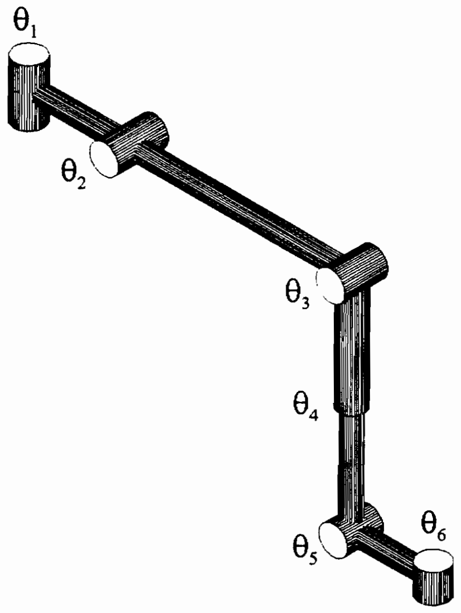

Figure 4.1 URSULA in the Zero Position

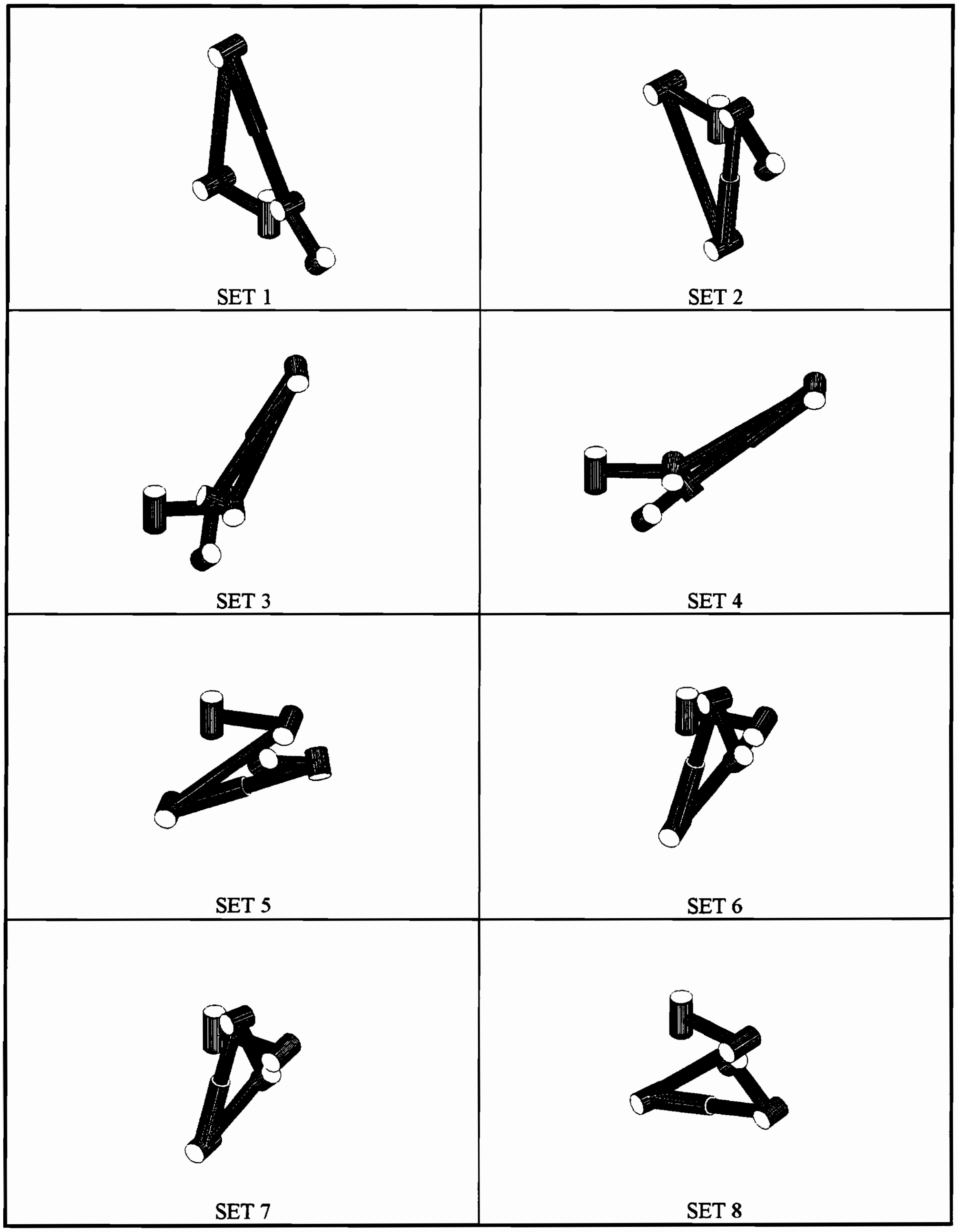

Figure 4.2 Sixteen Configurations Given in Table 4.1 for URSULA

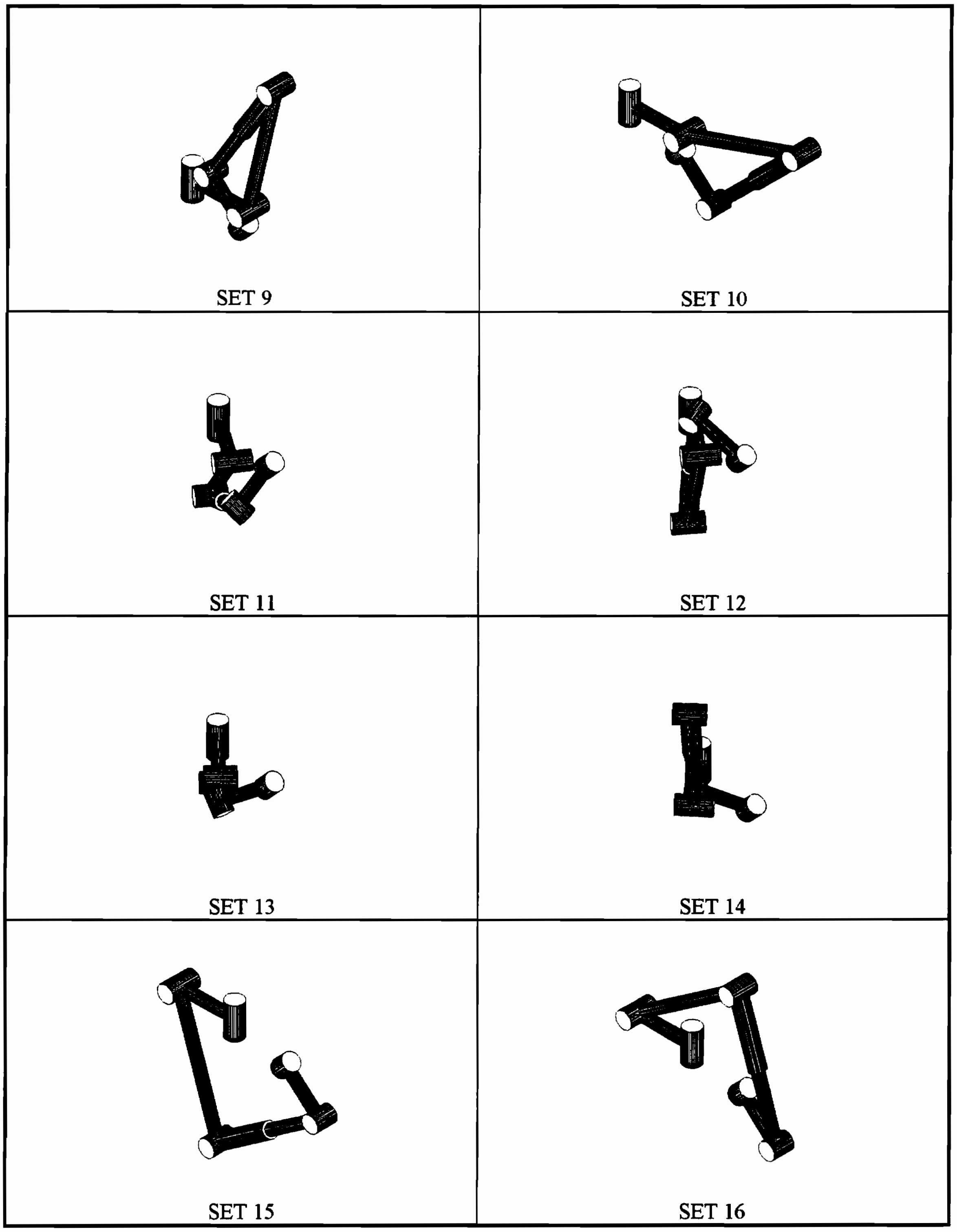

Figure 4.2 Continued

5. Future Work

5.1 Code Optimization

The inverse kinematics C code presently runs at 31 Hz on a 486/DX50 personal computer as measured by Borland's Turbo Profiler 2.0. The solution frequency could be increased significantly by changing platforms and/or optimizing the C code further. If this code were executed on a different platform, such as a workstation or a dedicated microprocessor, code execution speed would be sufficient for most tasks. The forward and inverse kinematics C code written for this study were written with efficiency and execution speed as priorities, but could be refined further. Rooting of the sixteenth degree univariate polynomial in the inverse kinematic code takes approximately fifty percent of the computing time for a single hand position and orientation.

Techniques other than that of rooting a 16th degree polynomial have been studied. In particular, the coefficient matrix used to derive the polynomial may be modified such that its eigenvalues are the negative of the polynomial roots. The eigenvalue analysis may prove to be a more computationally efficient method of solution [Manocha and Canny, 1992; Kholi and Osvatic, 1992].

5.2 Reduction of Solutions

Now that all physically realizable solutions for a desired hand position and orientation can be calculated, the task of choosing a solution needs to be addressed. This can be thought of as running the solutions through a series of filters, to eliminate unacceptable solutions. One possible method of selecting a solution set is depicted in the following figure.

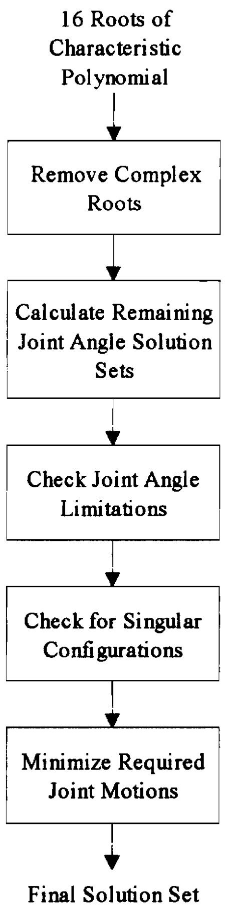
16 Roots of Characteristic Polynomial
Remove Complex Roots
Calculate Remaining Joint Angle Solution Sets
Check Joint Angle Limitations
Check for Singular Configurations
Minimize Required Joint Motions
Final Solution Set

Figure 5.1 Possible Solution Set Reduction Scheme

5.2.1 Complex Solutions

Although complex solutions for the roots of the characteristic polynomial are mathematically correct, they do not represent any physically obtainable manipulator configurations. Therefore, complex solutions should be removed from the collection of viable solutions prior to any further calculations.

5.2.2 Joint Limitations

The first step in the filtering process should be to eliminate all solutions that are not physically realizable. The solutions obtained from the inverse kinematics are valid for the kinematic model, which does not take into account link interference or actuator limits. By performing this task on the real valued sets first, unnecessary calculations in future filtering steps are reduced.

5.2.3 Singularities

A singularity is a position and orientation in which the manipulator loses one or more degrees of freedom. When the manipulator is moving to, through, or close to a singularity, joint velocities tend to infinity, as the remaining degrees of freedom attempt to reach the desired position and orientation. Finding singularities for a manipulator can be a very difficult problem. The term "singularity" is used because at these points the manipulator Jacobian matrix is singular. The Jacobian relates joint space velocities to Cartesian velocities. The angular velocity of the manipulator hand may be expressed as

$$\vec{w}_H = \dot{\theta}_1\hat{s}_1 + \dot{\theta}_2\hat{s}_2 + \dot{\theta}_3\hat{s}_3 + \dot{\theta}_4\hat{s}_4 + \dot{\theta}_5\hat{s}_5 + \dot{\theta}_6\hat{s}_6 \tag{5.1}$$

while the linear velocity may be written

$$\vec{v}_H = \vec{w}_H \times \vec{R}_H \tag{5.2}$$

where

$$\vec{R}_H = a_{12}\hat{a}_{12} + a_{23}\hat{a}_{23} + S_4\hat{s}_4 + a_{56}\hat{a}_{56} \tag{5.3}$$

as defined in equation (3.6). The Jacobian is the coefficient matrix of the system as shown below.

$$\begin{Bmatrix} \vec{w}_H \\ \vec{v}_H \end{Bmatrix} = [\mathbf{J}]\{\dot{\theta}\} \tag{5.4}$$

The above Jacobian relates the linear and angular velocities of the hand with respect to the base. This is desired if velocity control is to be used; but for singularity analysis, it is possible to express the Jacobian in another reference frame, towards the center of the manipulator, which would reduce the number of included terms. For example the following reference frame may be used.

$$\begin{bmatrix} \hat{s}_4 & \hat{a}_{34} & \hat{s}_3 \end{bmatrix} \tag{5.5}$$

If this reference frame is used, the Jacobian is written

$$\begin{Bmatrix} \vec{w}_H \cdot \hat{s}_4 \\ \vec{w}_H \cdot \hat{a}_{34} \\ \vec{w}_H \cdot \hat{s}_3 \\ \vec{v}_H \cdot \hat{s}_4 \\ \vec{v}_H \cdot \hat{a}_{34} \\ \vec{v}_H \cdot \hat{s}_3 \end{Bmatrix} = [\mathbf{J}]\{\dot{\theta}\} \tag{5.6}$$

Expanding the scalar and triple scalar products in the Jacobian gives the following result.

$$\mathbf{J} = \begin{bmatrix} -c_2c_3 + s_2s_3 & 0 & 0 & 1 & 0 & -c_5 \\ s_2c_3 + c_2s_3 & 0 & 0 & 0 & s_4 & c_4s_5 \\ 0 & 1 & 1 & 0 & -c_4 & s_4s_5 \\ 0 & -a_{23}c_3 & 0 & 0 & 0 & 0 \\ 0 & a_{23}s_3 & 0 & 0 & S_4c_4 & -s_4(S_4s_5 + a_{56}) \\ -a_{12} - a_{23}c_2 & 0 & 0 & 0 & S_4s_4 & c_4(S_4s_5 + a_{56}) \end{bmatrix} \tag{5.7}$$

A singularity occurs when the Jacobian is singular, which means that the Jacobian does not have an inverse and its determinant is zero. Therefore all singularities for this manipulator must satisfy the following relation:

$$\begin{aligned} |\mathbf{J}| &= f(\theta_2, \theta_3, \theta_4, \theta_5) \\ &= a_{23}c_3\left\{s_{2+3}S_4(S_4s_5 + a_{56}) + (a_{12} + a_{23}c_2)(S_4s_5 + a_{56}s_4^2)\right\} \\ &= 0 \end{aligned} \tag{5.8}$$

This equation defines singularities in joint space. The difficulty in applying this equation should not be underestimated. Due to the transcendental nature of the equation and the need to deal in Cartesian coordinates, this analysis is usually done off-line. This analysis typically falls into the realm of path planning and obstacle avoidance. A more in-depth study of workspace and singularity issues is presented by Lee, Woernle, and Hiller [1991].

5.2.4 Joint Motions

At this point, any remaining solution sets should provide a satisfactory result. If more than one set remains, it is necessary to reduce the number of sets until a single set remains. The solution requiring the least joint motion from the current configuration may be used. If the required change in joint angle θ_i is defined as $\Delta\theta_i$, an objective function can be defined as

$$\sqrt{\sum_{i=1}^{6} k_i(\Delta\theta_i)^2}, \tag{5.9}$$

where k_i is a weighting factor for each joint. This function may be evaluated for each solution set, and the set with the minimum result would be the final solution set.

This method of reducing the N sets of solutions for a given hand position and orientation is not unique.

6. Conclusion

A substantive application of a general inverse kinematic solution technique has been presented. The treatment of a 6R serial-chain manipulator as a 7R closed-loop chain has been explained in detail as well as the expansion of the closure equations via scalar and triple scalar products. The development of the characteristic polynomial in one joint variable and of the equations for the remaining joint variables was also enumerated.

It has been shown that the URSULA manipulator has the full complement of solutions in certain regions of its workspace. Methods for reducing the solutions have also been discussed. A foundation has been laid upon which future research, such as workspace and singularity analyses, can be based.

An efficient ANSI C code implementation of the URSULA inverse kinematic analysis was developed. Real-time implementation may be possible from a personal computer, certainly from a workstation or similar platform. This work should prove to be an invaluable resource to those unfamiliar with the application of general inverse kinematic analysis methods to manipulators with specific geometry.

7. References

Albala, H., and Angeles, J., 1979, "Numerical Solution to the Input-Output Displacement Equation of the General 7R Spatial Mechanism", Proceedings of the Fifth World Congress on Theory of Machines and Mechanisms, pp. 1008-1011.

Albala, H. and Pessen, D., 1983, "Displacement Analysis of a Special Case of the 7R, Single-Loop, Spatial Mechanism", Transactions of the ASME, Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 105, March, pp. 78-87.

Anton, H., 1987, *Elementary Linear Algebra*, John Wiley and Sons, New York.

Craig, John J., 1989, *Introduction to Robotics Mechanics and Control*, Addison Wesley, New York.

Duffy, J., 1980, *Analysis of Mechanisms and Robot Manipulators*, Edward Arnold Ltd., London

Duffy, J., and Crane, C., 1980,"A Displacement Analysis of the General Spatial 7R Mechanism", Mechanism and Machine Theory, Vol. 15, pp. 153-169.

Goldenberg, A. and Lawrence, D., 1985, "A Generalized Solution to the Inverse Kinematics of Robotic Manipulators", Journal of Dynamic Systems, Measurement, and Control, March, Vol. 107, pp. 103-106.

Kholi, D. and Osvatic, M., 1992, "Inverse Kinematics of General 6R and 5R,P Serial Manipulators", Flexible Mechanisms, Dynamics, and Analysis, DE-Vol. 47, pp. 619-627.

Lee, H.-Y., and Lin, W., and Duffy J., 1992, "A Method for the Forward Displacement Analysis of In-Parallel Platform Mechanisms", Accepted for publication in the International Journal of Mechatronics.

Lee, H.-Y., and Reinholtz, C.F., 1992, "Inverse Kinematics of Serial-Chain Manipulators", Submitted for publication in the ASME Journal of Mechanical Design.

Lee, H.-Y., Woernle, C., and Hiller, M., 1991, "A Complete Solution for the Inverse Kinematics Problem of the General 6R Robot Manipulator", ASME Journal of Mechanical Design, Vol. 113, No. 4, pp. 481-486.

Lee, H.-Y., 1990, "A Unified Theory for the Complete Solution to the Inverse Kinematics Problem of Industrial Robots with General Geometry" (in German), Ph.D. Dissertation, University of Duisburg, F.R. Germany.

Lee, H.-Y., and Liang, C.-G., 1988, "Displacement Analysis of the General Spatial 7-Link 7R Mechanism", Mechanism and Machine Theory, Vol. 23, No. 3, pp. 219-226.

Lee, H.-Y., and Liang, C.-G., 1987, "Displacement Analysis of the Spatial 7-Link 6R-P Linkages", Mechanism and Machine Theory, Vol. 22, No. 1, pp. 1-11.

Mabie, H.H., and Reinholtz, C.F., 1987, *Mechanisms and Dynamics of Machinery,* John Wiley and Sons, New York.

Manocha, D., and Canny, J., 1992, "Real Time Inverse Kinematics for General 6R Manipulators", Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, May.

McKerrow, P.J., 1991, *Introduction to Robotics*, Addison Wesley, Sydney, Australia.

Pieper, D.L., and Roth, B., 1969, "The Kinematics of Manipulators Under Computer Control", Proceedings of the 2nd International Congress for the Theory of Machines and Mechanisms, Zakopane Poland, Vol. 2, pp. 159-168.

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

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., 1992, *Numerical Recipes in C - The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, Cambridge.

Raghavan, M., and Roth, B., 1990a, "Inverse Kinematics of the General 6R Manipulator and Related Linkages", Proceedings of the 21st Biennial Mechanisms Conference, Chicago, IL, DE-Vol. 25 (Mechanism Synthesis and Analysis), pp. 59-65.

Raghavan, M., and Roth, B., 1990b, "Kinematic Analysis of 6R Manipulator of General Geometry", Proceedings of the 5th International Symposium on Robotics Research, ed. by Muria, H., and Arimoto, S., MIT Press, Cambridge.

Raghavan, M., and Roth, B., 1990c, "A General Solution for the Inverse Kinematics of All Series Chains", Proceedings of the Eighth CISM-IFTOMM Symposium on Robots and Manipulators (ROMANSY-90), Cracow, Poland.

Ralston, A., and Rabinowitz, P., 1978, *A First Course in Numerical Analysis*, 2nd Edition, McGraw-Hill, New York.

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

Salmon, G., 1885, *Lessons Introductory to the Modern Higher Algebra,* W. Metcalfe and Son, Cambridge.

Thomopoulos, S. and Tam, R., 1991, "An Iterative Solution to the Inverse Kinematics of Robotic Manipulators", Mechanism and Machine Theory, Vol 26, No. 4, pp. 359-373.

Tsai, L.-W., and Morgan, A., 1985, "Solving the Kinematics of the Most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods", Transactions of the ASME, Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 107, June, pp. 189-200.

Wolfram, Stephen, 1991, *Mathematica, A System for Doing Mathematics by Computer*, Addison Wesley, New York.

Appendix A
Sylvester's Dialytic Method
[Salmon, 1885]

"Sylvester gave his mode of elimination in the *Philosophical Magazine* for 1840, and called it 'dialytical', because the process as it were *dissolves* the relations which connect the different combinations of powers of the variables and treats them as simple independent quantities."

With k homogeneous equations in k unknowns or k non-homogeneous equations in k-1 unknowns, it is always possible to combine these equations into a single equation of the form $\Delta = 0$, which is free of unknown variables. The equation Δ is known as the eliminant of the system of equations, because it results from the elimination of the unknown variables.

Given the two simultaneous equations:

$$a_m x^m + a_{m-1} x^{m-1} y^1 + a_{m-2} x^{m-2} y^2 + \cdots + a_2 x^2 y^{m-2} + a_1 x y^{m-1} + a_0 y^m = 0 \tag{A.1}$$

$$b_n x^n + b_{n-1} x^{n-1} y^1 + b_{n-2} x^{n-2} y^2 + \cdots + b_2 x^2 y^{n-2} + b_1 x y^{n-1} + b_0 y^n = 0 \tag{A.2}$$

Multiply the equation of m^{th} degree by $x^{n-1},\ x^{n-2}y,\ x^{n-3}y^2, \ldots,\ x^{n-(n-1)}y^{n-2}$, and y^{n-1}, and the equation of n^{th} degree by $x^{m-1},\ x^{m-2}y,\ x^{m-3}y^2, \ldots,\ x^{m-(m-1)}y^{m-2}$, and y^{m-1}. The quantities $x^{m+n-1},\ x^{m+n-2}y,\ x^{m+n-3}y^2, \ldots,\ x^{(m+n)-(m+n-1)}y^{m+n-2}$, and y^{m+n-1} are considered as independent unknowns and can be linearly eliminated from the resulting $m+n$ equations. For the case of a quadratic and a cubic

$$a_2 x^2 + a_1 xy + a_0 y^2 = 0 \tag{A.3}$$

$$b_3 x^3 + b_2 x^2 y + b_1 x y^2 + b_0 y^3 = 0 \tag{A.4}$$

the quadratic is multiplied by x^2, xy, and y^2, and the cubic is multiplied by x and y. The resulting $m+n$ equations are

$$a_2 x^4 + a_1 x^3 y + a_0 x^2 y^2 = 0 \tag{A.5}$$

$$a_2 x^3 y + a_1 x^2 y^2 + a_0 x y^3 = 0 \tag{A.6}$$

$$a_2 x^2 y^2 + a_1 x y^3 + a_0 y^4 = 0 \tag{A.7}$$

$$b_3 x^4 + b_2 x^3 y + b_1 x^2 y^2 + b_0 x y^3 = 0 \tag{A.8}$$

$$b_3 x^3 y + b_2 x^2 y^2 + b_1 x y^3 + b_0 y^4 = 0 \tag{A.9}$$

The eliminant may be written as the determinant of the coefficient matrix of the following system:

$$\begin{bmatrix} a_2 & a_1 & a_0 & 0 & 0 \\ 0 & a_2 & a_1 & a_0 & 0 \\ 0 & 0 & a_2 & a_1 & a_0 \\ b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & b_3 & b_2 & b_1 & b_0 \end{bmatrix} \begin{bmatrix} x^4 \\ x^3 y \\ x^2 y^2 \\ xy^3 \\ y^4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.10}$$

Appendix B
Laguerre's Method

[Press, Teukolsky, Vetterling, and Flannery, 1992]

Laguerre's Method is a method of finding both real and complex roots of a polynomial of n^{th} degree with complex coefficients. For polynomials with real roots, convergence is guaranteed, but for complex roots little has been proved. Nonconvergence when dealing with complex roots is very unusual. A simple scheme is used to break a nonconverging cycle, if this should occur. It is known that when the method converges on a complex zero, the convergence is third order.

To motivate (although not rigorously derive) the Laguerre formulas, the following relations between the polynomial and its roots and derivatives can be noted:

$$P_n(x) = (x - x_1)(x - x_2)\cdots(x - x_n) \tag{B.1}$$

$$\ln|P_n(x)| = \ln|x - x_1| + \ln|x - x_2| + \cdots + \ln|x - x_n| \tag{B.2}$$

$$\frac{d\ln|P_n(x)|}{dx} = \frac{1}{x - x_1} + \frac{1}{x - x_2} + \cdots + \frac{1}{x - x_n} = \frac{P_n'}{P_n} \equiv G \tag{B.3}$$

$$\frac{d^2\ln|P_n(x)|}{dx^2} = \frac{1}{(x - x_1)^2} + \frac{1}{(x - x_2)^2} + \cdots + \frac{1}{(x - x_n)^2} = \left[\frac{P_n'}{P_n}\right]^2 - \frac{P_n''}{P_n} \equiv H \tag{B.4}$$

The root sought is assumed to be located some distance a from the current guess , while *all other roots* are assumed to be located at a distance b .

$$x - x_1 = a \qquad x - x_i = b \quad i = 2,3,\ldots,n \tag{B.5}$$

Then equations (B.3) and (B.4) can be expressed as

$$\frac{1}{a} + \frac{n-1}{b} = G \tag{B.6}$$

$$\frac{1}{a^2} + \frac{n-1}{b^2} = H \tag{B.7}$$

which yield a solution for a

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}} \tag{B.8}$$

where the sign should be taken to yield the largest magnitude for the denominator. Since the factor inside the square root can be negative, a can be complex.

The method operates iteratively. For a trial value x, a is calculated by equation (B.8). Then x-a becomes the next trial value. This continues until a is sufficiently small.

Appendix C
Mathematica Files

Two Mathematica input files are included in this Appendix. The first file, POLY16.M, is used to symbolically calculate the sixteenth degree characteristic polynomial. The file format follows the development in Section 3 - from the closure equations, (3.25), (3.26), and (3.27), to the final univariate polynomial, equation (3.61). All Mathematica functions used may be found in the Mathematica manual written by Stephen Wolfram. A deviation from the development of the closure equations given in Section 3.3 was necessary in order to keep symbolic results from getting too large. The three closure equations were rewritten in matrix form as

$$[\mathbf{D}]\begin{bmatrix} c_5c_6 \\ c_5s_6 \\ c_5 \\ s_5c_6 \\ s_5s_6 \\ s_5 \\ c_6 \\ s_6 \\ 1 \end{bmatrix} = \begin{bmatrix} 2a_{12}a_{23}s_2 \\ 2a_{12}a_{23}c_2 \\ 0 \end{bmatrix} \tag{C.1}$$

prior to the squaring and adding of the first two closure equations.

The output for POLY16.M is given in file POLY16.OUT. It is readily apparent from the output that performing these calculations by hand would be a formidable task. The output consists of the coefficients of the multivariate polynomials in equations (3.55) and (3.56) and the final sixteenth degree univariate polynomial in the tangent of the half-angle of θ_6, equation (3.61).

The second Mathematica input file, INVERSEM.M, uses results from POLY16.M to numerically calculate the sixteenth degree univariate polynomial and solves for all sets of solutions for a given hand position and orientation. The hand position and orientation used as an example results in sixteen real solutions, which are given in the output file, INVERSEM.OUT. Also included in the output file are the numerical coefficients to the sixteenth degree univariate polynomial.

POLY16.M

```
(* Input Desired Hand Position and Orientation Vectors *)
      Clear[rx,ry,rz,r11,r12,r13,r21,r22,r23,r31,r32,r33];
```

```
(* Link Lengths *)
       Clear[A12,A23,A34,A45,A56,A61];
       A34=0;
       A45=0;

(* Link Twists *)
       a12=Pi/2;
       a23=0;
       a34=Pi/2;
       a45=Pi/2;
       a56=Pi/2;

(* Link Offsets *)
       Clear[S1,S2,S3,S4,S5,S6];
       S1=0;
       S2=0;
       S3=0;
       S5=0;
       S6=0;

(* Rotation Matrices and Unit Vectors *)
       MH={{r11,r12,r13},{r21,r22,r23},{r31,r32,r33}};
              TMH=Transpose[MH];
       M12={{1,0,0},{0,Cos[a12],-Sin[a12]},{0,Sin[a12],Cos[a12]}};
              TM12=Transpose[M12];
       M23={{1,0,0},{0,Cos[a23],-Sin[a23]},{0,Sin[a23],Cos[a23]}};
              TM23=Transpose[M23];
       M34={{1,0,0},{0,Cos[a34],-Sin[a34]},{0,Sin[a34],Cos[a34]}};
              TM34=Transpose[M34];
       M45={{1,0,0},{0,Cos[a45],-Sin[a45]},{0,Sin[a45],Cos[a45]}};
              TM45=Transpose[M45];
       M56={{1,0,0},{0,Cos[a56],-Sin[a56]},{0,Sin[a56],Cos[a56]}};
              TM56=Transpose[M56];
       M1={{c1,-s1,0},{s1,c1,0},{0,0,1}};
              TM1=Transpose[M1];
       M2={{c2,-s2,0},{s2,c2,0},{0,0,1}};
              TM2=Transpose[M2];
       M3={{c3,-s3,0},{s3,c3,0},{0,0,1}};
              TM3=Transpose[M3];
       M4={{c4,-s4,0},{s4,c4,0},{0,0,1}};
              TM4=Transpose[M4];
       M5={{c5,-s5,0},{s5,c5,0},{0,0,1}};
              TM5=Transpose[M5];
       M6={{c6,-s6,0},{s6,c6,0},{0,0,1}};
              TM6=Transpose[M6];
       RH={{rx},{ry},{rz}};
              TRH=Transpose[RH];
       X={{1},{0},{0}};
              TX=Transpose[X];
       Y={{0},{1},{0}};
              TY=Transpose[Y];
```

```
        Z={{0},{0},{1}};
                TZ=Transpose[Z];

(* First Closure Equation *)
        LHSF1=-2*A12*(-A23*(TZ.M12.M2.X));
        RHSF1=-2*A12*(S4*(TZ.M45.M5.M56.M6.TMH.Z)+A56*(TX.M6.TMH.Z)+\
                S6*(TZ.TMH.Z)-rz);
        RHSF1=Expand[RHSF1];

(* Second Closure Equation *)
        LHSF2=2*A12*A23*(TX.M2.X);
        RHSF2=S4^2+A56^2+rx^2+ry^2+rz^2+2*(S4*A56*(TZ.M45.M5.X)+\
                S4*S6*(TZ.M45.M5.M56.Z)-S4*(TRH.MH.TM6.TM56.TM5.TM45.Z)-\
                A56*(TRH.MH.TM6.X))-A12^2-A23^2;
        RHSF2=Expand[RHSF2];

(* Third Closure Equation *)
        F3=A56*Det[{X,M6.TMH.Z,TM56.TM5.TM45.Z}]+
                S6*Det[{Z,M6.TMH.Z,TM56.TM5.TM45.Z}]-
        Det[{RH,Z,MH.TM6.TM56.TM5.TM45.Z}];
        F3=Expand[F3];

(* Build Matrix D *)
        D11=Coefficient[RHSF1,c5*c6];
        D21=Coefficient[RHSF2,c5*c6];
        D31=Coefficient[F3,c5*c6];

        D12=Coefficient[RHSF1,c5*s6];
        D22=Coefficient[RHSF2,c5*s6];
        D32=Coefficient[F3,c5*s6];

        s5=0;c6=0;s6=0;
        D13=Coefficient[RHSF1,c5];
        D23=Coefficient[RHSF2,c5];
        D33=Coefficient[F3,c5];
        Clear[s5,c6,s6];

        D14=Coefficient[RHSF1,s5*c6];
        D24=Coefficient[RHSF2,s5*c6];
        D34=Coefficient[F3,s5*c6];

        D15=Coefficient[RHSF1,s5*s6];
        D25=Coefficient[RHSF2,s5*s6];
        D35=Coefficient[F3,s5*s6];

        c5=0;c6=0;s6=0;
        D16=Coefficient[RHSF1,s5];
        D26=Coefficient[RHSF2,s5];
        D36=Coefficient[F3,s5];
        Clear[c5,c6,s6];

        c5=0;s5=0;s6=0;
        D17=Coefficient[RHSF1,c6];
        D27=Coefficient[RHSF2,c6];
        D37=Coefficient[F3,c6];
        Clear[c5,s5,s6];

        s5=0;c5=0;c6=0;
        D18=Coefficient[RHSF1,s6];
        D28=Coefficient[RHSF2,s6];
        D38=Coefficient[F3,s6];
        Clear[s5,c5,c6];

        c5=0;s5=0;c6=0;s6=0;
        D19=RHSF1;
        D29=RHSF2;
        D39=F3;
        Clear[c5,s5,c6,s6];


(* Save and Clear Variables *)
        !Del POLY16.OUT;
        Save["POLY16.OUT",
                D11,D12,D13,D14,D15,D16,D17,D18,D19,
                D21,D22,D23,D24,D25,D26,D27,D28,D29,
                D31,D32,D33,D34,D35,D36,D37,D38,D39];
        Clear[D11,D12,D13,D14,D15,D16,D17,D18,D19,
                D21,D22,D23,D24,D25,D26,D27,D28,D29,
                D31,D32,D33,D34,D35,D36,D37,D38,D39];

(* Square and Add First Two Closure Equations and Name F2 *)
        RHSF1=D11*c5*c6+D12*c5*s6+D13*c5+D14*s5*c6+D15*s5*s6
                D16*s5+D17*c6+D18*s6+D19;
        RHSF2=D21*c5*c6+D22*c5*s6+D23*c5+D24*s5*c6+D25*s5*s6
                D26*s5+D27*c6+D28*s6+D29;
        F2=RHSF1^2+RHSF2^2-(2*A12*A23)^2;

        F3=D31*c5*c6+D32*c5*s6+D33*c5+D34*s5*c6+D35*s5*s6
                D36*s5+D37*c6+D38*s6+D39;

(* Substitute Tan-Half-Angle Relations *)
        F2=F2/.{c5->((1-x5^2)/(1+x5^2)),c6->((1-x6^2)/(1+x6^2)),\
                s5->((2*x5)/(1+x5^2)),s6->((2*x6)/(1+x6^2))};
        F2=Expand[Together[F2]*((1+x5^2)*(1+x6^2))^2];

        F3=F3/.{c5->((1-x5^2)/(1+x5^2)),c6->((1-x6^2)/(1+x6^2)),\
                s5->((2*x5)/(1+x5^2)),s6->((2*x6)/(1+x6^2))};
        F3=Expand[Together[F3]*((1+x5^2)*(1+x6^2))];

(* Collect Coefficients of x5 *)
(* L Corresponds to Equation 3 and M Equation 2 *)
(* L and M are Used Instead of A and B Because A *)
(*      Is Used as a Link Parameter *)
        L=CoefficientList[F3,x5];
        L2=L[[3]];
        L1=L[[2]];
        L0=L[[1]];

        M=CoefficientList[F2,x5];
        M4=M[[5]];
        M3=M[[4]];
        M2=M[[3]];
        M1=M[[2]];
        M0=M[[1]];

(* Collect Coefficients of x6 *)
(* L2=L22*x6^2 + L21*x6 + L20 *)
        L=CoefficientList[L2,x6];
        L22=L[[3]];
        L21=L[[2]];
        L20=L[[1]];

        L=CoefficientList[L1,x6];
        L12=L[[3]];
        L11=L[[2]];
        L10=L[[1]];

        L=CoefficientList[L0,x6];
        L02=L[[3]];
        L01=L[[2]];
        L00=L[[1]];

        M=CoefficientList[M4,x6];
        M44=M[[5]];
        M43=M[[4]];
        M42=M[[3]];
        M41=M[[2]];
        M40=M[[1]];

        M=CoefficientList[M3,x6];
        M34=M[[5]];
        M33=M[[4]];
        M32=M[[3]];
        M31=M[[2]];
        M30=M[[1]];

        M=CoefficientList[M2,x6];
        M24=M[[5]];
        M23=M[[4]];
        M22=M[[3]];
        M21=M[[2]];
        M20=M[[1]];

        M=CoefficientList[M1,x6];
        M14=M[[5]];
        M13=M[[4]];
        M12=M[[3]];
        M11=M[[2]];
        M10=M[[1]];

        M=CoefficientList[M0,x6];
        M04=M[[5]];
        M03=M[[4]];
        M02=M[[3]];
        M01=M[[2]];
        M00=M[[1]];

(* Save and Clear Variables *)
        Save["POLY16.OUT",
                L22,L21,L20,L12,L11,L10,L02,L01,L00,
                M44,M43,M42,M41,M40,M34,M33,M32,M31,M30,
                M24,M23,M22,M21,M20,M14,M13,M12,M11,M10,
                M04,M03,M02,M01,M00];

        Clear[L22,L21,L20,L12,L11,L10,L02,L01,L00,
                M44,M43,M42,M41,M40,M34,M33,M32,M31,M30,
                M24,M23,M22,M21,M20,M14,M13,M12,M11,M10,
                M04,M03,M02,M01,M00,L2,L1,L0,M4,M3,M2,M1,M0];

(* Create 16th Degree Polynomial From L and M *)
        M={{L2,L1,L0,0,0,0},
                {0,L2,L1,L0,0,0},
                {0,0,L2,L1,L0,0},
                {0,0,0,L2,L1,L0},
                {M4,M3,M2,M1,M0,0},
                {0,M4,M3,M2,M1,M0}};
        P=Det[M];
        P=Simplify[P];

(* Save Reults *)
        Save["POLY16.OUT",P];
```

POLY16.OUT

```
D11={{0}}
D12={{0}}
D13={{2*A12*S4*r33}}
D14={{-2*A12*S4*r31}}
D15={{2*A12*S4*r32}}
D16={{0}}
D17={{-2*A12*A56*r31}}
D18={{2*A12*A56*r32}}
D19={{2*A12*rz}}
D21={{0}}
D22={{0}}
D23={{2*S4*r13*rx+2*S4*r23*ry+2*S4*r33*rz}}
D24={{-2*S4*r11*rx-2*S4*r21*ry-2*S4*r31*rz}}
D25={{2*S4*r12*rx+2*S4*r22*ry+2*S4*r32*rz}}
```

```
D26={{2*A56*S4}}
D27={{-2*A56*r11*rx-2*A56*r21*ry-2*A56*r31*rz}}
D28={{2*A56*r12*rx+2*A56*r22*ry+2*A56*r32*rz}}
D29={{-A12^2-A23^2+A56^2+S4^2+rx^2+ry^2+rz^2}}
D31={-(A56*r32)}
D32={-(A56*r31)}
D33={-(r23*rx)+r13*ry}
D34={r21*rx-r11*ry}
D35={-(r22*rx)+r12*ry}
D36={0}
D37={0}
D38={0}
D39={0}
L22=D31-D33
L21=-2*D32
L20=-D31-D33
L12=-2*D34
L11=4*D35
L10=2*D34
L02=-D31+D33
L01=2*D32
L00=D31+D33
M44=-4*A12^2*A23^2+D11^2-2*D11*D13+D13^2+D21^2-2*D21*D23+
    D23^2
M43=-4*D11*D12+4*D12*D13-4*D21*D22+4*D22*D23
M42=-8*A12^2*A23^2-2*D11^2+4*D12^2+2*D13^2-2*D21^2+4*D22^2+
    2*D23^2
M41=4*D11*D12+4*D12*D13+4*D21*D22+4*D22*D23
M40=-4*A12^2*A23^2+D11^2+2*D11*D13+D13^2+D21^2+2*D21*D23+
    D23^2
M34=-4*D11*D14+4*D13*D14-4*D21*D24+4*D23*D24
M33=8*D12*D14+8*D11*D15-8*D13*D15+8*D22*D24+8*D21*D25-
    8*D23*D25
M32=8*D11*D14-16*D12*D15+8*D21*D24-16*D22*D25
M31=-8*D12*D14-8*D11*D15-8*D13*D15-8*D22*D24-8*D21*D25-
    8*D23*D25
M30=-4*D11*D14-4*D13*D14-4*D21*D24-4*D23*D24
M24=-8*A12^2*A23^2-2*D11^2+4*D11*D13-2*D13^2+4*D14^2-
    2*D21^2+4*D21*D23-2*D23^2+4*D24^2
M23=8*D11*D12-8*D12*D13-16*D14*D15+8*D21*D22-8*D22*D23-
    16*D24*D25
M22=-16*A12^2*A23^2+4*D11^2-8*D12^2-4*D13^2-8*D14^2+
    16*D15^2+4*D21^2-8*D22^2-4*D23^2-8*D24^2+16*D25^2
M21=-8*D11*D12-8*D12*D13+16*D14*D15-8*D21*D22-8*D22*D23+
    16*D24*D25
M20=-8*A12^2*A23^2-2*D11^2-4*D11*D13-2*D13^2+4*D14^2-
    2*D21^2-4*D21*D23-2*D23^2+4*D24^2
M14=4*D11*D14-4*D13*D14+4*D21*D24-4*D23*D24
M13=-8*D12*D14-8*D11*D15+8*D13*D15-8*D22*D24-8*D21*D25+
    8*D23*D25
M12=-8*D11*D14+16*D12*D15-8*D21*D24+16*D22*D25
M11=8*D12*D14+8*D11*D15+8*D13*D15+8*D22*D24+8*D21*D25+
    8*D23*D25
M10=4*D11*D14+4*D13*D14+4*D21*D24+4*D23*D24
M04=-4*A12^2*A23^2+D11^2-2*D11*D13+D13^2+D21^2-2*D21*D23+
    D23^2
M03=-4*D11*D12+4*D12*D13-4*D21*D22+4*D22*D23
M02=-8*A12^2*A23^2-2*D11^2+4*D12^2+2*D13^2-2*D21^2+4*D22^2+
    2*D23^2
M01=4*D11*D12+4*D12*D13+4*D21*D22+4*D22*D23
M00=-4*A12^2*A23^2+D11^2+2*D11*D13+D13^2+D21^2+2*D21*D23+
    D23^2
P=L2^4*M0^2-L1*L2^3*M0*M1+L0*L2^3*M1^2+L1^2*L2^2*M0*M2-
    2*L0*L2^3*M0*M2-L0*L1*L2^2*M1*M2+L0^2*L2^2*M2^2-
    L1^3*L2*M0*M3+3*L0*L1*L2^2*M0*M3+L0*L1^2*L2*M1*M3-
    2*L0^2*L2^2*M1*M3-L0^2*L1*L2*M2*M3+L0^3*L2*M3^2+
    L1^4*M0*M4-4*L0*L1^2*L2*M0*M4+2*L0^2*L2^2*M0*M4-
    L0*L1^3*M1*M4+3*L0^2*L1*L2*M1*M4+L0^2*L1^2*M2*M4-
    2*L0^3*L2*M2*M4-L0^3*L1*M3*M4+L0^4*M4^2
```

INVERSEM.M

```
(* Clear All Previous Data *)
       Clear[X,THETA6,THETA5,THETA2,L2,L1,L0,M4,M3,M2,M1,M0];

(* Manipulator Geometry and Desired Hand Position and Orientation *)
       A12= 14.000;
       A23= 31.125;
       A34= 0.000;
       A45= 0.000;
       A56= 11.500;
       A67= 0.000;

       S1= 0.000;
       S2= 0.000;
       S3= 0.000;
       S4= 31.125;
       S5= 0.000;
       S6= 0.000;

       h11=-0.35947333851404;
       h21=-0.86861871849252;
       h31= 0.34099918003125;
       h13= 0.68198091541137;
       h23= 0.00487792400620;
       h33= 0.73135370161917;
       h12=h23*h31-h33*h21;
       h22=h11*h33-h31*h13;
       h32=h13*h21-h11*h23;

       rx= 13.00;
       ry= 0.00;
       rz=-4.00;
```

```
(* Calculate Matrix D *)
        D11 = -(A56*h32);
        D12 = -(A56*h31);
        D13 = -(h23*rx) + h13*ry;
        D14 = h21*rx - h11*ry;
        D15 = -(h22*rx) + h12*ry;
        D16 = 0;
        D17 = 0;
        D18 = 0;
        D19 = 0;
        D21 = 0;
        D22 = 0;
        D23 = 2*A12*S4*h33;
        D24 = -2*A12*S4*h31;
        D25 = 2*A12*S4*h32;
        D26 = 0;
        D27 = -2*A12*A56*h31;
        D28 = 2*A12*A56*h32;
        D29 = 2*A12*rz;
        D31 = 0;
        D32 = 0;
        D33 = 2*S4*h13*rx + 2*S4*h23*ry + 2*S4*h33*rz;
        D34 = -2*S4*h11*rx - 2*S4*h21*ry - 2*S4*h31*rz;
        D35 = 2*S4*h12*rx + 2*S4*h22*ry + 2*S4*h32*rz;
        D36 = 2*A56*S4;
        D37 = -2*A56*h11*rx - 2*A56*h21*ry - 2*A56*h31*rz;
        D38 = 2*A56*h12*rx + 2*A56*h22*ry + 2*A56*h32*rz;
        D39 = -A12^2 - A23^2 + A56^2 + S4^2 + rx^2 + ry^2 + rz^2;

(* Calculate Coefficients of h6 (X) in Subpolynomials *)
        L22 = D11 - D13;
        L21 = -2*D12;
        L20 = -D11 - D13;
        L12 = -2*D14;
        L11 = 4*D15;
        L10 = 2*D14;
        L02 = -D11 + D13;
        L01 = 2*D12;
        L00 = D11 + D13;
        M44 = -4*A12^2*A23^2 + D23^2 + 2*D23*D27 + D27^2 - 2*D23*D29 - 2*D27*D29 +
                D29^2 + D33^2 + 2*D33*D37 + D37^2 - 2*D33*D39 - 2*D37*D39 + D39^2;
        M43 = -4*D23*D28 - 4*D27*D28 + 4*D28*D29 - 4*D33*D38 - 4*D37*D38 + 4*D38*D39;
        M42 = -8*A12^2*A23^2 + 2*D23^2 - 2*D27^2 + 4*D28^2 - 4*D23*D29 + 2*D29^2 +
                2*D33^2 - 2*D37^2 + 4*D38^2 - 4*D33*D39 + 2*D39^2;
        M41 = -4*D23*D28 + 4*D27*D28 + 4*D28*D29 - 4*D33*D38 + 4*D37*D38 + 4*D38*D39;
        M40 = -4*A12^2*A23^2 + D23^2 - 2*D23*D27 + D27^2 - 2*D23*D29 + 2*D27*D29 +
                D29^2 + D33^2 - 2*D33*D37 + D37^2 - 2*D33*D39 + 2*D37*D39 + D39^2;
        M34 = 4*D23*D24 + 4*D24*D27 - 4*D24*D29 + 4*D33*D34 - 4*D33*D36 + 4*D34*D37 -
                4*D36*D37 - 4*D34*D39 + 4*D36*D39;
        M33 = -8*D23*D25 - 8*D25*D27 - 8*D24*D28 + 8*D25*D29 - 8*D33*D35 -
                8*D35*D37 - 8*D34*D38 + 8*D36*D38 + 8*D35*D39;
        M32 = -8*D24*D27 + 16*D25*D28 - 8*D33*D36 - 8*D34*D37 + 16*D35*D38 + 8*D36*D39;
        M31 = -8*D23*D25 + 8*D25*D27 + 8*D24*D28 + 8*D25*D29 - 8*D33*D35 +
                8*D35*D37 + 8*D34*D38 + 8*D36*D38 + 8*D35*D39;
        M30 = -4*D23*D24 + 4*D24*D27 + 4*D24*D29 - 4*D33*D34 - 4*D33*D36 +
                4*D34*D37 + 4*D36*D37 + 4*D34*D39 + 4*D36*D39;
        M24 = -8*A12^2*A23^2 - 2*D23^2 + 4*D24^2 + 2*D27^2 - 4*D27*D29 + 2*D29^2 -
                2*D33^2 + 4*D34^2 - 8*D34*D36 + 4*D36^2 + 2*D37^2 - 4*D37*D39 + 2*D39^2;
        M23 = -16*D24*D25 - 8*D27*D28 + 8*D28*D29 - 16*D34*D35 + 16*D35*D36 -
                8*D37*D38 + 8*D38*D39;
        M22 = -16*A12^2*A23^2 - 4*D23^2 - 8*D24^2 + 16*D25^2 - 4*D27^2 + 8*D28^2 +
                4*D29^2 - 4*D33^2 - 8*D34^2 + 16*D35^2 + 8*D36^2 - 4*D37^2 + 8*D38^2 +
                4*D39^2;
        M21 = 16*D24*D25 + 8*D27*D28 + 8*D28*D29 + 16*D34*D35 + 16*D35*D36 +
                8*D37*D38 + 8*D38*D39;
        M20 = -8*A12^2*A23^2 - 2*D23^2 + 4*D24^2 + 2*D27^2 + 4*D27*D29 + 2*D29^2 -
                2*D33^2 + 4*D34^2 + 8*D34*D36 + 4*D36^2 + 2*D37^2 + 4*D37*D39 + 2*D39^2;
        M14 = -4*D23*D24 + 4*D24*D27 - 4*D24*D29 - 4*D33*D34 + 4*D33*D36 +
                4*D34*D37 - 4*D36*D37 - 4*D34*D39 + 4*D36*D39;
        M13 = 8*D23*D25 - 8*D25*D27 - 8*D24*D28 + 8*D25*D29 + 8*D33*D35 - 8*D35*D37 -
                8*D34*D38 + 8*D36*D38 + 8*D35*D39;
        M12 = -8*D24*D27 + 16*D25*D28 + 8*D33*D36 - 8*D34*D37 + 16*D35*D38 + 8*D36*D39;
        M11 = 8*D23*D25 + 8*D25*D27 + 8*D24*D28 + 8*D25*D29 + 8*D33*D35 + 8*D35*D37 +
                8*D34*D38 + 8*D36*D38 + 8*D35*D39;
        M10 = 4*D23*D24 + 4*D24*D27 + 4*D24*D29 + 4*D33*D34 + 4*D33*D36 + 4*D34*D37 +
                4*D36*D37 + 4*D34*D39 + 4*D36*D39;
        M04 = -4*A12^2*A23^2 + D23^2 - 2*D23*D27 + D27^2 + 2*D23*D29 - 2*D27*D29 +
                D29^2 + D33^2 - 2*D33*D37 + D37^2 + 2*D33*D39 - 2*D37*D39 + D39^2;
        M03 = 4*D23*D28 - 4*D27*D28 + 4*D28*D29 + 4*D33*D38 - 4*D37*D38 + 4*D38*D39;
                M02 = -8*A12^2*A23^2 + 2*D23^2 - 2*D27^2 + 4*D28^2 + 4*D23*D29 + 2*D29^2 +
                2*D33^2 - 2*D37^2 + 4*D38^2 + 4*D33*D39 + 2*D39^2;
        M01 = 4*D23*D28 + 4*D27*D28 + 4*D28*D29 + 4*D33*D38 + 4*D37*D38 + 4*D38*D39;
        M00 = -4*A12^2*A23^2 + D23^2 + 2*D23*D27 + D27^2 + 2*D23*D29 + 2*D27*D29 +
                D29^2 + D33^2 + 2*D33*D37 + D37^2 + 2*D33*D39 + 2*D37*D39 + D39^2;

(* Build Polynomials L0,L1,L2 and M0,M1,M2,M3,M4 *)
(* L and M are Used Instead of A and B Because A *)
(*      Is Used as a Link Parameter *)
        L0=L02*X^2+L01*X+L00;
        L1=L12*X^2+L11*X+L10;
        L2=L22*X^2+L21*X+L20;

        M0=M04*X^4+M03*X^3+M02*X^2+M01*X+M00;
        M1=M14*X^4+M13*X^3+M12*X^2+M11*X+M10;
        M2=M24*X^4+M23*X^3+M22*X^2+M21*X+M20;
        M3=M34*X^4+M33*X^3+M32*X^2+M31*X+M30;
        M4=M44*X^4+M43*X^3+M42*X^2+M41*X+M40;

(* Build Final Univariate Polynomial in X *)
        P = L2^4*M0^2 - L1*L2^3*M0*M1 + L0*L2^3*M1^2 + L1^2*L2^2*M0*M2 -
                2*L0*L2^3*M0*M2 - L0*L1*L2^2*M1*M2 + L0^2*L2^2*M2^2 - L1^3*L2*M0*M3 +
                3*L0*L1*L2^2*M0*M3 + L0*L1^2*L2*M1*M3 - 2*L0^2*L2^2*M1*M3 -
                L0^2*L1*L2*M2*M3 + L0^3*L2*M3^2 + L1^4*M0*M4 - 4*L0*L1^2*L2*M0*M4 +
            2*L0^2*L2^2*M0*M4 - L0*L1^3*M1*M4 + 3*L0^2*L1*L2*M1*M4 + L0^2*L1^2*M2*M4 -
            2*L0^3*L2*M2*M4 - L0^3*L1*M3*M4 + L0^4*M4^2;

(* Calculate THETA6 *)
        P=Collect[P,X];
        TANTHETA6=Solve[P==0,X];
        TANTHETA6=X/.TANTHETA6;
        Clear[X];
        THETA6=2*ArcTan[TANTHETA6];
        T6DEG=THETA6*180/3.141592654;

(* Calculate THETA5 *)
        TANTHETA5={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
        tantheta51={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
        tantheta52={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        For[i=1,i<17,i++,{
                X=TANTHETA6[[i]];
                TANTHETA5[[i]] = (-L2^3*M0 + L0*L2^2*M2 - L0*L1*L2*M3 + L0*L1^2*M4 -
                        L0^2*L2*M4) / (L2^3*M1 - L1*L2^2*M2 + L1^2*L2*M3 - L0*L2^2*M3 -
                        L1^3*M4 + 2*L0*L1*L2*M4);
        }];

        THETA5=2*ArcTan[TANTHETA5];
        T5DEG=THETA5*180/3.141592654;

(* Calculate THETA2 *)
        c5=Cos[THETA5];
        s5=Sin[THETA5];
        c6=Cos[THETA6];
        s6=Sin[THETA6];

        c2 =-A12^2 - A23^2 + A56^2 + S4^2 + rx^2 + ry^2 + rz^2 + 2*(A56*S4*s5 +
                A56*(-c6*(h11*rx + h21*ry + h31*rz) + s6*(h12*rx + h22*ry + h32*rz)) +
                S4*(c5*(h13*rx + h23*ry + h33*rz) - c6*s5*(h11*rx +
                h21*ry + h31*rz) + s5*s6*(h12*rx + h22*ry + h32*rz)));
        s2 = 2*A12*(rz + A56*(h32*s6 - c6*h31) +
                S4*(c5*h33 + s5*(-c6*h31 + h32*s6)));

        THETA2=ArcTan[c2,s2];
        T2DEG=THETA2*180/3.141592654;

(* Calculate THETA3 *)
        c2=Cos[THETA2];
        s2=Sin[THETA2];

        s3 = (-A12*s2*(c5*h33 - s5*(c6*h31 - h32*s6)) -
                (c2*(S4 + A56*s5 + c5*(h13*rx + h23*ry + h33*rz - S6) -
                s5*c6*(h11*rx + h21*ry + h31*rz) + s5*s6*(h12*rx + h22*ry + h32*rz))))/
                (A23*c2 + A12);

        c3 = ((A23 + A12*c2)*(c5*h33 - s5*(c6*h31 - h32*s6)) -
            (s2*(S4 + A56*s5 + c5*(h13*rx + h23*ry + h33*rz - S6) -
             c6*s5*(h11*rx + h21*ry + h31*rz) + s5*s6*(h12*rx + h22*ry + h32*rz))))/
            (A23*c2 + A12);

      THETA3=ArcTan[c3,s3];
      T3DEG=THETA3*180/3.141592654;

 (* Caluculate THETA4 *)
      c3=Cos[THETA3];
      s3=Sin[THETA3];

      c4= -((-(c5*c6*h31) - h33*s5 + c5*h32*s6)/(c3*s2 + c2*s3));
      s4= -((-(c6*h32) - h31*s6)/(c3*s2 + c2*s3));

      THETA4=ArcTan[c4,s4];
      T4DEG=THETA4*180/3.141592654;

 (* Caluculate THETA1 *)
      c4=Cos[THETA4];
      s4=Sin[THETA4];

      s1 = -s4*(-c5*h23 + s5*(c6*h21 - h22*s6))/
            (c6*(h11*h23 - h13*h21) + s6*(h13*h22 - h12*h23));

      c1 = -s4*(-c5*h13 + s5*(c6*h11 - h12*s6))/
            (c6*(h11*h23 - h13*h21) + s6*(h13*h22 - h12*h23));

      THETA1=ArcTan[c1,s1];
      T1DEG=THETA1*180/3.141592654;

!DEL INVERSEM.OUT;
Save["INVERSEM.OUT",T1DEG,T2DEG,T3DEG,T4DEG,T5DEG,T6DEG,P];
```

INVERSEM.OUT

```
T1DEG = {-49.00635382239857, -37.18231794604589, 178.3324545510509,
   179.9032637860739, 0.6152643332174978, 13.46258555178579,
   25.15777820029928, 49.09852647728775, 44.05336881514538, 31.6256330981588,
   3.412008538073653, -179.3568204418134, 0.4349512132359644,
   -179.8984525526339, -34.56200042789544, -44.31604697128691}
T2DEG = {67.21288039439827, -121.7762655157024, -119.4351932203574,
   96.0748594739686, 77.18601364136914, -121.5988967872134,
   -121.6020291255381, 68.29417359724966, 35.55852175450018,
   -134.5336985877863, -135.0062841044153, -108.3634210669449,
   19.06556833988468, 145.3490229783739, -133.9531917592707, 37.0708287745248}
T3DEG = {-96.3342443638869, -74.60839988429674, -54.48568850388212,
   -125.5106538851755, -108.7958459168111, -71.51889289804564,
   -72.43241315583686, -96.9310628322546, -83.0657493892221,
   -107.3797830111226, -115.8500207324888, -16.29019224714765,
   -64.06308907990121, -163.7132694998171, -105.354428750584,
   -83.6684072576426}
T4DEG = {-86.5005374625514, 53.11995439514441, -177.6416360759162,
   179.6376456747154, 0.7094956819571085, -17.75439763812941,
   -33.98486702494515, 85.0196439597755, 113.6261334833841,
   -136.3585504216986, -175.6314022857869, 0.7442959668942423,
   179.5080156927684, -0.2395299979727924, 131.3193934241231,
   -112.4618517083648}
T5DEG = {31.2594680924033, -148.6594474128667, -143.074768817735,
   72.43445679486075, 11.39082919567854, -149.6610408362552,
   -149.285316930886, 30.94491135684386, 30.91447056910887,
   -149.193062454955, -152.0354582851038, -167.6521769337595,
   1.997597064166153, -61.3640437910521, -148.6354044865244,
   31.28747545410415}
T6DEG = {-165.4183227446371, -139.3202522523882, -120.3661198352513,
   -119.7388928027854, -119.4547411983012, -114.1758110876201,
   -108.5355393652511, -75.56725945286386, -37.88101884285953,
   -3.22391780042203, 53.94323817773084, 59.10224040890115,
   59.48992444070475, 60.11065333402549, 131.2579061595697, 160.1018891297819}
P = 369178046180899.9 + 1.31079834426321*10^16*X - 5088466105328056.*X^2 -
   1.666316861335004*10^17*X^3 + 4.348450668880128*10^16*X^4 +
   6.799196837852184*10^17*X^5 + 5.299981169283649*10^16*X^6 -
   1.243907297337772*10^18*X^7 - 6.167904665150016*10^17*X^8 +
   8.52539556321953*10^17*X^9 + 8.57777164461476*10^17*X^10 +
   1.114554392322798*10^17*X^11 - 1.427429405112863*10^17*X^12 -
   5.745328983268128*10^16*X^13 - 2465802583532064.*X^14 +
   1698120493221310.*X^15 + 176116158444713.5*X^16
```

Appendix D
C Files

The files INVERSE.C, POLY.C, COMPLEX.C, INVERSE.H, and COMPLEX.H were written to show a C code implementation of the inverse kinematics. The structure of INVERSE.C follows very closely that of the Mathematica input file INVERSEM.M. The primary differences are that the subpolynomials are defined in POLY.C. POLY.C also includes a subroutine which performs the multiplication of four second order polynomials and two fourth order polynomials, which is the form of the characteristic polynomial. INVERSE.C utilizes two subroutines, namely LAGUER and ZROOTS, taken with slight modification from *Numerical Recipes in C* [Press, Flannery, et al., 1992]. COMPLEX.C and COMPLEX.H provide the complex math support for these routines. ZROOTS is used to root the characteristic polynomial. INVERSE.H supplies the desired hand position and orientation to INVERSE.C. The output file INVERSE.OUT gives the number of real solutions and lists the solutions for the input hand position and orientation.

INVERSE.C

```
#include <stdio.h>
#include <math.h>
#include "inverse.h"
#include "complex.h"
#define EPSS 1.0e-14
#define MR 8
#define MT 10
#define MAXIT (MT*MR)
#define MAXM 17
#define EPS 2.0e-14
#define TRUE 1
#define FALSE 0

#define RAD (0.0174532925)

void POLYMULT(double *,double *,double *,double *,double *,double *,double *);

void POLY(double *,double *,double *,double *,double *,double *,double *,double *);

void POLYSOLVE (double [] , double [], int , double *);

void main(void)
{
       void zroots(fcomplex a[], int m, fcomplex roots[], int polish);

       double L0[3],L1[3],L2[3],M0[5],M1[5],M2[5],M3[5],M4[5],TEMP[17];
       double  l0,l1,l2,m0,m1,m2,m3,m4,denominator;
       double  TAN6[16],TAN5[16],T6[16],T5[16],T2[16],T3[16],
                      T4[16],T1[16],TN6,TN62,TN63,TN64;
       double s6,c6,s5,c5,c2,s2,c3,s3,c4,s4,c1,s1;
       extern double A12,A23,A56,S4,r11,r13,
              r21,r23,r33,r31,rx,ry,rz;
       double r12,r22,r32;
       int i,no_of_solns=0;


       fcomplex P[MAXM],roots[MAXM];

       FILE *output;

       r12=r23*r31-r33*r21;
       r22=r11*r33-r31*r13;
```

```
        r32=r13*r21-r11*r23;

        /* DEFINE SUBPOLYNOMIALS */
        POLY(L0,L1,L2,M0,M1,M2,M3,M4);

        /* MULTIPLY AND ADD SUBPOLYNOMIALS TO GET FINAL POLY */
        POLYMULT(TEMP,L2,L2,L2,L2,M0,M0);
                for(i=0;i<17;i++)P[i].r=TEMP[i];
        POLYMULT(TEMP,L1,L2,L2,L2,M0,M1);
                for(i=0;i<17;i++)P[i].r-=TEMP[i];
        POLYMULT(TEMP,L0,L2,L2,L2,M1,M1);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L1,L1,L2,L2,M0,M2);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L0,L2,L2,L2,M0,M2);
                for(i=0;i<17;i++)P[i].r-=2.0*TEMP[i];
        POLYMULT(TEMP,L0,L1,L2,L2,M1,M2);
                for(i=0;i<17;i++)P[i].r-=TEMP[i];
        POLYMULT(TEMP,L0,L0,L2,L2,M2,M2);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L1,L1,L1,L2,M0,M3);
                for(i=0;i<17;i++)P[i].r-=TEMP[i];
        POLYMULT(TEMP,L0,L1,L2,L2,M0,M3);
                for(i=0;i<17;i++)P[i].r+=3.0*TEMP[i];
        POLYMULT(TEMP,L0,L1,L1,L2,M1,M3);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L0,L0,L2,L2,M1,M3);
                for(i=0;i<17;i++)P[i].r-=2.0*TEMP[i];
        POLYMULT(TEMP,L0,L0,L1,L2,M2,M3);
                for(i=0;i<17;i++)P[i].r-=TEMP[i];
        POLYMULT(TEMP,L0,L0,L0,L2,M3,M3);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L1,L1,L1,L1,M0,M4);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L0,L1,L1,L2,M0,M4);
                for(i=0;i<17;i++)P[i].r-=4.0*TEMP[i];
        POLYMULT(TEMP,L0,L0,L2,L2,M0,M4);
                for(i=0;i<17;i++)P[i].r+=2.0*TEMP[i];
        POLYMULT(TEMP,L0,L1,L1,L1,M1,M4);
                for(i=0;i<17;i++)P[i].r-=TEMP[i];
        POLYMULT(TEMP,L0,L0,L1,L2,M1,M4);
                for(i=0;i<17;i++)P[i].r+=3.0*TEMP[i];
        POLYMULT(TEMP,L0,L0,L1,L1,M2,M4);
                for(i=0;i<17;i++)P[i].r+=TEMP[i];
        POLYMULT(TEMP,L0,L0,L0,L2,M2,M4);
                for(i=0;i<17;i++)P[i].r-=2.0*TEMP[i];
        POLYMULT(TEMP,L0,L0,L0,L1,M3,M4);
                for(i=0;i<17;i++)P[i].r-=TEMP[i];
        POLYMULT(TEMP,L0,L0,L0,L0,M4,M4);
                for(i=0;i<17;i++){
                        P[i].r+=TEMP[i];
                        P[i].i=0.0;
                }

/* SOLVE FINAL POLY FOR TAN6 */
        zroots(P,16,roots,FALSE);
/* CALCULATE JOINT ANGLES */
        /* ASSIGN REAL ROOTS TO TAN6 */
        for(i=1;i<=16;i++)if((roots[i].i*roots[i].i)<0.000001)
                        TAN6[no_of_solns++]=roots[i].r;

        /* CALCULATE TAN5 */
        for(i=0;i<no_of_solns;i++){

                TN6=TAN6[i];
                TN62=TN6*TN6;
                TN63=TN6*TN62;
                TN64=TN6*TN63;

                l0=L0[0]+L0[1]*TN6+L0[2]*TN62;
                l1=L1[0]+L1[1]*TN6+L1[2]*TN62;
        l2=L2[0]+L2[1]*TN6+L2[2]*TN62;
        m0=M0[0]+M0[1]*TN6+M0[2]*TN62+M0[3]*TN63+M0[4]*TN64;
        m1=M1[0]+M1[1]*TN6+M1[2]*TN62+M1[3]*TN63+M1[4]*TN64;
        m2=M2[0]+M2[1]*TN6+M2[2]*TN62+M2[3]*TN63+M2[4]*TN64;
        m3=M3[0]+M3[1]*TN6+M3[2]*TN62+M3[3]*TN63+M3[4]*TN64;
        m4=M4[0]+M4[1]*TN6+M4[2]*TN62+M4[3]*TN63+M4[4]*TN64;

        TAN5[i]=(-l2*l2*l2*m0+l0*l2*l2*m2-l0*l1*l2*m3+
                l0*l1*l1*m4-l0*l0*l2*m4)/(l2*l2*l2*m1-
                l1*l2*l2*m2+l1*l1*l2*m3-l0*l2*l2*m3-l1*l1*l1*m4+
                2.0*l0*l1*l2*m4);
}

for(i=0;i<no_of_solns;i++){
        T6[i]=2.0*atan(TAN6[i]);
        T5[i]=2.0*atan(TAN5[i]);
}

/* CALCULATE T2,T3,T4,T1 */

for(i=0;i<no_of_solns;i++){

        c5=cos(T5[i]);
        s5=sin(T5[i]);
        c6=cos(T6[i]);
        s6=sin(T6[i]);

/* T2 */
        c2 =-A12*A12 - A23*A23 + A56*A56 + S4*S4 + rx*rx + ry*ry + rz*rz +
                2.0*(A56*S4*s5 + A56*(-c6*(r11*rx + r21*ry + r31*rz) +
                s6*(r12*rx + r22*ry + r32*rz)) +
                S4*(c5*(r13*rx + r23*ry + r33*rz) - c6*s5*(r11*rx +
                r21*ry + r31*rz) + s5*s6*(r12*rx + r22*ry + r32*rz)));
        s2 = 2.0*A12*(rz + A56*(r32*s6 - c6*r31) +
                S4*(c5*r33 + s5*(-c6*r31 + r32*s6)));
        T2[i]=atan2(s2,c2);

        c2=cos(T2[i]);
        s2=sin(T2[i]);

/* T3 */
        denominator=A23*c2 + A12;
        s3 = (-A12*s2*(c5*r33 - s5*(c6*r31 - r32*s6)) -
                (c2*(S4 + A56*s5 + c5*(r13*rx + r23*ry + r33*rz) -
                s5*c6*(r11*rx + r21*ry + r31*rz) +
                s5*s6*(r12*rx + r22*ry + r32*rz))))/denominator;
        c3 = ((A23 + A12*c2)*(c5*r33 - s5*(c6*r31 - r32*s6)) -
                (s2*(S4 + A56*s5 + c5*(r13*rx + r23*ry + r33*rz) -
                c6*s5*(r11*rx + r21*ry + r31*rz) +
                s5*s6*(r12*rx + r22*ry + r32*rz))))/denominator;
        T3[i]=atan2(s3,c3);

        c3=cos(T3[i]);
        s3=sin(T3[i]);

/* T4 */
        denominator=c3*s2 + c2*s3;
        c4 = (c5*(c6*r31 - r32*s6) + r33*s5)/denominator;
        s4 = (c6*r32 + r31*s6)/denominator;
        T4[i]=atan2(s4,c4);

        c4=cos(T4[i]);
        s4=sin(T4[i]);

/* T1 */
        denominator=(c6*(r11*r23 - r13*r21) + s6*(r13*r22 - r12*r23));
        s1 = -s4*(-c5*r23 + s5*(c6*r21 - r22*s6))/denominator;
        c1 = -s4*(-c5*r13 + s5*(c6*r11 - r12*s6))/denominator;
        T1[i]=atan2(s1,c1);

}
```

```
        output=fopen("inverse.out","wt");

        fprintf(output,"Number of Solutions: %i\n",no_of_solns);
        fprintf(output,"        T1        T2        T3        T4        T5        T6\n");
        for(i=0;i<no_of_solns;i++){
                fprintf(output,"%15.9f,%15.9f,%15.9f,%15.9f,%15.9f,%15.9f\n",
                T1[i]/RAD,T2[i]/RAD,T3[i]/RAD,T4[i]/RAD,T5[i]/RAD,T6[i]/RAD);
        }

        fclose(output);


}
void zroots(fcomplex a[], int m, fcomplex roots[], int polish)
{
        void laguer(fcomplex a[], int m, fcomplex *x, int *its);
        int i,its,j,jj;
        fcomplex x,b,c,ad[MAXM];

        for(j=0;j<=m;j++)ad[j]=a[j];
        for(j=m;j>=1;j--){
                x=Complex(0.0,0.0);
                laguer(ad,j,&x,&its);
                if(fabs(x.i) <= 2.0*EPS*fabs(x.r))x.i=0.0;
                roots[j]=x;
                b=ad[j];
                for(jj=j-1;jj>=0;jj--){
                        c=ad[jj];
                        ad[jj]=b;
                        b=Cadd(Cmul(x,b),c);
                }
        }
        if(polish)
                for(j=1;j<=m;j++)
                        laguer(a,m,&roots[j],&its);
/*      for(j=2;j<=m;j++){
                x=roots[j];
                for(i=j-1;i>=1;i--){
                        if(roots[i].r<=x.r) break;
                        roots[i+1]=roots[i];
                }
                roots[i+1]=x;
        }
*/
}

void laguer(fcomplex a[], int m, fcomplex *x, int *its)
{
        double fmax (double value1, double value2);
        int iter,j;
        double abx,abp,abm,err;

        fcomplex dx,x1,b,d,f,g,h,sq,gp,gm,g2;
        static double frac[MR+1]={0.0,0.5,0.25,0.75,0.13,0.38,0.62,0.88,1.0};

        for(iter=1;iter<=MAXIT;iter++){
                *its=iter;
                b=a[m];
                err=Cabs(b);
                d=f=Complex(0.0,0.0);
                abx=Cabs(*x);
                for(j=m-1;j>=0;j--){
                        f=Cadd(Cmul(*x,f),d);
                        d=Cadd(Cmul(*x,d),b);
                        b=Cadd(Cmul(*x,b),a[j]);
                        err=Cabs(b)+abx*err;
                }
                err*=EPSS;
                if(Cabs(b)<=err) return;
                g=Cdiv(d,b);
                g2=Cmul(g,g);
                h=Csub(g2,RCmul(2.0,Cdiv(f,b)));
                sq=Csqrt(RCmul((double) (m-1),Csub(RCmul((double) m,h),g2)));
                gp=Cadd(g,sq);
                gm=Csub(g,sq);
                abp=Cabs(gp);
                abm=Cabs(gm);
                if(abp<abm) gp=gm;
                dx=((fmax(abp,abm)>0.0 ? Cdiv(Complex((double) m,0.0),gp)
                                :
        RCmul(exp(log(1+abx)),Complex(cos((double)iter),sin((double)iter)))));
                x1=Csub(*x,dx);
                if(x->r==x1.r&&x->i==x1.i)return;
                if(iter % MT) *x=x1;
                else *x=Csub(*x,RCmul(frac[iter/MT],dx));
        }
        printf("too many iterati'ns in laguer");
        return;
}

double fmax(double value1, double value2)
{
   return ( (value1 > value2) ? value1 : value2);
}
```

POLY.C

```
#include <math.h>

void POLY(double *L0, double *L1, double *L2, double *M0,
                        double *M1,double *M2, double *M3, double *M4)
{
        double D11,D12,D13,D14,D15,D23,D24,D25,D27,D28,
                        D29,D33,D34,D35,D36,D37,D38,D39;
        extern double A12, A23, A56, S4;
        extern double r11,r13,r21,r23,r33,r31,rx,ry,rz;
        double r12,r22,r32;

        r12=r23*r31-r33*r21;
        r22=r11*r33-r31*r13;
        r32=r13*r21-r11*r23;

        D11 = -A56*r32;
        D12 = -A56*r31;
        D13 = -r23*rx + r13*ry;
        D14 = r21*rx - r11*ry;
        D15 = -r22*rx + r12*ry;
        D23 = 2.0*A12*S4*r33;
        D24 = -2.0*A12*S4*r31;
        D25 = 2.0*A12*S4*r32;
        D27 = -2.0*A12*A56*r31;
        D28 = 2.0*A12*A56*r32;
        D29 = 2.0*A12*rz;
        D33 = 2.0*S4*(r13*rx + r23*ry + r33*rz);
        D34 = -2.0*S4*(r11*rx + r21*ry + r31*rz);
        D35 = 2.0*S4*(r12*rx + r22*ry + r32*rz);
        D36 = 2.0*A56*S4;
        D37 = -2.0*A56*(r11*rx + r21*ry + r31*rz);
        D38 = 2.0*A56*(r12*rx + r22*ry + r32*rz);
        D39 = -A12*A12 - A23*A23 + A56*A56 + S4*S4 + rx*rx + ry*ry + rz*rz;

        L2[2] = D11 - D13;
        L2[1] = -2.0*D12;
        L2[0] = -D11 - D13;
        L1[2] = -2.0*D14;
        L1[1] = 4.0*D15;
        L1[0] = 2.0*D14;
        L0[2] = -D11 + D13;
        L0[1] = 2.0*D12;
        L0[0] = D11 + D13;
        M4[4] = -4.0*A12*A12*A23*A23 + D23*D23 + 2.0*D23*D27 + D27*D27 -
                2.0*D23*D29 - 2.0*D27*D29 + D29*D29 + D33*D33 + 2.0*D33*D37 +
                D37*D37 - 2.0*D33*D39 - 2.0*D37*D39 + D39*D39;
        M4[3] = -4.0*D23*D28 - 4.0*D27*D28 + 4.0*D28*D29 - 4.0*D33*D38 -
                4.0*D37*D38 + 4.0*D38*D39;
        M4[2] = -8.0*A12*A12*A23*A23 + 2.0*D23*D23 - 2.0*D27*D27 + 4.0*D28*D28 -
                4.0*D23*D29 + 2.0*D29*D29 + 2.0*D33*D33 - 2.0*D37*D37 + 4.0*D38*D38 -
                4.0*D33*D39 + 2.0*D39*D39;
        M4[1] = -4.0*D23*D28 + 4.0*D27*D28 + 4.0*D28*D29 - 4.0*D33*D38 +
                4.0*D37*D38 + 4.0*D38*D39;
        M4[0] = -4.0*A12*A12*A23*A23 + D23*D23 - 2*D23*D27 + D27*D27 -
                2.0*D23*D29 + 2.0*D27*D29 + D29*D29 + D33*D33 - 2.0*D33*D37 +
                D37*D37 - 2.0*D33*D39 + 2.0*D37*D39 + D39*D39;
        M3[4] = 4.0*D23*D24 + 4.0*D24*D27 - 4.0*D24*D29 + 4.0*D33*D34 -
                4.0*D33*D36 + 4.0*D34*D37 - 4.0*D36*D37 - 4.0*D34*D39 + 4.0*D36*D39;
        M3[3] = -8.0*D23*D25 - 8.0*D25*D27 - 8.0*D24*D28 + 8.0*D25*D29 -
                8.0*D33*D35 - 8.0*D35*D37 - 8.0*D34*D38 + 8.0*D36*D38 + 8.0*D35*D39;
        M3[2] = -8.0*D24*D27 + 16*D25*D28 - 8.0*D33*D36 - 8.0*D34*D37 +
                16.0*D35*D38 + 8.0*D36*D39;
        M3[1] = -8.0*D23*D25 + 8.0*D25*D27 + 8.0*D24*D28 + 8.0*D25*D29 -
                8.0*D33*D35 + 8.0*D35*D37 + 8.0*D34*D38 + 8.0*D36*D38 + 8.0*D35*D39;
        M3[0] = -4.0*D23*D24 + 4.0*D24*D27 + 4.0*D24*D29 - 4.0*D33*D34 -
                4.0*D33*D36 + 4.0*D34*D37 + 4.0*D36*D37 + 4.0*D34*D39 + 4.0*D36*D39;
        M2[4] = -8.0*A12*A12*A23*A23 - 2.0*D23*D23 + 4.0*D24*D24 + 2.0*D27*D27 -
                4.0*D27*D29 + 2.0*D29*D29 - 2.0*D33*D33 + 4.0*D34*D34 - 8.0*D34*D36 +
                4.0*D36*D36 + 2.0*D37*D37 - 4.0*D37*D39 + 2.0*D39*D39;
        M2[3] = -16.0*D24*D25 - 8.0*D27*D28 + 8.0*D28*D29 - 16.0*D34*D35 +
                16.0*D35*D36 - 8.0*D37*D38 + 8.0*D38*D39;
        M2[2] = -16.0*A12*A12*A23*A23 - 4.0*D23*D23 - 8.0*D24*D24 + 16.0*D25*D25 -
                4.0*D27*D27 + 8.0*D28*D28 + 4.0*D29*D29 - 4.0*D33*D33 - 8.0*D34*D34 +
                16.0*D35*D35 + 8.0*D36*D36 - 4.0*D37*D37 + 8.0*D38*D38 + 4.0*D39*D39;
        M2[1] = 16.0*D24*D25 + 8.0*D27*D28 + 8.0*D28*D29 + 16.0*D34*D35 +
                16.0*D35*D36 + 8.0*D37*D38 + 8.0*D38*D39;
        M2[0] = -8.0*A12*A12*A23*A23 - 2.0*D23*D23 + 4.0*D24*D24 + 2.0*D27*D27 +
                4.0*D27*D29 + 2.0*D29*D29 - 2.0*D33*D33 + 4.0*D34*D34 + 8.0*D34*D36 +
                4.0*D36*D36 + 2.0*D37*D37 + 4.0*D37*D39 + 2.0*D39*D39;
        M1[4] = -4.0*D23*D24 + 4.0*D24*D27 - 4.0*D24*D29 - 4.0*D33*D34 + 4.0*D33*D36 +
                4.0*D34*D37 - 4.0*D36*D37 - 4.0*D34*D39 + 4.0*D36*D39;
        M1[3] = 8.0*D23*D25 - 8.0*D25*D27 - 8.0*D24*D28 + 8.0*D25*D29 + 8.0*D33*D35 -
                8.0*D35*D37 - 8.0*D34*D38 + 8.0*D36*D38 + 8.0*D35*D39;
        M1[2] = -8.0*D24*D27 + 16.0*D25*D28 + 8.0*D33*D36 - 8.0*D34*D37 +
                16.0*D35*D38 + 8.0*D36*D39;
        M1[1] = 8.0*D23*D25 + 8.0*D25*D27 + 8.0*D24*D28 + 8.0*D25*D29 + 8.0*D33*D35 +
                8.0*D35*D37 + 8.0*D34*D38 + 8.0*D36*D38 + 8.0*D35*D39;
        M1[0] = 4.0*D23*D24 + 4.0*D24*D27 + 4.0*D24*D29 + 4.0*D33*D34 + 4.0*D33*D36 +
                4.0*D34*D37 + 4.0*D36*D37 + 4.0*D34*D39 + 4.0*D36*D39;
        M0[4] = -4.0*A12*A12*A23*A23 + D23*D23 - 2.0*D23*D27 + D27*D27 +
                2.0*D23*D29 - 2.0*D27*D29 + D29*D29 + D33*D33 - 2.0*D33*D37 +
                D37*D37 + 2.0*D33*D39 - 2.0*D37*D39 + D39*D39;
        M0[3] = 4.0*D23*D28 - 4.0*D27*D28 + 4.0*D28*D29 + 4.0*D33*D38 - 4.0*D37*D38 +
                4.0*D38*D39;
         M0[2] = -8.0*A12*A12*A23*A23 + 2.0*D23*D23 - 2.0*D27*D27 + 4.0*D28*D28 +
                4.0*D23*D29 + 2.0*D29*D29 + 2.0*D33*D33 - 2.0*D37*D37 + 4.0*D38*D38 +
                4.0*D33*D39 + 2.0*D39*D39;
        M0[1] = 4.0*D23*D28 + 4.0*D27*D28 + 4.0*D28*D29 + 4.0*D33*D38 + 4.0*D37*D38 +
                4.0*D38*D39;
        M0[0] = -4.0*A12*A12*A23*A23 + D23*D23 + 2.0*D23*D27 + D27*D27 +
                2.0*D23*D29 + 2.0*D27*D29 + D29*D29 + D33*D33 + 2.0*D33*D37 +
                D37*D37 + 2.0*D33*D39 + 2.0*D37*D39 + D39*D39;
}

void POLYMULT( double *RESULT, double *A, double *B, double *C,
                              double *D, double *E, double *F)
{
        double temp,G[9];
        double t1,t2,t3,t4,t5,t6,t7,t8;
        double C2D2,C0D0,B0C0D0,B2C2D2;
        int i,j,k;

        C2D2=C[2]*D[2];
        C0D0=C[0]*D[0];
        B0C0D0=B[0]*C[0]*D[0];
        B2C2D2=B[2]*C[2]*D[2];

        t1=C[1]*D[0]+C[0]*D[1];
        t2=B[1]*C0D0+B[0]*t1;
        t3=C[2]*D[1]+C[1]*D[2];
        t4=C[2]*D[0]+C[1]*D[1]+C[0]*D[2];
        t5=B[2]*C0D0+B[1]*t1+B[0]*t4;
        t6=B[2]*t3+B[1]*C2D2;
        t7=B[2]*t1+B[1]*t4+B[0]*t3;
        t8=B[2]*t4+B[1]*t3+B[0]*C2D2;

        G[0]=A[0]*B0C0D0;
        G[1]=A[1]*B0C0D0+A[0]*t2;
        G[2]=A[2]*B0C0D0+A[1]*t2+A[0]*t5;
        G[3]=A[2]*t2+A[1]*t5+A[0]*t7;
        G[4]=A[2]*t5+A[1]*t7+A[0]*t8;
        G[5]=A[2]*t7+A[1]*t8+A[0]*t6;
        G[6]=A[0]*B2C2D2+A[2]*t8+A[1]*t6;
        G[7]=A[1]*B2C2D2+A[2]*t6;
        G[8]=A[2]*B2C2D2;

        for(i=0;i<17;i++)RESULT[i]=0;

        for(i=0;i<9;i++){
                for(j=0;j<5;j++){
                        temp=G[i]*E[j];
                        for(k=0;k<5;k++)RESULT[i+j+k]+=temp*F[k];
                }
        }
}
```

COMPLEX.C

```
#include <math.h>

typedef struct FCOMPLEX {double r,i;} fcomplex;

fcomplex Cadd(fcomplex a,fcomplex b)
{       fcomplex c;
        c.r=a.r+b.r;
        c.i=a.i+b.i;
        return c;
}

fcomplex Csub(fcomplex a,fcomplex b)
{       fcomplex c;
        c.r=a.r-b.r;
        c.i=a.i-b.i;
        return c;
}

fcomplex Cmul(fcomplex a,fcomplex b)
{       fcomplex c;
        c.r=a.r*b.r-a.i*b.i;
        c.i=a.i*b.r+a.r*b.i;
        return c;
}

fcomplex Complex(double re,double im)
{       fcomplex c;
        c.r=re;
        c.i=im;
        return c;
}

fcomplex Conjg(fcomplex z)
{       fcomplex c;
        c.r=z.r;
        c.i = -z.i;
```

```
	return c;
}

fcomplex Cdiv(fcomplex a,fcomplex b)
{	fcomplex c;
	double r,den;
	if (fabs(b.r) >= fabs(b.i)) {
		r=b.i/b.r;
		den=b.r+r*b.i;
		c.r=(a.r+r*a.i)/den;
		c.i=(a.i-r*a.r)/den;
	} else {
		r=b.r/b.i;
		den=b.i+r*b.r;
		c.r=(a.r*r+a.i)/den;
		c.i=(a.i*r-a.r)/den;
	}
	return c;
}

double Cabs(fcomplex z)
{	double x,y,ans,temp;
	x=fabs(z.r);
	y=fabs(z.i);
	if (x == 0.0)
		ans=y;
	else if (y == 0.0)
		ans=x;
	else if (x > y) {
		temp=y/x;
		ans=x*sqrt(1.0+temp*temp);
	} else {
		temp=x/y;
		ans=y*sqrt(1.0+temp*temp);
	}
	return ans;
}

fcomplex Csqrt(fcomplex z)
{	fcomplex c;
	double x,y,w,r;
	if ((z.r == 0.0) && (z.i == 0.0)) {
		c.r=0.0;
		c.i=0.0;
		return c;
	} else {
		x=fabs(z.r);
		y=fabs(z.i);
		if (x >= y) {
			r=y/x;
			w=sqrt(x)*sqrt(0.5*(1.0+sqrt(1.0+r*r)));
		} else {
			r=x/y;
			w=sqrt(y)*sqrt(0.5*(r+sqrt(1.0+r*r)));
		}
		if (z.r >= 0.0) {
			c.r=w;
			c.i=z.i/(2.0*w);
		} else {
			c.i=(z.i >= 0) ? w : -w;
			c.r=z.i/(2.0*c.i);
		}
		return c;
	}
}

fcomplex RCmul(double x,fcomplex a)
{	fcomplex c;
	c.r=x*a.r;
	c.i=x*a.i;
	return c;
```

```
}
```

INVERSE.H

```
double r11=-0.35947333851404,
       r21=-0.86861871849252,
       r31=0.34099918003125,
       r13=0.68198091541137,
       r23=0.00487792400620,
       r33=0.73135370161917,
       rx=13.000000000000,
       ry=0.000000000000,
       rz=-4.00000000000,

       A12= 14.000,
       A23= 31.125,
       A56= 11.500,
       S4= 31.125;
```

COMPLEX.H

```
typedef struct FCOMPLEX {double r,i;} fcomplex;

extern fcomplex Cadd(fcomplex a,fcomplex b);
extern fcomplex Csub(fcomplex a,fcomplex b);
extern fcomplex Cmul(fcomplex a,fcomplex b);
extern fcomplex Complex(double re,double im);
extern fcomplex Conjg(fcomplex z);
extern fcomplex Cdiv(fcomplex a,fcomplex b);
extern double Cabs(fcomplex z);
extern fcomplex Csqrt(fcomplex z);
extern fcomplex RCmul(double x,fcomplex a);
```

INVERSE.OUT

```
Number of Solutions:16
T1,T2,T3,T4,T5,T6
-49.006353885,67.212880480,-96.334244487,-86.500537573,31.259468132,-165.418322955
-44.316047028,37.070828822,-83.668407364,-112.461851852,31.287475494,160.101889334
-37.182317994,-121.776265671,-74.608399979,53.119954463,-148.659447602,-139.320252430
-34.562000472,-133.953191930,-105.354428885,131.319393591,-148.635404676,131.257906327
178.332436651,-119.435177651,-54.485536789,-177.641618494,-143.074718376,-120.366119716
179.903252759,96.074374834,-125.508425033,179.637635787,72.433425546,-119.738893995
0.615266258,77.186030872,-108.795801157,0.709498085,11.390898248,-119.454740572
-179.898452782,145.349023621,-163.713269538,-0.239529997,-61.364044360,60.110653413
13.462585610,-121.598896925,-71.518892884,-17.754397690,-149.661040976,-114.175811246
25.157778246,-121.602029279,-72.432413241,-33.984867083,-149.285317116,-108.535539502
0.434951218,19.065568006,-64.063089177,179.508015917,1.997597361,59.489924509
49.098526540,68.294173684,-96.931062956,85.019644068,30.944911396,-75.567259549
-179.356822847,-108.363180826,-16.291543991,0.744297201,-167.652314928,59.102240489
44.053368871,35.558521800,-83.065749495,113.626133628,30.914470608,-37.881018891
3.412008542,-135.006284274,-115.850020882,-175.631402509,-152.035458480,53.943238246
   31.625633138,-134.533698759,-107.379783148,-136.358550595,-149.193062645,-3.223917805
```

Vita

Will was born on the 23rd of August 1968 in Hampton, Virginia. There he was raised by his parents, Leonard, an electronics technician, and Marna, a technical staff assistant. He attended Hampton Christian High School where he excelled in math and science. It was in high school that he developed an interest in engineering. The author chose Virginia Tech for its strong engineering reputation and its scenic area. There he continued his interest in engineering, specializing in Aerospace Engineering. During his first semester of graduate school, Will took a Mechanical Engineering robotics course, sparking his interest in robotics. Subsequently, he took advantage of the opportunity to take part in robotics research being performed at the NASA Langley Research Center as a summer intern. Upon returning to Virginia Tech, Will changed his graduate studies from Aerospace to Mechanical Engineering in order to study manipulator kinematics under the tutelage of Dr. Charles F. Reinholtz.

William H. Cordle, III